자바 프로그램의 이벤트 모니터링*

최윤정° 창병모
숙명여자대학교 컴퓨터과학과
{swany°, chang}@sookmyung.ac.kr

Monitoring Events for Java Programs

Yoon-jeong Choi° Byeong-Mo Chang
Dept. of Computer Science, Sookmyung Women's Univ.

요 약

현재 많이 사용되는 대부분의 J2ME 프로그램은 모바일 기기의 특성상 제한된 자원을 사용하며 입력 키 등에서 발생하는 이벤트(event)들을 처리하는 이벤트-구동 프로그램이다. 이벤트-구동 프로그램에서 이벤트의 효과적인 처리는 전체 프로그램의 안전성과 신뢰성뿐만 아니라 효율에 영향을 미칠 수 있으며 보통 디버깅이 어려운 특성을 가지고 있다. 본 연구에서는 실행 중에 실시간으로 이벤트 발생 및 처리 정보를 보여줄 수 있는 모니터링 시스템을 설계 개발하였다. 이 시스템은 사용자 옵션에 따라 사용자가 관심 있는 이벤트만을 실행 중에 추적할 수 있으며 실행 후에 이벤트 관련 요약 프로파일 정보를 제공한다. 또한, 이 시스템은 코드 인라인 기법을 이용하여 실행시간 부담을 크게 줄였다.

1. 서 론

최근에 이동전화, PDA, 핸드폰 등과 같은 모바일 기기 사용이 보편화되면서 이를 위한 Java 언어인 J2ME(Java 2 Micro Edition)가 널리 사용되고 있다. 대부분의 J2ME 프로그램은 모바일 기기의 특성상 제한된 자원을 사용하며 입력 키 등에서 발생하는 이벤트(event)들을 처리하는 이벤트-구동 프로그램이다. 이벤트-구동 프로그램에서 이벤트의 효과적인 처리가 전체 프로그램의 안전성과 신뢰성뿐만 아니라 효율에 영향을 미칠 수 있으며 보통 디버깅이 어려운 특성을 가지고 있다[1].

현재 JVM(Java Virtual Machine)이 제공하는 JVMPI(Java Virtual Machine Profiler Interface)는 메쏘드 호출이나 객체 할당 같은 실행 중 발생하는 사건들을 실시간으로 트레이스, 즉 추적하는 기능을 제공한다[2]. 그러나 이 시스템은 방대한 양의 트레이스로 인하여 실행시간의 지연이 너무 크며 라이브러리들을 포함하여 모든 코드가 트레이스 되므로 프로그램 내의 사용자가 관심 있는 부분만을 트레이스 하기 힘들다. 또한, JVMPI는 쓰레기 수거, 객체 할당, 예외 발생, 메쏘드 호출 등에 대한 트레이스를 제공하지만, 이벤트 처리에 대한 트레이스 정보를 전혀 제공하지 않고 있다[2,3,4]. 따라서 현재까지 프로그램 개발자는 이벤트 관련해서 효과적인 디버깅이 매우 어려우며 이는 프로그램 신뢰성 향상에 저해 요인이 될 수 있다.

본 연구에서는 이러한 문제점을 해결하기 위해 실행 중에 실시간으로 java.awt.event에 정의된 이벤트들의 발생 및 처리 정보를 보여줄 수 있는 *동적 이벤트 모니터링 시스템*을 설계 개발한다. 이 시스템은 사용자 옵션에 따라 사용자가 관심 있는 이벤트만을 실행 중에 추적해 볼 수 있으며 실행 후에 이벤트 관련 요약 프로파일 정보를 제공한다. 이 시스템에서 제공하는 옵션의 종류로는 이벤트 종류, 이벤트 발생 가능한 컴포넌트, 프로파일 생성 여부 등이 있다. 사용자가 옵션을 선택하면 해당 이벤트의 발생 및 처리 과정을 실시간으로 트레이스, 즉 추적할 수 있다. 또한 실행 후에는 실행 중 발생한 이벤트에 대한 요약 정보인 이벤트 프로파일을 생성한다. 이 시스템을 이용하여 사용자는 관심 있는 이벤트를 중심으로 이벤트의 발생 및 처리 과정을 살펴봄으로써 효과적인 디버깅을 도울 수 있을 것이며 결과적으로 프로그램의 신뢰성 및 효율성 향상에 기여할 수 있을 것이다.

