ホンダにおける
RTミドルウェア開発と標準化活動

株式会社 本田技術研究所　基礎技術研究センター
関谷 眞

目次

- 知能ロボットシステム概要
- コンポーネント指向ミドルウェア
- HRTMの開発
- ASIMOへの適用
- HRTMとOpenRTM-aistの連携動作
- 標準化活動

知能ロボットシステム概要

センサーやアクチュエーターは追加や変更される

↓

システム構成は変更したくない

↓

センサー，アクチュエーターの関係を抽象化した
データ処理（知能アルゴリズム）が必要となる

知能アルゴリズムの抽象化

階層設計によって知能アルゴリズムの抽象化を実現する
知能ロボットシステムアーキテクチャー

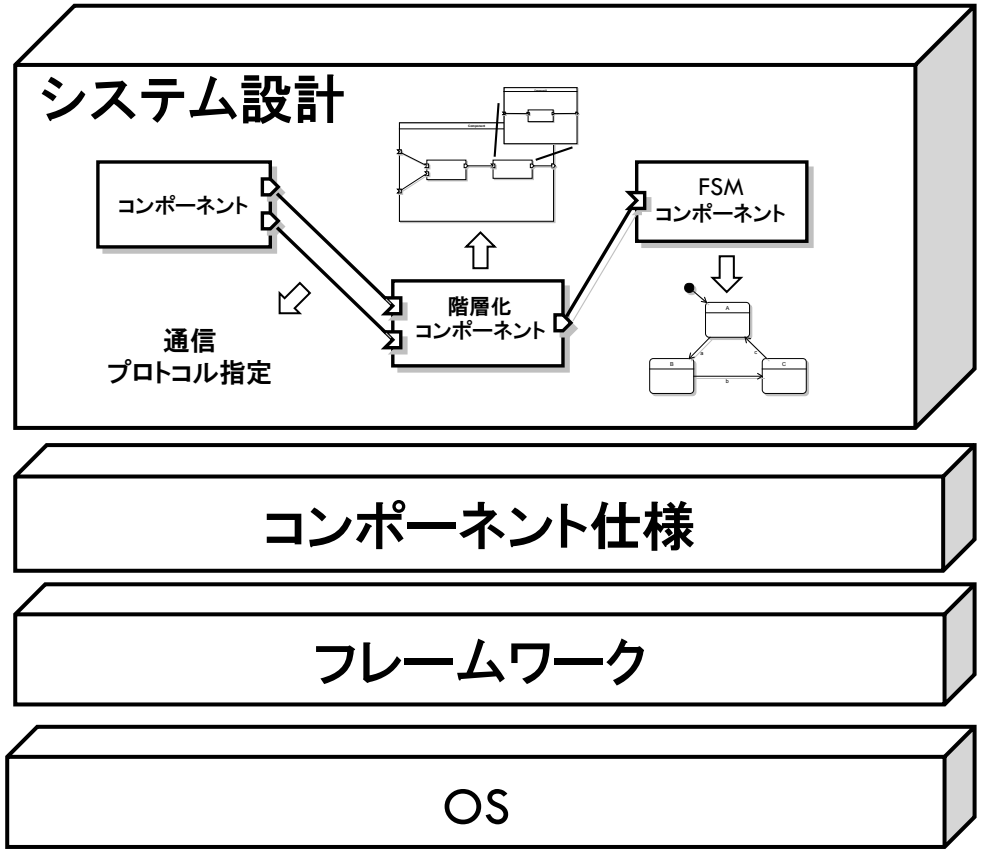
抽象化モデル層 （知能アルゴリズム）	• データフロー • ステートマシン • 抽象化センサー • 抽象化アクチュエーター
基盤機能層 （フレームワーク）	• 実行制御 • 通信
OS	
ハードウェア	

コンポーネント指向開発

- アルゴリズムの抽象化のための
  オブジェクト・コンポーネント指向開発
  - 共通のインターフェース仕様に従う
  - 同一インターフェースを持つコンポーネント同士は入れ替え可能
- コンポーネントの特徴
  - オブジェクトは開発言語仕様として定義される(例えばC++)が，コンポーネントは言語とは独立した仕様として定義することができる

コンポーネントは抽象化モデルの
実現手段として適している

コンポーネント指向ミドルウェア



コンポーネント指向ミドルウェアは
システム設計ツール，コンポーネント仕様，フレームワークで構成される

知能ロボットミドルウェアの要件

構成要素	要件
システム設計ツール	• 標準的なモデル言語でシステムを設計できること
コンポーネント仕様	• 標準的なコンポーネント仕様に準拠すること • コンポーネントを階層的に設計できること • 状態遷移型コンポーネントを設計できること
フレームワーク	• コンポーネントを様々なOS上で実行できること • コンポーネント間通信に複数の手法を利用できること • 処理時間に対するオーバーヘッドが最小限であること

ロボットミドルウエアの比較

Name	OpenRTM-aist	ROS	OROCOS
階層化 コンポーネント	実装は2階層まで	no	no
FSM コンポーネント	no	yes (Python)	no
サポートOS	Linux Windows Mac OSX	Linux	Linux Windows Mac OSX
通信プロトコル	CORBA	TCP/IP UDP	TCP/IP UDP
システム設計言語	独自ブロック図	no	no
コンポーネント仕様	RTC	no	独自仕様

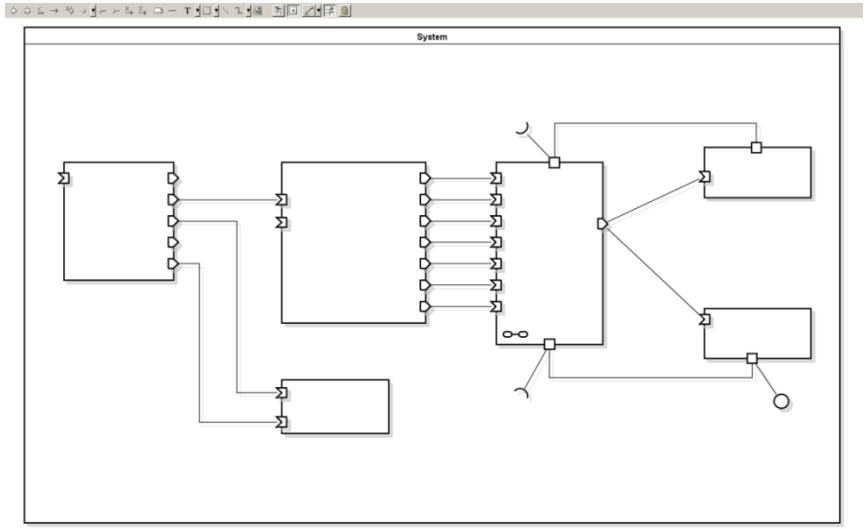
2009年当時は全ての要件を満たす
ロボットミドルウェアが存在しなかったため
RTC仕様に準拠した独自のミドルウェア開発を開始した

