

RTミドルウェア SUMMER CAMP 2012

日時: 2012年7月30日(月)～8月3日(金)
場所: 産業技術総合研究所 つくばセンター中央第二
本部・情報棟1F ネットワーク会議室



RTミドルウェア SUMMER CAMP 2012(第2日目)

10:00- 10:30	RTコンポーネントの開発手順について
	担当: 安藤 慶昭(産業技術総合研究所)
10:30- 11:30	RTM開発ツール群の紹介
	担当: 坂本 武志(株式会社 グローバルアシスト)
11:30- 12:30	OpenHRIの使い方
	担当: 松坂 要佐(MID)
12:30- 13:30	休憩
13:30- 17:00	グループ実習 1
	担当: 各講師陣

第2部 RTM開発ツール群の紹介

株式会社 グローバルアシスト
坂本 武志



2012.7.30-8.3 RTミドルウェア SUMMER CAMP 2012

3

OpenRT Platform



- ロボット知能ソフトウェアプラットフォーム
 - <http://www.openrtp.jp/wiki/>
 - システム設計, シミュレーション, 動作生成, シナリオ生成などをサポート
- OpenRT Platformツール群
 - コンポーネント開発, システム開発における各開発フェーズの作業支援
 - 開発プラットフォームにEclipseを採用
- 構成
 - RTCビルダ
 - RTCデバッグ
 - RTシステムエディタ
 - ロボット設計支援ツール
 - シミュレータ
 - 動作設計ツール
 - シナリオ作成ツール

など



2012.7.30-8.3 RTミドルウェア SUMMER CAMP 2012

4

RTCBuilder,RTSystemEditorの 新機能について

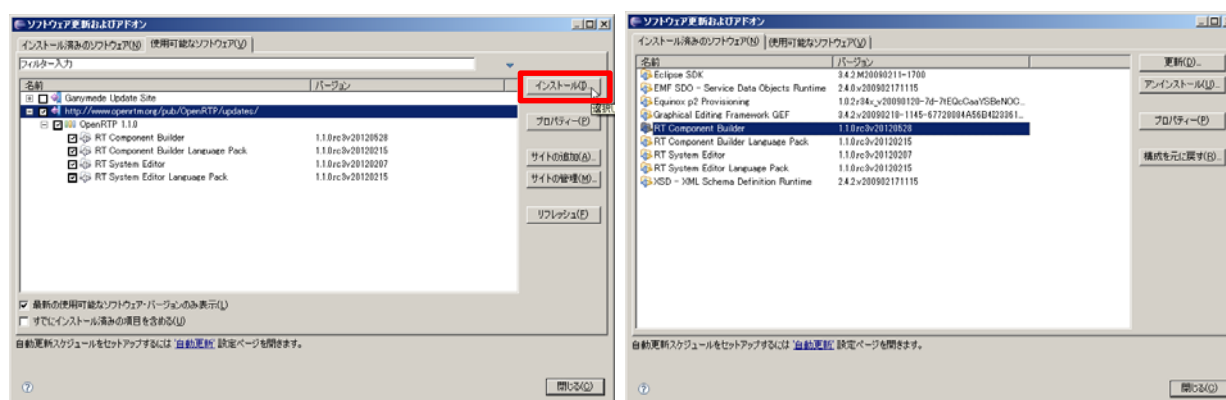


共通



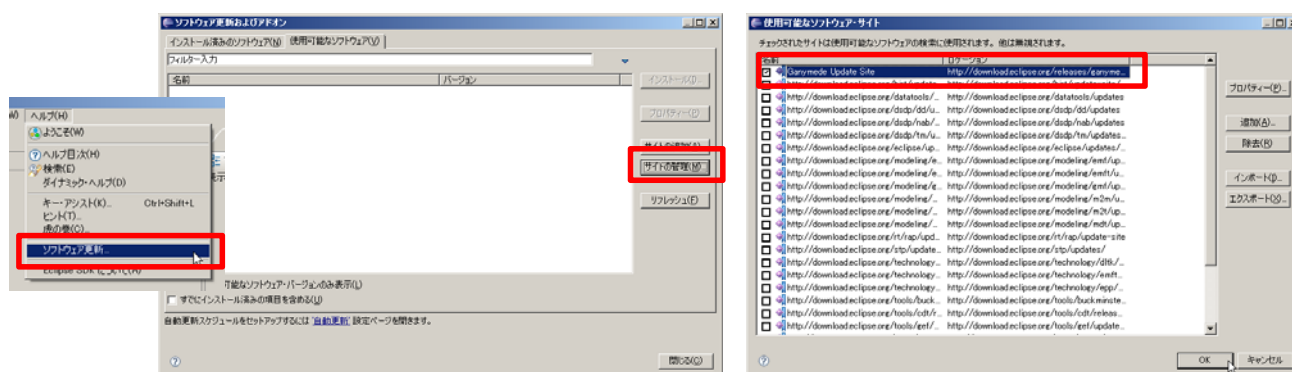
■ 更新サイトへの対応

- 上部メニューから「ヘルプ」→「ソフトウェア更新」を選択
- 表示された【ソフトウェア更新およびアドオン】画面にて、「サイトの追加」を選択し、以下のURLを入力
- <http://www.openrtm.org/pub/OpenRTP/updates/>

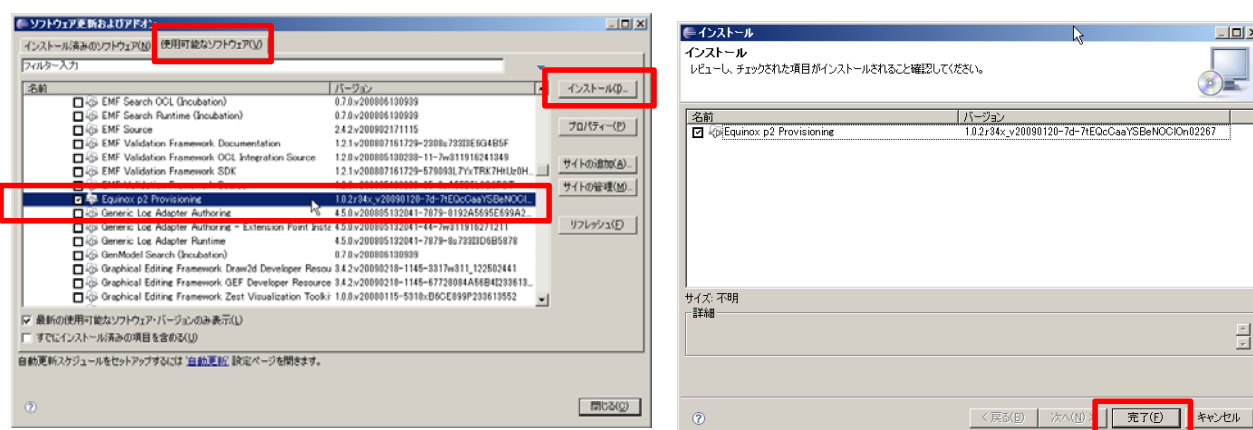


- ※途中、インストール内容の確認や、ライセンス条項の確認画面が表示されます。
- ※インストールが完了すると、再起動を促すメッセージが表示されます。
- ※一度インストールを行うと、「インストール済みのソフトウェア」一覧に表示されるようになります。次回以降はこちらの画面からツールの更新が可能となります。

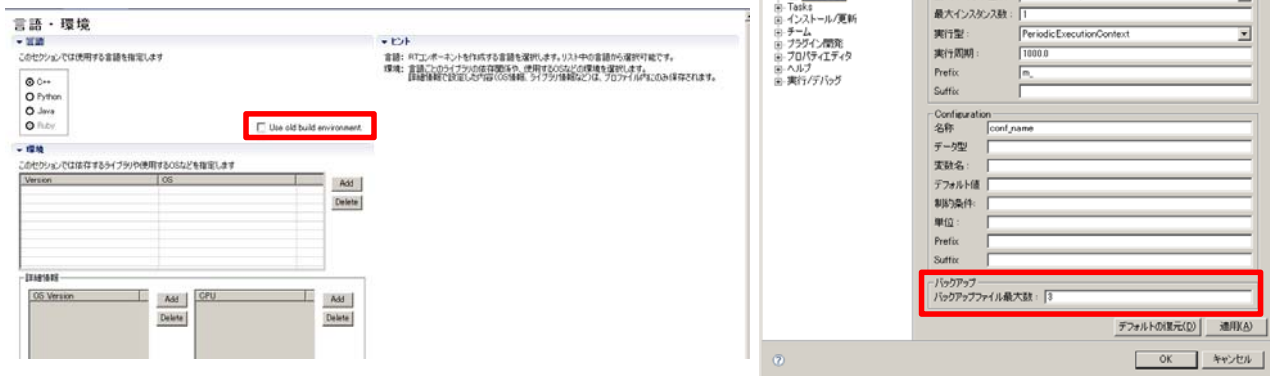
- RC3以前のバージョンの場合、プラグインの追加が必要
- プラグインの追加手順
 - 上部メニューから「ヘルプ」→「ソフトウェア更新」を選択
 - 表示された【ソフトウェア更新およびアドオン】画面にて、「サイトの管理」を選択
 - 表示された【使用可能なソフトウェア・サイト】画面にて、「Ganymede Update Site」をチェックし、「OK」を選択



- 【ソフトウェア更新およびアドオン】画面にて「使用可能なソフトウェア」タブを選択
- 「Ganymede Update Site」→「カテゴリーなし」→「Equinox p2 Provisioning」をチェックし、「インストール」を選択
- 【インストール】画面にて「Equinox p2 Provisioning」が選択されている事を確認し、「完了」を選択
 - 再起動を促す画面が表示されるので、再起動を実行

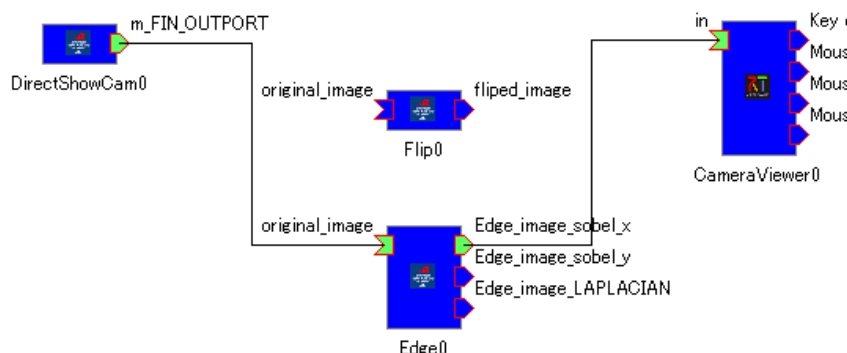


- CMakeへの対応
 - 旧バージョン(VsualStudioプロジェクト形式)のコード生成を行いたい場合は、「言語・環境」タブの「Use old build environment」をチェック
- バックアップファイル数設定機能
 - 設定画面中の「RTCBuilder」-「コード生成」「バックアップファイルの最大数」にてバックアップファイルの数を設定可能



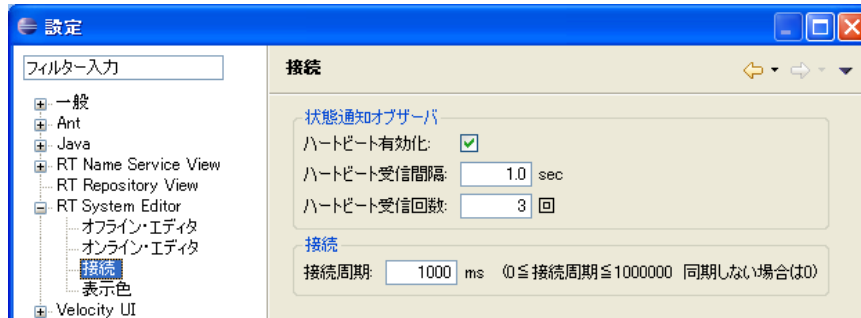
RTSystemEditor

- ポート名の表示
 - エディタ画面内のコンポーネントのポート名を表示
- アイコン表示
 - コンポーネント内に指定したアイコンを表示可能に
 - 表示するアイコンの設定は、設定画面にて指定
 - ID毎, カテゴリ毎の指定が可能



■ 接続一状態通知オブザーバ

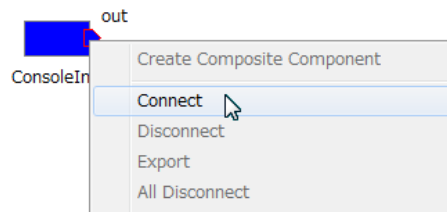
- RTCの生存確認用オブザーバに関する設定
 - RTSE側から生存確認を行うのではなく、RTC側から通知(ハートビート)を行う形
 - OpenRTM-aist-1.1以降で対応



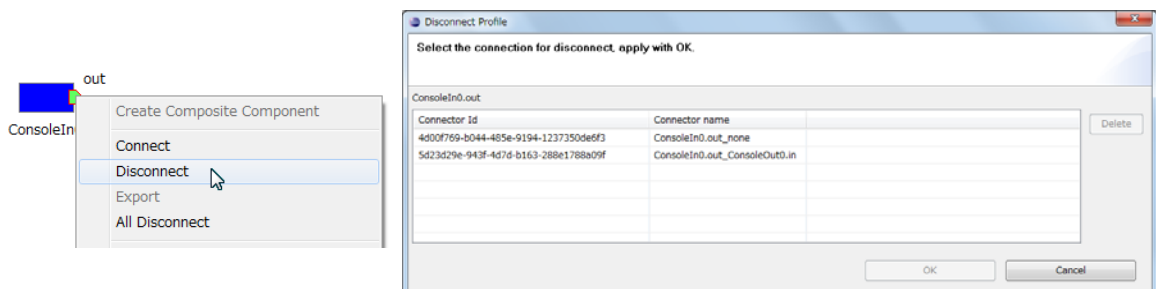
- ハートビート有効化: ハートビートによる生存確認機能の有効化
- ハートビート受信間隔: ハートビートの受信間隔. この間隔以内にRTC側からハートビートが送られてこないで生存確認失敗と判断
- ハートビート受信回数: この回数を超えて生存確認に失敗した場合, 対象RTCに異常が発生したと判断

■ 単独ポートの接続

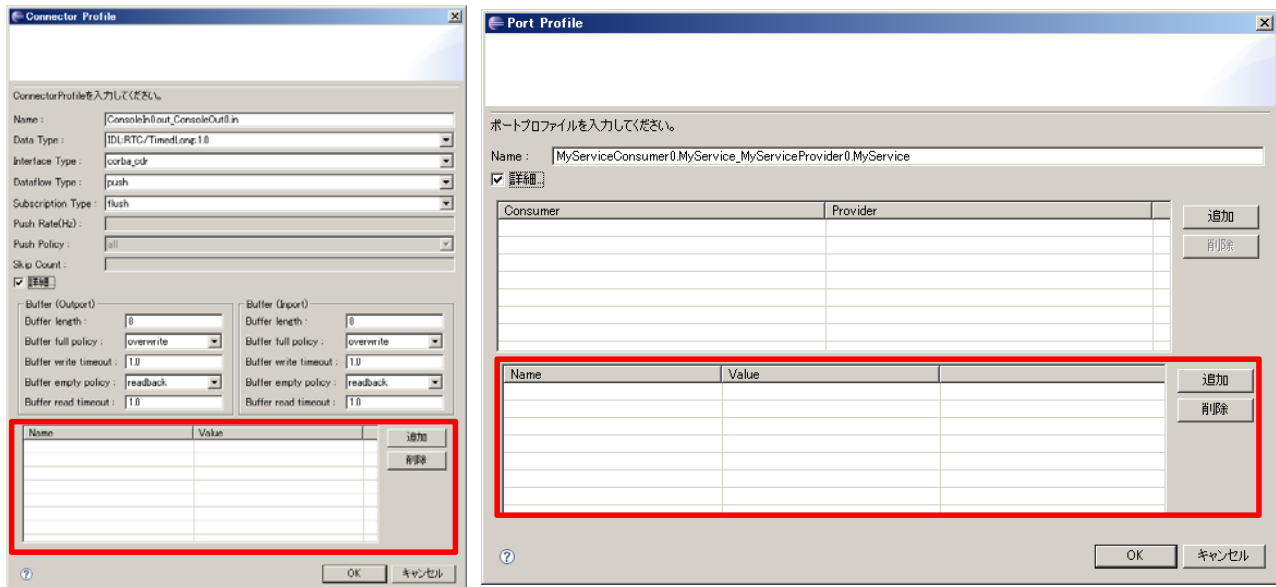
- 単独のポートに対してConnectorProfileを設定可能に



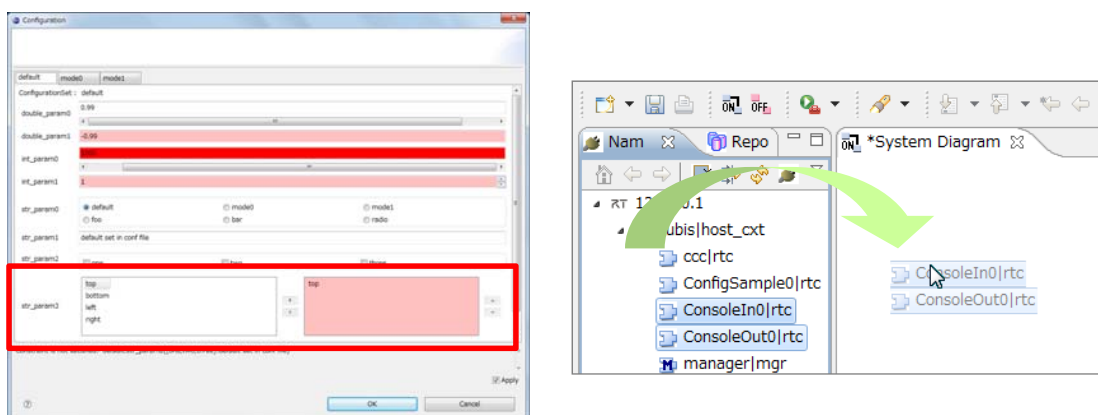
- 切断時には, コンテキストメニュー中から「Disconnect」を選択
 - 切断対象のConnectorProfileを選択



- コネクタプロファイルへ任意の項目が設定可能に
 - 単独のポートに対してConnectorProfileを設定可能に
 - 「詳細」ボタンを選択した際に表示される拡張エリアにて、任意のNVListを指定可能に

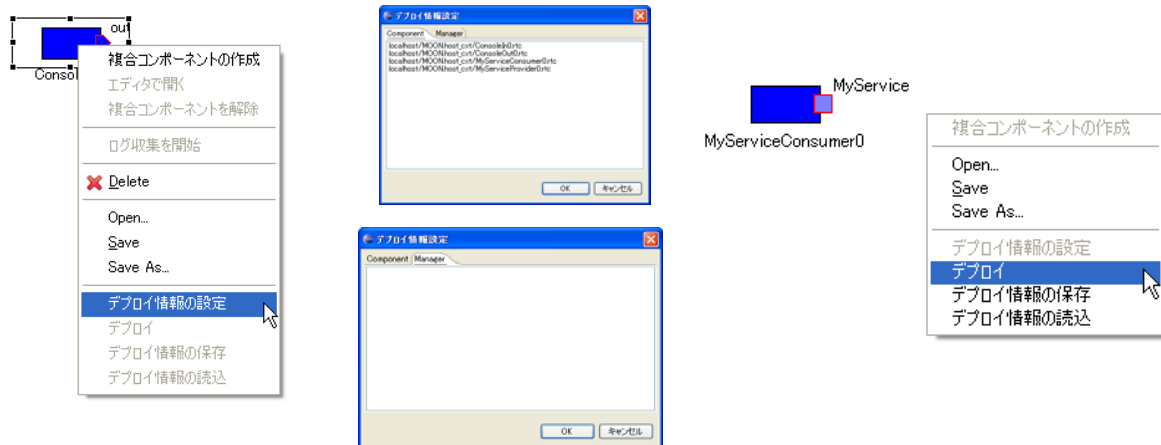


- ConfigurationSet設定時UIにリストボックスを追加
 - 複数項目を設定するためのリストボックスが利用可能に
 - リストボックスの指定は、他のUIと同様RTCBuilderにて設定可能
- 複数RTCのDnD
 - NameServiceViewにて複数のRTCを選択し、エディタ画面に配置可能に
 - RTCの選択は、「Ctrl」キーを押しながら実行