본 연구에서는 또한 동적 분석에 의한 실행 시간 부담을 줄이기 위해, 실행 시간에 부담을 주는 JVMPI를 사용하지 않고 코드 인라인 기법을 기반으로 설계하였다[5]. 이 시스템은 사용자 옵션에 따라 관심 있는 정보만을 출력하도록 코드를 삽입하여 입력 프로그램을 변환한다. 이 변환된 프로그램은 원래 프로그램과 동일하게 동작하면서 실행 중에 트레이스 정보를 출력하고 실행 후에는 이벤트 프로파일을 출력한다.

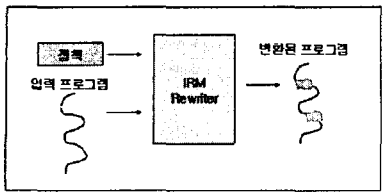
본 논문의 구성은 다음과 같다. 2장에서는 시스템의 설계에 대해 설명 하고 3장에서는 구현 4장에선 실험 결과를 제시한다. 5장에서는 결론 및 향후 연구 과제에 대한 소개를 한다.

2. 설 계

본 연구에서 개발할 시스템은 사용자가 동적 이벤트 모니터링을 위한 옵션을 선택할 수 있으며 이를 바탕으로 사용자는 관심 있는 이벤트만의 발생 및 처리 과정을 실시간으로 추적할 수 있도록 설계하였다. 또한 실행 후에 실행 중 발생 혹은 처리된 이벤트들에 대한 요약 정보인 이벤트 프로파일을 제공하여 사용자가 이벤트 처리 과정을 요약해서 살펴볼 수 있도록 하였다.

본 시스템은 실행 시간 부담을 줄이기 위해 코드 인라인 방법을 기반으로 설계하였다[5]. 코드 인라인 기법이란 실행될 프로그램 내에 모니터링 기능을 하는 레퍼런스 모니터(RM)를 위한 코드를 삽입하고 변환된 프로그램을 실행하면 삽입된 코드에 의해 실행과정을 모니터링 하는 것이다. 이 방법은 [그림 1]처럼 IRM Rewriter가 원래 입력 프로그램과 모니터링 하고자 하는 성질 혹은 정책을 입력으로 받아 원래 프로그램에 모니터링 할 수 있는 코드가 삽입된 프로그램을 생성한다.

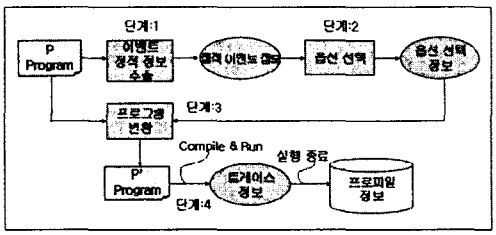
* 본 연구는 한국과학재단 특정기초과제 "정적 분석을 이용한 모바일 자바 프로그램의 효율적인 자원 사용을 위한 환경 연구" (R01-2002-000-003630-0)의 지원에 의해 수행되었음.



[그림 1] 인라인 레퍼런스 모니터

코드 인라인 방법은 각 응용에 따라서 필요한 부분만을 효과적으로 모니터링 할 수 있는 방법으로 그 효율성으로 인해 최근 들어 주목받고 있다. 본 연구에서는 이 기법을 기반으로 하여 사용자가 좀 더 효과적으로 실행 중 발생하는 이벤트를 트레이스 할 수 있는 시스템을 개발한다.

시스템의 전체적인 구조는 [그림 2]과 같다. 사용자 옵션에 따라 해당 트레이스 정보만을 출력하도록 입력 프로그램 P에 코드를 삽입하여 변환된 프로그램 P'를 생성한다. 이 변환된 프로그램은 원래 프로그램과 동일하게 동작하면서 실행 중에 트레이스 정보를 출력하고 실행 후에 이벤트 요약 프로파일을 출력한다.



[그림 2] 시스템 구조

본 시스템은 크게 네 단계로 구성되어 있다.