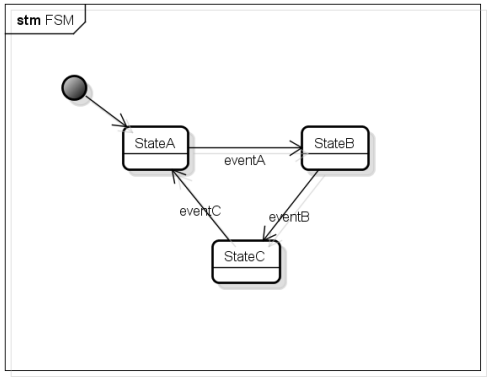
Honda Robotic Technology Middleware(HRTM)の特徴

構成要素	特徴
システム設計ツール	• UMLでシステムを設計できる
コンポーネント仕様	• RTC仕様に準拠する • コンポーネントを階層的に設計できる • 状態遷移型コンポーネントを設計できる
フレームワーク	• コンポーネントのソースコードはOSに依存しない • コンポーネント間通信 　• 複数のプロトコルを選択できる 　• ネットワーク・プロセス間・プロセス内通信モードを選択できる • ハードリアルタイム性を確保

システム設計ツール

- UMLによる視覚的な設計
  - ポートの種類や接続関係設計
  - コンポーネントの実行順指定
  - ステートチャートによるFSM設計
  - ソースコード自動生成機能
- コンポーネント実行状態のモニタリング



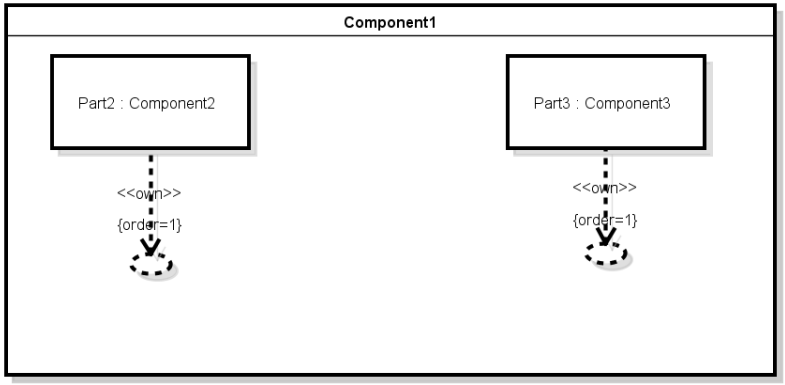


コンポーネント仕様

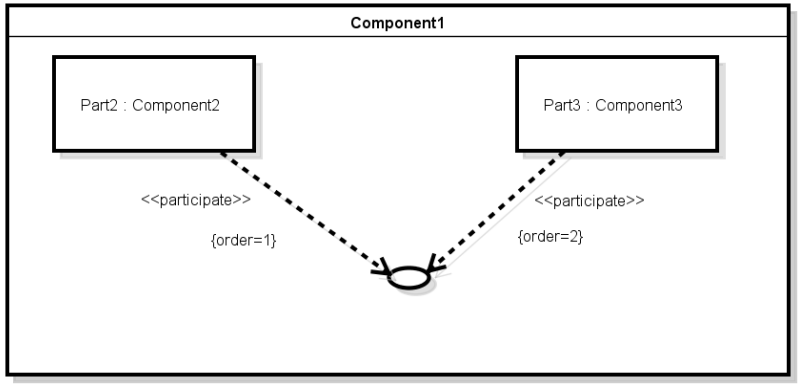
- RTC仕様（CORBA PSM）に準拠
- OpenRTM-aistとの相互接続が可能
- HRTM独自の拡張を実装