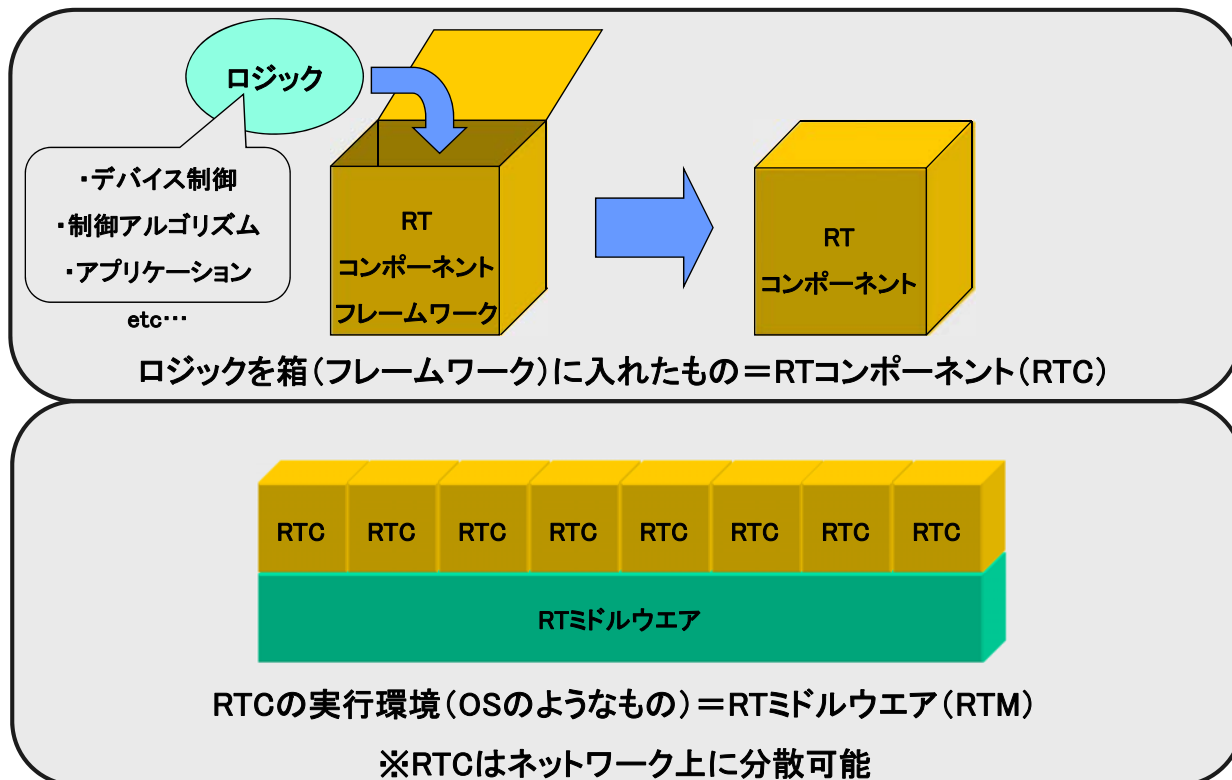


■ デプロイ機能

- オフラインエディタで作成したシステム情報から実際のシステムを構築
- デプロイ元情報として、実際に動作しているRTCもしくはManagerを設定可能
- 設定したデプロイ情報は、システム情報(RtsProfile)とは独立に保存・再利用が可能



モデリングについて



オブジェクト指向とは？

■ 元々は…ソフトウェアを開発するときの**考え方**のひとつ

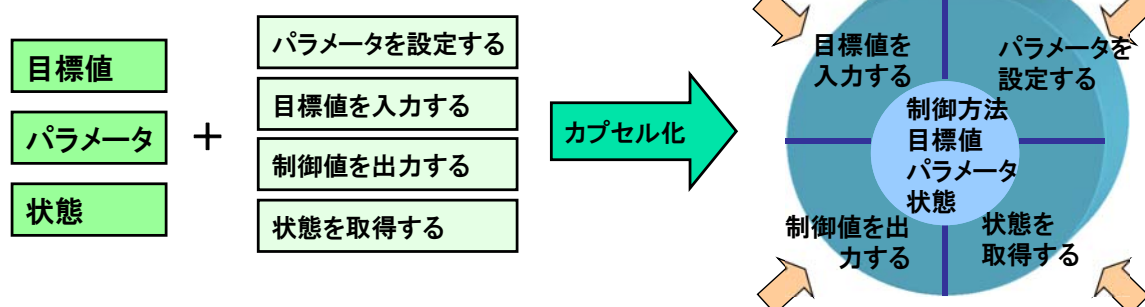
● ソフトウェア開発手法のひとつ

- 他には構造化手法、手続き指向、アスペクト指向 などがある
- 過去の考え方の良い部分を組み合わせて、更に発展させた考え方

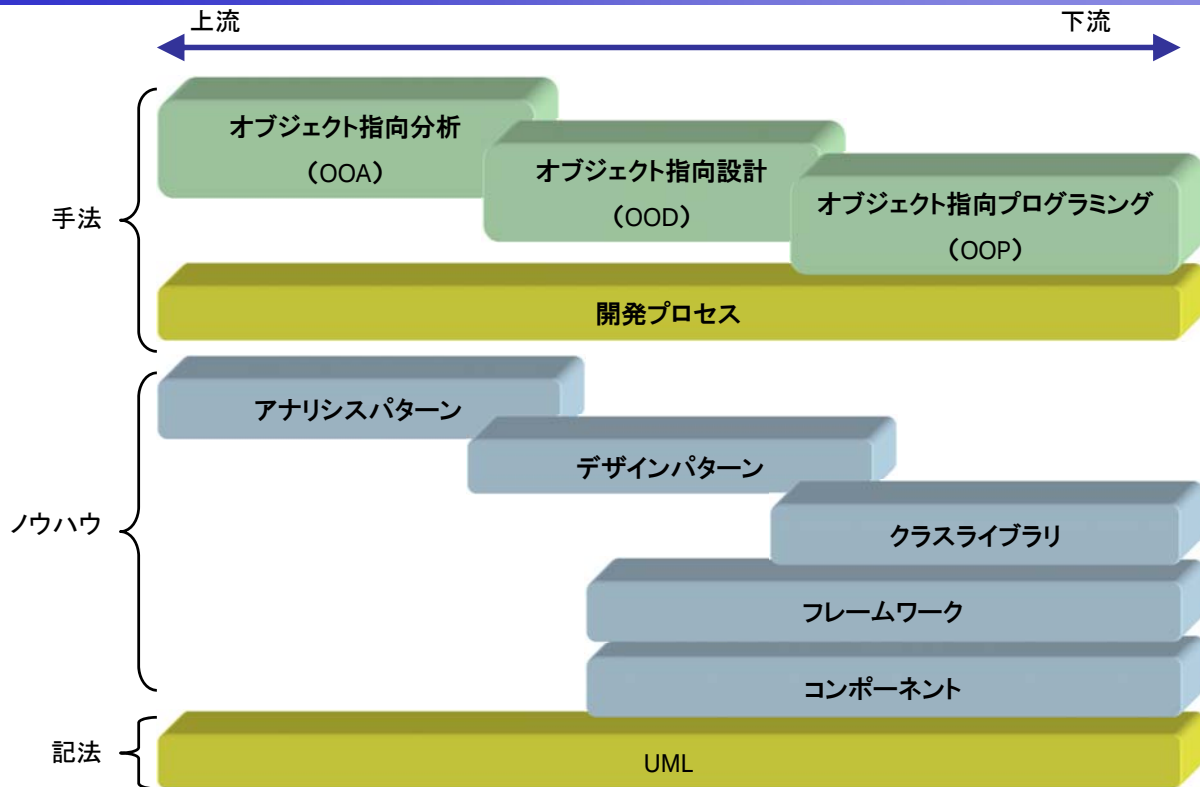
● **データとその操作方法をまとめてオブジェクト(クラス)を定義し、オブジェクトの**組み合わせ**でシステムを構成する方法**

● 現実世界を「そのまま」表現するのではなく「**モデル**」を構築する

- 対象となる業務や装置の構造や振舞いに関心事に注目して整理



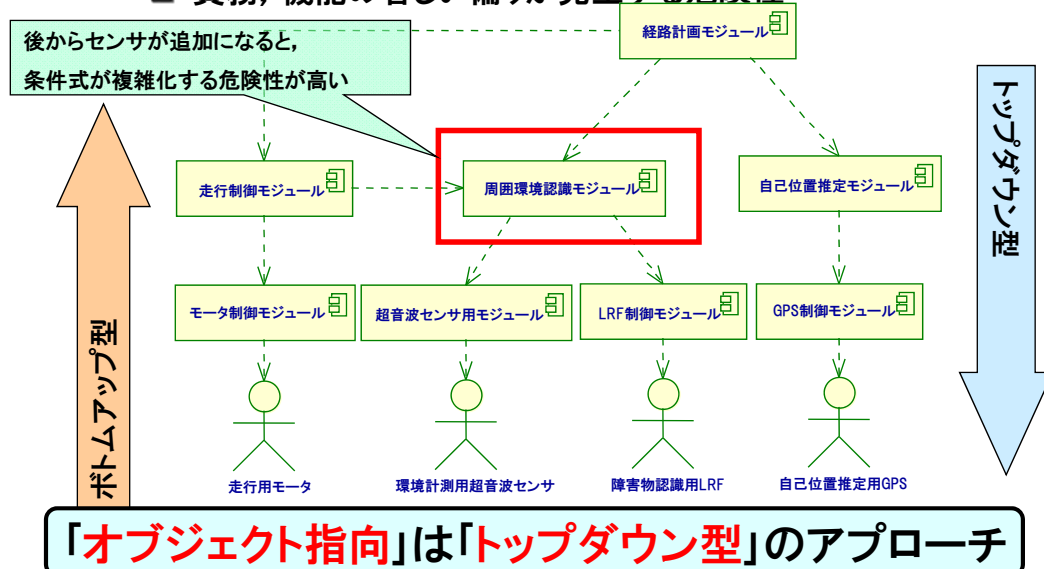
オブジェクト指向技術の全体像



ソフトウェア開発時のアプローチ

■ 「ボトムアップ型」と「トップダウン型」

- 組み込み系では、各要素技術の集合と考えることが多い？
- 「ボトムアップ型」が多い？
 - ☑ スパゲッティ・コードになる危険性
 - ☑ 責務、機能の著しい偏りが発生する危険性

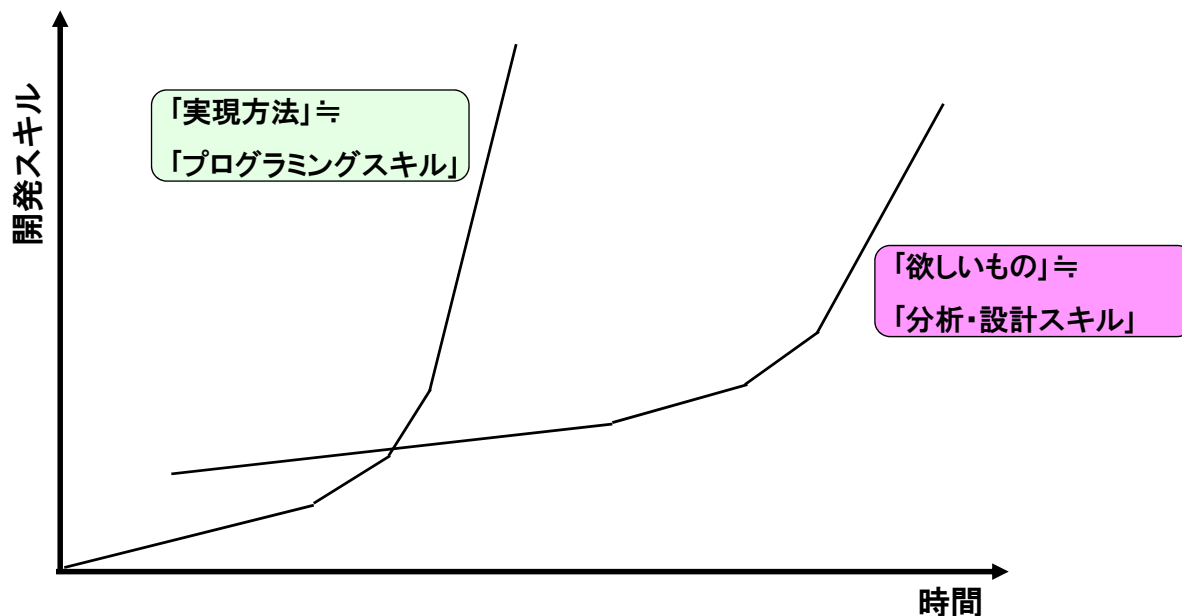


- 着目点/関心事の明確化
 - 検討すべき内容の明確化
 - 着目点の絞り込み
- 共通イメージの形成
 - 関係者間での認識のズレをなくす
- 7±2の法則
 - 人間が一度に記憶できる要素の数
 - 対象物の粒度/抽象度とは無関係
 - 「分割して統治せよ」(Devide and conquer)

システム開発の現状

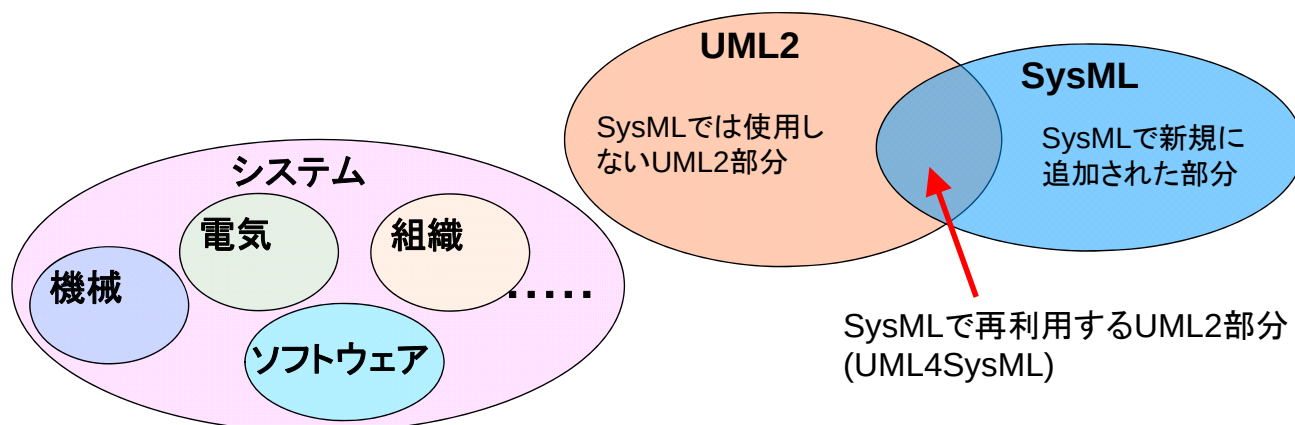
- システムそのもの大規模化, 高機能化, 複雑化
 - 使用可能なハードウェアの低価格化, 高機能化
 - ユーザニーズの高度化
 - 開発規模の大規模化
 - 開発チームの大規模化
 - 専門領域(ドメイン)の細分化, 各ドメイン毎の高度化
- 
- システムの全体を把握するのが困難
 - 大規模化に伴い, 全員が全体を理解するのが困難に
 - 各ドメイン内の状況把握で手一杯
 - 全体最適の検討がより困難に
 - 専門領域(ドメイン)の細分化, 高度化によるドメイン毎の部分最適
 - ドメインを跨がった最適化を行いたい場合, コミュニケーションが困難
 - 各ドメイン固有の用語, モデル

- 実現方法としての難しさ
- 「欲しいもの」を作る難しさ

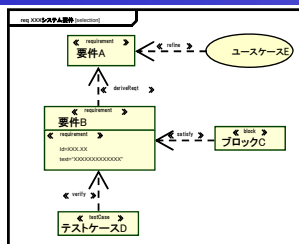


SysMLとは？

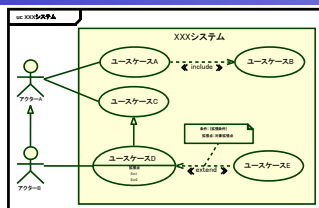
- System Modeling Language: システムモデリング言語
 - 「システム全体」の「モデル」を表現する言語(表記法+意味)である。
 - 「システム工学」における問題を扱うためのモデリング言語
 - 「UML」「オブジェクト指向」がベースとなっている。



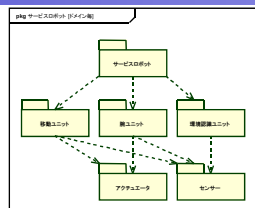
「オブジェクト指向」の考え方にに基づき、
「システム全体」を「モデリング」するための「表記法」



