Robotic Component Specification (RTC)

Noriaki Ando, Geoffrey Biggs, Tetsuo Kotoku
National Institute of
Advanced Industrial Science and Technology (AIST)
What is RT?

• RT = Robot Technology  cf. IT
  – not only standalone robots, but also robotic elements (sensors, actuators, etc....)

RT-Middleware developed by AIST

OpenRTM-aist

• RT-Middleware
  – middleware and platform for RT-element integration
• RT-Component
  – basic software unit in RT-Middleware
Outline

• RTC Specification
  – Overview
  – Implementations
• OpenRTM-aist
• RTMSafety
  – Specification VnV
• Summary
Robotics DTF

- Founded in 2004 as a SIG
- Promoted to DTF in 2005
- Current active member
  - AIST, ATR, ETRI, Honda R&D, JARA, Univ. (SIT, UEC, Tsukuba)
- Web: http://robotics.omg.org

- Specifications
  - Robotic Technology Component Specification (RTC), with MARS PTF
  - Robot Localization Specification (RLS)
  - Robot Interaction Specification (RoIS)

- In progress
  - Dynamic Deployment and Configuration for RTC (DDC4RTC), FTF
  - Finite State Machine for RTC (FSM4RTC), reviewing RFP
OMG RTC Specification

- Atlanta meeting (2005.9)
  - RFP (Request For Proposal) : Robot Technology Components (RTCs) issued
- Burlingame meeting (2005.12)
  - Robotics DTF started
- Tampa meeting (2006.2)
  - Response : PIM and PSM for RT-Component submitted
  - Submitter: AIST, RTI
- Sent Louis meeting (2006.4)
  - Proposals were merged and re-submitted
- Boston meeting (2006.6)
  - RTC Spec. was submitted to AB (rejected).
- Anaheim meeting (2006.9)
  - RTC Spec. was submitted to AB (approved).
  - FTF organized
- Jacksonville meeting (2007.9)
  - FTF report was approved
- Officially available **formal-08-04-04** (2008.4)
- Version 1.1 **formal/12-09-01** (2012.9)
Features of RTC

• Provides rich component lifecycle to enforce state coherency among components
• Defines data structures for describing components and other elements
• Supports fundamental design patterns
  – Collaboration of fine-grained components tightly coupled in time (e.g. Simulink)
  – Stimulus response with finite state machines
  – Dynamic composition of components collaborating synchronously or asynchronously
RTC Specification
formal/12-09-01

 Specification consists of 3 packages:
- Lightweight RTC
- Execution Semantics
- Introspection
PIM Overview: Lightweight RTC

- **Lightweight RTC**
  - Stereotypes and constraints for components, ports, and connectors
  - Component lifecycle
  - Baseline support for component execution: EC
  - No reflection or introspection for dynamic system construction
  - Mainly used for static component
PIM Overview: Execution

- Execution Semantics
  - Provides behavioral design patterns commonly used in robotic and control systems
  1. Periodic synchronous execution ("data flow")
  2. Stimulus response/event-driven execution (FSMs)
  3. Multi-modal behavior
PIM Overview: Introspection

- **Introspection**
  - Query and modify component properties and connections at runtime
  - Based on Super-Distributed Objects (SDO)
  - Mainly used for dynamic component system integration
PSM (Platform-Specific Models)

- CORBA IDL
  - CORBA 2.x compliant IDL is provided.
- Lightweight CORBA Component Model
  - Distributed CORBA-based components.
- Local components
  - Low-overhead communication in a single process.
  - C++ mapping is provided.

