Timed CARDMI의 Real-time Java Virtual Machine 에서의 실현 가능성에 관한 연구

온진호 O, 우수정, 이문근

전북대학교 컴퓨터공학과

jjinghott@gmail.com, wpig04@gmail.com, moonkun@jbnu.ac.kr

A Study on Feasibility for Realization of Timed CARDMI in Real-time Java Virtual Machine

Jinho On O, Sujeong U, Moonkun Lee

Department of Computer Engineering, Chonbuk National University

요 약

Timed CARDMI는 실시간 속성을 만족해야 하는 분산/ 이동/ 실시간 시스템 In The Large 관점의 행위 및 시간속성을 분석하기 위해 정의된 정형기법이다. Timed CARDMI로 정의된 복잡한 시스템의 분석과 검증을 위한 CASE 툴인 SAVE는 RTOS 상에서 CARDMI가 지니는 다양한 행위와 시간속성들에 대한 분석, 명세, 검증, 시뮬레이션을 위한 도구로, 본 논문에서는 RTOS와 Timed CARDMI의 인터페이스 역할을 수행할 Real-time Java의 속성들이 Timed CARDMI의 다양한 시간속성, 행위들에 대한 실행 조건들을 만족하는지에 대한 효용성을 분석한다.

1. 서 론

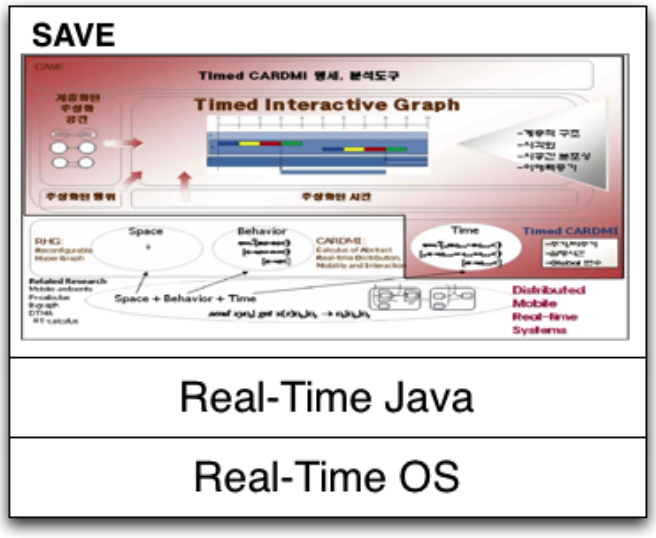
본 논문은 분산, 이동, 실시간 시스템(DMRS: Distributed Mobile Real-time Systems)의 공간과 행위적 특성들을 명세하기 위해 제안된 CARDMI[1]에 시간속성을 추가한 프로세스 대수 기반의 정형기법(Timed CARDMI)[2]의 In The Large(ITL) 관점에서의 다양한 행위 및 시간 속성들의 시뮬레이션 실행과정을 위한 Real-Time Java[3]의 속성들에 대한 분석을 목적으로 한다. RTOS 상의 Real-Time Java의 실행 속성과 Timed CARDMI를 지원하기 위한 CASE 도구인 Specification, Analysis, Verification, and Evaluation (SAVE)[2] 간의 속성 매핑에 대한 분석을 통해 실제 구현에 있어서의 효용성을 분석하였다.

2. Timed CARDMI

CARDMI는 시스템을 행위정보와 공간정보로 구분하여 명세 하는 프로세스 대수 정형기법으로 공간정보와 에이전트들의 행위정보(이동, 통신 등)를 구분하였고, 행위는 이동과 인터액션으로 구분하였다. 진화하는 DMRS에서 에이전트의 순차성, 병렬성, 분산성, 이동성 등에 대한 모델링, 추상화, 동일성 검증 등을 제안하였다.

본 논문은 2007년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2007-521-D00451)

이를 확장한 Timed CARDMI는 tick-time을 기반으로 하며, 시간의 속성을 이동과 통신을 수행하는 에이전트에 맞도록 대기시간, 실행시작 만족시간, 실행시간, 실행완료 만족시간으로 분류하고, 주기/비주기적 시간을 지니는 속성을 가진 정형기법으로, 이러한 세부 시간속성을 통해 다른 정형기법에서는 제공하지 못하는 즉각적인 실행과 실행시간이 있는 실행을 명세할 수 있으며, 시간에 따른 반복적인 행위를 표현할 수 있는 장점을 지닌다.