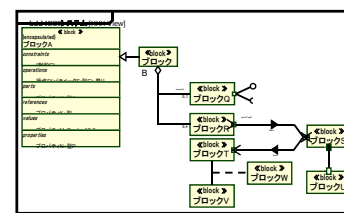
要求図



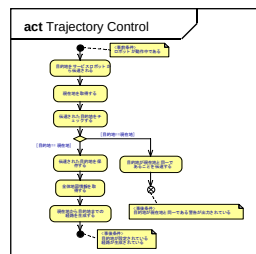
ユースケース図



パッケージ図



ブロック定義図



アクティビティ図

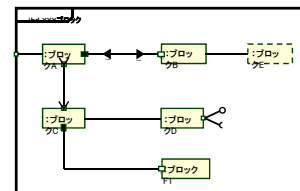
要件
要求図
ユースケース図

構造
パッケージ図
ブロック定義図
内部ブロック図

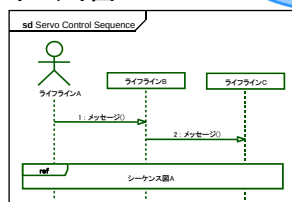
振る舞い
アクティビティ図
シーケンス図
ステートマシン図

制約
パラメトリック図

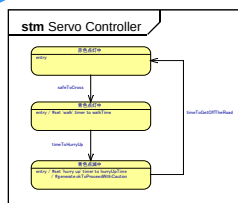
SysML



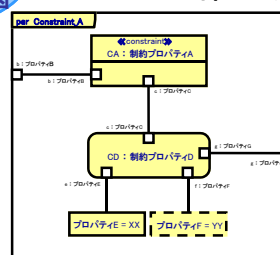
内部ブロック図



シーケンス図



ステートマシン図

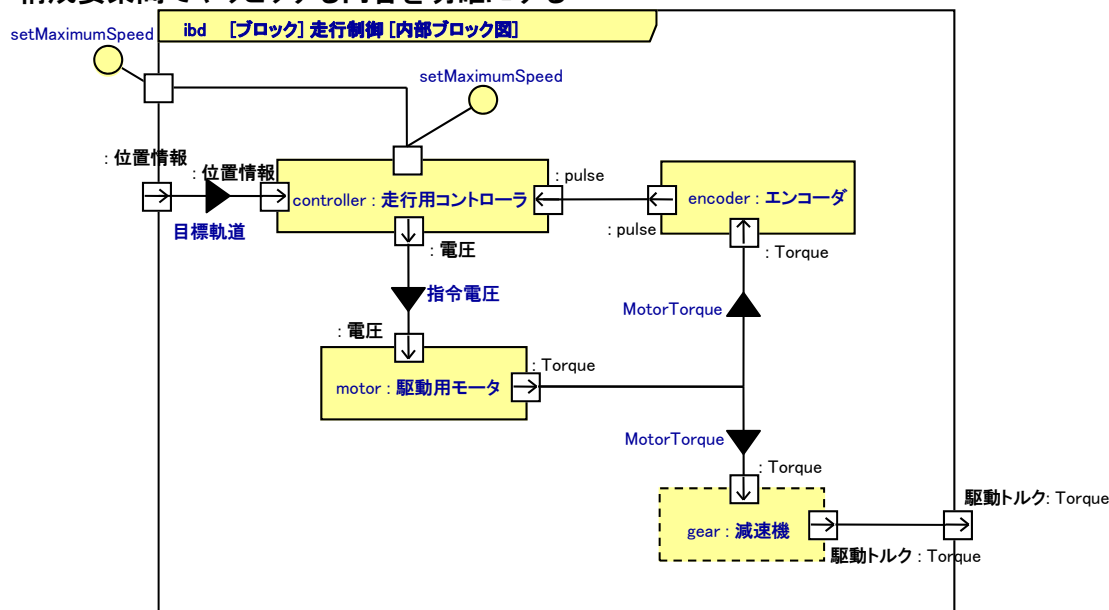


パラメトリック図

内部ブロック図

■ システムを構成する要素間でやり取りする内容を表現

- システム構成要素間の接続関係を明確にする
- 構成要素間でやり取りする内容を明確にする

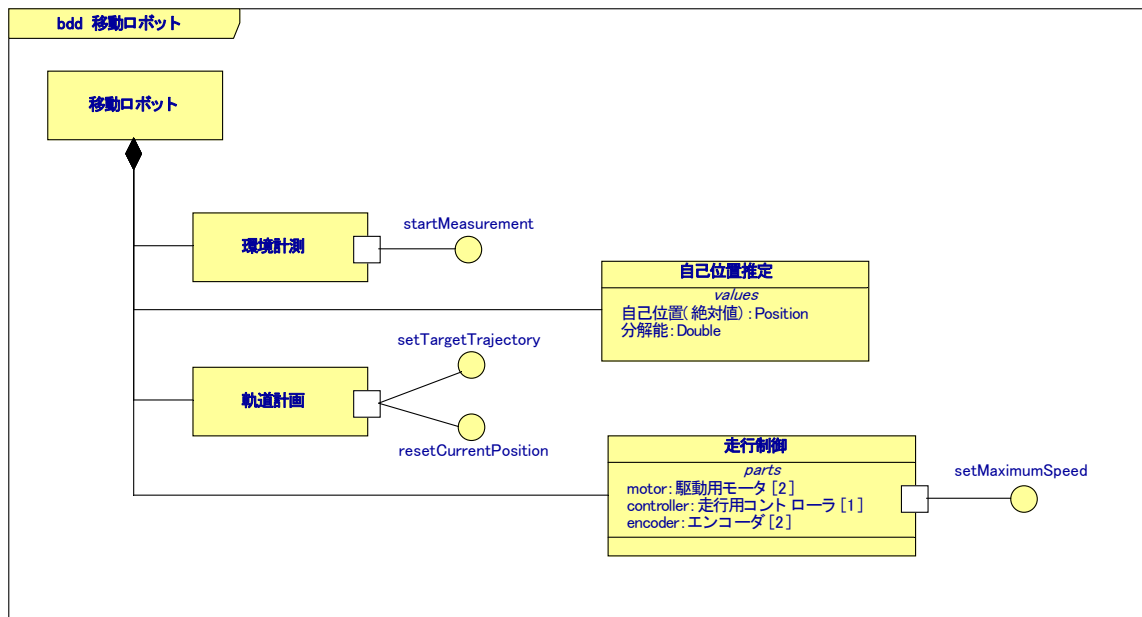


◆ 連続にやり取りするデータ, 離散的にやり取りするイベントの両者を表現可能

ブロック定義図

■ システムの静的な構造を表現

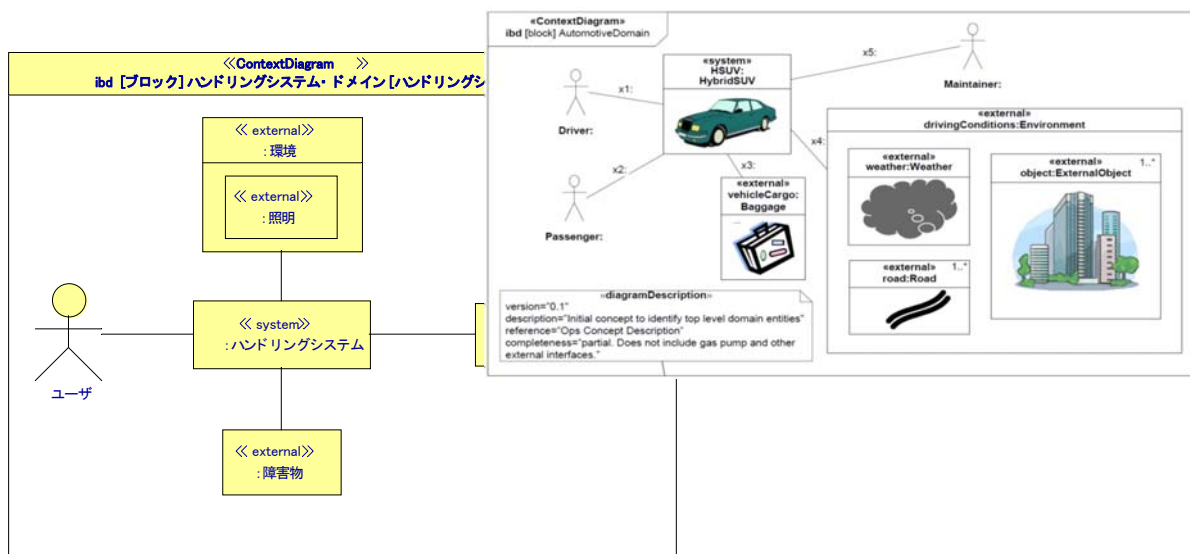
- システムを構成する要素を明確にする
- 構成要素間の関係がどのようなになっているかを表現する



◆ 静的な要素, 構造, 関係, 数的関係を表現

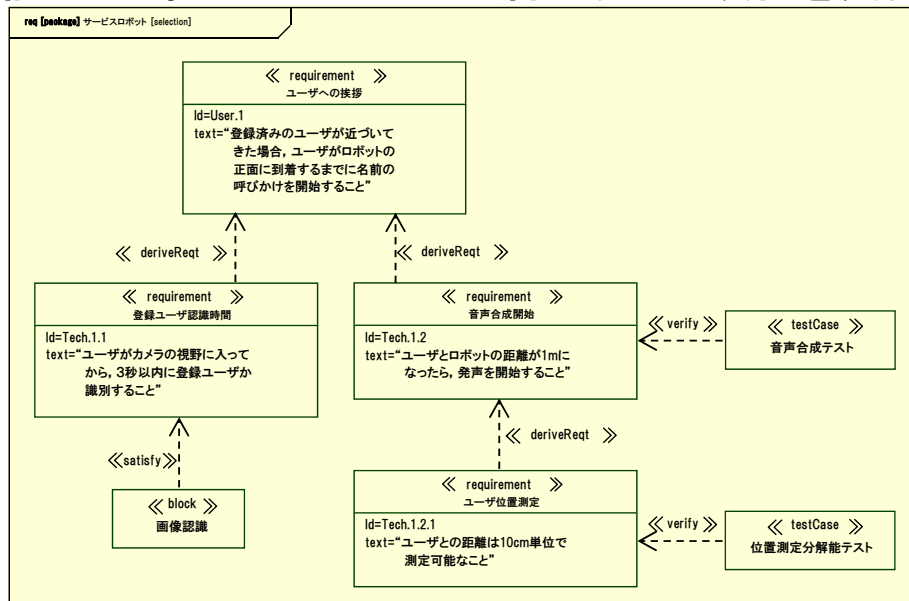
コンテキスト図

■ システムが動作する環境を表現



- ◆ 対象システムと相互作用する要素, 影響を与える要素を抽出する
- ◆ 以降の分析/設計作業にて, 検討する必要がある要素を洗い出す
 - ◆ 記述していない要素については, 検討対象外
 - ☑ SysML仕様内で明示的に定義されたダイアグラムではない

■ 性能なども含めたシステムが満たすべき項目を表現



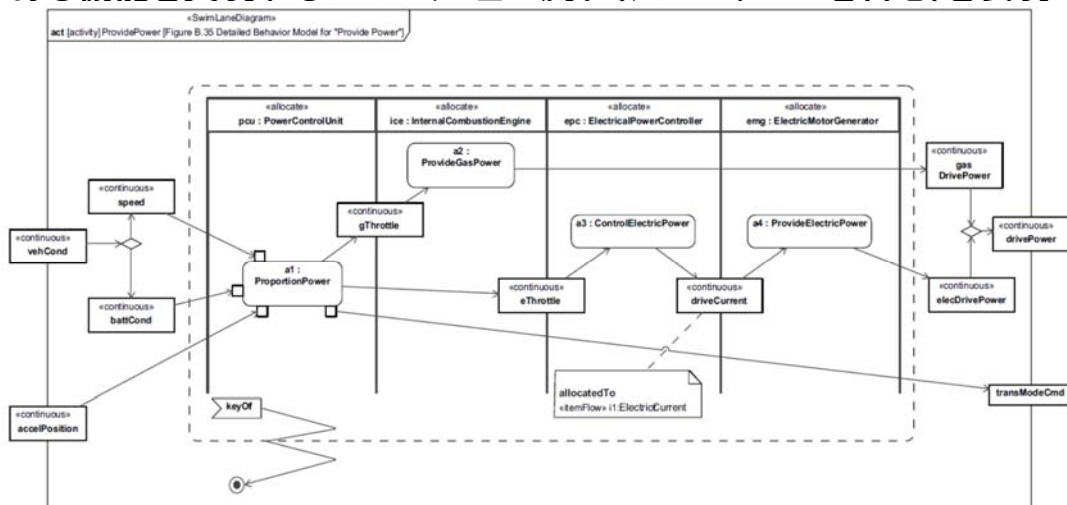
- ◆ 非機能要件も含めた要件・要求の階層化, 追跡が可能
- ◆ 要件・要求の内容については, 文章として記述
- ◆ 表形式での表現も可能

アクティビティ図

■ 処理の流れを表現

現実世界の処理流れを整理する(ビジネスプロセスも含む)

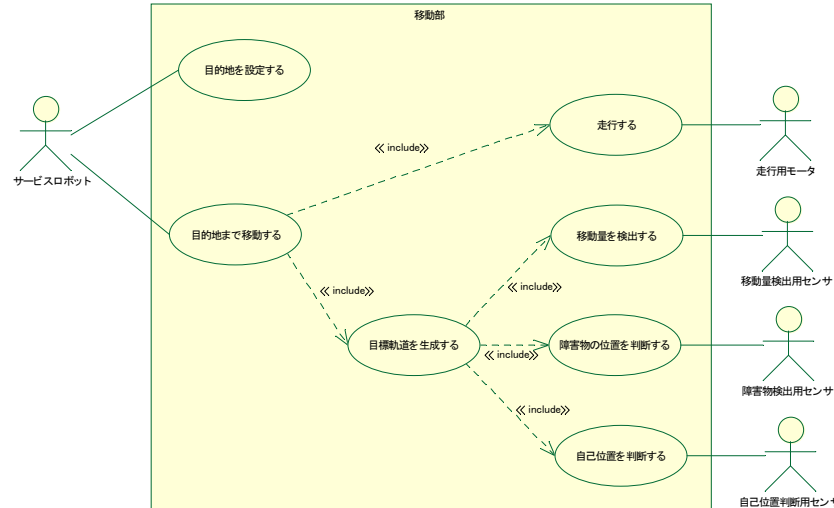
ある機能を実現するための処理の流れ(アルゴリズムを含む)を表現



- ◆ 処理の実行順序, 各処理の実行主体を表現
- ◆ 各処理間でやり取りするデータを表現
- ◆ 粒度に応じて処理を階層化して表現することも可能

■ 対象範囲(システム)が提供する機能の明確化

- 開発対象の範囲(システム化範囲)を明確にする。
- 機能とそのユーザの関係を明確にする。
- 対象範囲(システム)が必要とする外部要素を明確にする。

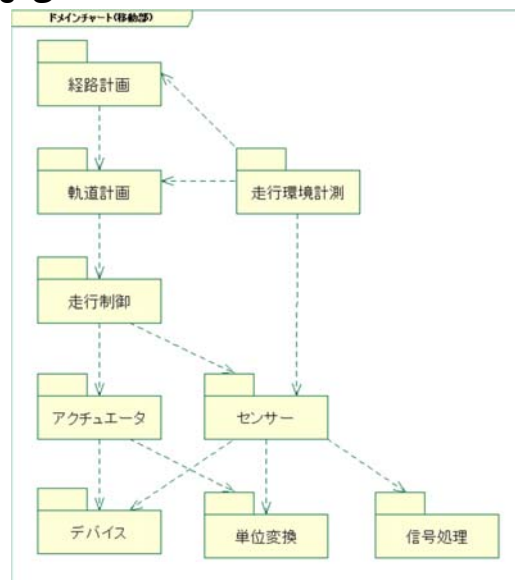


◆提供機能のカタログ化

◆開発を行う優先順位の決定にも利用される

■ 静的なおおまかな構造および構造間の関係を表現

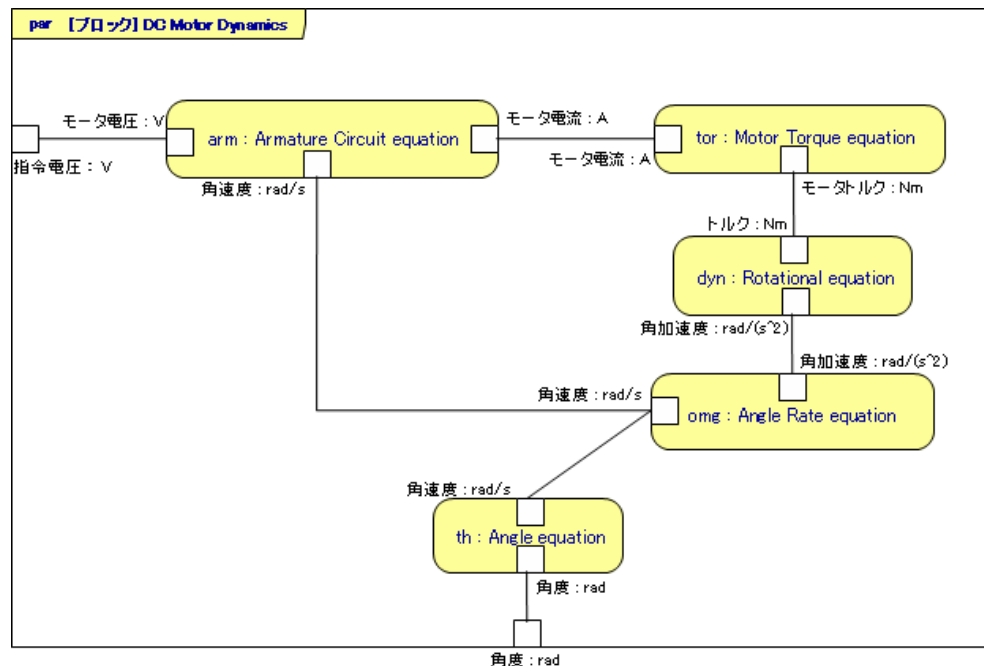
- パッケージは再利用を行う際の一番大きな単位であり、モデル要素をグループ化する単位となる



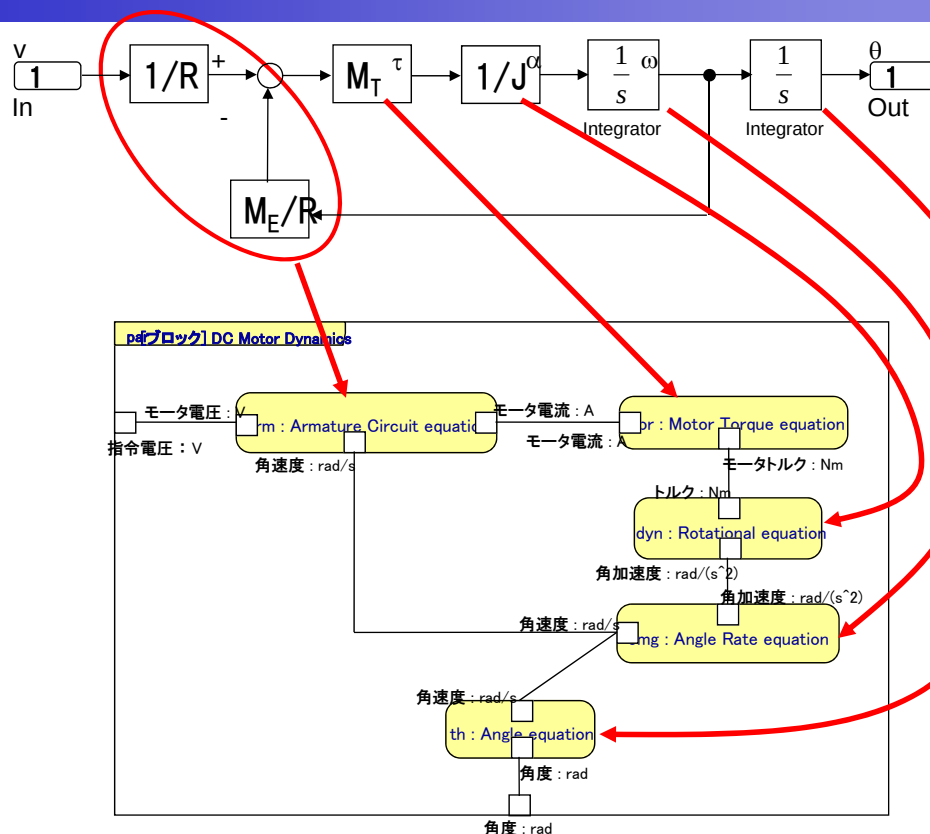
☑ 専門分野(ドメイン)の分割, 依存関係を表現する「ドメイン・チャート」としても利用可能