階層型コンポーネント

- 多階層化が可能
- プロパティによって，サブコンポーネントの並列実行，直列実行及び実行順序を指定可能



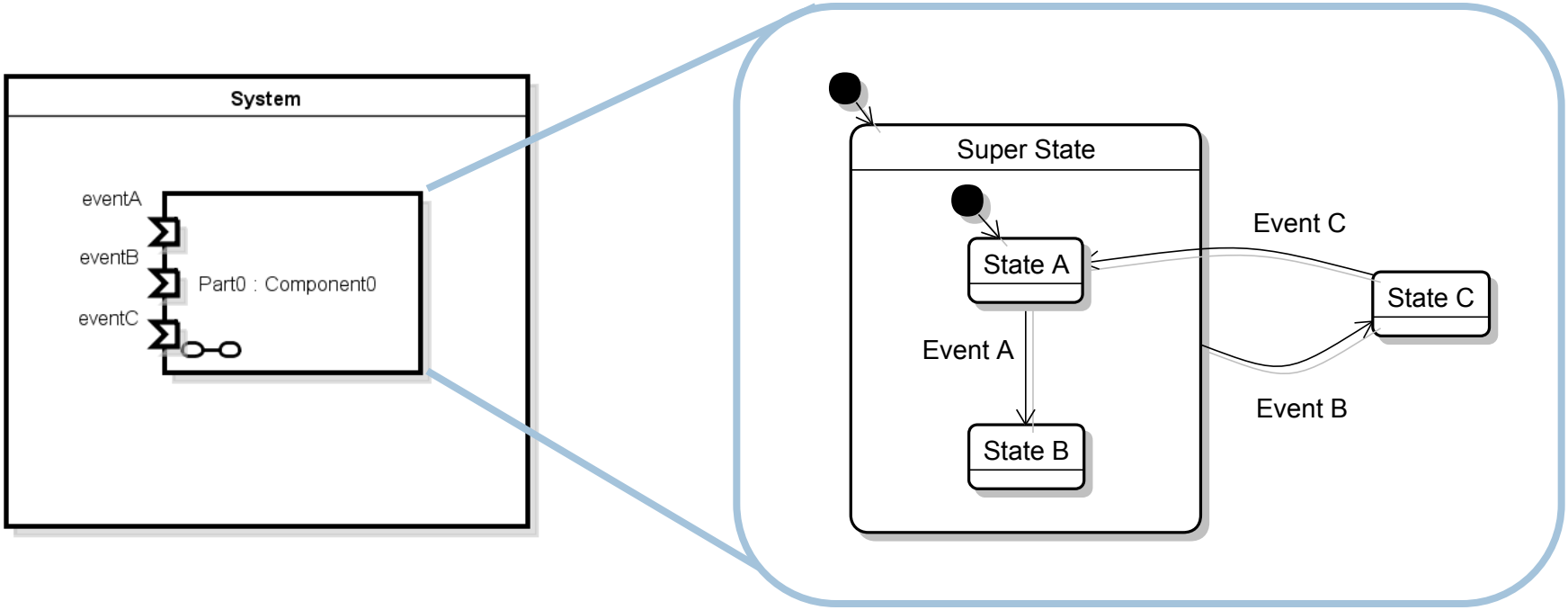
並列実行



直列実行

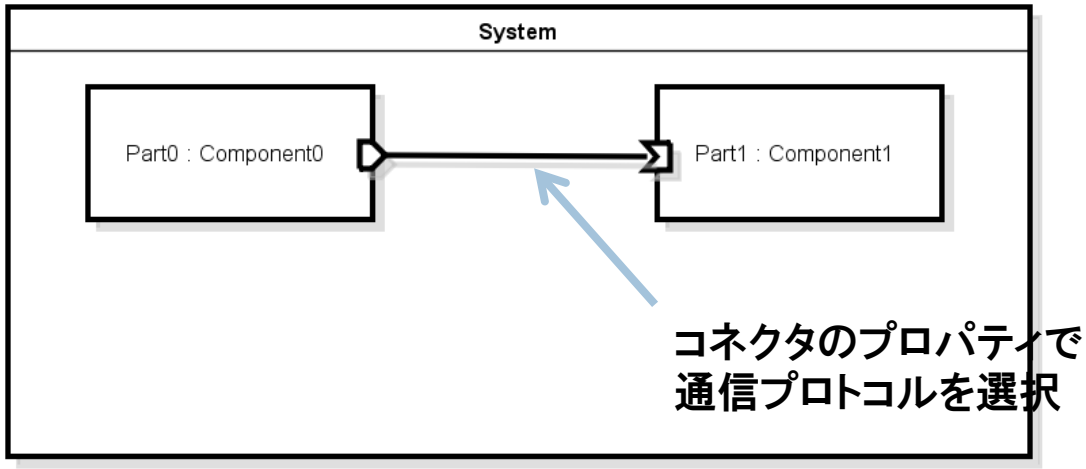
状態遷移型コンポーネント

- UML Statechartによる視覚的な状態遷移モデル設計を実現し，複雑な状態遷移を簡潔に表現できるようにした
- 設計した状態遷移図からソースコードを自動生成できる



フレームワーク

- マルチプラットフォーム対応
  - OS依存APIの抽象化層を実装し，コンポーネントのソースコードをOS非依存とした
- マルチ通信プロトコル対応
  - コンポーネント間の通信プロトコルをプロパティとして，実行時に選択する機能を実装した



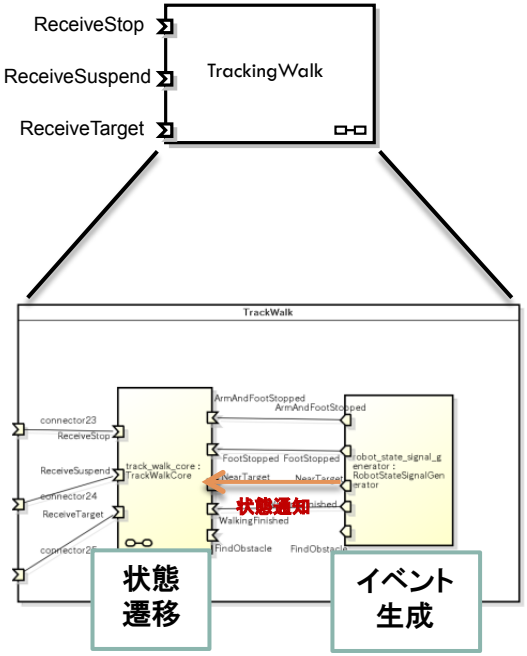
ハードリアルタイム性

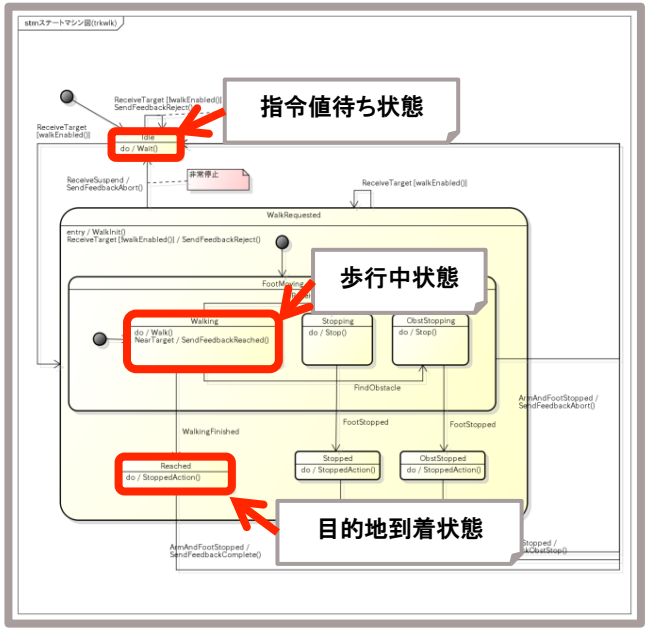
- 実行コンテキストに優先度指定を追加した
- OpenRTM-aist互換通信処理を拡張し，3つの通信モードを実装した
  - ネットワーク
  - プロセス間
  - プロセス内
- 1000バイト通信時間測定結果

