UML-RT 모델의 시나리오 기반 다중 쓰레딩의 실험적 평가를 위한 PBX 시스템 사례 연구

김세화°, Michael Buettner, 홍성수, 박선희
서울대학교 전기컴퓨터공학부
{ksaehwa°, buettner, sshong, shpark}@redwood.snu.ac.kr

A Case Study of PBX Systems for Experimental Assessment of Scenario-Based Multithreading of UML-RT Models

Saehwa Kim, Michael Buettner, Seongsoo Hong, Sunheui Park
School of Electrical Engineering and Computer Science, Seoul National University

요 약

본 논문에서는 실시간 객체 지향 모델의 시나리오 기반 다중 쓰레딩에 대한 실험적 평가를 위해 PBX (Private Branch eXchange, 사설 교환기) 시스템에 대한 사례 연구 결과를 제시한다. 사용된 PBX 시스템은 재구성 가능한 동적 구조와 전형적인 계층 구조와 같은 실세계의 응용들에서 발견되는 많은 특징을 보인다. 이 실험적 연구는 (1) 설계자의 사용 편리성과 (2) 결과적인 실행 바이너리의 성능이 얼마나 향상되는지를 평가하는 것을 목적으로 한다. 본 논문에서는 우리의 도구를 통해 시나리오 기반 다중 쓰레딩 실행 바이너리를 생성하는 것과 함께 시나리오를 모델링하기 위해 이를 어떻게 적용하는지를 보인다. 이 사례 연구는 우리의 방법이 규모의 복잡한 모델을 다룰 수 있으며 시나리오 기반 다중 쓰레딩이 실세계 모델에 대하여 성능 향상을 가져오는 것을 명백하게 보여준다.

1. 서론

실시간 객체 지향 모델과 객체 지향 CASE 툴은 실시간 시스템 소프트웨어 설계에 널리 사용되고 있다. 그러나 현재의 객체 지향 모델링을 위한 모델링 도구는 종종 예측 가능하고 검증가능한 타이밍 행동을 제공하지 못하고 있으며 자동적으로 생성된 실행 바이너리는 만족스러울 만하지 못한 경우가 많다. 실시간 내장형 시스템에서는 제한된 자원에서 타이밍 요구 사항을 만족시키는 실행 바이너리를 생성하는 것이 매우 중요하다. 현재 설계자들은 설계 수준에서의 객체를 구현 수준에서의 태스크로 임시 방편적인 방식으로 매핑하고 있다. 태스크의 유도는 실시간 스케줄 가능성에 매우 중요한 영향을 미치기 때문에 이러한 방식으로 시스템을 조율하는 것은 종종 매우 지겹고 시간 소모적인 일이다.

우리는 선행 연구[1, 2, 3]에서 실시간 객체 지향 모델을 다중 쓰레드 구현으로 스케줄 가능성을 고려하여 매핑하는 체계적이고 자동화된 방법을 제시하였다. 이는 시나리오의 개념에 바탕을 두고 있다. 시나리오는 외부 입력 이벤트로부터 야기된 액션들의 연속이다 [1]. 현재 대부분의 모델링 도구에서는 객체들의 집합을 쓰레드에 매핑하는 캡슐 기반 구조를 지원하는, 반면 우리의 방법은 시나리오의 집합을 쓰레드에 매핑한다[2]. [3]에서 우리는 이러한 시나리오 기반 구현 구조를 지원하는 도구 사슬을 구현하고 실험 결과를 제시하였다.

본 논문에서는 실시간 객체 지향 모델에 대한 우리의 시나리오 기반 다중 쓰레딩에 대한 사례 연구를 통한 실험적 평가를 보인다. 이 실험적 연구의 목적은 설계자의 사용 편리성과 결과 실행 바이너리의 성능 향상을 평가하는 것이다. 이 연구를 위하여 대상 내장형 실시간 시스템으로 PBX (Private Branch eXchange, 사설 교환기) 시스템을 선정하였다. 사례 연구에 사용된 시스템은 재구성 가능한 동적 구조와 전형적인 계층 구조와 같은 실세계 응용의 많은 특징들을 포함하고 있다.

본 논문은 우리의 도구가 설계와 구현을 별개로 구별하고, 사용자가 감지 가능한 타이밍 제약 조건과 연관된 시나리오를 모델링 하는 방법을 제공함으로써 다중 쓰레딩에 관련하여 모델링을 어떻게 간단히 만들었는지를 보인다. 또한 우리의 도구를 통해 생성된 실행 바이너리가 어떠한 성능 향상을 가져오는지 실험 결과를 통해 제시한다.