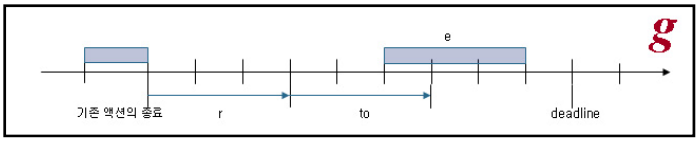
(그림 1) Timed CARDMI의 실행환경

Timed Action Graph[2] 는 시간 흐름기반의

에이전트의 행위를 가시적으로 표현할 수 있으며, 행위에 대한 공간, 시간을 동시에 표현할 수 있는 장점을 지닌다. 또한 계층적 공간에 대한 시간명세 방법을 제공하며, 시간/ 공간/ 행위를 추상화할 수 있도록 정의되었다. SAVE는 공간정보와, TAG, 텍스트 기반 명세를 모두 포함하며, 명세, 수정, 분석, 검증에서의 일관성을 유지할 수 있는 장점을 지닌다. 또한, Timed CARDMI에 최적화된 GUI 명세방법을 제안한다.

이러한 방법은 공간과 시간, 그리고 행위로 구분되어 있었던 명세에 대해 통합된 명세환경을 제공하여 복잡한 DMRS에 대한 분석 및 검증을 용이하게 할 수 있다.

2.1 Timed CARDMI의 시간속성



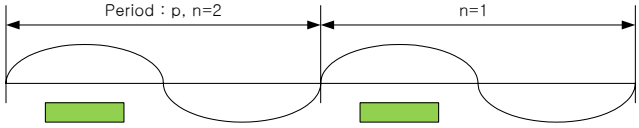
(그림 2) 일반적인 시간의 흐름

[정의 2.1] Timed Action

$$Act_T := {}_{[g]}Act^{p,n}_{[r,to,e,d]}$$

[정의 2.1] 에서 Act_T 에 대한 각각의 설명은 다음과 같다:

- 시간 g 는 글로벌 타이머의 시간을 나타내며, 명세에서는 행위가 어떠한 시점에서 시작되어야 하는지를 명시하며, 실행의 순간에는 시작 이후에 변경되는 타이머의 시간을 나타낸다. g 가 생략되면 행위의 시작 시점이 모든 시간에 대해 오픈 되어 있음을 나타내며, 실행의 순간에 결정되어 짐을 나타낸다.
- r 은 액션이 실행되기 위해 최소한으로 만족되어야 하는 준비/대기 시간을 나타낸다.
- to 는 타임아웃(timeout)의 줄임 표현으로 준비시간 r 이 지난 후 액션이 실행되기 위해 대기할 수 있는 최대 시간을 나타낸다. Tick이 한번씩 흐를 때마다 to 은 0부터 증가하며, 최대 timeout 시간을 초과하면, 액션은 실패한다.
- e 는 행위가 실행되는데 소모되는 시간을 나타낸다. 실행시간이 0으로 표현되면, 행위의 실행이 순간적으로 실행됨을 의미한다.
- d 는 데드라인(deadline)을 나타내며 액션이 실행된 시점부터 d 까지의 시간을 말하며, 타임아웃 안에 시작된 행위가 실행 시간 e 만큼을 소모한 후 d 안에 완료되어야 함을 의미한다.
- p 와 n 은 주기적인 행위를 위한 식별자로 그 행위는 그림 4와 같고, 특성은 다음과 같다. p 는 Act 의 총 실행 주기를 말하며, $p=0$ 이면 생략 가능하다. 또한, n 은 p 의 주기를 n 번 반복함을 나타나며, $n=0$ 이면 생략 가능하다.



(그림 3)

${}_{[g]}Act_{[r,to,e,d]}$ 은 행위가 g 시간에 동작하기 시작하여 r 의 시간을 준비, 대기한 후 to 안에 동작이 시작되어 e 만큼의 실행시간 최소 실행시간 동안 실행을 한 후 d 안에 완료 해야 함을 나타낸다.