○ 첫 번째 단계는 입력 프로그램을 간단하게 정적 분석하여 이벤트 관련 프로그램 구조를 추출하며 이 정보를 이용하여 이벤트 관련 옵션 정보를 제공한다.

○ 두 번째 단계는 시스템의 전 단계에서 정적 분석을 통해 얻은 정보를 통해 옵션 선택 기능을 제공하면 사용자는 관심 있는 이벤트 혹은 컴포넌트를 선택할 수 있다.

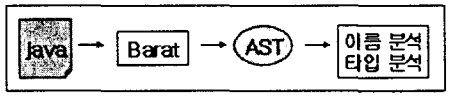
○ 세 번째 단계는 사용자가 선택한 옵션을 바탕으로 입력 프로그램 P에 기존의 프로그램과 똑같이 동작하면서 동적 이벤트 정보를 출력할 수 있도록 추가적인 코드를 삽입하여 새로운 프로그램 P'으로 변환시킨다.

○ 네 번째 단계는 프로그램 실행 중의 이벤트 발생과 처리에 대한 트레이스 정보를 제공하는 단계로 변환된 프로그램을 실행하면 프로그램의 실행 중에 사용자가 원하는 이벤트 정보를 실시간으로 얻을 수 있다. 또한, 프로그램 실행 후 이벤트 실행에 대한 요약 정보인 이벤트 프로파일을 제공한다.

3. 구현

본 연구에서는 이벤트 정보 추출과 소스코드 변환을 위해 자바 컴파일러 전단부인 Barat을 이용하였다. Barat은 Java 프로그램의 정적 분석기 구현을 지원하는 컴파일러 전단부로 자바 소스 파일이나 클래스 파일을 입력 받아 이름과 타입 정보를 포함하는 AST (Abstract Syntax Tree)를 구성한다[6].

이렇게 구성된 AST의 각 노드를 비지터(Visitor)라는 디자인 패턴을 이용한 트리 횡단 루틴을 이용하여 방문하면서 필요한 연산을 수행할 수 있다. Barat이 기본적으로 제공하는 비지터에는 AST의 모든 노드들을 깊이 우선 탐색을 해주는 DescendingVisitor, 모든 노드를 검색한 후 AST정보를 기반으로 소스 코드를 다시 생성해 주는 OutputVisitor 등이 있다. 따라서 이러한 비지터를 확장하여 AST 노드 방문 시 원하는 연산을 수행하는 비지터를 구현할 수 있다[6]. 따라서 이러한 비지터들의 메쏘드를 재정의함으로써 AST 노드를 방문 시 원하는 연산을 수행하도록 재구성할 수 있다.



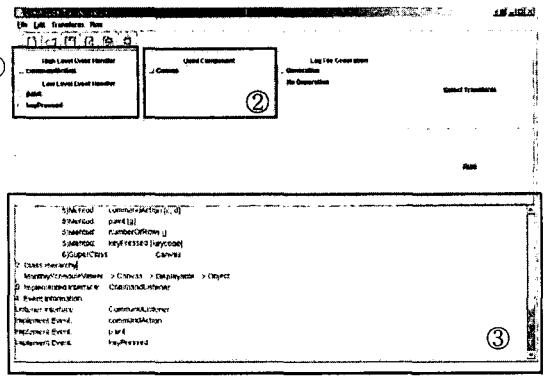
[그림 3] Barat의 구조

본 논문에서는 Barat에서 제공하는 DescendingVisitor와 OutputVisitor의 확장하여 이벤트 정보 추출과 소스코드 변환을 구현하였다. [그림 2]의 이벤트 정보 추출 단계는 노드들을 깊이 우선 탐색을 해주는 DescendingVisitor를 확장하여 구현하였다. 이 분석기는 사용자의 옵션 선택을 위해 입력 프로그램을 간단하게 정적 분석하여 프로그램 내의 발생 가능한 이벤트 종류와 생성된 컴포넌트 등의 정적 이벤트 정보를 추출한다. 특히 구현된 이벤트 리스너와 이벤트 처리 메쏘드들에 대한 정보도 추출한다.