■ システムに付随する制約の関係を表現

- 各ブロックで定義された制約間の関係を表現
- ブロック線図と似た内容を表示することが可能

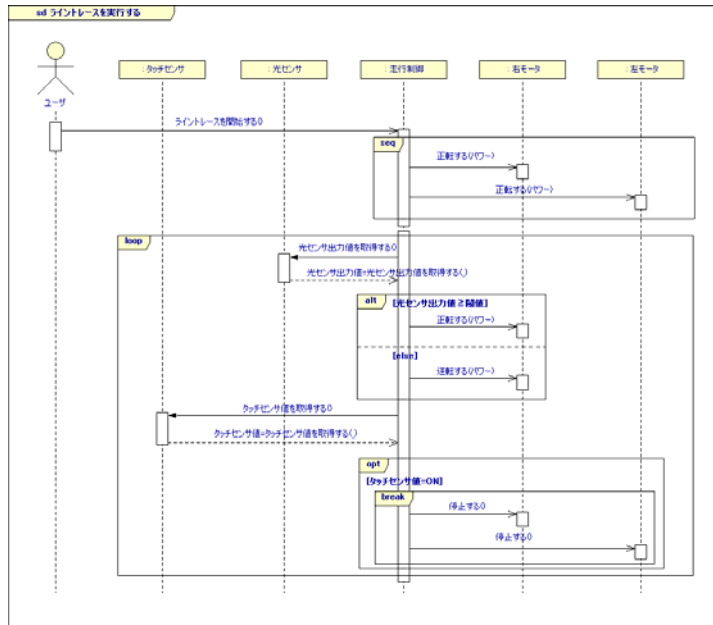


ブロック線図とパラメトリック図



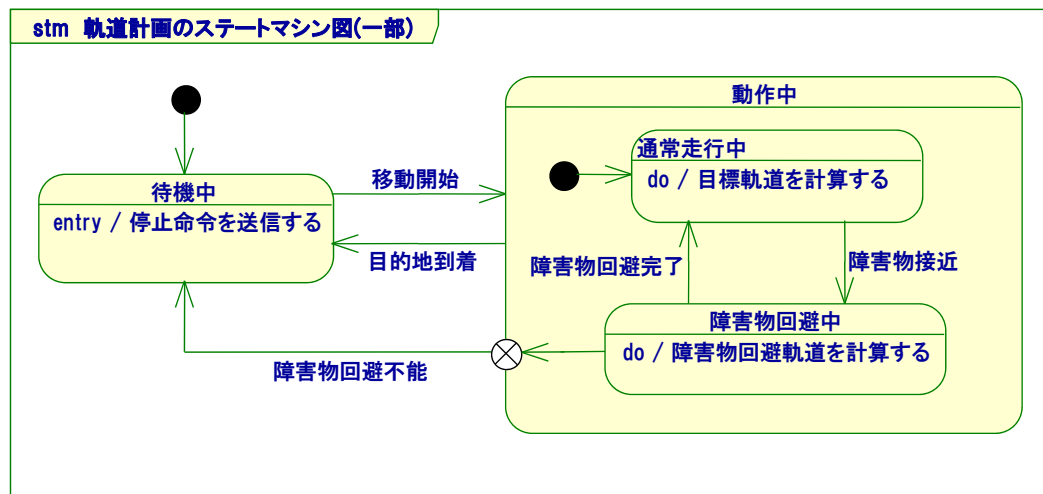
シーケンス図

- システムの相互作用を表現する
 - 要素間でやり取りする情報とその手順を表現
- ある“機能”を実現するための処理の流れを**時系列**に沿って表現する
 - 上から下に向かって時間が流れる



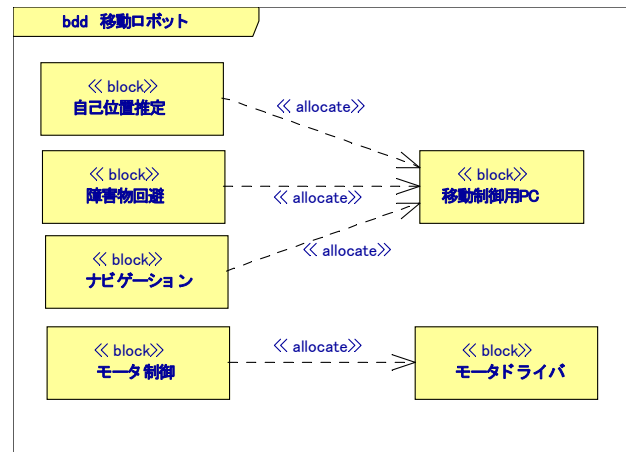
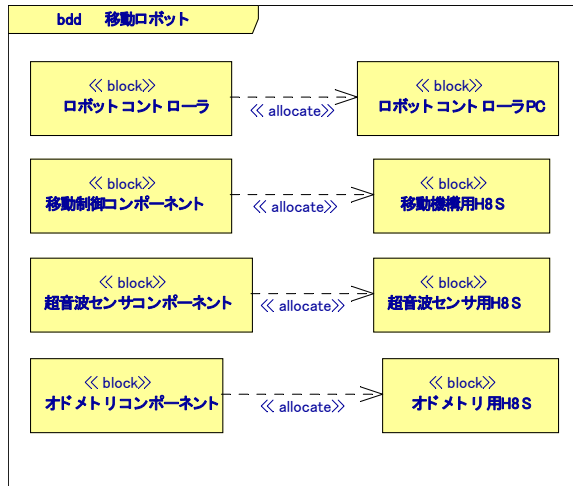
ステートマシン図

- ある要素が生成されてから破棄されるまでの状態の移り変わりを表現
 - 状態が変化する流れと、そのきっかけを表現可能
 - 各状態で実行する振る舞い(アクティビティ)も表現可能



■ 要素間の対応関係を表現

- ソフトウェアコンポーネントのノード配置, アクティビティのブロックへの振り分けなどを表現
- 「機能」と「構造」の分離を実現



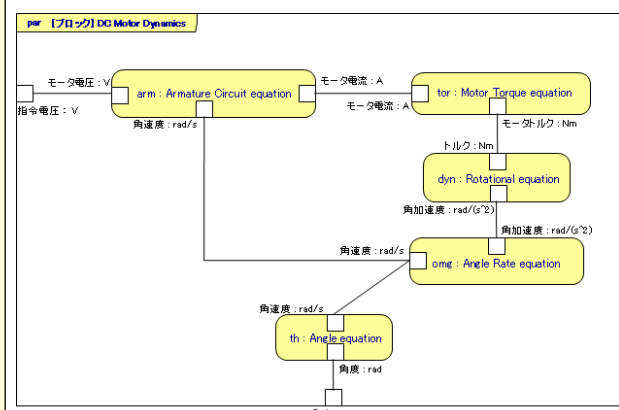
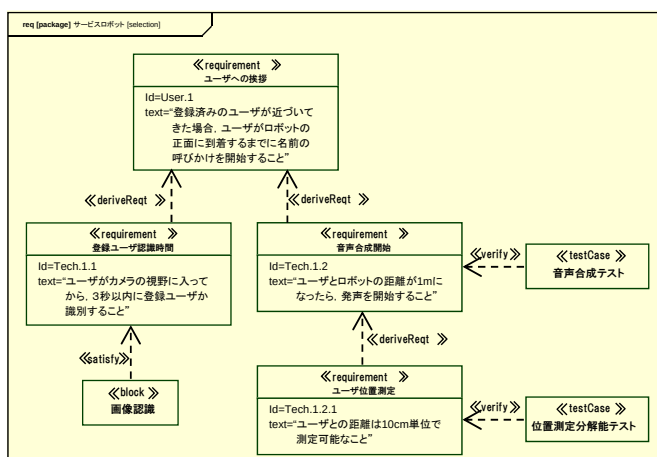
SysMLの特徴

■ ハードウェアも含めたシステム設計全体をサポート

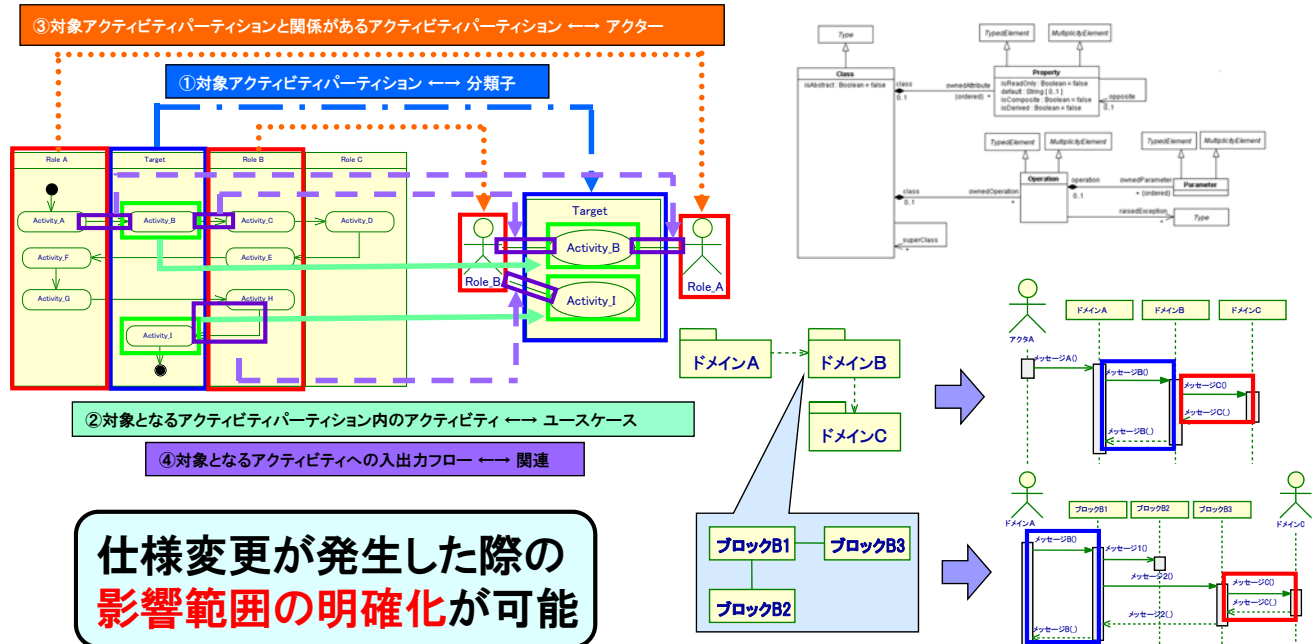
- 既存モデルの置き換えを狙っているのではなく、あくまでも他ドメインの専門家とのコミュニケーションの円滑化が狙い
- 必要に応じて、処理、データ、人、組織などの要素も表現可能

■ システム開発の広範な工程をサポート

- ユーザ要求を明確化する部分から、実際の制御系まで広範囲をサポート



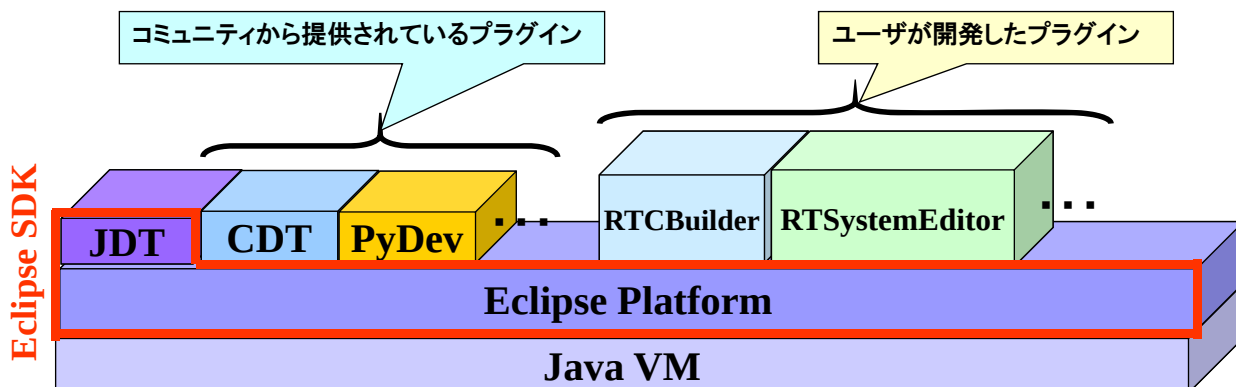
- ダイアグラム間の関係が明確
 - メタモデル定義により, 仕様自体を定義
 - 粒度が異なる要素間の関係も明確化可能



補足説明

■ オープンソース・コミュニティで開発されている統合開発環境

- マルチプラットフォーム対応. WindowsやLinuxなど複数OS上で利用可能
- 「Plug-in」形式を採用しており, 新たなツールの追加, 機能のカスタマイズが可能
- RCP(Rich Client Platform)を利用することで, 簡単に単独アプリ化が可能



RTCBuilder, RTSystemEditorのインストール

■ ダウンロードし, 解凍するだけ

※Javaの実行環境については, 別途インストールが必要

OpenRTM-aist

The power to connect

ホーム ダウンロード ドキュメント コミュニティ 研究・開発 プロジェクト ハードウェア

Google Translate

言語を選択

Powered by Google 翻訳

ナビゲーション

- ホーム
- ダウンロード
 - C++版
 - Python版
 - Java版
- ツール
 - Eclipse tools 1.1.0-RC2
 - Eclipse tools 1.1.0-RC1
 - Eclipse tools 1.0-RELEASE
 - Eclipse tools 1.0.0-RC1
 - Eclipse tools 0.4.2
 - rtshel(CUIツール)
 - Pythonライブラリ (rtctree/rtprofile)
 - RtLink: RtcTemplate
- コンポーネント
 - RTC/RTS仕様記述方式
- ドキュメント
- コミュニティ
- 研究・開発
- プロジェクト
- ハードウェア

ホーム > ダウンロード > ツール > Eclipse tools 1.1.0-RC2

OpenRTM Eclipse tools 1.1.0-RC2

投稿者: y-kuribara 投稿日時: 11月, 2011-10-11 18:36

これは、OpenRTM-aistのツールとして開発された RTCBuilder (旧 RtcTemplate) および RTSystemEditor (旧 RtLink) は、OpenRSPやその他のツールと統合開発環境を構成する OpenRT Platform に組み込まれることになりました。これらでは、RTSystemEditor 及び RTCBuilder のみを提供していますが、将来的には様々なツールを一通り提供する予定です。

現在の RTSystemEditor 及び RTCBuilder の最新バージョンは 1.1.0 です。

Table of contents

- 全部入りパッケージ
- バイナリ
- Eclipse/JDK/JRE等
- 過去のバージョン

全部入りパッケージ

Eclipse-3.4.2 [Gangmode SR2]		
Eclipse] 4.2+RTSE+RTCB	eclipse342_rtmtools110-rc2_win32_ja.zip	2011.07.22
Windows/全部入り	MD5:2e6f9fa3e370b6e7ac1f934036c7abf	

- Ubuntu8.04, Ubuntu9.10, Ubuntu10.04でLinux(Eclipse) 4.2が動作しない不具合が報告されています。
- Ubuntu8.04では、apt-get install xulrunner-1.9として xulrunnerをアップデートしてください。
- Ubuntu9.10, Ubuntu10.04では、以下の方法を利用するか、Eclipse3.3f(11.2.5)をご利用ください。

```
$ su
# vi /etc/apt/sources.list
1行追加 - deb http://jp.archive.ubuntu.com/ubuntu/ jaunty main restricted
# apt-get update
# apt-get install xulrunner-1.9
# dpkg -i | grep xulrunner-1.9
$ xulrunner-1.9
```

RTSystemEditorについて

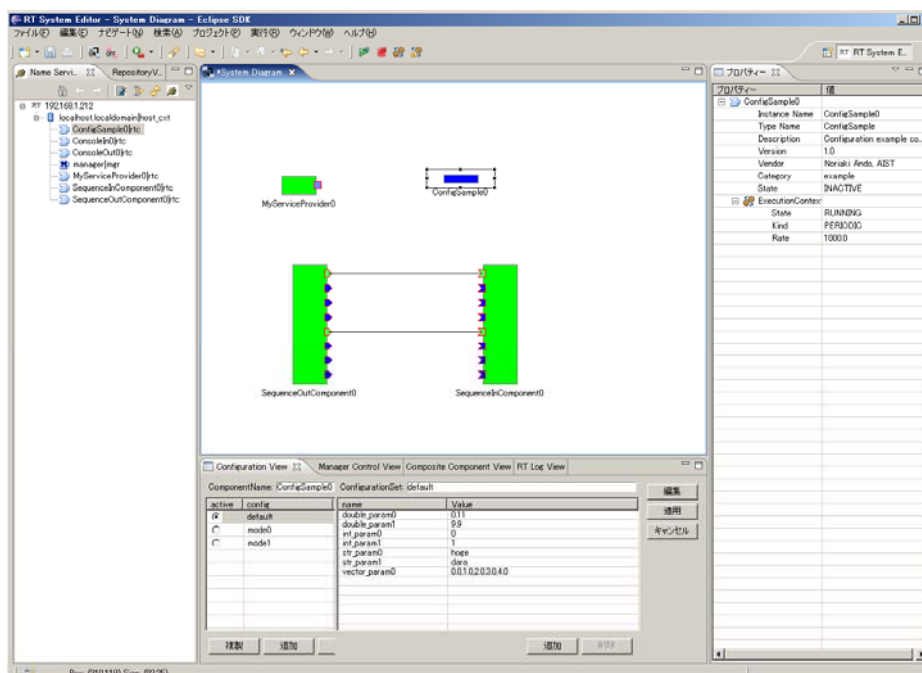


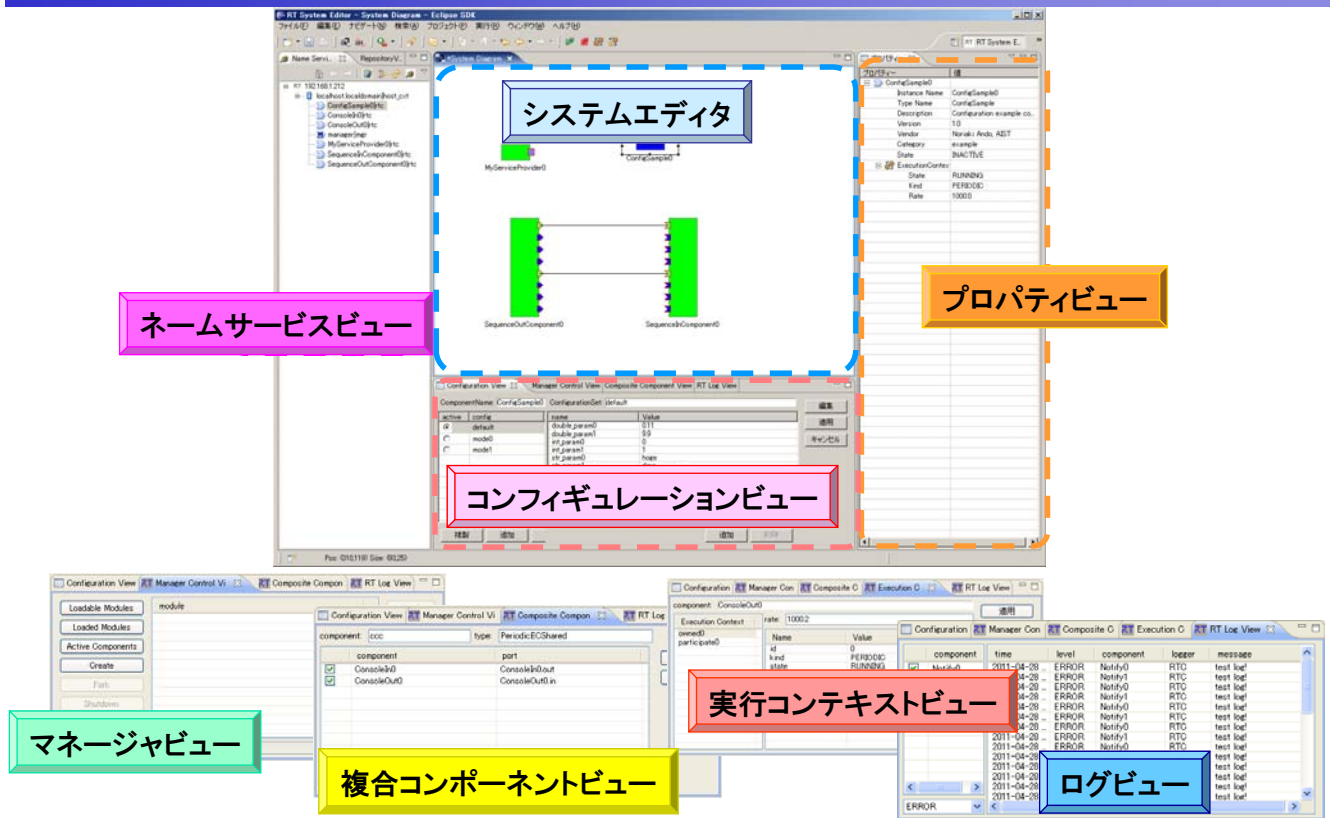
RTSystemEditor概要



■ RTSystemEditorとは？

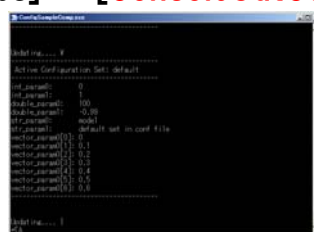
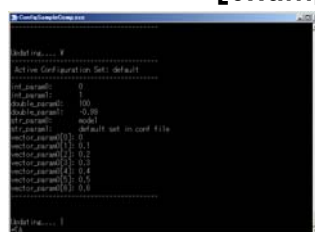
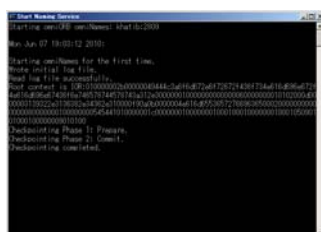
- RTコンポーネントを組み合わせて、RTシステムを構築するためのツール