通信モード	受信時間 [us]	送信時間 [us]	合計 [us]
ネットワーク	123	188	311
プロセス内	16	7	23

ASIMOへの適用

- HRTMの機能実証例として，ASIMOシステムの経路追従歩行部のコンポーネント化に成功した



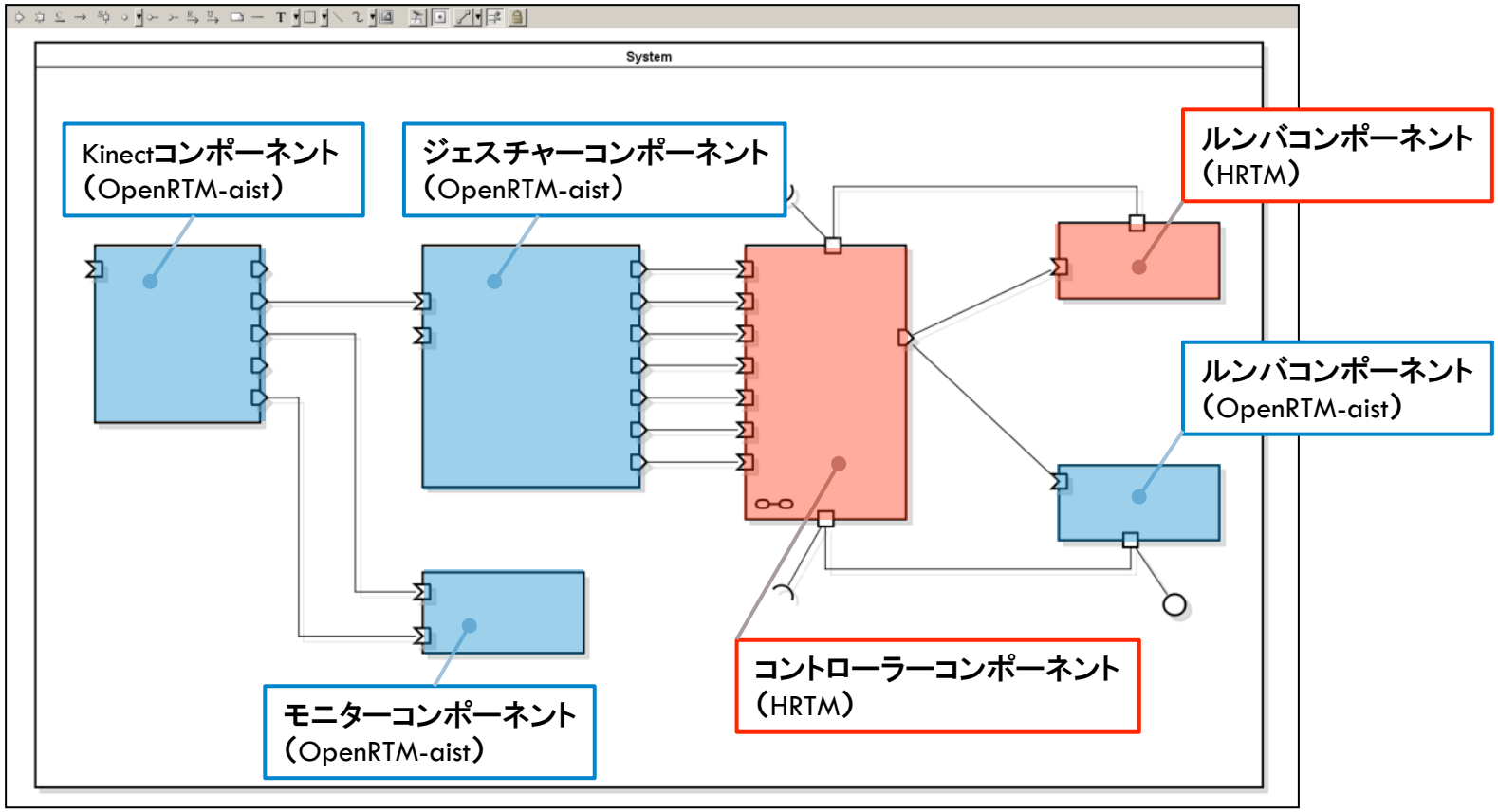


コンポーネント化後の処理時間　→ 32 μs(10%)増加

※ポート数を最小限にすることで，処理時間は減らすことができる

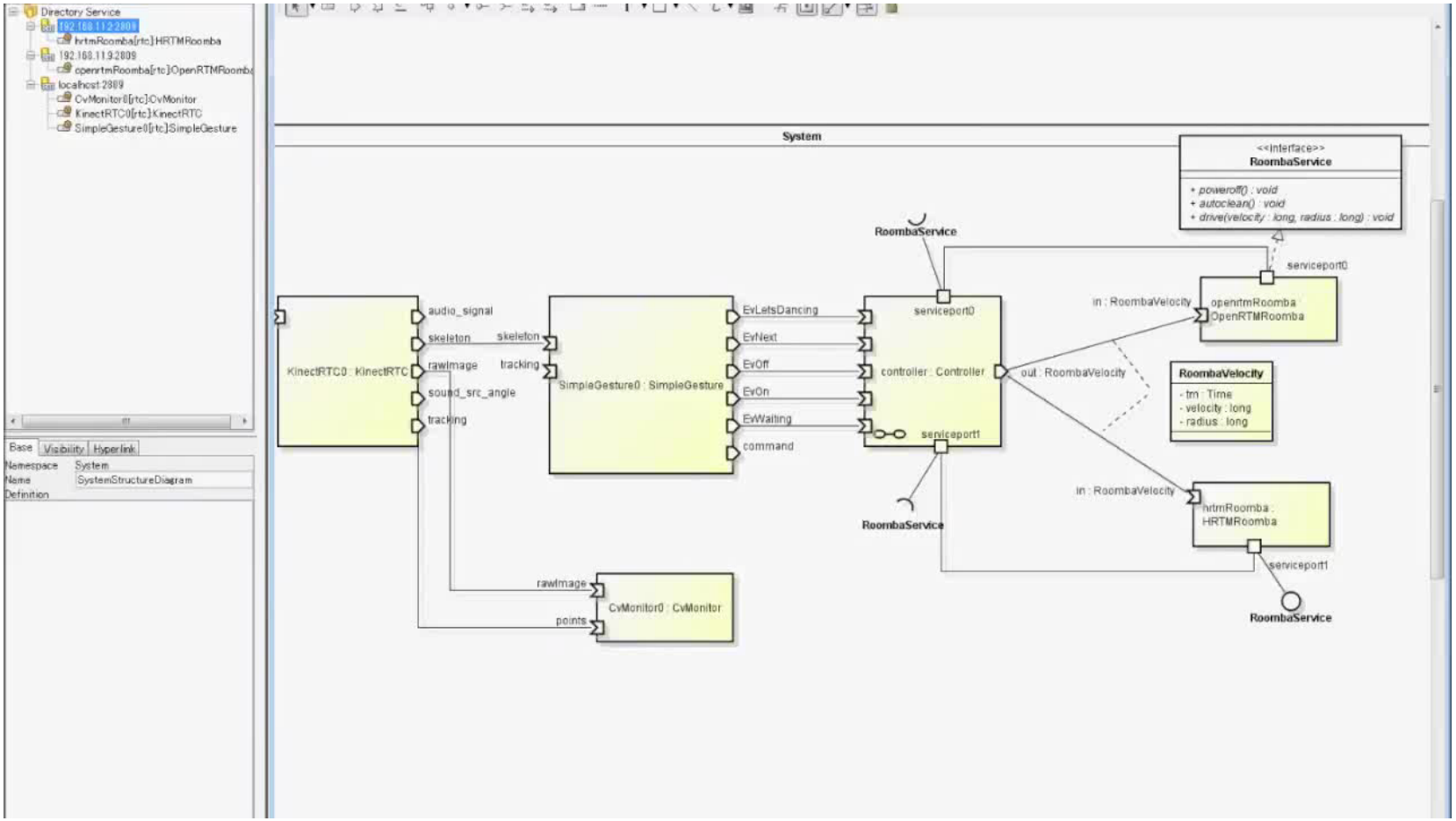
HRTMとOpenRTM-aistとの連携システム概要

- HRTMコンポーネントとOpenRTM-aistコンポーネントが相互接続して2台のルンバを操作するシステムを構築した



HRTMとOpenRTM-aistとの連携動作