선택 옵션은 추출된 정적 이벤트 정보를 기반으로, 프로그램 실행 시 발생 가능한 이벤트의 목록을 보여주고, 사용자가 이들 중 관심 있는 이벤트만을 선택 가능하게 한다. 또한 현재 프로그램 내에서 만들어진 컴포넌트들의 목록 또한 선택 가능하게 한다. 사용자는 자신이 원하는 이벤트의 종류와 컴포넌트의 종류를 선택하면 관련 정보만을 트레이스 하는 프로그램 P'을 생성한다.

이렇게 정적 분석된 이벤트 정보를 이용하여 만들어진 옵션 정보를 이용하여 [그림 4]와 같은 화면을 구성한다. 화면을 구성하는 예제 프로그램은 J2ME 프로그램으로 달력을 제공하여 약속 정보를 저장하고 정보를 디스플레이 하는 MonthlyScheduleViewer 프로그램이다. 데이터베이스와 연결된 프로그램으로 이벤트에 관련 트레이스 정보가 프로그램을 디버깅하는 과정에서 필요한 예이다.

[그림 4]는 사용자가 원하는 옵션을 선택할 수 있는 화면으로써, MonthlyScheduleViewer 프로그램의 기본적인 정보가 [그림 4]의 ③에 보여 지며, ①부분에 해당하는 것은 상위 레벨 이벤트와, 하위 레벨 이벤트로 구분되어 현재 프로그램에 구현되어 있는 이벤트를 옵션 정보로 제공한다. 또한 현재 사용되어지는 컴포넌트 정보 역시 옵션으로 ②부분에 제공된다. 사용자는 관심 있는 이벤트나 컴포넌트만을 선택하여 실행 시 원하는 정보를 제공 받을 수 있다.



[그림 4] 옵션 선택 화면

5. 실험 결과

본 연구에서는 사용되는 실험 환경은 Windows2000 Professional과 Wireless Toolkit 2.0 버전, SDK 1.4.1 버전이다[3]. 본 연구에서는 총 7개의 프로그램을 이용하여 실험하였다. 3개의 MIDlet 프로그램과 2개의 J2SE 프로그램을 이용하여 실험하였다.

3개의 MIDlet 프로그램은 다음과 같다. 첫 번째는 위의 MonthlyScheduleViewer 프로그램이다. 두 번째, 세 번째는 J2ME의 벤치마크 프로그램으로 두 번째는 RMStress 프로그램으로 RMS를 테스트하는 프로그램으로 create, open, write, read, set, delete하는 데 걸리는 시간을 측정한다[7]. 세 번째는 keycodes 프로그램으로 각각의 key들의 코드 값을 구하는 프로그램이다.

본 연구에서는 2개의 J2SE 프로그램을 실험하였다. 첫 번째는 EventsLog 애플릿 프로그램이다. 이벤트-구동 프로그램의 대표적인 애플릿 프로그램으로 EventsLog 프로그램은 각 이벤트가 발생하였을 때 이벤트의 이름을 기록하는 프로그램이다. 두 번째 프로그램은 GEditor 애플리케이션 프로그램이다. 이 프로그램은 버튼, 레벨, 텍스트 필드, 텍스트 영역을 사용자가 버튼을 클릭 하여 쉽게 그릴 수 있는 에디터 프로그램이다.

각 프로그램을 변환하고 실행했을 때의 실험 결과를 [표 1], [표 2]에 정리하였다. [표 1]은 코드 변환 전의 줄 수와 변환 후의 줄 수를 비교해 놓았다.

본 연구에서는 사용한 프로그램들은 대부분 사용자가 종료시킬 때까지 수행되는 이벤트 구동 프로그램들이다. 따라서 프로그램의 전체 실행 시간은 측정할 수 없다. 따라서 본 실험에서는 원래 프로그램과 변환된 프로그램의 실행 시간을 조사하기 위하여 몇 개의 이벤트들에 대해서 이벤트 실행 시간 즉 발생부터 처리까지 걸리는 시간을 측정하였다. [표 2]는 변환 전과 변환 후의 이벤트 실행 시간을 비교해 서 보여주고 있다. 이 실험을 통해서 변환된 프로그램의 이벤트 처리 시간이 현실적으로 사용할 수 있는 것을 확인할 수 있었다.