準備

- Naming Serviceの起動
 - [スタート]メニューから
[プログラム]→[OpenRTM-aist 1.1]→[C++]→[tools]→[Start Naming Service]
- CosoleInCompの起動
 - [スタート]メニューから起動
[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]
→[examples]→ [ConsoleInComp.exe]
- CosoleOutCompの起動
 - [スタート]メニューから起動
[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]
→[examples]→ [ConsoleOutComp.exe]



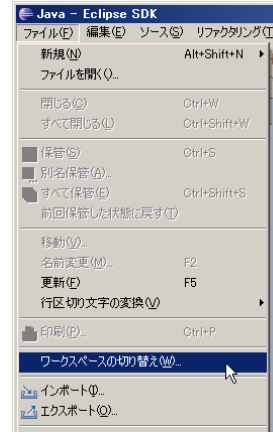
ツールの起動

- Windowsの場合
 - Eclipse.exeをダブルクリック
- Unix系の場合
 - ターミナルを利用してコマンドラインから起動
 - Ex) \$ /usr/local/Eclipse/eclipse

■ ワークスペースの選択(初回起動時)



■ ワークスペースの切替(通常時)



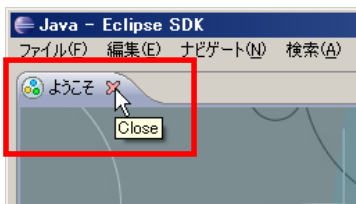
※ワークスペース

Eclipseで開発を行う際の作業領域

Eclipse上でプロジェクトやファイルを作成するとワークスペースとして指定したディレクトリ以下に実際のディレクトリ, ファイルを作成する

準備

- 初期画面のクローズ
 - 初回起動時のみ



※パースペクティブ

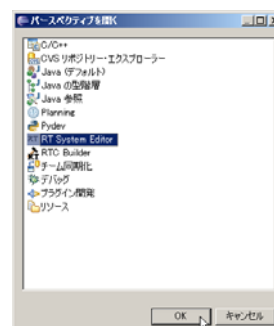
Eclipse上でツールの構成を管理する単位
メニュー, ツールバー, エディタ, ビューなど
使用目的に応じて組み合わせる
独自の構成を登録することも可能

■ パースペクティブの切り替え

- ①画面右上の「パースペクティブを開く」を選択し、一覧から「その他」を選択

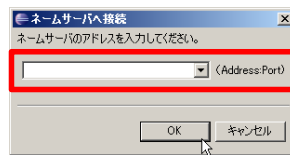
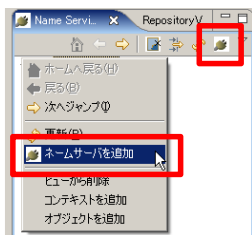


- ②一覧画面から対象ツールを選択

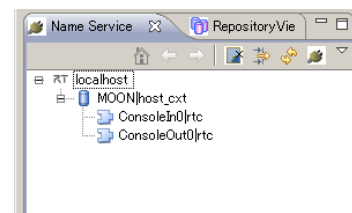


RTシステム構築の基本操作

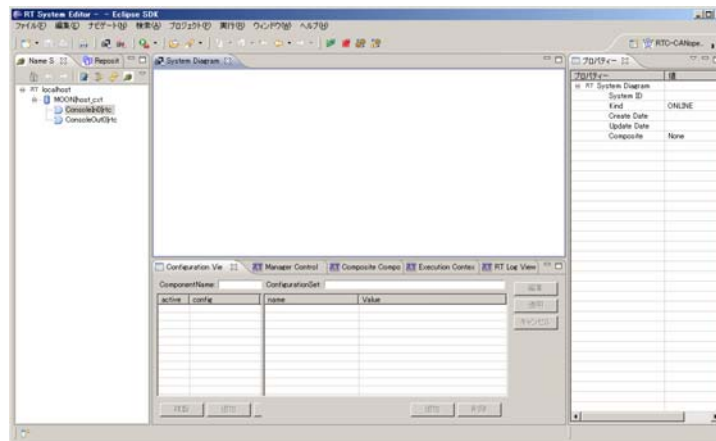
■ ネームサービスへ接続



※対象ネームサーバのアドレス、ポートを指定
→ポート省略時のポート番号は
設定画面にて設定可能

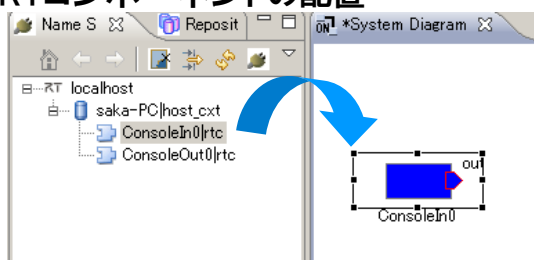


■ システムエディタの起動

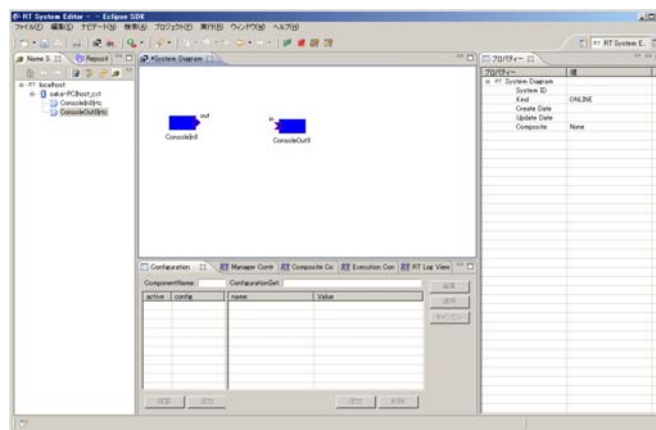


RTシステム構築の基本操作

■ RTコンポーネントの配置

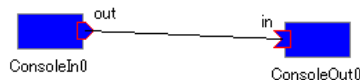


※ネームサービスビューから対象コンポーネントをドラッグアンドドロップ

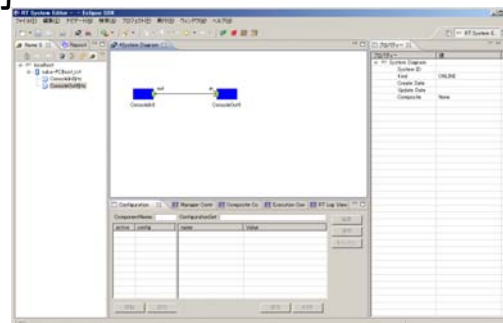


■ ポートの接続

①接続元のポートから接続先の②接続プロファイルを入力
ポートまでドラッグ



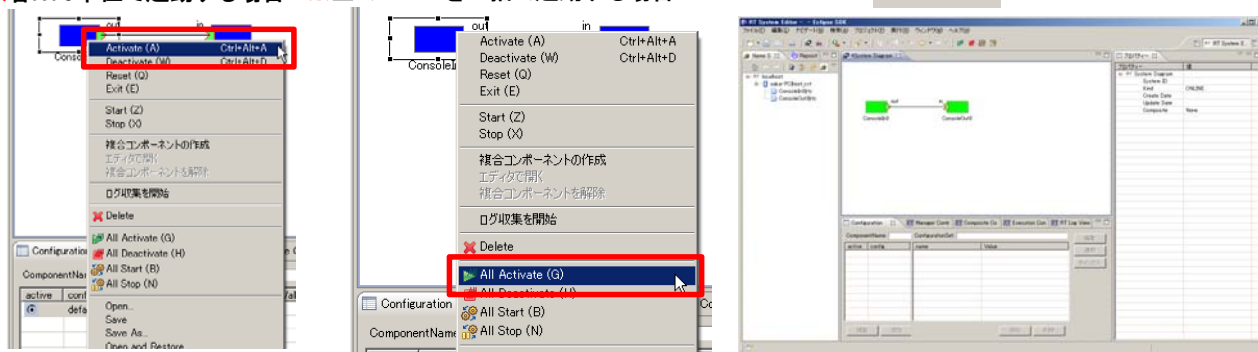
※ポートのプロパティが異なる場合など、
接続不可能なポートの場合にはアイコンが変化



RTシステム構築の基本操作

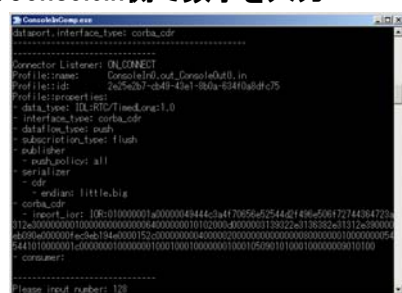
■ コンポーネントの起動

※各RTC単位で起動する場合 ※全てのRTCを一括で起動する場合

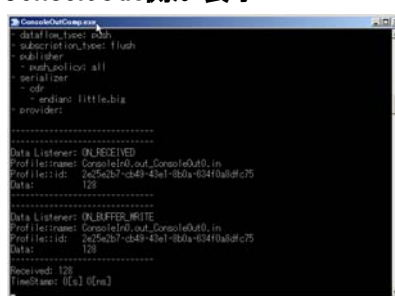


■ サンプルの実行

① ConsoleIn側で数字を入力



② ConsoleOut側が表示



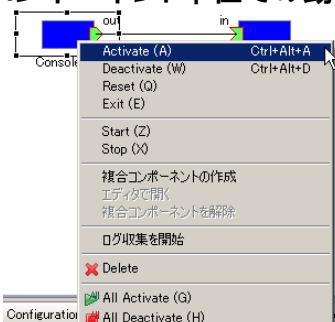
※停止はDeactivateを実行

※RTC間の接続を切る場合には
接続線をDelete
もしくは、右クリックメニューから「Delete」を選択

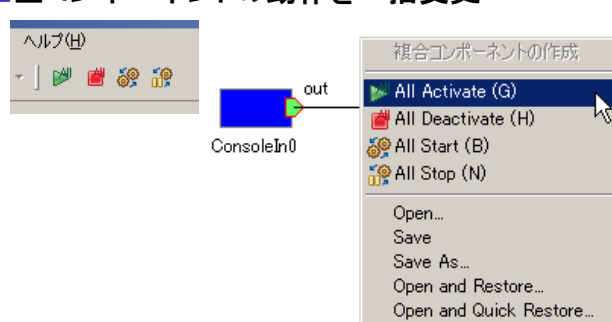
RTコンポーネントの動作

アクション名	説明
Activate	対象RTCを活性化する
Deactivate	対象RTCを非活性化する
Reset	対象RTCをエラー状態からリセットする
Exit	対象RTCの実行主体(ExecutionContext)を停止し、終了する
Start	実行主体(ExecutionContext)の動作を開始する
Stop	実行主体(ExecutionContext)の動作を停止する

■ 各コンポーネント単位での動作変更



■ 全コンポーネントの動作を一括変更



※ポップアップメニュー中でのキーバインドを追加

※単独RTCのActivate/Deactivateについては、グローバルはショートカットキー定義を追加

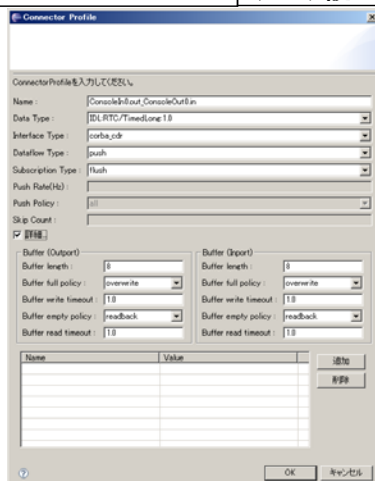
接続プロファイル(DataPort)について

項目	設定内容
Name	接続の名称
DataType	ポート間で送受信するデータの型. ex)TimedOctet, TimedShortなど
InterfaceType	データを送受信するポートの型. ex)corba_cdrなど
DataFlowType	データの送信方法. ex)push, pullなど
SubscriptionType	データ送信タイミング. 送信方法がPush の場合有効. New, Periodic, Flushから選択
Push Rate	データ送信周期(単位:Hz). SubscriptionTypeがPeriodic の場合のみ有効
Push Policy	データ送信ポリシー. SubscriptionTypeがNew, Periodic の場合のみ有効. all, fifo, skip, newから選択
Skip Count	送信データスキップ数. Push PolicyがSkip の場合のみ有効

- SubscriptionType
 - New : バッファ内に新規データが格納されたタイミングで送信
 - Periodic : 一定周期で定期的にデータを送信
 - Flush : バッファを介さず即座に同期的に送信
- Push Policy
 - all : バッファ内のデータを一括送信
 - fifo : バッファ内のデータをFIFOで1個ずつ送信
 - skip : バッファ内のデータを間引いて送信
 - new : バッファ内のデータの最新値を送信(古い値は捨てられる)

接続プロファイル(DataPort)について

項目	設定内容
Buffer length	バッファの大きさ
Buffer full policy	データ書き込み時に, バッファフルだった場合の処理. overwrite, do_nothing, blockから選択
Buffer write timeout	データ書き込み時に, タイムアウトイベントを発生させるまでの時間(単位:秒)
Buffer empty policy	データ読み出し時に, バッファが空だった場合の処理. readback, do_nothing, blockから選択
Buffer read timeout	データ読み出し時に, タイムアウトイベントを発生させるまでの時間(単位:秒)

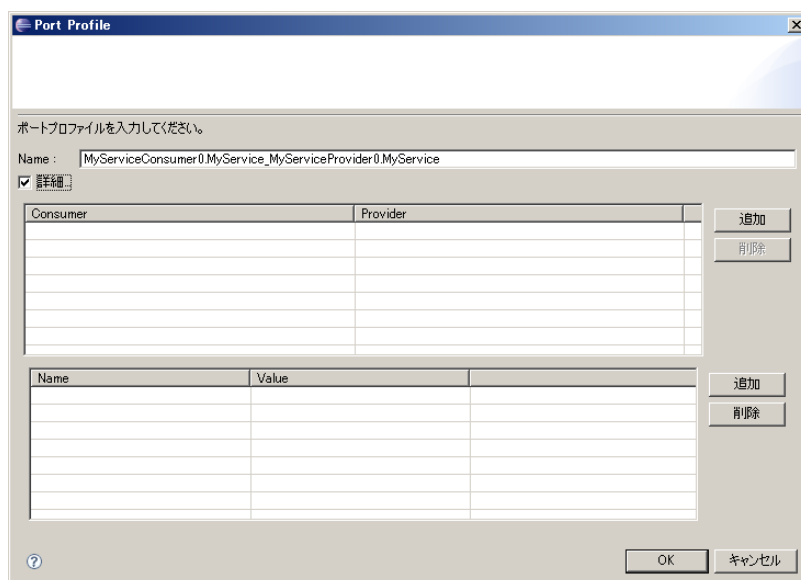


- ※OutPort側のバッファ, InPort側のバッファそれぞれに設定可能
- ※timeoutとして「0.0」を設定した場合は, タイムアウトしない

- Buffer Policy
 - overwrite : 上書き
 - readback : 最後の要素を再読み出し
 - block : ブロック
 - do_nothing : なにもしない

- ※Buffer Policy = Block+timeout時間の指定で, 一定時間後読み出し/書き込み不可能な場合にタイムアウトを発生させる処理となる

項目	設定内容
Name	接続の名称
インターフェース情報	接続するインターフェースを設定。 接続対象のServicePortに複数のServiceInterfaceが定義されていた場合、どのインターフェースを実際に接続するかを指定



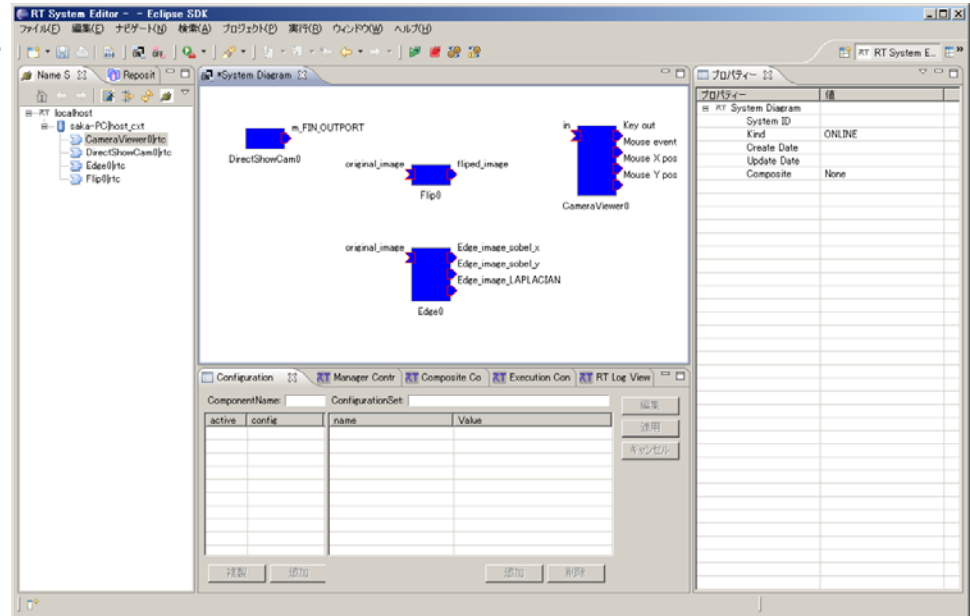
準備

- CameraViewerCompの起動
 - [スタート]メニューから起動
 - [プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]
 - [opencv-rtcs]→ **[CameraViewerComp.exe]**
- DirectShowCamCompの起動
 - [スタート]メニューから起動
 - [プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]
 - [opencv-rtcs]→ **[DirectShowCamComp.exe]**
- 画像処理用コンポーネントの起動
 - [スタート]メニューから起動
 - [プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]
 - [opencv-rtcs]→ **[FlipComp.exe]**
 - [プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]
 - [opencv-rtcs]→ **[EdgeComp.exe]**

システムの構築

■ 以下のコンポーネントをエディタ上に配置

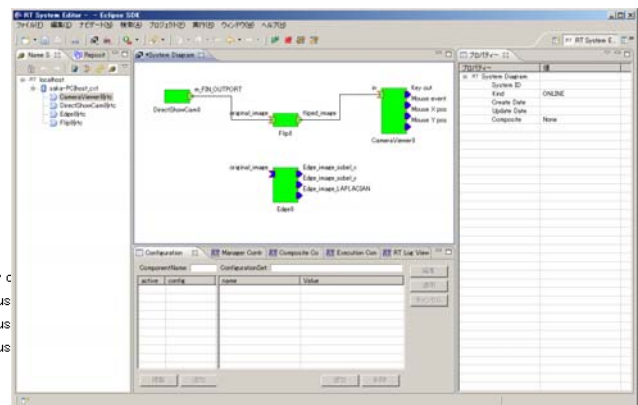
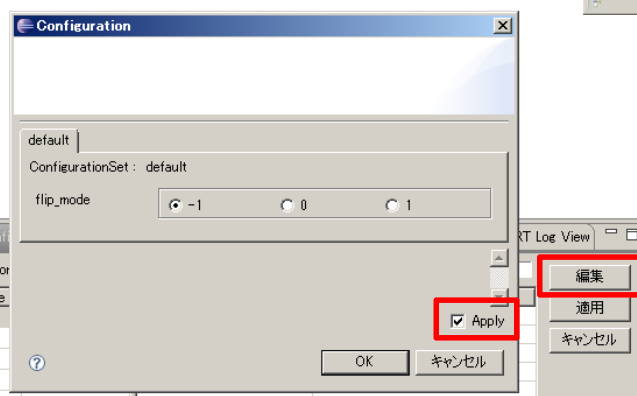
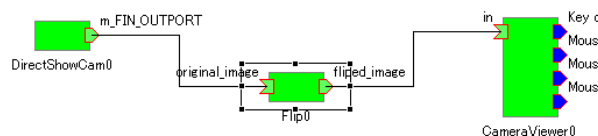
- DirectShowCam
- Flip
- Edge
- CameraViewer