2.2 Timed CARDMI의 행위

[정의 2.3] Syntax

(1) Primitive action:

$$E_T := {}_{[g]}x(Z)^{p,n}_{[r,to,e,d]} \bullet E_T \mid {}_{[g]}\bar{x}(M)^{p,n}_{[r,to,e,d]} \bullet E_T \mid E_T \setminus (vx) \mid E_{T_1} + E_{T_2} \mid 0 \mid \varepsilon \mid \mathrm{M} \mid {}_{[g]}[E_T]_{[r,d]} \mid < E_{T_1}, E_{T_2} > \bullet E_{T_3}$$

(2) Composition: $\varepsilon := E_T \mid E_T \mid E_T \mid E_T$

(3) Movement:

$$\mathrm{M} := {}_{[g]}in\ A^{p,n}_{[r,to,e,d]} \bullet E_T \mid {}_{[g]}out\ A^{p,n}_{[r,to,e,d]} \bullet E_T$$

(4) Communication Entity:

Send: $M := m', M \mid m'$

$m' := A \mid x_i \mid y_i \mid m_i$.

Receive: $Z := z', Z \mid z'$

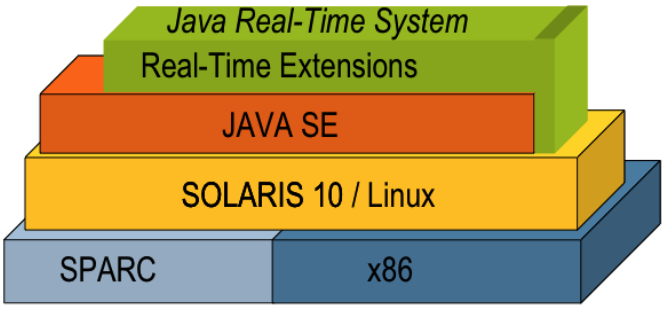
$$z' := z : agent \mid z : msg \mid z : port$$

□

3. Real-time Specification for Java (RTSJ)

Real-time Specification for Java (RTSJ)[4] 는 오픈 스펙으로, 실시간 시스템을 구현하여 자바 언어를 확장시켰다. RTSJ를 구현하기 위해서는 OS, JRE, Java Class Library (JCL)에서 Real-time 조건을 만족해야 한다. 이렇게 정의된 RTSJ는 자바 자체가 지니는 다음과 같은 문제로 인해 보급되지 못했다[5]:

- 스레드관리: 표준 자바는 스레드 스케줄링이나 스레드 우선순위를 보장하지 않는다.
- 클래스 로딩: 자바 순응 JVM은 프로그램에 의해 첫 번째로 참조될 때까지 클래스 로딩을 지연시켜야 한다.
- 가비지 콜렉션: 포인터 안정성, 누수방지, 커스텀 메모리 관리 툴의 필요성 제거 등 애플리케이션에서 GC의 장점은 증명되고 있지만 GC는 RT 시스템에서는 상당한 문제를 발생시킨다.
- 컴파일: 컴파일이 발생할 시기를 예견할 수 없기 때문에 애플리케이션의 액티비티를 효율적으로 설계하는데 어려움이 발생한다.



(그림 4) IBM WebSphere Real Time VM의 시스템 블록도

하지만, IBM의 WebSphere Real Time[6], JamaicaVM[7] 등과 같은 RTSJ를 만족하는 VM의 개발로 Real-time Java가 사용이 증가되고 있다.

4. Timed CARDMI 속성의 Real-time Java의 변환

javax.realtime[8] 패키지에 포함되어 있는 시간 속성에 관련된 클래스는 표 1과 같다.