Other PSM can be defined and implemented if you want
## OMG RTC families

<table>
<thead>
<tr>
<th>Name</th>
<th>Vendor</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenRTM-aist</td>
<td>AIST</td>
<td>C++, Python, Java</td>
</tr>
<tr>
<td>OpenRTM.NET</td>
<td>SEC</td>
<td>.NET(C#, VB, C++,/CLI, F#, etc..)</td>
</tr>
<tr>
<td>RTM for VxWorks</td>
<td>SEC</td>
<td>OpenRTM-aist port to VxWorks</td>
</tr>
<tr>
<td>RTM for TOPPERS</td>
<td>AIST</td>
<td>OpenRTM-aist port to uITRON embedded OS</td>
</tr>
<tr>
<td>RTM for T-Kernel</td>
<td>NEC software</td>
<td>OpenRTM-aist port to T-Kernel embedded OS</td>
</tr>
<tr>
<td>miniRTC, microRTC</td>
<td>SEC</td>
<td>RTC for embedded system with CAN</td>
</tr>
<tr>
<td>RTC CANOpen</td>
<td>SIT, CIA</td>
<td>Based on CANOpen standard and RTC standard</td>
</tr>
<tr>
<td>PALRO</td>
<td>Fujisoft</td>
<td>C++ PSM impl. for small humanoid robot</td>
</tr>
<tr>
<td>OPRoS</td>
<td>ETRI</td>
<td>Korean national project</td>
</tr>
<tr>
<td>GostaiRTC</td>
<td>GOSTAI, THALES</td>
<td>Based on C++ PSM running on URBI</td>
</tr>
<tr>
<td>H-RTM (Tentative name)</td>
<td>Honda R&amp;D</td>
<td>Compatible with OpenRTM, with FSM type RTC</td>
</tr>
</tbody>
</table>

**Implementation diversity by same specification**

- guarantees continuity of products
- makes easy to bridge between implementations
OMG Standards and OpenRTM-aist

PIM

PSM

Extended specification

Impl

OpenRTM-aist's design policy

Component developer never touch IDL (except service port)
Applications

DAQ-Middleware: KEK/J-PARC
KEK: High Energy Accelerator Research Organization
J-PARC: Japan Proton Accelerator Research Complex

HRP-4: Kawada/AIST

TAIZOU: General Robotics Inc.
HIRO: Kawada/GRX
HRP-4C: Kawada/AIST
OpenRTM-aist’s RTC Architecture

- **RT-Component**
  - Service ports (Require, consume)
  - Provider
  - Consumer

- **Activity**
  - Created
  - Inactive
  - Active
  - Error

- **Execution context**
  - Execution

- **Data ports**
  - InPort Buffer
  - OutPort Buffer

- **Interfaces**
  - SDO Interfaces
  - RTC Interfaces

- **Service ports** (Provide)

- **Execution context**
  - Interfaces to obtain metadata, state, configuration etc.

- **OpenRTM-aist’s RTC Architecture**
**Activity, Execution context**

- Common state machine
  - Inactive \(\rightarrow\) Active
  - Error

- Life cycle management, core logic execution

- Composite execution
- Real-time execution

**Data Port**

- Data centric communication
- Continuous data transfer
- Dynamic connection/disconnection

**Service Port**

- User defined interface
- Access to detailed functionality of RTC
  - Getting/setting parameters
  - Changing modes
  - etc...

**Configuration**

- Function for internal parameter
- Multiple parameter sets
- They can be changed from remote in run-time

- RTC can have several configuration sets. Runtime reconfiguration and dynamic switching are supported
Data Port

- Data-centric communication port
  - Type
    • Defined by user in IDL
  - Output: OutPort
  - Input: InPort
- Connection management
  - Interface type
    • CORBA, TCP socket, other protocol, etc...
  - Data flow type
    • push/pull
  - Subscription type
    • Flush, New, Periodic

(a) Push (publisher-subscriber) communication model
(b) Pull communication model
(c) “flush” type subscription
Execution of RTC logic

- Execution context (EC) = entity of logic execution ≅ thread
- RTC⇔EC: Dynamic binding

Execution scheme can be changed by attaching appropriate EC
Ex. External triggered EC

- **External triggered EC**
  - Extended interface
  - Calling tick() operation from outside
  - Logic is progressed by one tick

- **Execution timing is controlled from outside**

- **This feature is convenient for**
  - Simulator
  - Debugger

**Execution timing of all the components in dynamics simulator should be controlled by simulator’s scheduler**

**Same binary component can be used in simulator and actual hardware**
RTMSafety

- IEC61508 (Functional safety standard) capable RTM
- Developed by SEC and AIST
- Based on lightweight RTC specification
- Implemented in C, runs on QNX safety kernel
OpenRTM and RTMSafety

(a) Conventional RT-Middleware

(b) d-RTM

Utilize safety functions provided safety OS
LwRTCs on RTMSafety implicitly/explicitly use these safety functions
Learning from experience

- Execution context gives diversity of execution of component and system structure.
- Port service and combination of data-port/service-port extend the range of application.
- RTC based system shows good conformity to SysML’s IBD.
  - Block = RTC, flow-port = data port
- Specification should be defined from point of view such as functional safety and cording standards
  - RTC has some bugs as specification
  - RTC specs expandability helps applying to various platform and field.