HIJA High-Integrity Java Application

Real-Time Communication with Direct Publish/Subscribe Event Service

Marc Schanne

Workshop Java Technologies for Real-Time and Embedded Systems
17. Oktober 2005, San Diego, CA

High-Integrity Java Applications (HIJA)

- **Development distributed, high-integrity systems**
  - Research project in the 6th framework program of the European Commission
- **Architecturally Neutral** Design of Applications for **Real-Time Systems (ANRTS)** in the domains of
  - **safety-critical** (e.g., avionics, automobile, automation control) and
  - **business-critical** (e.g., telecommunication, multimedia) systems.

Outline

- Networking support
  - Motivation
- Publish/Subscribe event service
  - Relationship to scheduling
- Design pattern
  - ONE pattern for hard and soft real-time
- Application development
  - Declarative programming

Improvements with Distribution

- **Motherhood & Apple Pie** for **Embedded Systems!**
  - **Processing of data** closer to source and sink with modern, smart sensors and actors
  - **Higher dependability** of the complete system with possible local error handling in components
  - Design for **more complex systems** with re-use of well-tested components
  - **Better scalability** with easy addition of components for replicated functionality
  - **Simplified maintainability** with modular design for component replacement

HIJA Networking Support

- Forms of communication
  - **connection-oriented** built on existing end-to-end network connections
  - **message-based** with transmission of events
- Mechanisms in synchronisation
  - **synchronous** processing in remote threads (possible blocking)
  - **asynchronous** interaction w/o blocking

⇒ **Definition of network support in HIJA**

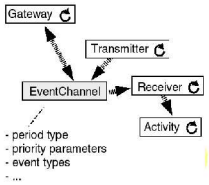
- connection-oriented, synchronous remote method invocation (RMI)
- Asynchronous publish/subscribe event service

Real-Time Specification for Java (RTSJ)

- Basis for HRTJ and FRTJ computational models
  - Minor changes
    - Java and RTSJ programming model with additional definition for architecturally neutral program descriptions
  - Tools support
    - Static & dynamic verification, runtime environment, and libraries
  - Extensions
    - Integration of real-time garbage collection for FRTJ
  - Restrictions
    - Philosophy of Ada Ravenscar profile in HRTJ
    - Separation phases for initialization and mission
    - Pre-emptive, fixed priority scheduling with priority ceiling for periodic or sporadic (i.e., minimum inter-arrival time) threads

Publish/Subscribe Event Service

- Two different implementations, with one API
  - Hard Real-Time Java (HRTJ)
  - Flexible (Soft and Soft/Hard) Real-Time Java (FRTJ)
    - 1st version in a former the research project HIDOORS
  - asynchronous, connection-less message interaction
  - transient push communication
  - topic-based publish/subscribe protocol
  - logical event channel with system's configuration
  - periodic/sporadic events



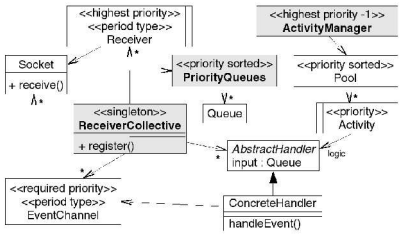
Addressed Networks/Fieldbuses

- Selection ordered by domain
  - Avionics: ARINC629
    - ARINC629 – databus standard for cabling in Boeing 777
  - Automobile: CAN, TT-CAN, TTP/C, FlexRay
    - Event based Controller Area Network (CAN), Time Triggered Protocol (TTP) or mixed implementation with FlexRay protocol
  - Automation Control: ProfiBus, Ethernet/IP
    - Fieldbuses and development for Internet technology
- Used common attribute
  - Communication with broadcast or one-to-many (multicast) interaction

Related Work

- Common Object Request Broker Architecture (CORBA)
  - Specification collection of object/service information model for heterogeneous applications
    - RT-CORBA – real-time extension for ORB
    - CORBA Messaging – asynchronous event communication
    - CORBA Event-Service, Notification-Service – event services with CORBA 3.0
    - Minimum CORBA – adaptation for embedded systems
  - More or less implementations
    - ROFES – 1st work to mix real-time and minimum (C++, CAN)
    - ZEN – CORBA for Java / RTSJ
- Open System Architecture – Platform for Universal Services (OSA+)
  - Hardware-related micro kernel, service-oriented arch.

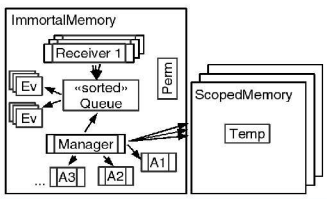
Design Pattern



- Receiver Collective – only receive and sort in queues
- Activity Manager – start handling logic
- Queues – priority sorted

Implementation with HRTJ

- Memory (heap) and Threads
  - Object creation in ImmortalMemory in initialization phase
  - Re-used buffer for events receiving, simple (de-) serialization (low protocol overhead)
  - Non-nested ScopedMemory for handling logic per period



Integration in HIJA Scheduling

- Concurrent threads needs scheduling
- Use for Fixed Priority Scheduling
  - Priority assignment rate-monotonic with length of period or time for next deadline
  - Sporadic and periodic receivers with maximum priority (sporadic server)
  - Periodic execution of activity manager
  - Handling logic with calculated priority (event channel)
  - Definition of round for periodic execution (maximum static buffer size)
- Verification with static feasibility and scheduling analysis for the whole system

Application Development (1/3)

- Static information for verification used for declarative development, definition of
  - System node
  - Network access point

```
<socket name="ttp">
  <code>TTPSocket.instance()</code><buffer>200</buffer>
</socket>
```

  - Event channel

```
<channel id="13" name="work">
  <periodic>200ms</periodic>
  <deadline>100ms</deadline>
  <event id="1" name="status">
    <size>120</size>
  <event>
</channel>
<channel id="42" name="emergency">
  <sporadic>500ms</sporadic>
  <deadline>50ms</deadline>
</channel>
```


Application Development (2/3)

- Publisher with reference to event channels

```
<server name="send">
  <channel ref="work" />
</server>
```

- Subscriber with handling logic and requirements

```
<client name="rec">
  <handler name="control">
    <event ref="system.send.work.status" />
    <code>CtrlAction.instance()</code>
    <wcct>50ms</wcct>
  </handler>
</client>
```

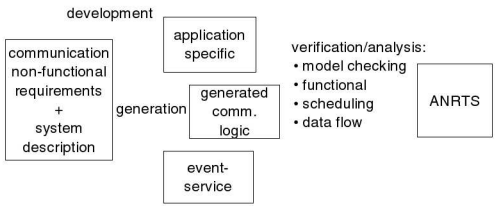
- Reference to system profile

```
<hw-spec-data>url</hw-spec-data>
```

- Program logic and network access point driver are Java classes with fabric methods to integrate in declaration

Application Development (3/3)

- Development process



Current Summary – Next Goals

- Asynchronous communication with real-time
  - Publish/Subscribe protocol
  - Uniform API for HRTJ and FRTJ
  - Static and dynamic implementation
  - Integration in scheduling
  - Code generation from description
  - Peer-to-peer API over publish/subscribe event service
    - soft real-time implementation
  - DDS with publish/subscribe event service
    - even hard real-time?

Thank you! – Any questions?

- HIJA project website: http://www.hija.info

HIJA High-Integrity Java Application

- event channel network: http://www.eventchannelnetwork.org

- FZI Karlsruhe: http://www.fzi.de/ajc/

JAVA Authorized Java Center Sun