2012年12月のOMG Robotics Information Dayで実施



標準化活動

- FSM4RTC(Finite State Machine for RTC)仕様
  - RTC仕様に対する拡張仕様
    - コンポーネントの状態変化を外部に通知するインターフェース仕様
    - FSMコンポーネントのための拡張インターフェース仕様
    - データポートインターフェース仕様
  - 2015年7月 現在採択仕様として公開中
    - http://www.omg.org/spec/FSM4RTC
  - 2015年9月 正式仕様化予定

まとめ

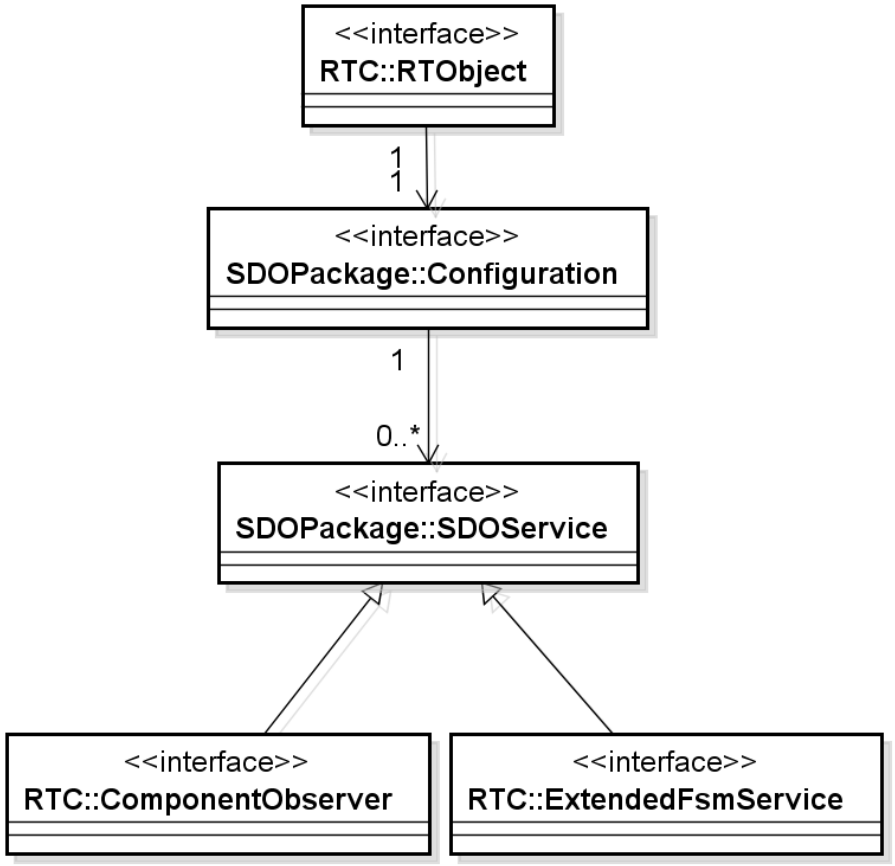
- ホンダ製ミドルウェアの狙いと特徴を示し，ASIMOシステムへの適用例を紹介した
- 拡張仕様を産総研と共に標準化作業中
- 拡張機能はOpenRTM-aistへの適用を検討中

