J2ME 프로그램의 동적 이벤트 분석기의 구현

(Implementation of Dynamic Event Analysis Tool for J2ME Programs)

최 윤 정 † 창 병 모 ††

(Yoon-jeong Choi) (Byeong-Mo Chang)

요 약 J2ME 모바일 프로그램은 모바일 기기가 보편화되면서 그 사용이 증가하고 있으며 J2ME 프로그램은 자원이 제한된 모바일 환경에서 실행되므로 자원의 효율적인 사용이 매우 중요하다. 또한 J2ME 프로그램은 대부분 이벤트-구동 프로그램이며 효과적인 이벤트 관리가 프로그램의 신뢰성 및 효율적인 자원 사용에 많은 영향을 미치며 이벤트 관련 디버깅은 매우 어려운 부분이다. 본 연구에서는 실행 중에 실시간으로 이벤트 발생 및 처리 정보를 보여줄 수 있는 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.

키워드 : 이벤트, J2ME, 동적 분석, 실시간, 트레이스, 옵션, 프로파일

Abstract J2ME mobile programs have been widely used as mobile devices like mobile phones become popular. Efficient use of resources in mobile programs is very important because mobile programs are executed in mobile environment with insufficient resources. Moreover, most J2ME programs are event-driven, so effective event handling is important for reliability and efficient use of resource . In this research, we develop a dynamic event analysis system, which can show event handling in real-time. In this system, users can trace only interesting events by selecting some options, and can get event profile after execution.

Key words : Event, J2ME, Dynamic event analysis, real-time, trace, options, profile

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 [3,4]. 대부분의 J2ME 프로그램은 모바일 기기의 특성상 제한된 자원을 사용하며 입력 키 등에서 발생하는 이벤트(event)들을 처리하는 이벤트-구동(event-driven) 프로그램이다. J2ME 프로그램에서는 이러한 이벤트의 효과적인 처리가 전체 프로그램의 안전성과 신뢰성뿐만 아니라 효율에 영향을 미칠 수 있으며 이러한 이벤트-구동 프로그램은 보통 디버깅이 매우 어려운 특성을 가지고 있다.

† 정 회 원 : 삼성전자 연구원
yoonjeong.choi@samsung.com
†† 종신회원 : 숙명여자대학교 정보과학부 교수
chang@sookmyung.ac.kr
(Corresponding author임)
논문접수 : 2004년 5월 13일
심사완료 : 2006년 8월 10일

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. 그러나 이 시스템은 방대한 양의 트레이스로 인하여 실행 시간의 지연이 너무 크다. 또한 라이브러리들을 포함하여 모든 코드가 트레이스 되므로 프로그램 내의 사용자가 관심 있는 부분만을 트레이스 하기 힘들다는 단점을 갖고 있다. 또한, JVMPI[9]를 이용한 J2ME Wireless Toolkit이나 AdaptJ 등은 쓰레기 수거, 객체 할당, 예외 발생, 메소드 호출 등에 대한 트레이스를 제공하지만, 이벤트 처리에 대한 트레이스 정보를 전혀 제공하지 않고 있다[5,7]. 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 이벤트 발생 및 처리 정보를 보여줄 수 있는 동적 이벤트 분석 시스템을 설계 개발한다. 이 시스템은 사용자 옵션에 따라 관심 있는 이벤트만을 실

행 중에 추적해 볼 수 있다. 옵션을 통해 이벤트 종류, 이벤트 발생 가능한 컴포넌트, 프로파일 생성 여부 등을 선택할 수 있다. 사용자가 옵션을 선택하면 해당 이벤트의 발생 및 처리 과정을 실시간으로 트레이스, 즉 추적할 수 있다. 이 시스템을 이용하여 사용자는 관심 있는 이벤트를 중심으로 이벤트의 발생 및 처리 과정을 살펴봄으로써 J2ME 프로그램의 효과적인 디버깅을 도울 수 있을 것이며 결과적으로 프로그램의 신뢰성 및 효율성 향상에 기여할 수 있을 것이다.