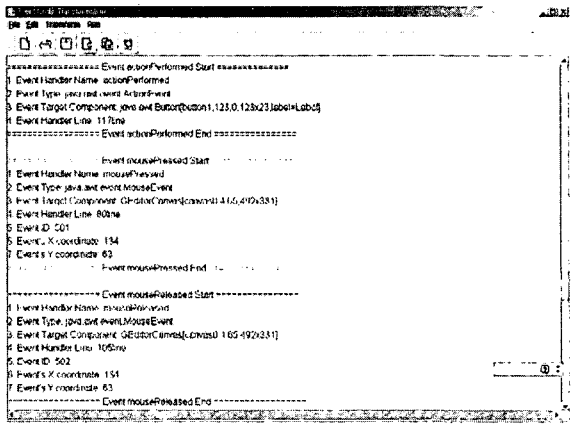
프로그램	줄 수	
	변환 전	변환 후
MonthlyScheduleViewer	659	759
RMStress	1337	1354
keycodes	168	173
EventsLog	136	289
GEditor	723	983

[표 1] 벤치마크 프로그램과 예제 프로그램 변환 후 결과

J2ME	command Action		paint		keyPressed		keyReleased	
	전	후	전	후	전	후	전	후
	0.01	0.10	0.09	0.14	0.01	0.08	·	·
	0.04	0.18	·	·	·	·	·	·
	·	·			0.02	0.09	0.01	0.11
J2SE	component Shown		mouse Moved		keyPressed		focusLost	
	전	후	전	후	전	후	전	후
	0.00	0.01	0.01	0.03	0.01	0.01	0.01	0.04
J2SE	mouse		mouse		mouse		mouseClick	

	Pressed		Dragged		Released			
	전	후	전	후	전	후	전	후
	0.00	0.01	0.00	0.01	0.01.	0.01	0.00	0.00

[표 2] 벤치마크 프로그램과 예제 프로그램 코드 변환 결과



[그림 5] J2SE 변환된 소스코드 실행 결과 화면

[그림 5]는 변환된 J2SE 프로그램 GEditor를 실하였을 때 보여 지는 화면이다. 발생하는 이벤트에 대한 트레이스 정보로 실시간 제공 되어진다.

4. 결론 및 향후 과제

자바 프로그램의 이벤트 처리에 대한 분석은 자원의 효과적인 사용과 더불어 이벤트 디버깅에 있어서 꼭 필요한 부분이다. 본 연구에서는 이를 위하여 프로그램 변환을 통하여 실행 시간에 이벤트 발생 및 처리과정을 추적할 수 있는 시스템을 설계 개발하였다. 특히 사용자 옵션에 따라 관심 있는 이벤트만을 선택적으로 추적할 수 있도록 하였다. 이를 이용하여 사용자는 이벤트 처리 과정 중 관심 있는 부분을 중심으로 추적할 수 있으며 이러한 기능은 자바 이벤트-구동 프로그램의 신뢰성 및 효율성 향상에 기여할 수 있을 것으로 기대된다.

현재까지는 프로그램의 변환 및 변환된 프로그램의 자동 실행 부분을 중심으로 구현하였다. 향후 연구에서는 실행 결과인 트레이스 정보를 시각화하여 보다 효과적인 디버깅이 가능하게 프로파일 역시 시각화하여 보다 효과적으로 프로그램 실행의 특성 보여주는 등의 연구를 수행할 것이다.

5. 참고 문헌

1. James White. An Introduction to Java 2 Micro Edition (J2ME); Java in Small Things. ICSE'01 May 12-19, 2001, Toronto, Canada
2. http://java.sun.com/j2se/1.4.2/docs/guide/jvmpi/index.hml
3. http://java.sun.com/products/j2mewtoolkit/
4. McGill Univ. *J: A Tool for Dynamic Analysis of Java Programs, ACM OOPSLA'03, October, 2003, Anaheim, CA.
5. Ulfar Erlingsson, The Inlined Reference Monitor Approach to Security Policy Enforcement, PhD thesis, Department of Computer Science, Cornell University, 2003. Available as Technical Report 2003-1916.
6. Boris Bokowski, André Spiegel. Barat - A Front-End for Java. Technical Report B-98-09 December 1998
7. http://sourceforge.net/projects/j2metest