(표 1) Timed CARDMI에서 사용되는 Real-time Java의 주요 클래스

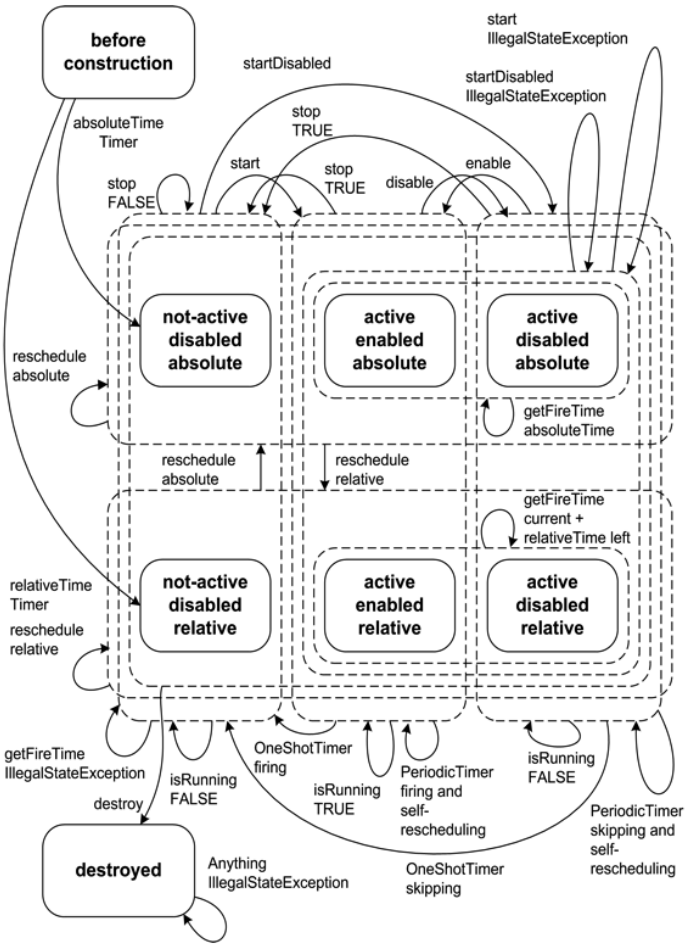
클래스 명	설명
AbsoluteTime	Clock으로부터 생성된 시간으로부터의 특정 절대 point를 ms, ns의 형태로 지정
HighResolutionTime	ms형태의 시간을 ns 형태로 표현하기 위한 클래스
RalativeTime	현재 시간으로부터의 상대시간을 ms 또는 ns의 형태로 지정
Timer	Clock으로부터 주어진 시간으로 timed event를 발생시키기 위한 클래스
OneShotTimer	Timer를 상속 받으며 특정 시간에 이벤트를 한번 fire 시키기 위한 Timer
PeriodicTimer	Timer를 상속 받으며 특정 주기로 시간 간격으로 이벤트를 발생시키기 위한 Timer
AsyncEventHandler	RT 스레드로 동작하며 Timer에 의해 비동기적으로 실행되어 발생된 이벤트를 처리하기 위한 코드를 정의하는 클래스
AperiodicParameters	RT 스레드의 스케줄링을 결정하기 위해 사용되며 한번 실행되는 스레드에 대한 속성을 결정한다. 이를 통해 스레드의 cost, deadline, overrunHandler, missHandler등을 지정할 수 있다.
PeriodicParameters	RT 스레드의 스케줄링을 결정하기 위해 사용되며 주기적으로 실행되는 스레드에 대한 속성을 결정한다. 속성은 AperiodicParameters와 같지만, deadline이 한 주기 시간에 제한된다.

4.1 노드 생성/종료

Timed CARDMI 행위의 가장 기본 단위인 노드는 병행성을 지니며 이동될 수 있다. 이러한 속성을 RT-Java에서 표현하기 위해서는 하나의 노드를 RT 스레드로 구현해야 함을 의미한다. 자바의 RT 스레드는 javax.realtime.RealtimeThread의 인스턴스이다. RTSJ는 우선순위가 자바의 기본 우선순위 단계인 0~10의 값보다 큰 값을 가지며 최소한 28개 이상을 제공해야 한다고 요구[5]하고 있다. 실제로는 구현체[6][7]에 따라 28개 이상의 다양한 단계로 지정된다.

4.2 시간속성

(그림 5)는 Real-time Java에서의 Timer의 상태전이 과정을 설명하며 Timed CARDMI에서의 시간 속성의 가장 기본이 된다.



(그림 5) Timer의 상태전이

Timed CARDMI에서의 각 행위들은 다음과 같은 실행코드에 의해 생성될 수 있으며 각 시간의 특성에 의해 정해진 시간, 또는 정해진 간격으로 실행될 수 있다.