본 연구에서는 또한 모니터링에 의한 실행 시간 부담을 줄이기 위해, 실행 시간에 부담을 주는 JVMPI를 사용하지 않고 코드 인라인(code inline) 기법을 기반으로 설계하였다. 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. 2장은 본 연구와 관련된 동적 모니터링 연구를 소개하고 3장에서는 시스템의 설계에 대해 설명한다. 4장에서는 구현, 5장은 실험 결과를 제시하고 6장에서 결론을 맺는다.

2. 동적 프로그램 모니터링

프로그램의 실행 과정을 동적으로 모니터링을 할 수 있게 하는 기법으로 지금까지 크게 3가지 방법이 제안되었으며 그 기본적인 아이디어는 그림 1과 같다.

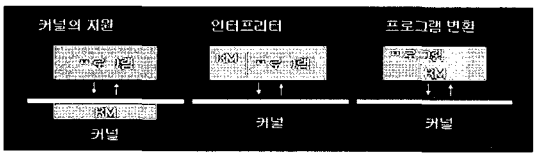

그림 1 3가지 모니터링 방법

- 첫 번째 방법은 운영체제 커널의 지원을 받아 실행 과정을 모니터링 하는 방법으로 시스템호출과 같이 관심 있는 연산을 할 때마다 커널 내의 레퍼런스 모니터(RM)를 수행함으로써 실행 과정을 모니터링 한다. 그러나 이 방법은 운영체제 커널의 지원이 있어야만 가능한 방법이며 문맥-전환(Context switching)으로 인한 오버헤드가 크다.
- 두 번째 방법은 자바 언어처럼 인터프리터에 의해 실행되는 경우에 인터프리터 내에 모니터링 하는 기능을 하는 레퍼런스 모니터(RM)를 포함시키는 것이다. 자바가상기계(JVM)도 이 기능을 제공하고 있으며 JVMPI를 통하여 실행 과정에서 발생하는 이벤트들을 추적할 수 있다. 그러나 이 방법은 실행 속도를 크게 저하시키는 단점을 가지고 있다.
- 세 번째 방법은 실행될 프로그램 내에 모니터링 기능을 하는 레퍼런스 모니터(RM)를 코드 인라인(code inline) 기법을 이용하여 코드를 삽입하고 변환된 프로그램을 실행하면 삽입된 코드에 의해 실행과정을 모니터링 하는 것이다. 이 방법은 그림 2처럼 IRM Rewriter가 원래 입력 프로그램(Original Application)과 모니터링 하고자 하는 성질 혹은 정책(Policy)을 입력으로 받아 원래 프로그램에 모니터링 할 수 있는 코드가 삽입된 프로그램을 생성한다.

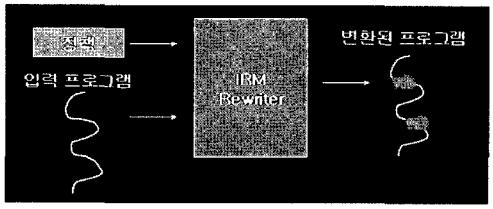

그림 2 인라인 레퍼런스 모니터

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 개발할 수 있다.

3. 설 계

본 연구에서 개발할 시스템은 사용자가 동적 이벤트 분석을 위한 옵션을 선택할 수 있으며 이를 바탕으로 사용자는 관심 있는 이벤트만의 발생 및 처리 과정을 실행 중 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 동적 분석에 의한 실행 시간 부담을 줄이기 위해 코드 인라인 방법을 기반으로 설계하였다. 코드 인라인 기법을 사용하여 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.

시스템의 전체적인 구조는 그림 3과 같다

본 시스템은 크게 네 단계로 구성되어 있다. 각 단계의 기능은 아래와 같다.