논문의 나머지 부분은 다음과 같이 구성되어 있다. 2절에서는 UML-RT와 우리의 시나리오 기반 다중 쓰레딩에 대하여 개관한다. 3절에서는 본 논문의 사례 연구 대상인 PBX 시스템에 대하여 설명하고, 4절에서는 우리의 도구가 시나리오 기반 수행 바이너리를 생성하기 위하여 어떻게 적용되는 지를 설명한다. 5절에서는 실험 결과를 제시하고 마지막으로 6절에서 결론을 내린다.

2. UML 2.0과 시나리오 기반 다중 쓰레딩

이 절에서는 우리의 방법을 적용시킨 언어로 선택된 UML-RT와 우리의 시나리오 기반 다중 쓰레딩에 대하여 개관한다.

2.1. UML 2.0 모델링 언어

UML-RT(UML Real-Time)[4]는 UML에 이벤트 중심이며 분산 처리의 가능성이 있는 시스템에 필요한 새로운 개념을 추가한 것이다. UML-RT의 가장 기본 요소는 캡슐(capsule)이다. 캡슐은 특정 작업을 수행하는 수행 가능한 개체이다. 캡슐들은 기본적으로 서로 병렬적으로 동시 수행이 가능한 것으로 가정한다. 캡슐은 포트(port)라 불리는 인터페이스 오브젝트를 통해서만 메시지를 주고 받을 수 있다. 캡슐들 간의 포트들의 연결을 통해서 모델의 구조가 규정된다. 모델의 행동 양식은 FSM(finite state machine)로 정의된 각 캡슐에 따라 규정된다.

캡슐은 고정형, 선택형, 플러그-인의 세 가지 타입이 있다. 시스템이 초기화될 때 시스템에 포함된 모든 고정형 캡슐은 인스턴스화 된다. 선택형과 플러그-인 캡슐은 설계자의 필요에 따라 동적으로 인스턴스화 된다. 플러그-인 캡슐은 실제 인스턴스가 아니라 모델에 있는 다른 캡슐 인스턴스에 대한 참조로 고정형이나 생성된 플러그-인 캡슐에 대한 이입(import)을 통해 생성된다. 그림 1의 예에서 빗금이 있는 캡슐은 선택형이고, 속이 찬 캡슐은 플러그 인 캡슐이며 4개의 최상위 캡슐은 고정형이다. 이 예에서 *PhoneProxy* 인스턴스는 *ProxyManager* 캡슐에 의하여 명시적으로 생성되며 후에 *DeviceManager* 캡슐로 이입된다.

2.2. 시나리오 기반 다중 쓰레딩

실시간 객체 지향 모델링에서 행위 모델은 이벤트 처리에 기반을 두고 있다. 다중 쓰레드 구현에서 각 쓰레드는 관련된 메시지를 받게 되고 이를 하나 하나 처리하는 식으로 동작하게 된다. 캡슐 기반 다중 쓰레딩에서는 한 객체에 관련된 모든 이벤트가 하나의 쓰레드에서 처리된다. 반면 시나리오 기반 다중 쓰레딩에서는 한 시나리오에 관련된 모든 이벤트가 하나의 쓰레드에 매핑된다.

캡슐 기반 다중 쓰레딩에서는 하나의 캡슐에 들어오는 여러 메시지들이 다른 쓰레드에서 처리되는 것이 불가능하다. 반면 시나리오 기반 다중 쓰레딩에서는 각각의 메시지가 아니라 완전한 메시지 사슬에 대해 쓰레드로 매핑하고 우선순위를 매기는 것이 가능하다. 이에 따라 캡슐은 어떤 시나리오가 수행되느냐에 따라 순간 순간 다른 쓰레드에서 수행될 수 있다. 이는 설계자가 문제를 개념화하는 데 도움을 줄 뿐만 아니라 모델을 설계하는데 훨씬 더 큰 융통성을 제공한다.

뿐만 아니라 캡슐 기반 다중 쓰레딩은 불필요한 블록킹을 초래함으로써 실시간 시스템의 성능을 저해할 수 있다. 캡슐 기반 구현에서 블록킹의 원인은 [5]에서 언급되었듯이 (1) 두 수준 스케줄링, (2) 쓰레드 간 메시지 발송, 그리고 (3) 완전 수행 조건으로 나열할 수 있다. 첫 번째 블록킹은 쓰레드의 우선순위를 처리하는 메시지의 우선순위에 따라 동적으로 변화시킴으로써 사라지게 할 수 있다. 두 번째 블록킹은 IIP(Immediate Priority Inheritance Protocol [6])를 통해 한번의 블록킹으로 한정지을 수 있다. 그러나 완전 수행 조건에 의한 블록킹은 사라지게 할 수도 한번으로 한정 지을 수도 없다.