システム構成の変更

■ Flip側との接続

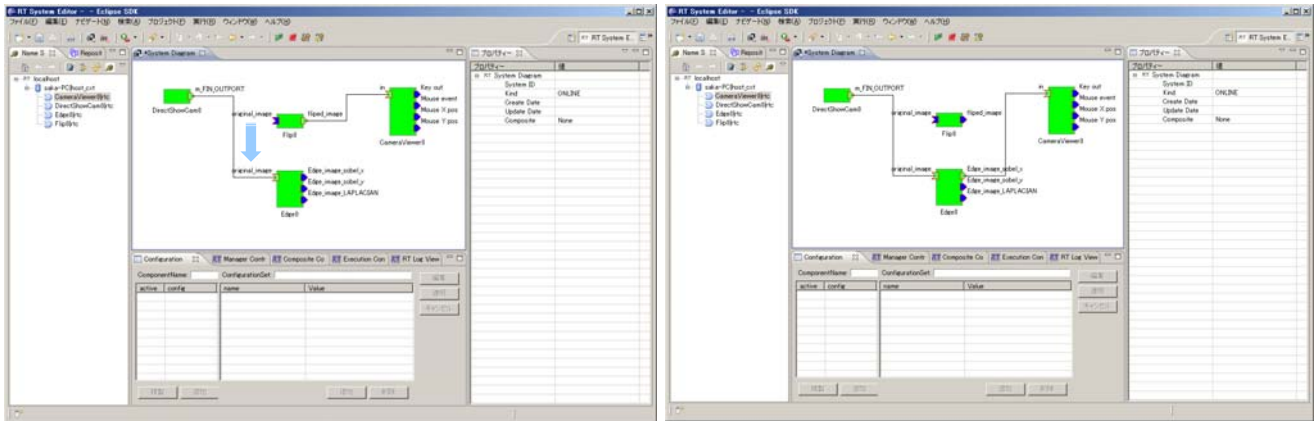
- DirectShowCam → Flip
→ CameraViewerと接続
(接続プロファイルはデフォルト設定)
- AllActivateを実行



- ConfigurationViewの「編集」
- 表示されたダイアログ内で「flip_mode」の値を変更
- 「Apply」のチェックボックス

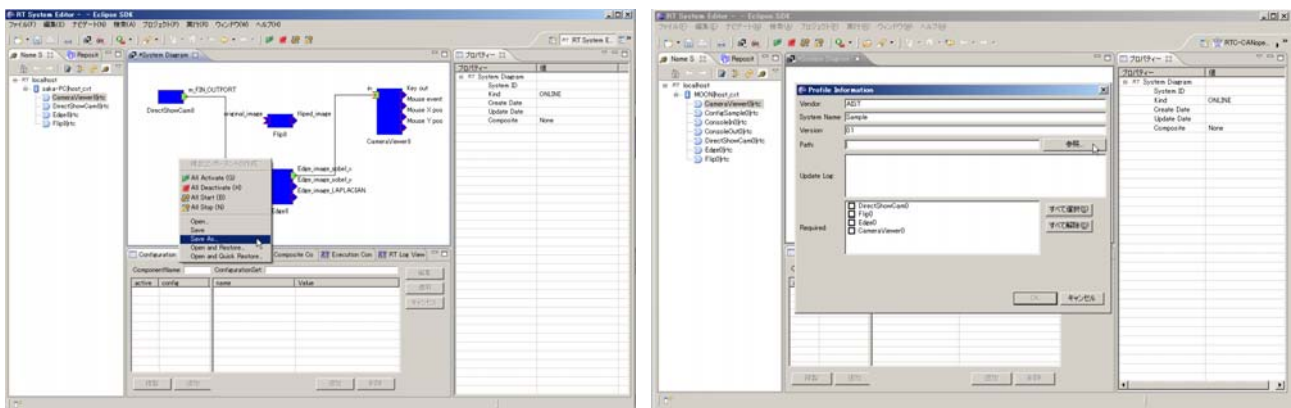
システム構成の変更

- Edge側への差し替え
 - Flipに繋がっている接続線を選択
 - Flip側のPort部分に表示されているハンドルをEdge側のPortに繋ぎ替え
 - 接続プロファイルはデフォルト設定のまま



システム構成の保存・復元

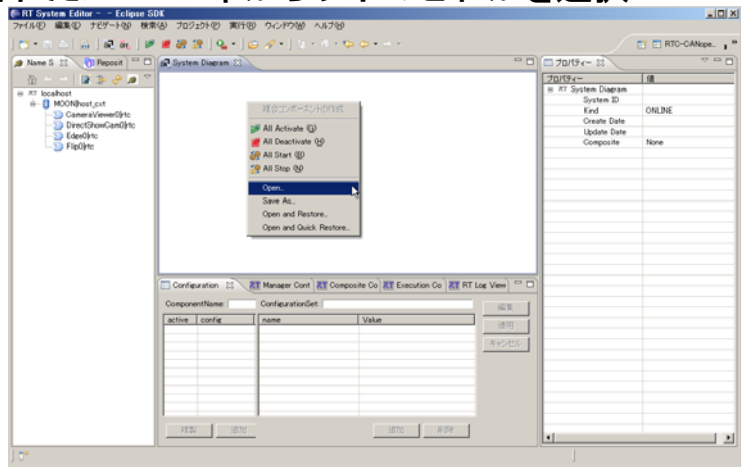
- エディタ上で右クリック
- 表示されたメニュー内から「Save As...」を選択
- 【Profile Information】画面にて、ベンダ名、システム名、バージョン番号、保存先ファイル名を指定



※指定したファイルにXML形式(RtsProfile)で保存

システム構成の保存・復元

- システムエディタを閉じる
- 各コンポーネントを一度終了し、再起動
- エディタ上で右クリックし、表示されたメニュー中から以下のどれかを選択
 - 「Open」
 - 「Open and Restore」
 - 「Open and Quick Restore」
- 【ファイル選択画面】にて、先ほど保存したファイルを指定



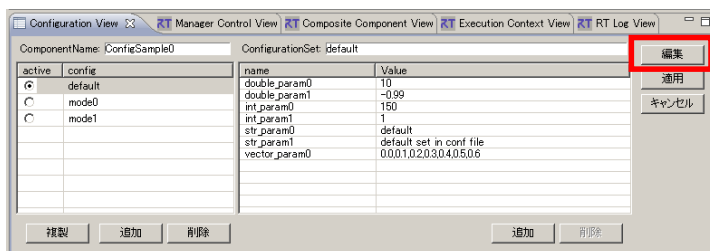
※Open: 使用していたRTCのみを読み込み

Open and Restore: 使用していたRTCを読み込むと同時に、接続、コンフィギュレーションセットの内容も復帰

Open and Quick Restore: 読み込み内容はOpen and Restoreと同様。該当RTCを検索する際にIORのみを使用

コンフィギュレーションビュー

- RTコンポーネントのコンフィギュレーション情報の確認/編集



※「編集」ボタンにより、各種コントロールを用いた一括編集が可能

※「Apply」チェックボックスがONの場合、設定値を変更すると即座にコンポーネントに反映
→ テキストボックスからフォーカス外れる、ラジオボタンを選択する、スライダーを操作する、スピナを変更する、などのタイミング

※コンフィギュレーション情報を複数保持している場合、上部のタブで編集対象を切り替え



コンフィギュレーション情報の設定方法

- rtc.conf内

[カテゴリ名]. [コンポーネント名]. config_file: [コンフィギュレーションファイル名]

※例) example.ConfigSample.config_file: configsample.conf

- コンフィギュレーションファイル内

- コンフィギュレーション情報

conf. [コンフィグセット名]. [コンフィグパラメータ名]: [デフォルト値]

※例) conf.mode0.int_param0: 123

- Widget情報

conf. __widget__. [コンフィグパラメータ名]: [Widget名]

※例) conf.__widget__.str_param0: radio

- 制約情報

conf. __constraints__. [コンフィグパラメータ名]: [制約情報]

※例) conf.__constraints__.str_param0: (bar,foo,foo,dara)

conf. __[コンフィグセット名]. [コンフィグパラメータ名]: [制約情報]

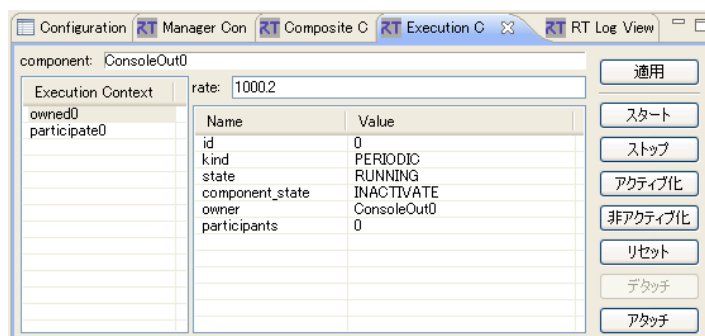
※例) conf._mode1.str_param0: (bar2,foo2,dara2)

RTCの利用者が設定するのではなく, RTC開発者, RTC管理者が設定することを想定.

RTBuilderを使用することで設定可能

実行コンテキストビュー

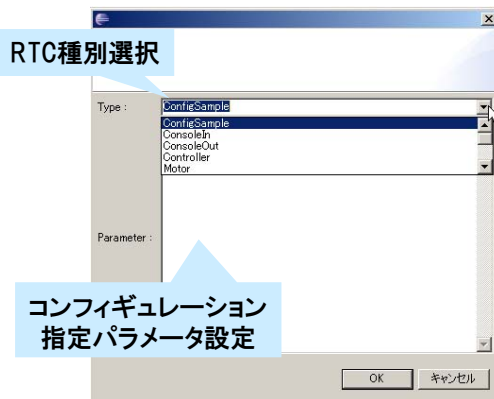
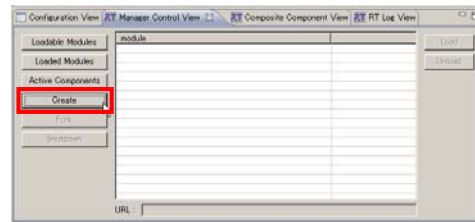
- RTコンポーネントが属する実行コンテキスト(EC)を一覧表示



属性名	説明
id	ECのID. オンラインの場合には, context_handleを表示
kind	ECの種別(PERIODIC/EVENT_DRIVEN/OTHER)
state	ECの状態(RUNNING/STOPPING)
component state	対象RTCの状態(ACTIVE/INACTIVE/ERROR)
owner	対象ECを所有しているオーナーRTCのインスタンス名
participants	対象ECに参加中のRTCの数

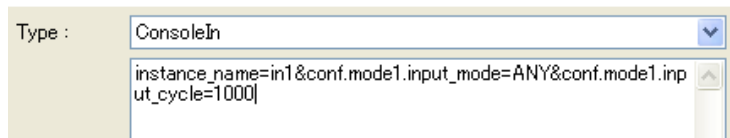
※対象ECの実行周期の変更, EC自身の動作開始/終了, 新規RTCへのアタッチ, アタッチ済みRTCのデタッチも可能

■ RTコンポーネントの新規インスタンスの生成



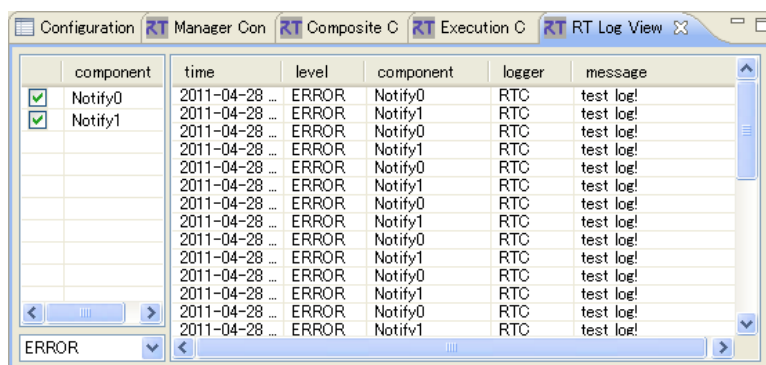
● コンフィギュレーション指定パラメータ

- conf. [ConfigSet名]. [Configパラメータ名]=[設定値]
の形式にてConfigurationSetの値も設定可能



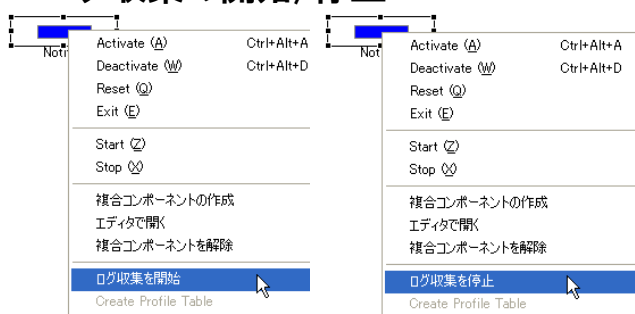
ログビュー

■ 選択したRTCから収集したログ情報を一覧表示

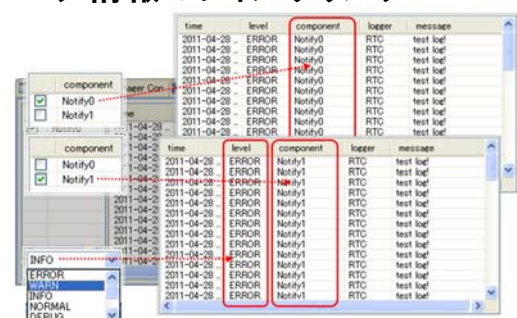


※近日機能追加予定

● ログ収集の開始/停止



● ログ情報のフィルタリング

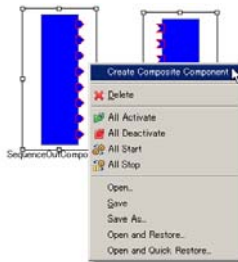


複合コンポーネント

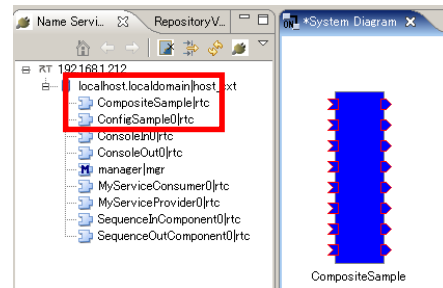
- 複数のRTCをまとめて、1つのRTCとして扱うための仕組み

- 複合コンポーネントの作成方法

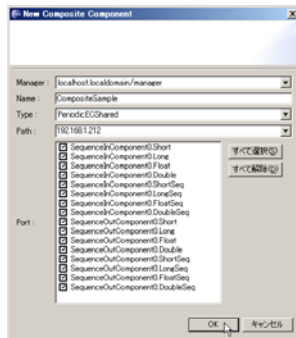
① 複数RTCを選択している状態で右クリック



③ 複合コンポーネントを生成



② 複合コンポーネントのプロパティを設定



項目	設定内容
Manager	複合コンポーネントを制御するマネージャを選択
Name	複合コンポーネントのインスタンス名を入力
Type	複合コンポーネントの型を選択
Path	複合コンポーネントのパスを入力
Port	外部に公開するポートを選択

※生成対象複合コンポーネント外部と接続されているPortは強制的に公開されます

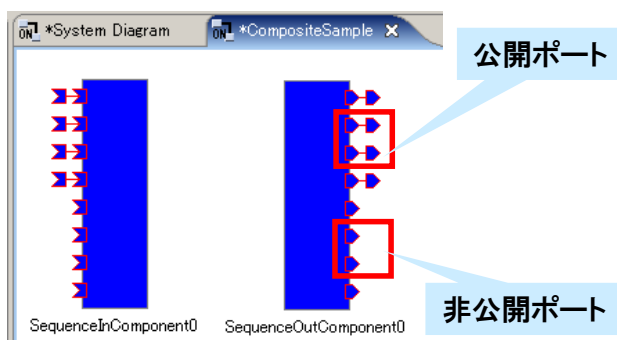
複合コンポーネント

- 複合コンポーネントのタイプについて

タイプ名	説明
PeriodicECShared	実行主体であるExecutionContextのみを共有. 各子コンポーネントはそれぞれの状態を持つ
PeriodicStateShared	実行主体であるExecutionContextと状態を共有
Grouping	便宜的にツール上のみでグループ化

- 複合コンポーネントエディタ

- 複合コンポーネントをダブルクリックすることで表示



※エディタ内に別RTCをDnDすることで、子コンポーネントの追加が可能
→追加したRTCのポートは
全て非公開に設定

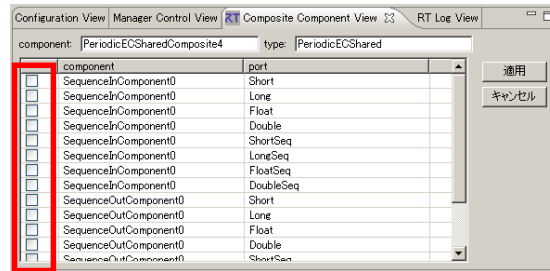
※エディタ内のRTCを削除することで、子コンポーネントの削除が可能
→削除されたRTCは、親エディタに表示

複合コンポーネント

■ 公開ポートの設定

● 複合コンポーネントビュー

ポート公開情報

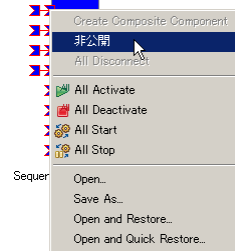
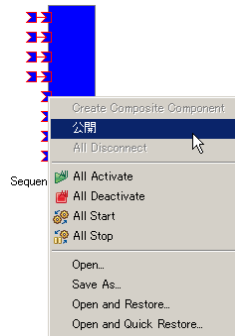