Appendix　FSM4RTC仕様

FSM4RTCの概要

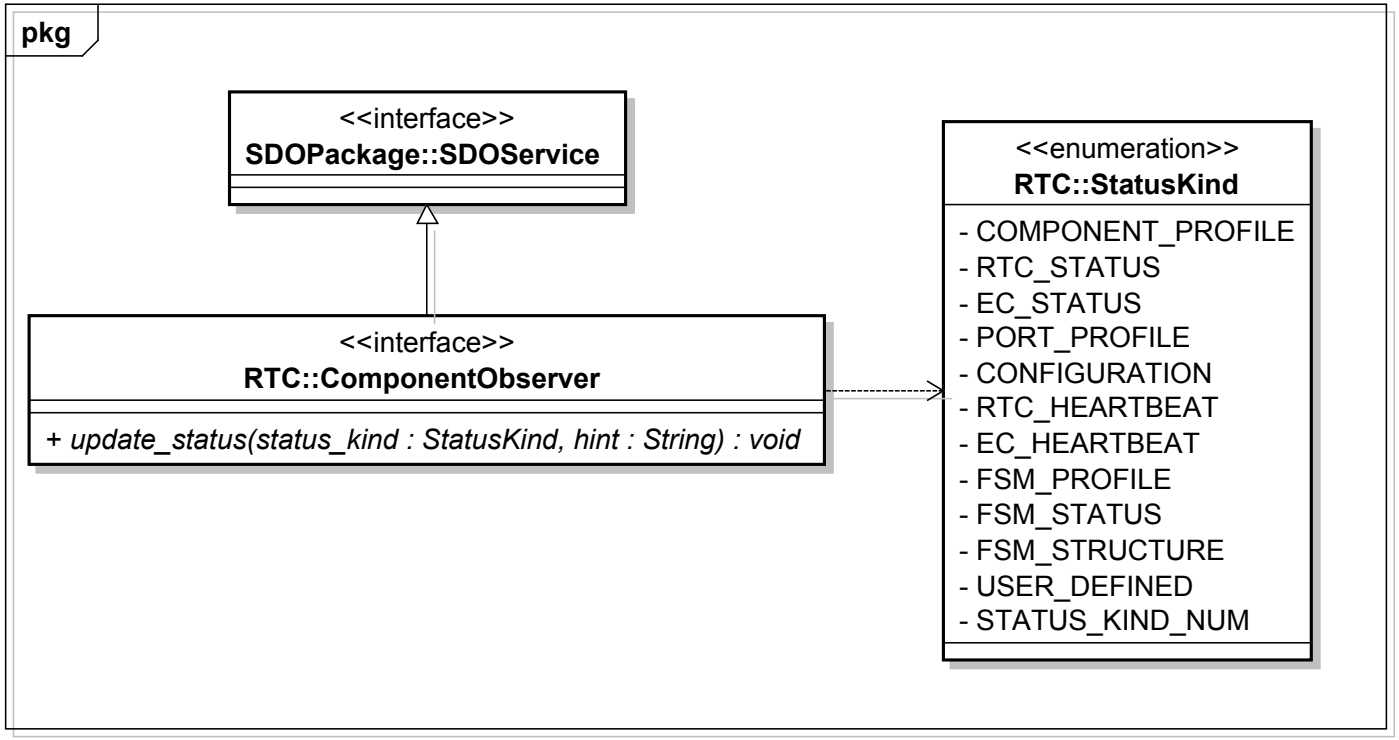
- RTCに対する拡張項目
  - ComponentObserver（SDOServiceを継承）
    - コンポーネントの状態変化を外部に通知するためのインターフェース
  - ExtendedFsmService（SDOServiceを継承）
    - FSMコンポーネントのための拡張インターフェース
  - DataPortの追加
    - コンポーネント間でイベントを送受信するためのインターフェース

SDOServiceによる機能拡張



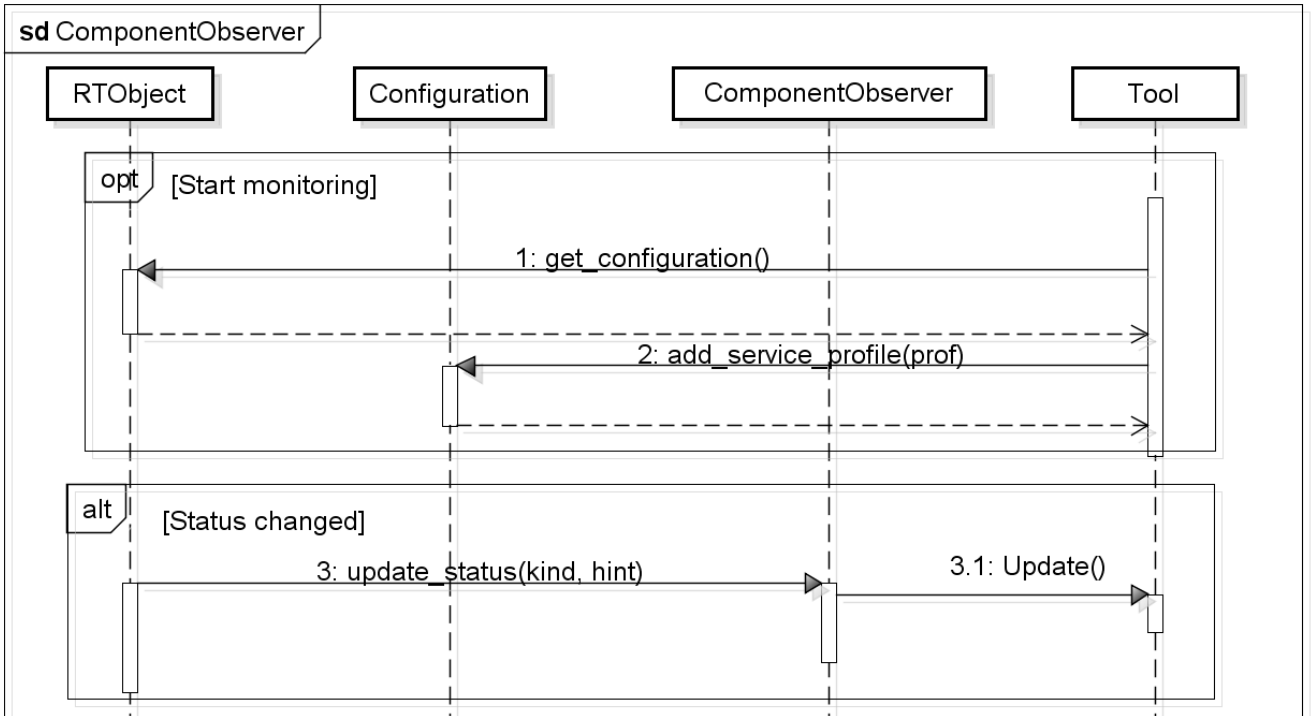
FSM4RTCはSDOServiceを継承したインターフェースによる
コンポーネント拡張の仕組みを利用している