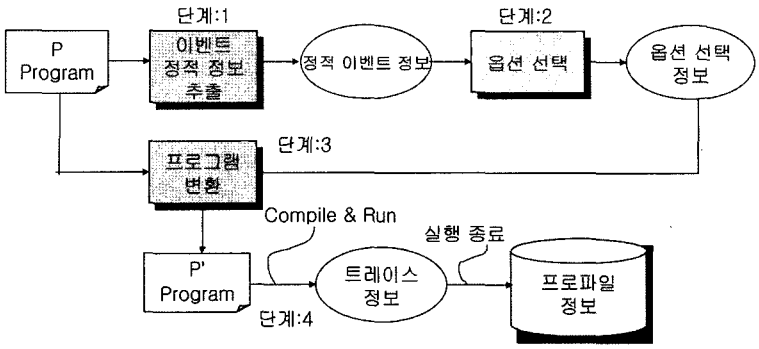

그림 3 시스템 구조

- 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 옵션에 따라 입력 프로그램 P와 똑같이 동작하면서 동적 이벤트 정보를 출력할 수 있도록 추가적인 코드를 삽입하여 새로운 프로그램 P'으로 변환 시킨다.
- 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.

4. 구 현

4.1 구현 환경

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[1].

이렇게 구성된 AST의 각 노드를 비지터(Visitor) 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 AST 정보를 기반으로 소스 코드를 다시 생성하는 OutputVisitor 등이 있다. 이러한 비지터를 확장하여 AST 노드 방문 시 원하는 연산을 수행하는 비지터를 구현할 수 있다[2]. 따라서 이러한 비지터들의 메쏘드를 재정의함으로써 AST노드를 방문 시 원하는 연산을 수행하도록 재구성할 수 있다. 본 논문에서는 Barat에서 제공하는 DescendingVisitor와 OutputVisitor의 확장하여 이벤트 정보 추출과 소스코드 변환을 구현하였다.

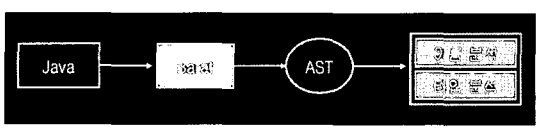

그림 4 Barat의 구조

4.2 구 현

각 단계의 구현 내용은 다음과 같다. 그림 3의 이벤트 관련 정보 추출을 위해 정적 분석기를 구현하였다. 이 정적 분석기는 입력 프로그램을 정적 분석하여 프로그램 내의 발생 가능한 이벤트 종류와 생성된 컴포넌트 등의 정적 이벤트 정보를 추출하고 이 분석정보는 사용자 선택 옵션을 제공에 사용된다.

이 정적 분석기는 Barat에서 제공하는 노드들을 깊이 우선 탐색을 해주는 DescendingVisitor를 확장하여 구현하였다. 이 정적 분석기는 입력 프로그램을 받아서 AST의 각 노드를 방문하면서 해당 노드를 방문할 때마다 발생 가능한 이벤트 종류, 구현된 이벤트 리스너, 이벤트 생성 컴포넌트 정보 등을 추출하여 구성한다.

선택 옵션은 추출된 정적 이벤트 정보를 기반으로, 프로그램 실행 시 발생 가능한 이벤트나 컴포넌트의 목록을 보여주고, 사용자가 이들 중 관심 있는 것만을 선택 가능하게 한다. 이렇게 정적 분석된 이벤트 정보를 가지고 만들어진 옵션을 이용하여 그림 5와 같은 화면을 구성하며 사용자는 관심 있는 이벤트나 컴포넌트만을 선택하여 실행 시 원하는 정보를 제공 받을 수 있다.