※ポート公開情報を変更し、「適用」をクリック

● 複合コンポーネントエディタ

※非公開ポートを「公開」

※公開ポートを「非公開」



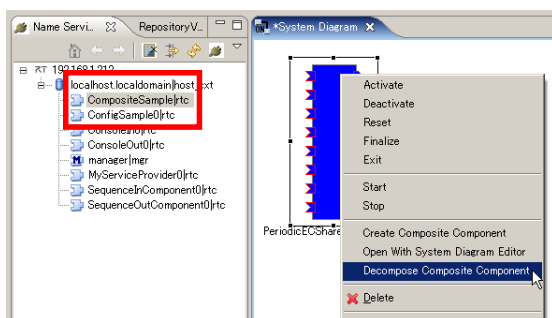
外部コンポーネントと接続されているポートを「非公開」に設定することはできません

複合コンポーネント

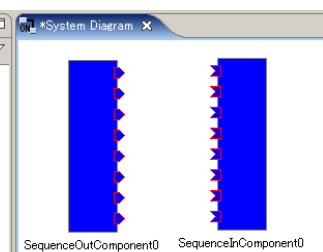
■ 複合コンポーネントの解除

①複合RTCを右クリックし、複合コンポーネントの解除を選択

②複合コンポーネントが分解され、内部のRTCが表示

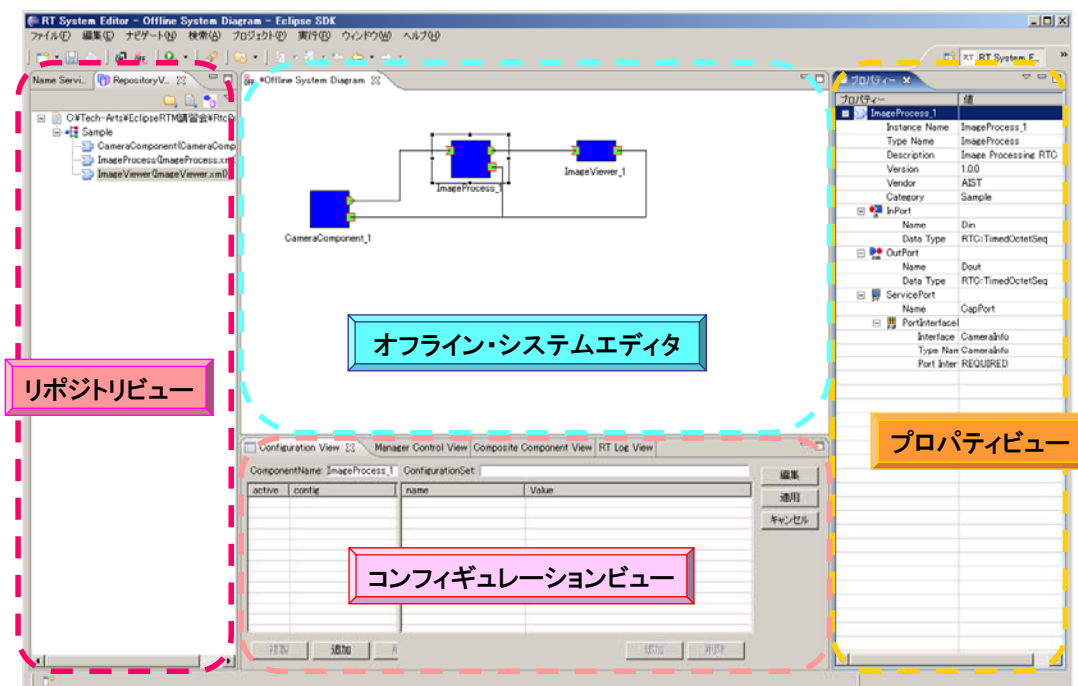


ネームサーバの登録も解除される



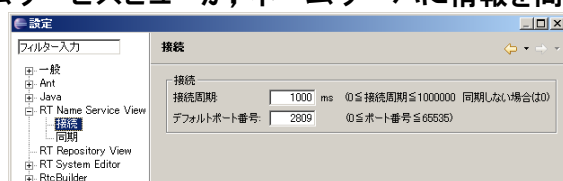
※エディタ上で、(Deleteキーなどで)単純に削除した場合は、エディタから表示が消えるのみ複合コンポーネントは解除されない

- RTコンポーネントの仕様を用いてRTシステムを構築
- 実際のRTコンポーネントが動作している必要はない

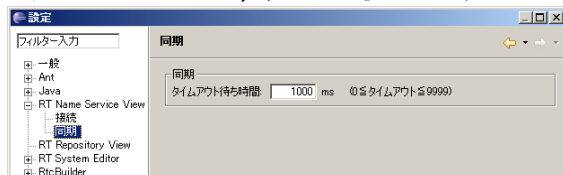


設定画面

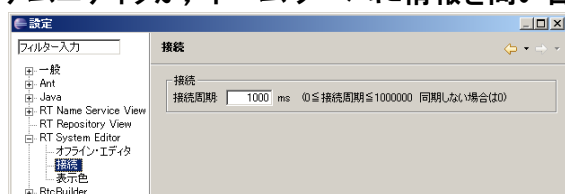
- 「RT Name Service View」-「接続」【接続周期】
 - ネームサービスビューが、ネームサーバに情報を問い合わせる周期



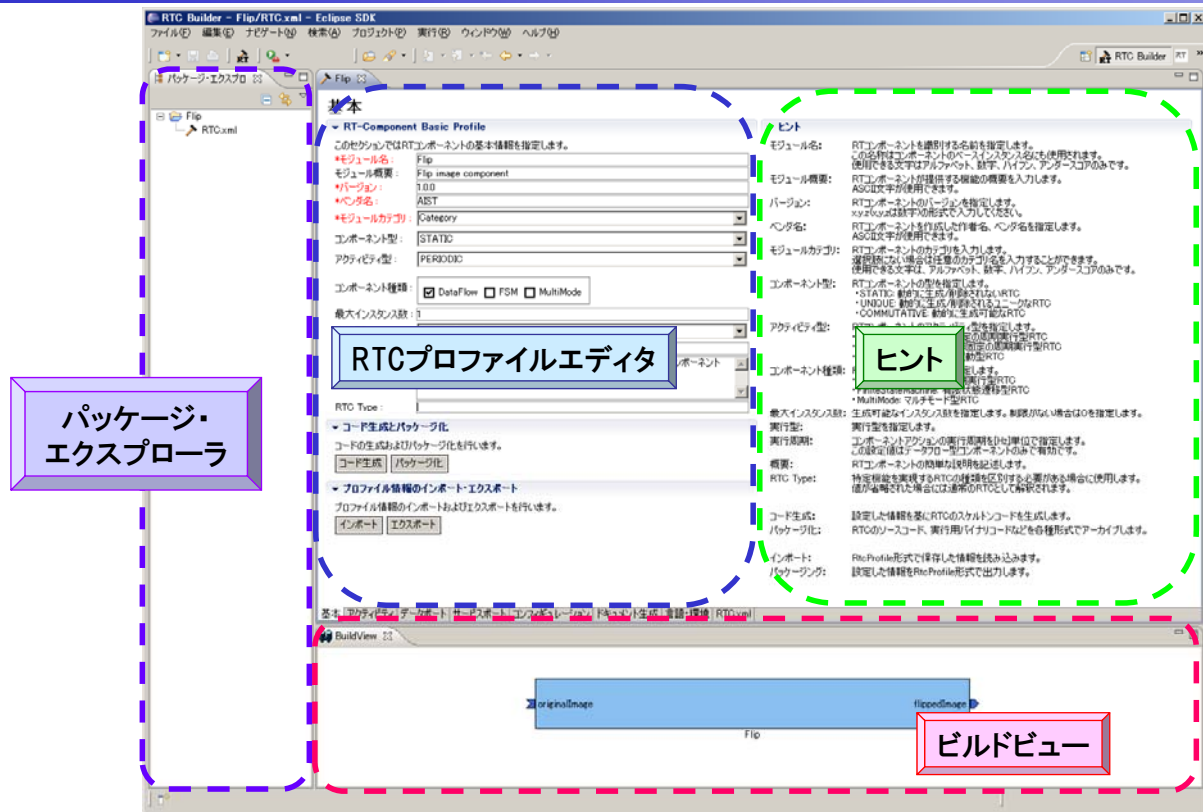
- 「RT Name Service View」-「同期」【タイムアウト待ち時間】
 - ネームサービスビューが、リモートオブジェクトのレスポンスを待つ時間



- 「RT System Editor」-「接続」【接続周期】
 - システムエディタが、ネームサーバに情報を問い合わせる周期



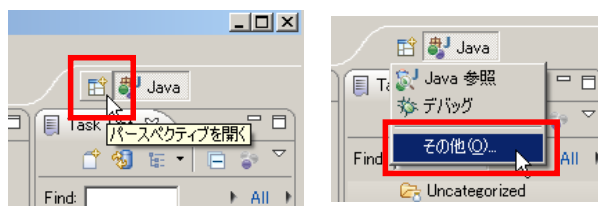
【接続周期】をゼロに設定すると
ネームサーバとの同期を行わない



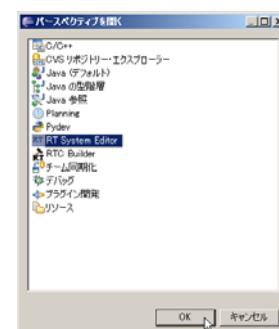
準備

■ パースペクティブの切り替え

①画面右上の「パースペクティブを開く」を選択し、一覧から「その他」を選択



②一覧画面から対象ツールを選択

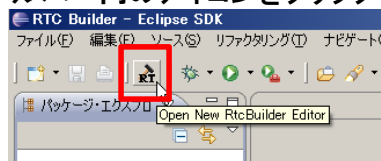


※パースペクティブ

Eclipse上でツールの構成を管理する単位
メニュー、ツールバー、エディタ、ビューなど
使用目的に応じて組み合わせる
独自の構成を登録することも可能

プロジェクト作成/エディタ起動

① ツールバー内のアイコンをクリック

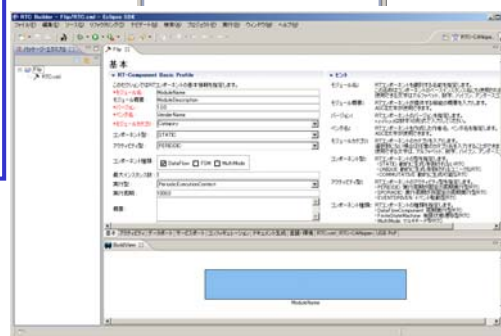
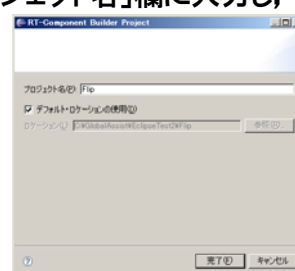


- ※メニューから「ファイル」-「新規」-「プロジェクト」を選択
【新規プロジェクト】画面にて「その他」-「RtcBuilder」を選択し、「次へ」
- ※メニューから「ファイル」-「Open New Builder Editor」を選択

- ※任意の場所にプロジェクトを作成したい場合
②にて「デフォルト・ロケーションの使用」チェックボックスを外す
「参照」ボタンにて対象ディレクトリを選択
→物理的にはワークスペース以外の場所に作成される
論理的にはワークスペース配下に紐付けされる

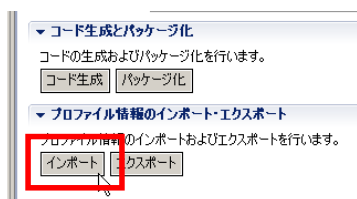
プロジェクト名: Flip

② 「プロジェクト名」欄に入力し, 「終了」

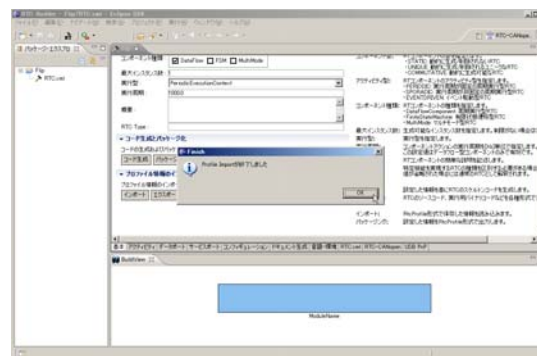


プロファイル インポート

① 「基本」タブ下部の「インポート」ボタンをクリック



② 【インポート】画面にて対象ファイルを選択



- 作成済みのRTコンポーネント情報を再利用
 - 「エクスポート」機能を利用して出力したファイルの読み込みが可能
 - コード生成時に作成されるRtcProfileの情報を読み込み可能
 - XML形式, YAML形式での入出力が可能

■ コード生成

RTC Type :

▼ コード生成とパッケージ化

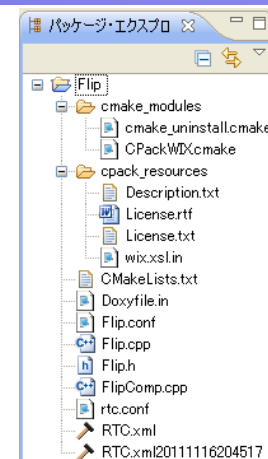
コードの生成およびパッケージ化を行います。

コード生成 パッケージ化

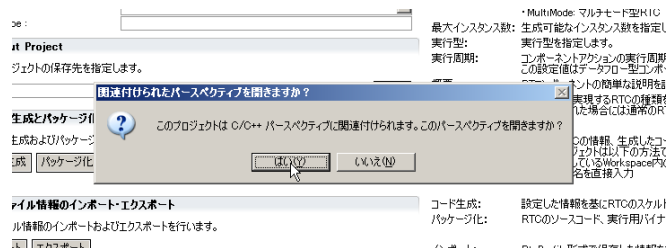
▼ プロファイル情報のインポート・エクスポート

プロファイル情報のインポートおよびエクスポートを行います。

インポート エクスポート



■ コード生成実行後、パースペクティブを自動切替



※生成コードが表示されない場合には、「リフレッシュ」を実行

C++版RTC → CDT

Java版RTC → JDT

(デフォルトインストール済み)

Python版 → PyDev

RTCプロファイルエディタ

画面要素名	説明
基本プロファイル	RTコンポーネントのプロファイル情報など、コンポーネントの基本情報を設定。 コード生成、インポート/エクスポート、パッケージング処理を実行
アクティビティ・プロファイル	RTコンポーネントがサポートしているアクティビティ情報を設定
データポート・プロファイル	RTコンポーネントに付属するデータポートに関する情報を設定
サービスポート・プロファイル	RTコンポーネントに付属するサービスポートおよび各サービスポートに付属するサービスインターフェースに関する情報を設定
コンフィギュレーション	RTコンポーネントに設定するユーザ定義のコンフィギュレーション・パラメータセット情報およびシステムのコンフィギュレーション情報を設定
ドキュメント生成	生成したコードに追加する各種ドキュメント情報を設定
言語・環境	生成対象コードの選択やOSなどの実行環境に関する情報を設定
RTC.xml	設定した情報を基に生成したRTC仕様(RtcProfile)を表示

RT-Component Basic Profile

このセクションではRTコンポーネントの基本情報を指定します。

*モジュール名:

Flip

モジュール概要:

Flip image component

*バージョン:

1.0.0

*ベンダ名:

AIST

*モジュールカテゴリ:

Category

コンポーネント型:

STATIC

アクティビティ型:

PERIODIC

コンポーネント種類:

☒ DataFlow ☐ FSM ☐ MultiMode

最大インスタンス数:

1

実行型:

PeriodicExecutionContext

実行周期:

0.0

概要:

OpenCVライブラリのうち、cvFlip関数を用いて画像の反転を行うコンポーネント

RTC Type:

ヒント

モジュール名:

RTコンポーネントを識別する名前を指定します。
この名前はコンポーネントのベースインスタンス名にも使用されます。
使用できる文字はアルファベット、数字、ハイフン、アンダースコアのみです。

モジュール概要:

RTコンポーネントが提供する機能の概要を入力します。
ASCII文字が使用できます。

バージョン:

RTコンポーネントのバージョンを指定します。

モジュール名: Flip

モジュール概要: 任意(Flip image compon

バージョン: 1.0.0

ベンダ名: 任意(AIST)

モジュールカテゴリ: 任意(Category)

コンポーネント型: STATIC

アクティビティ型: PERIODIC

コンポーネントの種類: DataFlow

最大インスタンス数: 1

実行型: PeriodicExecutionContext

実行周期: 1000.0

81

アクティビティ

このセクションでは使用するアクショングループを指定します。

コンポーネントの起動時に終了処理に関するアクション

onInitialize onFinalize

実行エンタキの起動と停止に関するアクション

onStartup onShutdown

alive状態でのコンポーネントアクション

onActivated onDeactivated onAborting

onError onReset

Dataflow型コンポーネントのアクション

onExecute onStateUpdate onRateChanged

FSM型コンポーネントのアクション

onAction

Mode型コンポーネントのアクション

onModeChanged

▼ Documentation

このセクションでは各アクションの概要を説明するドキュメントを記述します。
上段のアクションを選択すると、それと対応するドキュメントが記述されます。

アクティビティ名: ON OFF

動作概要: コンポーネント自身の各種初期化処理

事前条件: なし

事後条件: コンポーネントの初期化処理が正常に完了している

① 設定対象のアクティビティを選択

初期化処理です。コンポーネントライフサイクルが開始時に一度だけ呼びます。常に有効、終了処理です。コンポーネントライフサイクルが終了時に一度だけ呼びます。
ExecutionContext切替時も開始するときに一度だけ呼びます。
非アクティブ状態からアクティブ化されるときに一度だけ呼びます。
アクティブ状態から非アクティブ化されるときに一度だけ呼びます。
ERROR状態に入る前に一度だけ呼びます。
ERROR状態から回復するときに一度だけ呼びます。
ERROR状態からリセットされるアクティブ状態に移行するときに一度だけ呼びます。
アクティブ状態時に定期的に呼びます。
onExecuteの準備処理を呼びます。
ExecutionContextが変更されたときに呼びます。
対応する状態に応じた動作を実行するため呼びます。
onModeChanged モードが変更されたときに呼びます。

動作概要: アクティビティの概要説明を記述します。
事前条件: アクティビティを実行する前に成立すべき事前条件を記述します。
事後条件: アクティビティを実行した後に成立すべき事後条件を記述します。

② 使用/未使用を設定

☒ ON ☐ OFF

以下をチェック:
onActivated
onDeactivated
onExecute

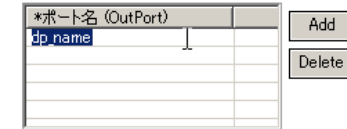
82

■ 生成対象RTCに付加するDataPortの情報を設定

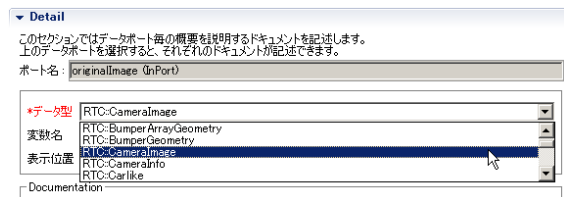


① 該当種類の欄の「Add」ボタンをクリックし、ポートを追加後、直接入力で名称設定

ポート)の情報を設定します。

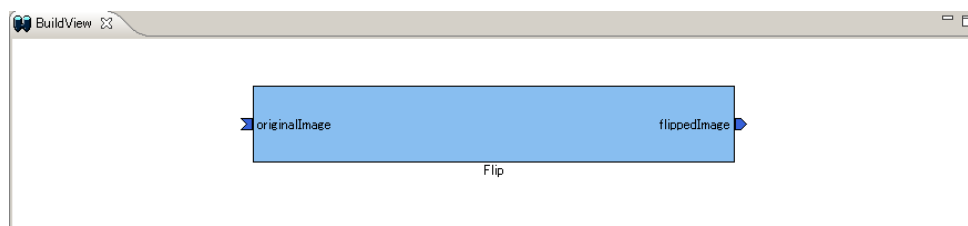


② 設定する型情報を一覧から選択



- ※データ型は、型定義が記載されたIDLファイルを設定画面にて追加することで追加可能
- ※OpenRTM-aistにて事前定義されている型については、デフォルトで使用可能
→ [RTM_Root]rtm/idl 以下に存在するIDLファイルで定義された型
- ※各ポートに対する説明記述を設定可能
→ 記述した各種コメントは、生成コード内にDoxygen形式で追加される

※Portの設定内容に応じて、下部のBuildViewの表示が変化



- InPort
ポート名: **originalImage**
データ型: **RTC::CameraImage**
変数名: **originalImage**
表示位置: **left**
- OutPort
ポート名: **flippedImage**
データ型: **RTC::CameraImage**
変数名: **flippedImage**
表示位置: **right**

サービスポート・プロファイル

■ 生成対象RTCに付加するServicePortの情報を設定

■ サービスインターフェースの指定

- IDLファイルを指定すると、定義されたインターフェース情報を表示

今回のサンプルでは未使用

コンフィギュレーション・プロファイル

■ 生成対象RTCで使用する設定情報を設定

コンフィギュレーション・パラメータ

- ①「Add」ボタンをクリックし、追加後、直接入力で名称設定

- ②詳細画面にて、型情報、変数名などを設定

名称: flipMode
データ型: int
デフォルト値: 0
変数名: flipMode
制約条件: (-1, 0, 1)
Widget: radio

- ※データ型は、short,int,long,float,double,stringから選択可能(直接入力も可能)
- ※制約情報とWidget情報を入力することで、RTSystemEditorのコンフィギュレーションビューの表示を設定することが可能

制約条件, Widgetの設定方法

■ 制約条件について

- データポートとコンフィギュレーションに設定可能
- チェックはあくまでも**コンポーネント開発者側の責務**
 - ミドルウェア側で検証を行っているわけではない

■ 制約の記述書式

- 指定なし: 空白
- 即値: 値そのもの
 - 例) 100
- 範囲: <, >, <=, >=
 - 例) 0<=x<=100
- 列挙型: (値1, 値2, ...)
 - 例) (val0, val1, val2)
- 配列型: 値1, 値2, ...
 - 例) val0, val1, val2
- ハッシュ型: { key0: 値0, key1: 値1, ... }
 - 例) { key0: val0, key1: val1 }

■ Widget

- text(テキストボックス)
 - デフォルト
- slider(スライダー)
 - **数値型**に対して**範囲指定**の場合
 - 刻み幅をstepにて指定可能
- spin(スピナ)
 - **数値型**に対して**範囲指定**の場合
 - 刻み幅をstepにて指定可能
- radio(ラジオボタン)
 - 制約が**列挙型**の場合に指定可能

※ 指定したWidgetと制約条件がマッチしない場合は、テキストボックスを使用

言語・環境・プロファイル

■ 生成対象RTCを実装する言語, 動作環境に関する情報を設定

言語・環境

▼ 言語
このセクションでは使用する言語を指定します

☒ C++
☐ Python
☐ Java
☐ Ruby

☐ Use old build environment.