반면 시나리오 기반 다중 쓰레딩에서는 두 수준 스케줄링과 쓰레드 간 메시지 메시지 발송에 의한 블록킹이 존재하지 않으며, 완전 수행조건에 의한 블록킹도 단 한번으로 한정 지을 수 있다. 시나리오 기반 다중쓰레딩에서 우선순위 역전은 블록킹을 초래하는 시나리오에 의한 하나의 메시지의 처리 시간으로 한정된다.

3. PBX 시스템: 사례 연구 대상 시스템

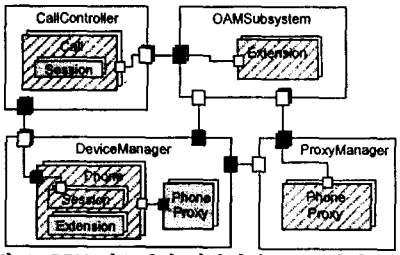


그림 1. PBX 시스템의 간략화된 구조 다이어그램

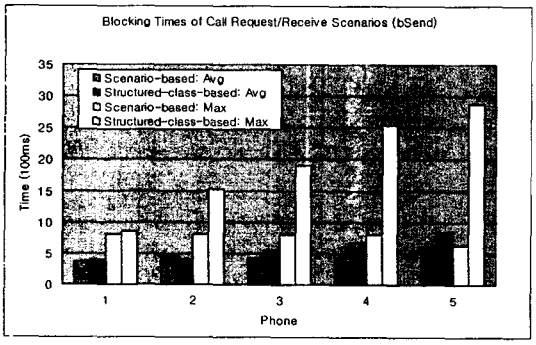
사용된 PBX 시스템 모델은 29개의 캡슐로 구성되며 하부 하드웨어 계층이 외부 입력을 처리하는 전형적인 계층 구조를 따른다. 모델은 그림 1에서 보여지는 바와 같이 상위 구조에서 *ProxyManager*, *DeviceManager*, *OAMSubsystem*, 그리고 *CallController*의 4 개의 캡슐로 이루어져 있다.

*ProxyManager*는 무선 송수화기 프록시의 집합을 관리하며 물리적인 셀 폰 디바이스와 PBX 시스템의 인터페이스 역할을 한다. PBX 시스템이 초기화될 때 시스템에 존재하는 내선 수만큼의 *PhoneProxy*가 생성된다. *PhoneProxy*는 또한 연결된 셀 폰 기기의 고장을 감지하기 위하여 연결 상태를 감시하는 역할도 수행한다. *PhoneProxy*에 대한 입력 시그널은 숫자 버튼들, bSnd/bEnd 버튼, 전원 버튼, 그리고 *ping* 출력 시그널에 대한 응답인 *ack* 가 있다.

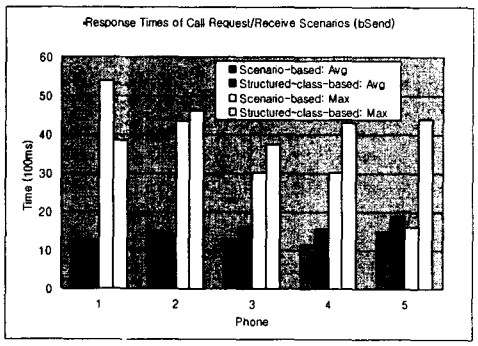
*DeviceManager*는 PBX 시스템 내에서 폰 기기를 추상화하기 위하여 사용되는 *Phone* 캡슐의 인스턴스를 관리한다. 셀 폰 기기가 전원을 켜는 시그널을 보내면 폰 캡슐의 인스턴스가 생성된다. 그러면 *DeviceManager*는 전원을 켜는 시그널을 받은 *PhoneProxy*를 이입한다. 이 *PhoneProxy* 는 새로 생성된 *Phone* 인스턴스와 연결될 것이다. *Phone* 인스턴스는 콜을 시작하던지 혹은 콜을 요청받았을 때 특화된 *Session* 캡슐을 생성하며 *OAMSubsystem*으로부터 *Extension* 인스턴스를 이입한다. 이는 개개의 *Phone* 인스턴스와 *OAMSubsystem*과의 통신 채널이 된다.