그림 3의 프로그램 변환은 OutputVisitor를 확장하여 구현하였다. 이 부분은 입력 프로그램 P를 옵션 선택에

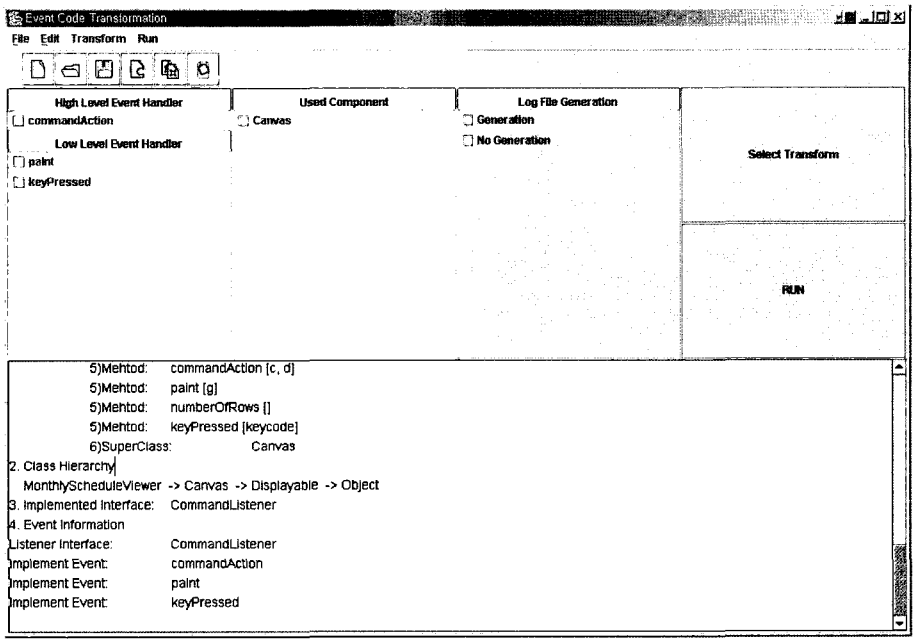

그림 5 옵션 선택 화면

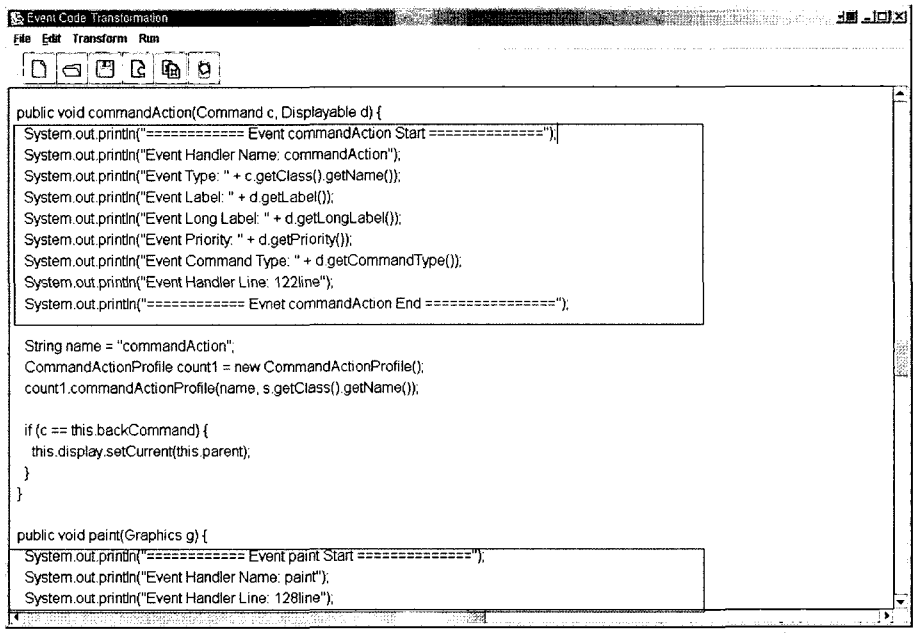

그림 6 변환된 프로그램 P′ 화면

따라 해당 이벤트들의 처리 과정을 추적할 수 있도록 코드를 추가하여 프로그램 P′을 생성한다.

코드 삽입은 다음과 같이 이루어진다. J2ME에서 가능한 이벤트는 라이브러리 형태로 미리 정해져 있고 이 벤트가 발생할 때 자동으로 수행되는 이벤트 리스너도 미리 정의되어 있다. 예를 들어 명령어 버튼을 누르면 발생되는 이벤트는 "Command" 이벤트이고 이때 실행되는 리스너는 "commandAction"이다. 따라서 옵션에서 선택한 이벤트를 위한 해당 리스너를 찾을 수 있고 이렇게 찾은 해당 리스너 앞부분에 이벤트 종류에 따라 그 특성을 보여줄 수 있는 코드를 삽입한다.