▼ ヒント
言語: RTコンポーネントを作成する言語を選択します。リスト中の言語から選択可能です。
環境: 言語ごとのライブラリの依存関係や、使用するOSなどの環境を選択します。
詳細情報で設定した内容(OS情報、ライブラリ情報など)は、プロファイル内にのみ保存されます。

▼ 環境
このセクションでは依存するライブラリや使用するOSなどを指定します

Version	OS

Add
Delete

詳細情報

OS Version	CPU

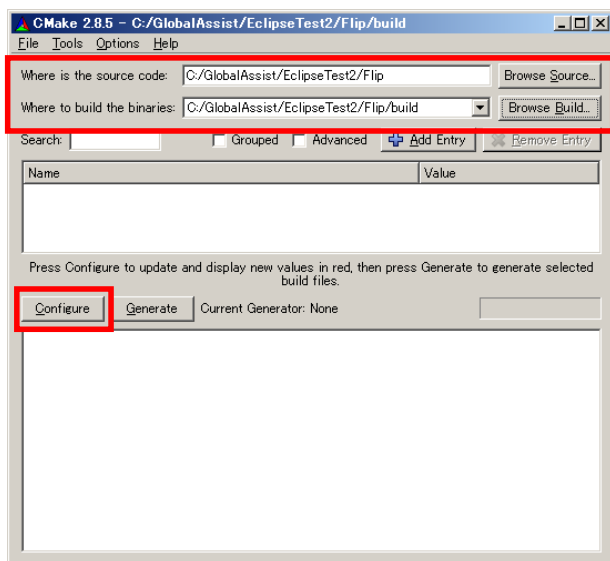
Add
Delete

このチェックボックスをONにすると、旧バージョンと同様なコード(Cmakeを利用しない形式)を生成

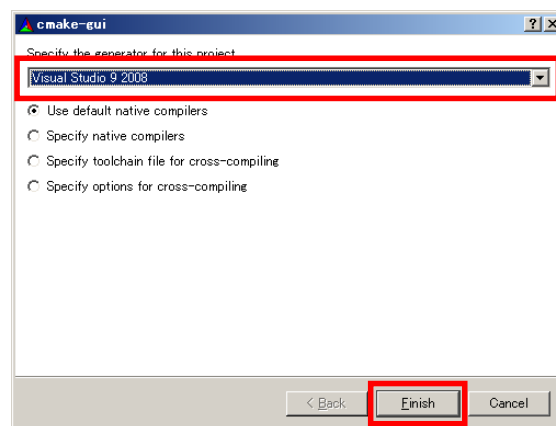
「C++」を選択

コンパイル(Windows,CMake利用)

①GUI版Cmakeを起動し, source, binaryのディレクトリを指定



②「Configure」を実行し, 使用するプラットフォームを選択

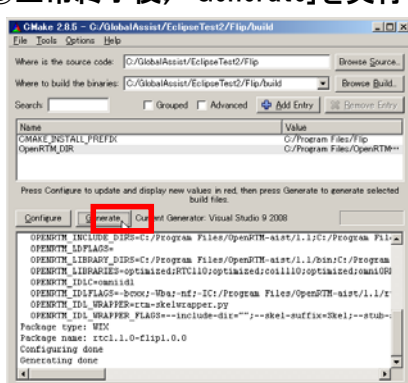


※binaryには, sourceとは別のディレクトリを指定する事を推奨

※日本語は文字化けしてしまうため英数字のみのディレクトリを推奨

コンパイル(Windows,CMake利用)

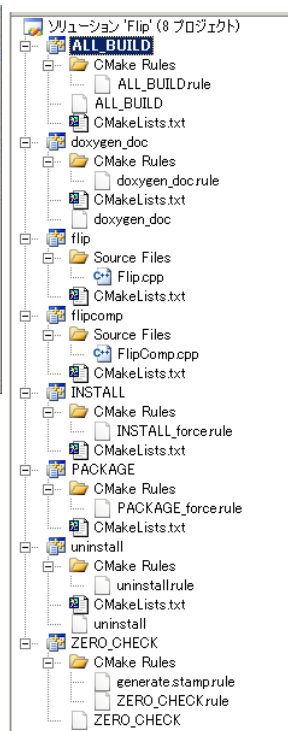
③正常終了後, 「Generate」を実行



④binaryとして指定したディレクトリ内にあるソリューションファイルを開く

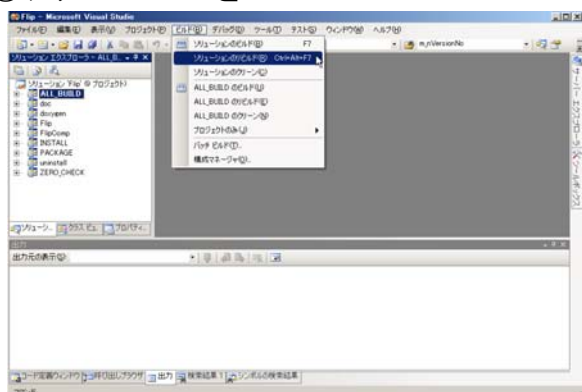


9 Flip
Microsoft Visual Studio Solution
Version: Visual Studio 2008

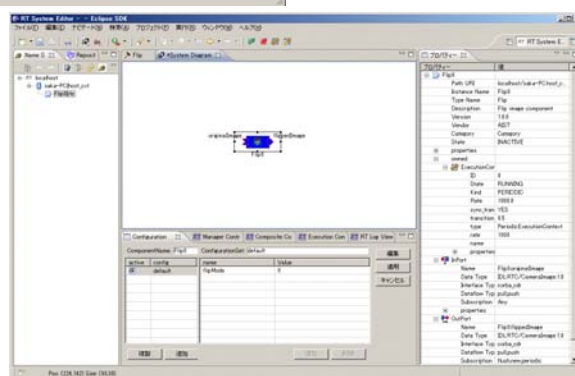
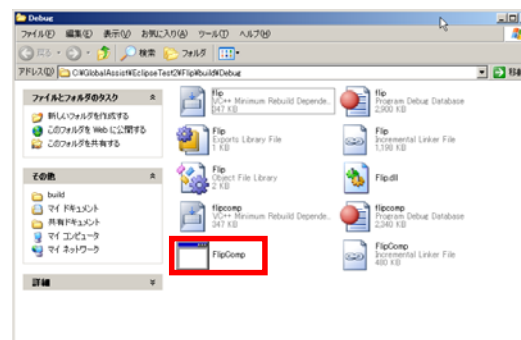


コンパイル・実行(Windows,CMake利用)

⑤ソリューションをビルド



⑥binaryにて指定したディレクトリ以下のSrc/Debug内のFlipComp.exeを起動



ドキュメント作成(Windows,CMake利用)

※binaryにて指定したディレクトリ以下のdoc/html/doxygen/html以下にドキュメント



■ 生成されたドキュメントの例

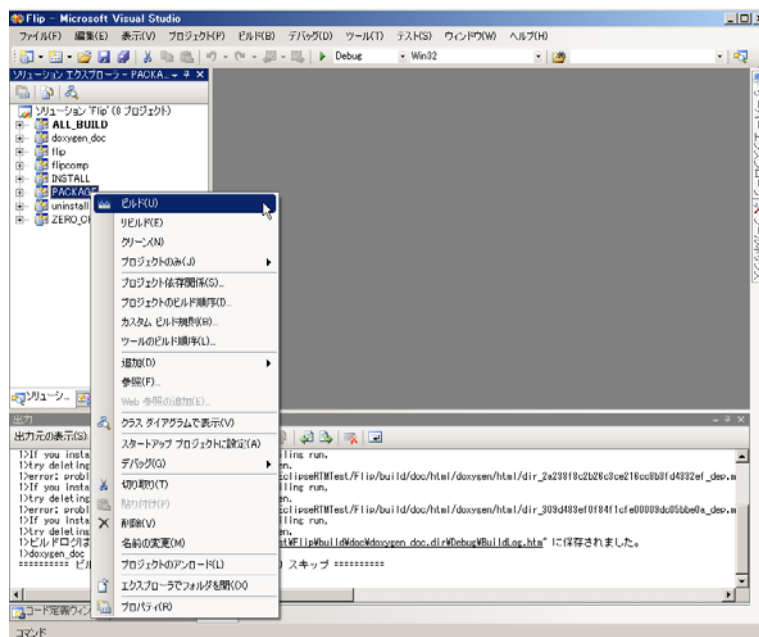
flip 1.0.0



flip 1.0.0



■ ソリューション中の「PACKAGE」をビルド



- binaryにて指定したディレクトリ直下にmsi形式のインストールパッケージを生成
 - コンポーネントのインストール先
C:¥Program Files¥OpenRTM-aist¥1.1¥components¥<言語>/<パッケージ名>

既存コンポーネントの再利用

■ プロジェクトとは

- ユーザが作成した様々なコンポーネントやツールの公開場所
- ユーザ登録すれば、誰でも自分の成果物の紹介ページを作成可能
- 他のユーザに自分のコンポーネント等を紹介することができる

■ プロジェクトのカテゴリ

- RTコンポーネント: 1つのコンポーネントまたは複数のコンポーネント群などが登録されています。
- RTミドルウェア: OpenRTM-aistや他のミドルウェア、ミドルウェア拡張モジュール等が登録されています。
- ツール: 各種ツール(RTSystemEditorやrtshellを含む)ツールはこのカテゴリになります。
- 関連ドキュメント: 関連ドキュメントとは、各種インターフェースの仕様書やマニュアル等を含みます。

既存コンポーネントの再利用

■ プロジェクトから対象コンポーネントを取得

■ 「顔検出コンポーネント」

<http://www.openrtm.org/openrtm/ja/project/facedetect>

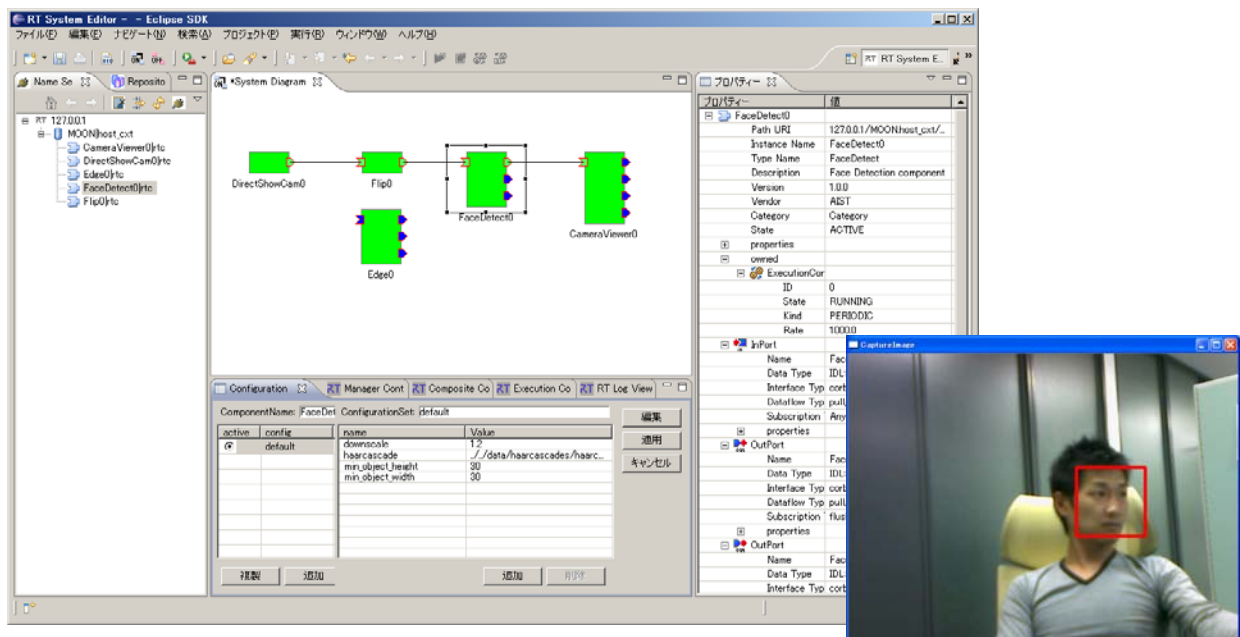
対象コンポーネントをダウンロード



既存コンポーネントの再利用

■ ダウンロードしたファイル(FaceDetect.zip)を解凍

■ 解凍したディレクトリ内の以下のファイルを実行し、システムエディタ上に配置 \$(FaceDetect_Root)/build/Release/FaceDetectComp.exe



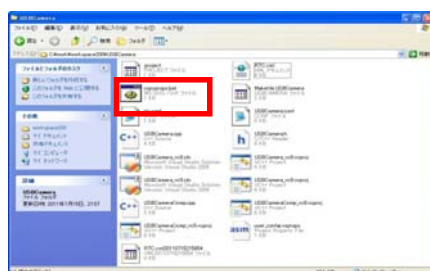
RTCBUILDER補足説明



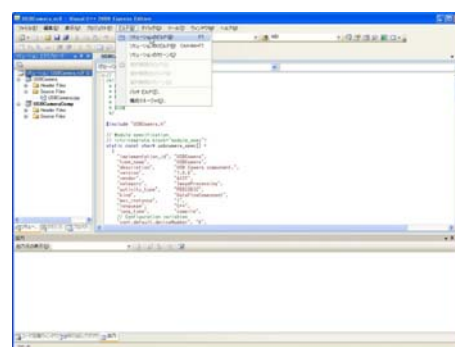
コンパイル・実行(Windows)



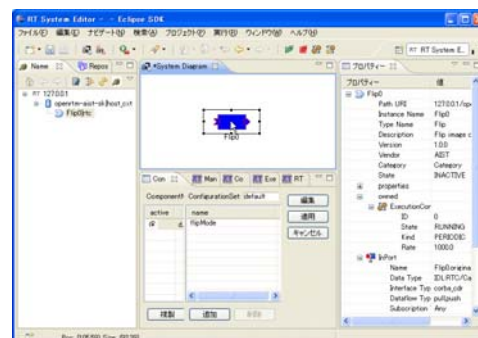
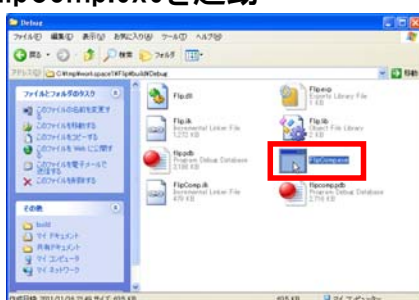
- ①コード生成先ディレクトリ内の「copyprops.bat」をダブルクリックして、設定ファイルをコピー



- ②VisualStudioを用いたビルド

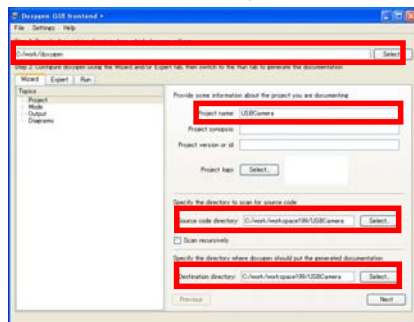


- ③FlipComp¥¥Debug内のFlipComp.exeを起動

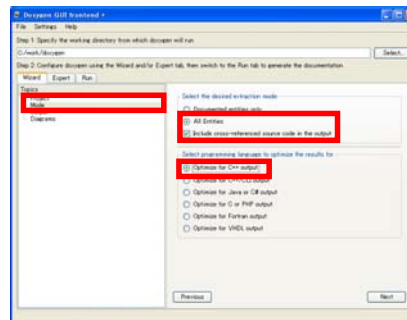


ドキュメント作成(Windows)

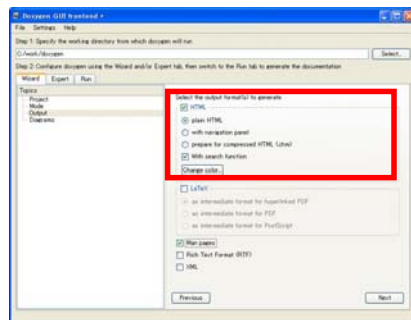
- ① Doxygen用GUIツールを起動
作業用ディレクトリ,ソース格納場所,
生成ファイル出力先,プロジェクト名を指定



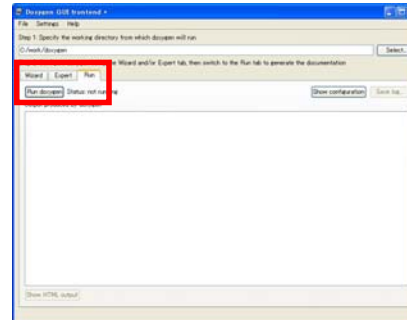
- ② 「Mode」セクションにて,
出力内容,使用言語を指定



- ③ 「Output」セクションにて, html出力を指定



- ③ 「Run」タブにて, 「Run doxygen」を実行

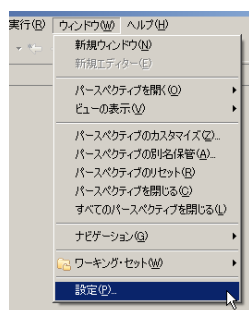


各種設定

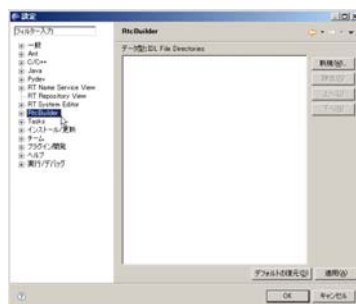
- DataPortにて利用するデータ型の指定

→データ型を定義したIDLファイルが格納されているディレクトリを指定

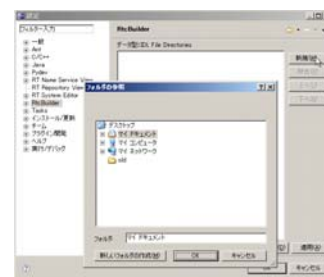
- ① メニューから
「ウインドウ」-「設定」



- ② 「RtcBuilder」を選択



- ③ 「新規」ボタンにて表示される
ディレクトリ選択ダイアログ
にて場所を指定



※ 独自に定義したデータ型を使用する場合のみ必要な設定

OpenRTM-aistにて標準で用意されている型のみを使用する場合には設定不要

・標準型の定義内容格納位置: [RTM_Root]rtm/idl

→ BasicDataType.idl, ExtendedDataTypes.idl など

→ デフォルト設定では, [RTM_Root]=C:/Program Files/OpenRTM-aist/1.1/

RTミドルウェア SUMMER CAMP 2012

日時: 2012年7月30日(月)～8月3日(金)
場所: 産業技術総合研究所 つくばセンター中央第二
本部・情報棟1F ネットワーク会議室