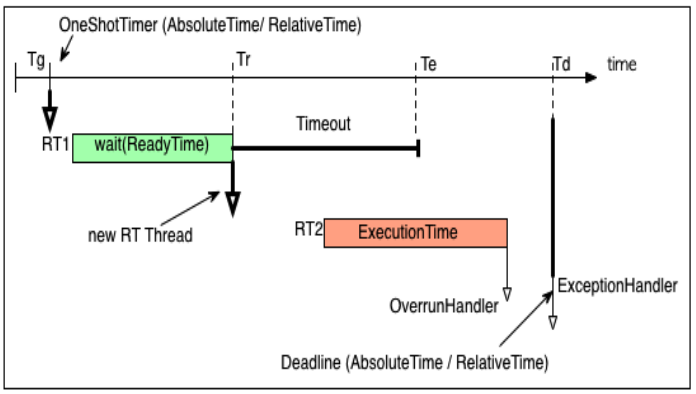
일반적 실행의 생성

```
OneShotTimer(HighResolutionTime time,
    AsyncEventHandler(SchedulingParameters s,
        AperiodicParameters(RelativeTime cost,
                            RelativeTime deadline,
                            overrunHandler,
                            missHandler),
                MemoryParameter memory,
                ProcessingGroupParameter pg,
                Boolean nonheap)
)
```

주기적 실행의 생성

```
PeriodicTimer(HighResolutionTime time,
    AsyncEventHandler(SchedulingParameters s,
        PeriodicParameters(RelativeTime start,
                           RelativeTime period,
                           RelativeTime cost,
                           RelativeTime deadline,
                           overrunHandler,
                           missHandler),
                MemoryParameter memory,
                ProcessingGroupParameter pg,
                Boolean nonheap)
)
```

Timed CARDMI의 [정의 2.1]에 의해 Real-time Java에서는 다음과 같은 GlobalTime, ReadyTime, Timeout, ExecutionTime, Deadline의 시간속성을 만족해야 한다.



(그림 6) Real Time Thread의 CARDMI 시간 속성

4.2.1 Global Time

g 는 Timer의 자식 클래스인 OneShotTimer[9]에 의해서 정의될 수 있다.

g 속성의 실행

```
public OneShotTimer(HighResolutionTime time,
                    AsyncEventHandler handler)
```

절대시간	AbsoluteTime(java.util.Data date)
상대시간	RelativeTime(long millis, int nanos)

특정 행위가 특정시간에 실행되어야 한다면 RT Java에서는 Date클래스를 통해 특정 시간에 대한 인스턴스를 생성한 후 특정 시간에 대한 실행을

보장한다. 또한, 현재 시간으로부터의 상대 시간을 통해 행위를 시작하기 위한 g 는 RelativeTime 클래스를 통해서 실행을 보장받을 수 있다.

4.2.2 Ready Time

g 에 의해 실행된 AsyncEventHandler[10]는 실행 즉시 ReadyTime 의 시간만큼 sleep 상태로 빠지게 된다. ReadyTime이 종료되면 Handler는 RT 스레드를 생성하고 start() 메소드를 실행하게 된다.

$$wait(\mathrm{Re}adytime) = (\mathrm{Tg} + r\mathrm{e}adytime) - RT1 \text{ 시작시간}$$

$$Tr = Tg + wait(\mathrm{Re}adytime)$$

4.2.3 Timeout

ReadTime 후 실행된 RT 스레드가 실제 행위를 하는 스레드로 초기 Handler에 정의된 Timeout 시간 안에 실행이 시작되어야 한다.

$$Tr + timeout$$

4.2.4 Execution Time

ExecutionTime은 분기된 RT 스레드의 코드의 행위가 지니는 실제 실행시간을 의미하며, ReleaseParameters 클래스의 cost 시간에 의해서 결정된다. ExecutionTime이 초과되었을 경우 AsyncEventHadler인 OverrunHandler에서 오류에 대처하거나 추가 행위를 수행할 수 있다. ExecutionTime의 cost 값은 초기 Handler에 의해서 결정된다.

4.2.5 Deadline

Deadline은 분기된 RT 스레드의 모든 실행이 완료되어야 할 시간으로 초기 Handler의 실행 시간인 g 부터의 경과 시간을 의미한다. 일반적 실행의 경우에는 AperiodicParameters에 정의된 상대시간인 deadline 파라메터에 의해 결정되며, 주기적 실행에서는 일반적으로 다음 주기 전까지를 의미하고, PeriodicParameters의 deadline 파라메터에 의해서 지정될 수 있다. 분기된 RT 스레드의 Deadline은 다음 식에 의해 계산된다.