ComponentObserver
クラス図



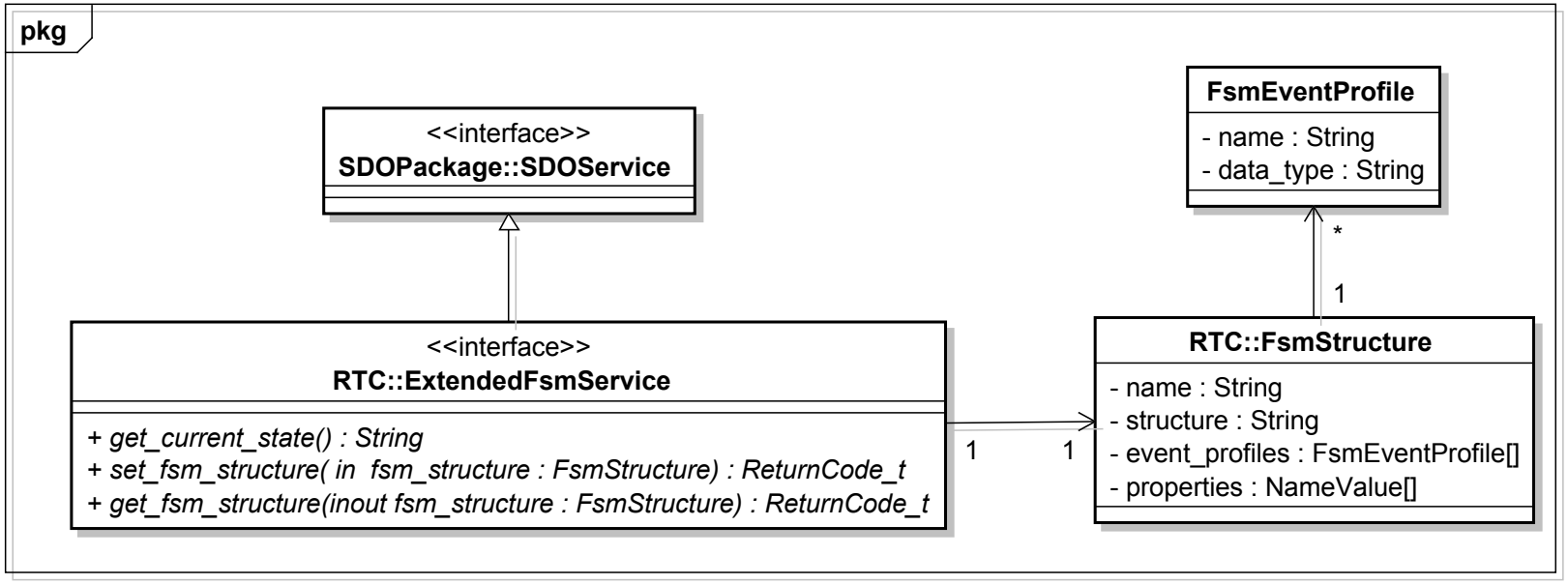
コンポーネントが外部に通知を行うためのupdate_statusメソッドを持つ

ComponentObserver
シーケンス図



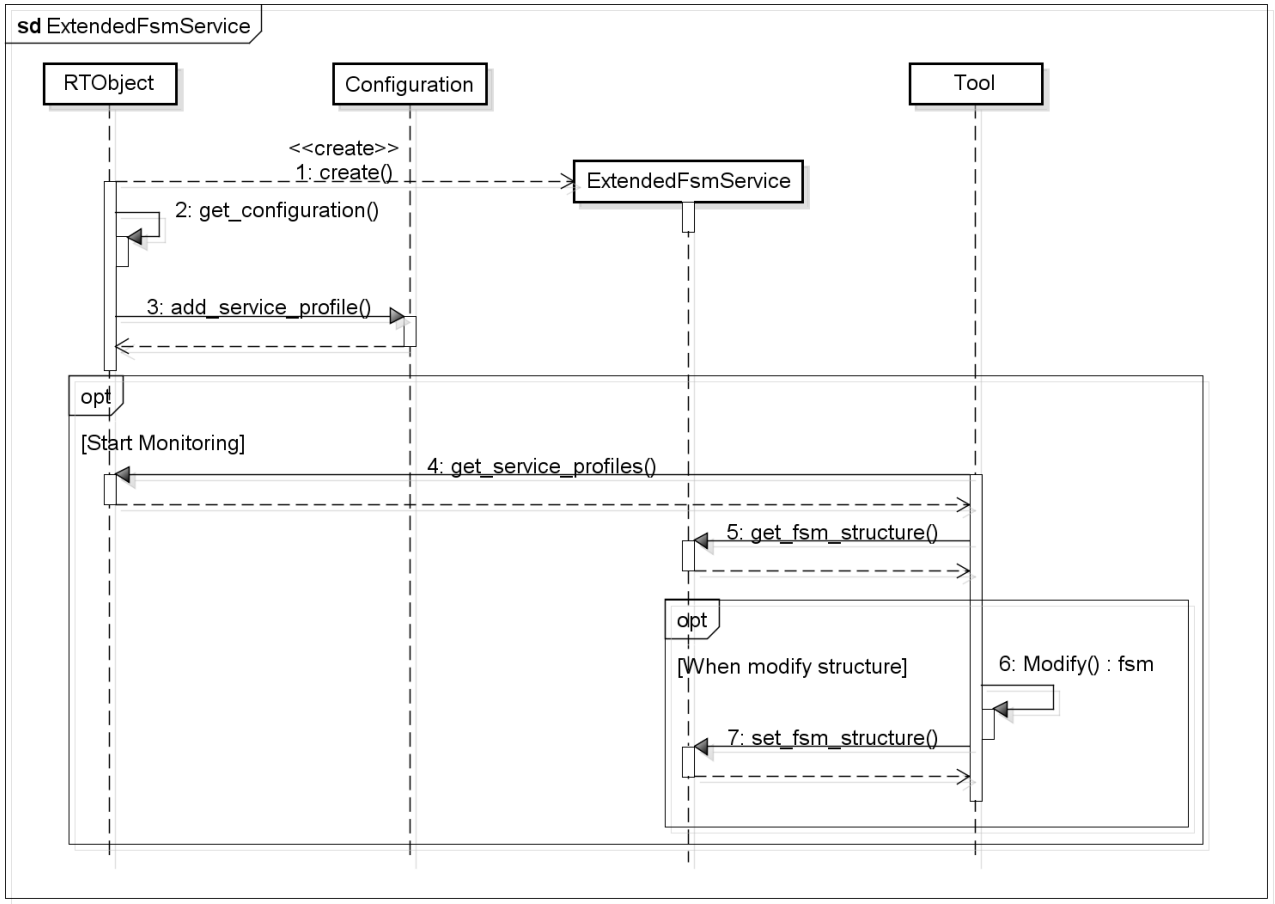
外部ツールなどがComponentObserverをコンポーネントに登録し，状態の通知を待つ

