Migrating a Representative U.S. Naval Combat Systems Application to Real-Time Java

Tim Childress
Barbara Doyal

Naval Surface Warfare Center
Dahlgren Division
The OA Surface Application Domain

- Large distributed real-time weapon system applications
- Both soft and hard real-time applications
- Timescale for deadlines typically in the O(10) – O(100) millisecond range
- Long system life spans, often for several decades
- Consequences of not meeting requirements for time-deterministic behavior could be severe
Established Surface Domain OA Computing Environment (OACE)

Standards and Middleware Isolate Applications From Technology Change
Java, one of the two programming languages approved for new development in U.S. Navy surface domain Open Architecture applications, offers considerable advantages for software development:

- Designed from ground up to be object-oriented
- Automatic memory management reduces debugging time
- Rich API set encourages programmer productivity
- Java applications are portable
- Large base of Java developers and third-party Java products

But standard Java is not conducive to real-time programming:

- Automatic garbage collection can cause apps to pause at any time
- No mechanism to prevent priority inversion
- No facilities for asynchronous transfer of control, safe asynchronous thread termination, or direct access to physical memory
Java Technologies for Real-Time

- **Standard Java**
  - May suffice for some real-time applications if usage of features that impact determinism is severely restricted

- **Standard Java with Real-Time Garbage Collection**
  - IBM J9 JVM with Metronome garbage collector

- **The Real-Time Specification for Java (RTSJ)**
  - Sun Mackinac, AICAS Jamaica
  - RTSJ and real-time garbage collection are not mutually exclusive

- **“Java-Like” Virtual Machines with Real-Time GC**
  - Aonix PERC
Objectives of this Investigation

- Obtain information on which technologies, and which subsets of features within them, are appropriate for different regions of the real-time application space
  - For example: Is there a class of soft real-time applications for which real-time garbage collection is sufficient, without having to use RTSJ-specific features (even if using an RTSJ JVM)?

- Gain insight on appropriate techniques and design patterns for application of real-time Java technology in the OA domain
  - Languages typically provide many alternative ways to implement solutions, but which are best suited to the problems we encounter?

- Identify possible deficiencies in existing real-time Java technologies for our domain of applications
  - Such knowledge could be applied toward the development of a Mission-Critical Java
Metrics of Interest

- **Timeliness**
  - Degree to which application meets real-time deadlines

- **Jitter**
  - Lower jitter means more consistent real-time behavior and consequently more predictability

- **Resource usage**
  - Memory, CPU utilization, etc., as a function of time

- **Load**
  - Application behavior under heavy processing scenarios

- **Relevant characteristics associated with application development:**
  - Portability
  - Maintainability
  - Scalability
  - Ease of development
Auto-Special in Context

Demo 01 Block Diagram

[Diagram showing various components and connections related to Auto-Special in Context, including SHOOTERS, FIRE SIM XXI, OTH, LA BROKER, Truth Display, MISSION PRIORITIES, Amaranth, MSHN, QUO, TO RESOURCE MANAGEMENT, UAV VIDEO, DISPLAY SERVER, MANUAL ENG CTRL, CALL FOR FIRE, AAW ENG SVR, Engagement Server - TBMD, LA ENG SVR, ENGAGEMENT SERVER - TBMD, Nav Data Svr, UAV, CALL FOR FIRE, GUN SIM, FCS SIM, VLS SIM, AAW DWC, Tracking and Filtering, Tracking Updates, Radar Data, Track Corr, & Filter, ID Request, ID SYS, Remos, AAW/ TBMD, Fault tolerant and/or Scalable, Land Attack, DARPA, Network QoS, Nav Data Svr, TRK NUM SERVER, DECON 3D, DECONFLICT SERVER, DECONFLICT REQUEST, SPIY-DECLARED AUTO SPECIAL, Track Updates, ID SYS, From APPS, Instrumentation, Resource MGT, Control Actions, Missle Flyout, DARPA, Call for Fire, and other related components and processes.]
Aegis Auto Special Timeline Example

1. Initiates a track & sends A/S Detect Notify to CWS.
2. Checks bearing & does range vs. range rate check for interceptability.
3. If track passes, Tent. A/S Engage Order is sent to WCS. An ID Msg is sent to SPY and an IFF Mode 4 interrogation is issued.

1. Receives Second Track Msg and sends a New Track Msg (mt362) to the ID Function which sends a New Firm Track Msg (mt365) to the CWS module.
2. A full interceptability check against the A/S doctrine parameters is performed.
3. If the track passes, a second A/S Confirm Engage Order is sent to WCS.

“Real-Time” implies meeting the auto special reaction time requirement from detection to deployment of missile, following a deterministic critical path mission.
Auto-Special Migration Phases

- **Phase 1: Analyze Existing Application**
  - Analyze existing C/C++ A-S code, using both automated and manual analysis techniques
  - Develop support environment, e.g., track generator

- **Phase 2: Port to Standard Java**
  - Translate existing application design into Java, with only minor refactoring from original design
  - Collect metrics of interest to compare with C/C++ version

- **Phase 3: Refactor**
  - Refactor Java code base to utilize both standard and real-time design patterns
  - Collect metrics for JVMs (both standard and RTSJ) with real-time garbage collection

- **Phase 4: Create RTSJ Version**
  - Modify design to leverage RTSJ-specific features
  - Collect metrics for RTSJ solution
Auto-Special Scenarios

1. “Spy Declared” Auto-Special

2. “C&D-Determined” Auto-Special Type 1
   - Current track received is compared against doctrine

3. “C&D-Determined” Auto-Special Type 2 (Analysis of all track files on a set interval)
   - Tracks in track table are analyzed against the doctrine periodically to check if track is auto-special
Current Status and Plans

- **Current Status**
  - Analysis of existing application complete
  - Design of initial Java implementation complete
  - Skeleton of initial Java implementation is complete; currently filling in application logic
  - Design and development of track generator in progress
  - Examining RT Java technologies for proper insertion into migrated Auto-Special application

- **Plans for Subsequent Work**
  - Summarize and report findings
  - Extend migration to include additional areas of the Aegis-based U.S. Navy OA prototype
  - Evaluate real-time middleware products in Java environment
  - Provide results and experience as input to the OACE document update process