새롭게 생성된 프로그램은 그림 6과 같이 현재 프로

그램 창에 보여지며 박스 안의 부분이 추가적으로 삽입된 코드이다. MonthlyScheduleViewer 프로그램을 예제로 하여 이벤트 중 commandaction 이벤트에 대한 정보를 보여줄 수 있는 코드가 삽입되고, 아래 부분의 paint 이벤트에 대한 정보를 보여줄 수 있는 코드가 삽입된다. 이렇게 변환된 프로그램을 WTK 상에서 자동으로 실행시켜 준다. 변환 전의 프로그램과 동일하게 동작하면서 실행 중에 자동으로 우리가 원하는 정보를 트레이스 할 수 있고 마지막으로 프로파일을 얻을 수 있다.

5. 실험 결과

본 연구에서는 사용되는 실험 환경은 Windows2000 Professional과 Wireless Toolkit 2.0 버전, SDK 1.4.1 버전이다. 본 연구에서는 총 7개의 프로그램을 이용하여 실험하였다. 5개의 MIDlet 프로그램과 2개의 J2SE 프로그램을 이용하여 실험하였다.

먼저 예를 들어 구현한 시스템의 사용 과정을 살펴보자. 예로 사용할 프로그램은 J2ME 프로그램으로 달력을 제공하여 약속 정보를 저장하고 그 정보를 디스플레이 하는 MonthlyScheduleViewer 프로그램이다. 이 프로그램은 그림 7의 MonthlyScheduleViewer.java, ScheduleViewerTest.java와 MonthlyScheduleViewer_DB.java로 구성되어 있으며 데이타베이스와 연결된 프로그램으로 이벤트에 관련 트레이스 정보가 프로그램을 디버깅하는 과정에서 필요한 예이다. 여러 이벤트 핸들러 중 commandAction, KeyPressed, Paint 등과 같은 이벤트 핸들러가 구현되어 있다.

변환된 프로그램 P′은 컴파일 후에 실행되어 사용자에게 이벤트 처리 과정을 추적할 수 있는 정보를 제공한다. 변환된 프로그램을 실행하면 실행 중에 실시간으로 발생된 이벤트 정보와 처리과정 등의 트레이스 정보를 그림 8와 같이 제공한다. 또한 프로그램이 종료되었을 때 그림 9과 같이 실행 중에 발생된 이벤트들의 요약 프로파일 정보를 제공한다.

그림 8의 화면은 변환된 프로그램 P′ 실행 중 발생하는 이벤트에 대한 정보 실시간으로 사용자에게 보여주는 화면이다. 사용자가 이벤트를 발생시키고 이벤트 트레이스 정보를 통해 프로그램이 제대로 동작하는가의 여부를 알 수 있다. 현재 어떠한 이벤트가 발생하고 있으며, 그 이벤트가 적용되는 대상 컴포넌트를 알 수 있다. 또한 각 이벤트의 특성에 맞게 필요한 정보를 동시에 제공한다. 각 이벤트 마다 공통적으로 Event Handler 이름과 Event에 대한 코드의 라인 번호를 제공하고, CommandAction Event와 같은 경우는 Type, LongLevel Type, Priority, Command Type 등을 제공한다. KeyPressed Event와 같은 경우는 눌려지는 키의 이름을 제공한다.