ExtendedFsmService
クラス図



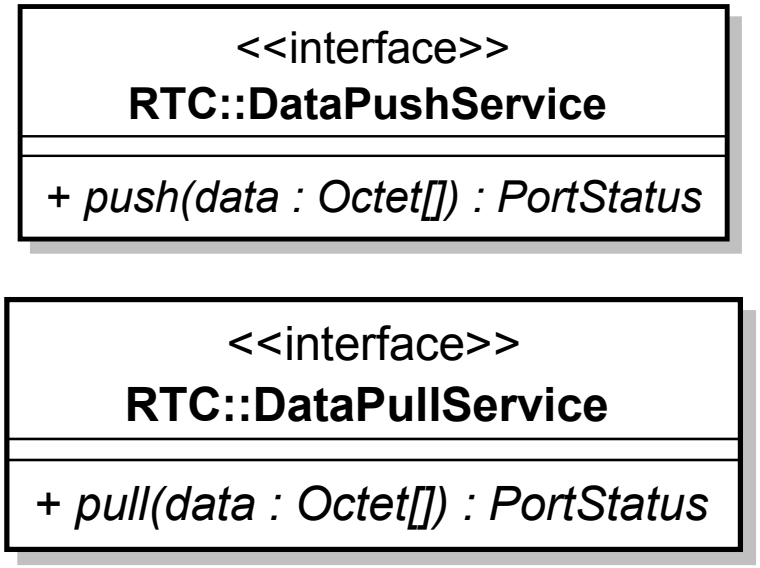
- FSM状態を取得するためのget_current_state
- FSMの内部構造を設定するためのset_fsm_structure
- FSMの内部構造を取得するためのget_fsm_structure

ExtendedFsmService
シーケンス図



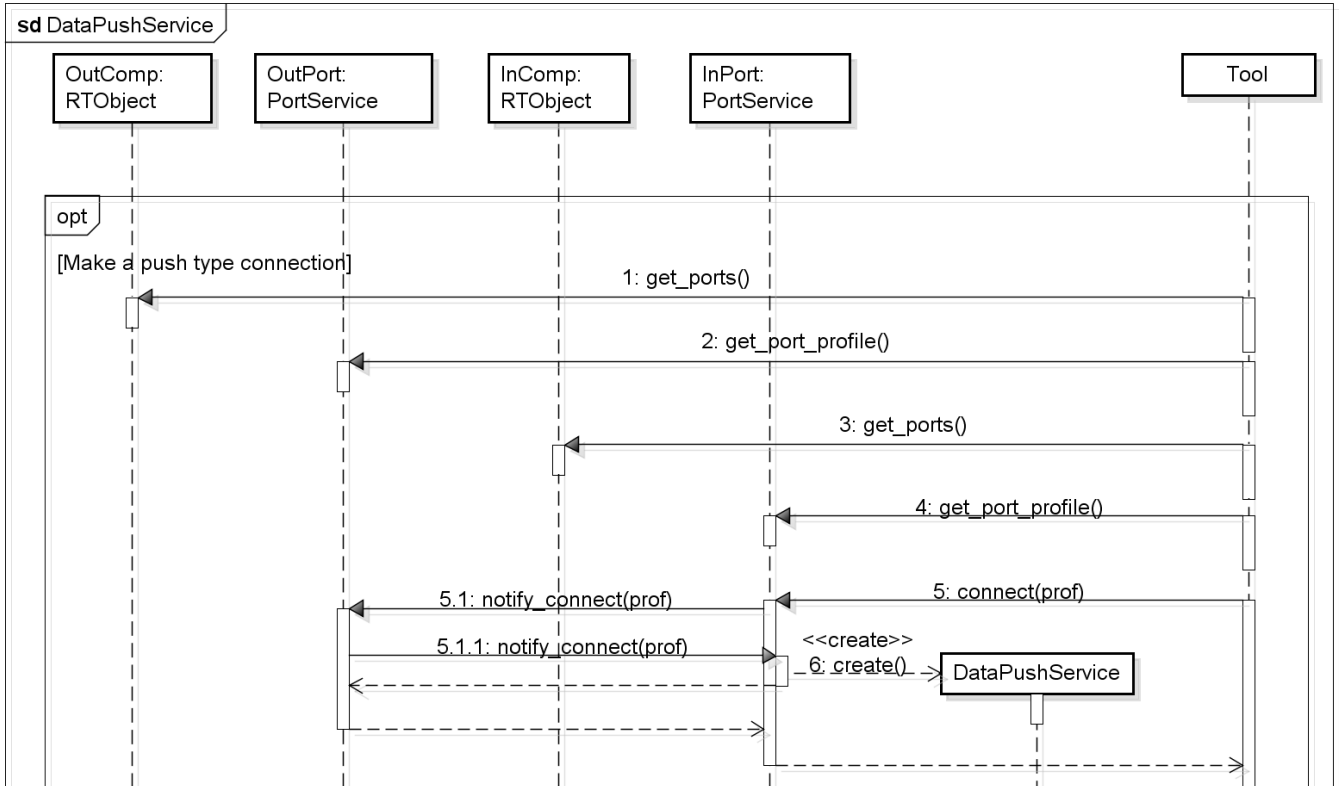
FSMコンポーネントはExtendedFsmServiceを
外部インターフェースとして自ら登録する

DataPort
クラス図



プッシュ型通信のDataPushServiceはpush
プル型通信のDataPullServiceはPullメソッドを持つ

DataPort
シーケンス図



PortServiceはPortProfileに基づいて
DataPushServiceまたはDataPullServiceを生成する