*OAMSubsystem*은 폰 번호를 내선 인덱스와 매핑시키는 것을 관리하며 *PhoneProxy*와 같은 개수의 *Extension* 인스턴스를 포함한다. 셀 폰의 전원이 켜지면 새로운 *Phone* 인스턴스는 파워 온 시그널을 받은 *PhoneProxy*에 해당하는 *Extension* 인스턴스를 이입하게 된다. 이런식으로 *OAMSubsystem*은 각 *Phone* 인스턴스와 직접 통신할 수 있다.

*CallController*는 하나의 콜에서 두 개의 폰을 연결하는



(ㄱ)



(ㄴ)

그림 2. 콜 요청/응답 시나리오의 (ㄱ) 블록킹 시간과 (ㄴ) 응답 시간

역할을 한다. 하나의 콜에 대하여 하나의 *Call* 인스턴스가 생성되며 전화를 건 쪽의 *Session*이 이입된다. 그런 다음 *Call* 인스턴스는 전화를 건 셀 폰 번호가 존재하는지 검사를 하고 해당 폰이 통화 중이 아니라면 해당 폰의 *Session* 인스턴스를 이입하고 통화가 연결되게 된다.

4. 시나리오 기반 도구의 적용

사례 연구를 위하여 [3]에서 제시된 RoseRT [4] 모델링 도구에 기반한 우리의 시나리오 기반 도구를 적용하였다. 구체적으로 (1) RoseRT IDE를 이용하여 PBX 시스템 모델을 입력 시그널을 생성하는 우리의 테스트 기기 모델과 통합시키고, (2) 우리의 분석기 도구를 통해 RoseRT를 통해 생성된 코드로부터 시나리오 모델을 자동적으로 생성해 내고, (3) 우리의 코드 변환기를 통해 단일 쓰레딩 소스 코드를 시나리오 기반 다중 쓰레딩 코드로 변환하고, (4) 이를 RoseRT 런타임 시스템에 기반하여 구성된 우리의 시나리오 기반 런타임 시스템과 같이 컴파일하여 실행 바이너리를 얻는다. 보다 자세한 내용은 [3]의 논문을 참조한다.

5. 실험 성능 결과

실험에서는 폰의 개수를 5개에서 100개로 변화시키면서 각 시나리오에 대한 블록킹 시간과 응답 시간을 측정하였다. 응답 시간은 시발 외부 메시지가 인큐될 때부터 수행 사슬의 마지막 메시지가 처리될 때까지의 시간이다. 블록킹 시간은 시나리오가 낮거나 같은 우선순위의 태스크가 수행될 때 기다려야 하는 시간이다. 본 논문에서는 콜 요청/응답 (*bSnd*) 시나리오에 대한 결과만 제시한다. 이는 다른 시나리오들이 유사한 결과를 보여주기 때문이다. 블록킹 시간과 응답 시간 결과는 폰의 개수가 변화하더라도 비슷한 결과를 보여주기 때문에 다섯 개의 폰에 대한 결과를 제시한다. 이와 함께 규모의 시스템에 대한 지원성을 살펴보기 위하여 폰의 개수를 변화시켜 얻은 결과를 제시한다.

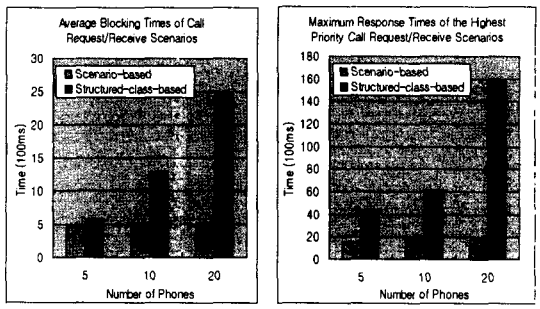
5.1. 블록킹 시간과 응답 시간

그림 2는 콜 요청/응답 시나리오에 대한 평균과 최대 블록킹 시간((ㄱ))과 응답 시간((ㄴ))을 보여준다. 그림에서 보이듯이 시나리오 기반 구현에서의 최대 블록킹 시간과 응답 시간은 캡슐 기반 구현에 비하여 현저히 작으며 평균 시간에 대하여서도 대체로 작은 것을 볼 수 있다. 캡슐 기반 구현에서의 블록킹 시간과 응답 시간이 우선순위가 높아짐에 따라 증가하는데 이는 높은 우선순위의 태스크에 대하여 보다 많은 수의 낮은 우선순위의 태스크가 존재하기 때문이다. 캡슐 기반 접근법에서는 메시지를 FIFO(First-In-First-Out) 방식으로 처리하기 때문에 더 많은 수의 낮은 우선순위의 태스크가 더 큰 블록킹 시간을 초래한다. 반면 시나리오 기반 다중 쓰레딩에서는 높은 우선순위의 태스크가 항상 낮은 우선순위의 태스크보다 먼저 수행되기 때문에 블록킹 시간이 우선순위가 높아짐에 따라 크게 증가하지 못한다.