AbsoluteTime =

$$Td - (RT2 \text{ 시작 시간})$$

RelativeTime =

$$(Tg + deadline) - RT2\text{시작 시간}$$

4.3 오류처리

$< E_{T_1}, E_{T_2} > \bullet E_{T_3}$ 의 실행에 있어 $< E_{T_1}, E_{T_2} >$ 의 의미는 T_1 의 시간조건을 만족하지 못하는 경우 즉, Deadline를 넘겨서 실행되는 경우에 대한 오류처리를 E_{T_2} 가 처리하겠다는 것을 의미한다. RT-Java에서는 이와 같은 행위에 대해서 AsyncEventHandler를 할당하여 오류를 처리할 수 있도록 하였다. 비주기적 실행에 대한 오류처리는 다음과 같이 구현된다.

```
AsyncEventHandler exceptionHandler = new
                AsyncEventHander(S.., R.., ..);

new RealtimThread(SParameter,
        AperiodicParameter(cost, deadline,
                overrunHandler,
                exceptionHandler)
        );
```

5. Timed CARDMI 속성에 대한 Real-time Java의 효용성 비교

RTSJ의 다양한 구현체 등이 직접 RTOS를 사용하는 경우보다 더 많은 문제점을 내포하고 있지만, 다양한 알고리즘으로 많은 부분이 극복되어가고 있다. 또한, Java가 지니는 응용프로그램의 확장성 및 편의성이 Real-time 환경에서의 Java의 사용을 증대시키고 있다.

Timed CARDMI SAVE는 툴의 일반적인 실행에 있어서는 일반적인 Java 스레드로 생성되며, 시뮬레이션 환경에서만 RT 스레드로 실행을 하게 된다[6]. 4장에서 살펴 본 Timed CARDMI의 RT Java의 변환을 통해서 Timed CARDMI의 시뮬레이션이 RT Java를 통해서 실행될 때 시간 속성이 어떻게 변환될 수 있는지를 분석했다.

하지만, 제안된 방법으로는 Timed CARDMI의 하나의 노드의 실행을 위해서는 (그림 6)과 같이 2개의 RT 스레드가 필요하다는 문제점이 발생한다.

6. 결론

본 논문에서는 Timed CARDMI가 지니는 시간 속성을

분석하고 Real-time Java의 스펙상의 시간 및 행위 속성을 분석하여 Timed CARDMI를 Real-time Java로 시뮬레이션 할 경우의 시간 속성의 만족 여부를 분석하였다.

CASE 도구의 개발 관점에서 Java가 갖는 다양한 장점들을 RTOS와 결합하여 다양한 분석 기능을 제공하며 정확한 예측을 수행할 수 있는 시뮬레이션 툴을 위한 SAVE의 시간 속성과 Real-time Java 와의 속성 매핑은 현 단계의 SAVE 뿐만 아니라 추후 개발될 다양한 분석 모듈을 위해서도 매우 중요한 분석과정이다.

추후 연구에서는 Real-time Java상에서 Timed CARDMI가 지니는 다양한 속성들에 대한 정밀한 분석이 진행되어야 할 것이다.

[참고문헌]

[1] J. Choi, "A Calculus for Equivalence Analysis and Verification of Distributed Mobile System Based on Abstraction," PhD Dissertation, Chonbuk national Univ, 2007.

[2] 온진호, 최정란, 이문근, “공간 프로세스 대수를 이용한 정형 명세와 분석에서의 시간속성의 시각화,” 정보처리학회논문지, 제 16-D 권, 제 3 호, pp.339-352, 2009.

[3] Eric J. Bruno, Greg Bollella, “Real-Time Java Programming: With Java RTS,” Prentice Hall, 2009.

[4] “JSR 1: Real-time Specification for Java,” http://jcp.org/en/jsr/detail?id=1.

[5] Bollella, G., Canham, T., Carson, V., Champlin, V., Dvorak, D., Giovannoni, B., Indictor, M., Meyer, K., Murray, A., and Reinholtz, K. 2003. “Programming with non-heap memory in the real time specification for Java,” In Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (Anaheim, CA, USA, October 26 - 30, 2003). OOPSLA '03. ACM Press, New York, NY.

[6] “WebSphere Real Time,” http://www-01.ibm.com/software/webservers/realtime/.

[7] “JamaicaVM-Java Technology for Realtime,” http://www.aicas.com/jamaica.html.

[8] “javax.realtime,” http://www.rtsj.org/specjavadoc/javax/realtime/package-summary.html.

[9] “OneShotTimer,” http://www.rtsj.org/specjavadoc/javax/realtime/OneShotTimer.html.

[10] “AsyncEventHandler,” http://www.rtsj.org/specjavadoc/javax/realtime/AsyncEventHandler.html.