그림 2 (ㄴ)은 캡슐 기반 버전의 최대 응답 시간이 모든 우선순위에 대하여 상당히 일정한 것을 보여준다. 반면 시나리오 기반 구현에서는 최대 응답 시간이 우선순위가 증가함에 따라 일관적으로 감소하는 것을 볼 수 있다. 결과적으로 시나리오 기반 구현이 캡슐 기반 구현에 비하여 응답 시간이 거의 항상 작으며 이러한 현상은 높은 우선순위의 태스크에 대하여 크게 나타나는 것을 보여준다.

5.2. 규모 지원성

두 다중 쓰레딩 방법의 규모 지원성을 비교하기 위하여, 폰의 개수를 변화시켜가면서 *bSnd* 시나리오의 평균 블록킹 시간(그림 3 (ㄱ))과 최대 응답 시간 (그림 3 (ㄴ)) 결과를 살펴본다. 최대 블록킹 시간이나 평균 응답 시간과 같은 결과는 5.1절과 5.2절의 결과로부터 예측될 수 있기 때문에 생략한다. 그림 3에서 보여지듯이, 시나리오 기반 다중 쓰레딩에서는 폰의 개수가 변화하면서 결과 시간들이 거의 상수로 일정한 것을 볼 수 있다. 이는 시나리오 기반 다중 쓰레딩이 캡슐 기반 방법에 비하여 훨씬 더 규모의 시스템을 잘 지원한다는 것을 명백하게 보여준다.



(ㄱ) (ㄴ)

그림 3. 최고 우선순위 시나리오의
(ㄱ) 평균 블록킹 시간과 (ㄴ) 최대 응답 시간.

6. 결론

본 논문에서는 UML-RT 모델에 대한 우리의 시나리오 기반 다중 쓰레딩을 실험적으로 평가하기 위한 사례 연구를 제시하였다. 이를 위해 우리는 실세계 예제로서 PBX 시스템을 사용하였다. 사례 연구는 우리의 방법이 대규모의 복잡한 모델을 다룰 수 있으며 시나리오 기반 다중 쓰레딩이 실세계 모델에 대하여 성능 향상을 가져오는 것을 명백하게 보여주었다. 또한 모델링 환경에 있어서도 설계자의 사용 편리성이 향상되는 것도 보여주었다. 이는 원래의 모델을 수정하지 않고 원하는 행동을 보이는 실행 바이너리를 빠르게 자동적으로 생성함으로써 가능했다. 성능 실험 결과는 응답 시간과 블록킹 시간의 향상을 분명하게 보여주었다. 또한 이러한 성능 향상이 많은 수의 쓰레드를 가진 규모의 시스템에서 더 크게 나타남을 보였다.

향후 연구에서는 분산 시스템을 포함한 실세계 응용에 대한 연구를 지속시킬 것이다. 또한 QoS 지원 응용에 대한 연구도 고려하고 있다.

참고 문헌

1. S. Kim, S. Cho, and S. Hong. Schedulability-aware mapping of real-time object-oriented models to multithreaded implementations, In Proceedings of International Conference on Real-Time Computing Systems and Applications, pp. 7-14, 2000.
2. S. Kim, S. Hong, and N. Chang. Scenario-based implementation architecture for real-time object-oriented models, In Proceedings of IEEE International Workshop on Object-oriented Real-time Dependable Systems, 2002.
3. J. Masse, S. Kim, and S. Hong. Tool set implementation for scenario-based multithreading of UML-RT models and experimental validation. In Proceedings of IEEE Real-Time/Embedded Technology and Applications Symposium. pp. 70-77, 2003.
4. IBM Rational Software Corporation. Rational Rose RealTime User Guide: Revision 2001.03.00, 2000.
5. M. Saksena, P. Freeman, and P. Rodziewicz. Guidelines for automated implementation of executable object oriented models for real-time embedded control systems, In Proceedings of IEEE Real-Time Systems Symposium, pp. 240-251, Dec. 1997.
6. Institute for Electrical and Electronic Engineers. IEEE Std. 1003.1c-1995 POSIX Part 1: System Application Program Interface-Amendment 2: Threads Extension, 1995.