그림 9는 프로그램이 종료되면서 제공되는 프로파일 정보로 발생한 이벤트의 종류와 그 이벤트의 발생 빈도가 포함된다. 또한, 어떠한 컴포넌트에서 어떠한 이벤트

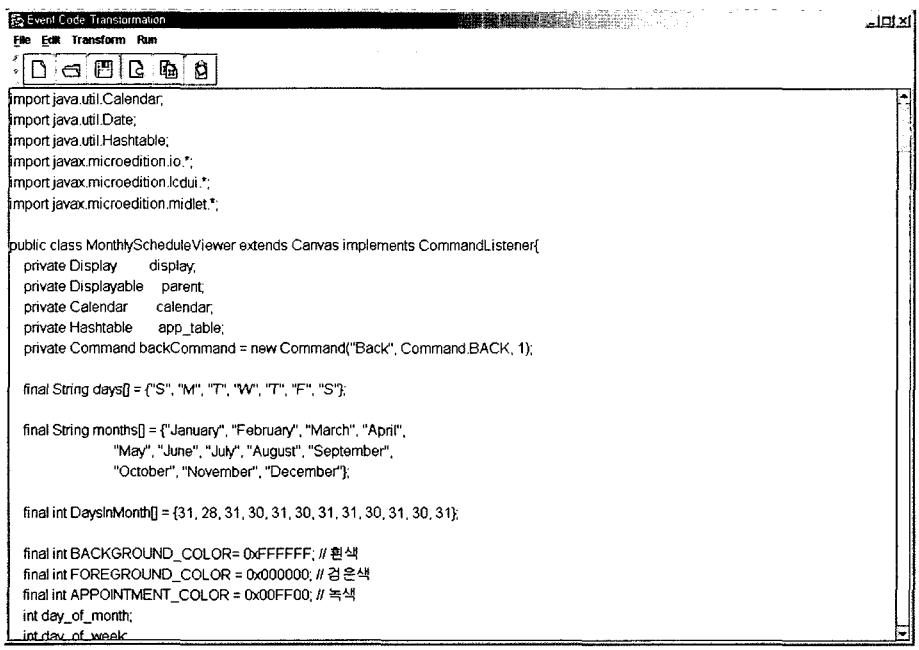

그림 7 MonthlyScheduleViewer 프로그램

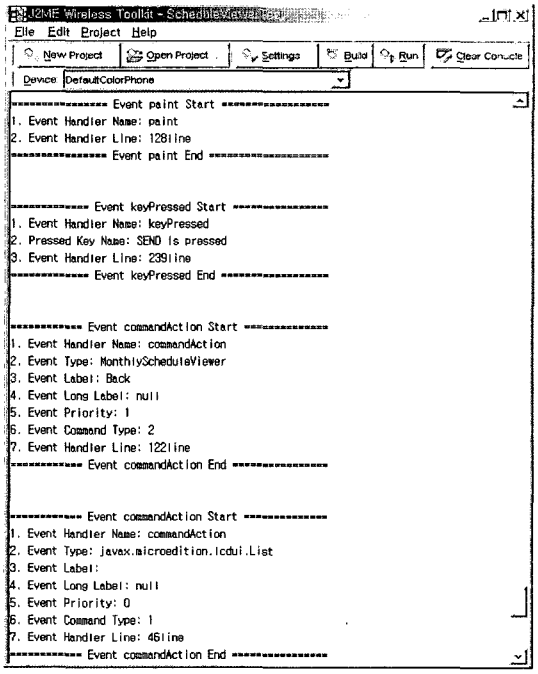

그림 8 이벤트 트레이스 정보

가 발생하였는지 또한 해당 이벤트에 대한 발생 빈도 등을 알려준다. KeyPressed 이벤트와 같은 이벤트는 Key 이벤트의 특성에 맞게 어떠한 키가 눌렸는지에 대한 정보도 제공해 줄 수 있다. 또한 구현된 상위레벨 이벤트의 정보가 제공된다.

본 연구에서는 총 7개의 프로그램을 이용하여 실험하였다. 5개의 MIDlet 프로그램과 2개의 J2SE 프로그램을 이용하여 실험하였다. 5개의 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. 세 번째는 keycodes 프로그램으로 각각의 key들의 code 값을 구하는 프로그램이다. 네 번째는 CalcMIDlet 프로그램으로 MIDlet 계산기 프로그램이다. 마지막으로 AddrBook-MIDlet 프로그램으로 주소록 프로그램이다. 또한 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.

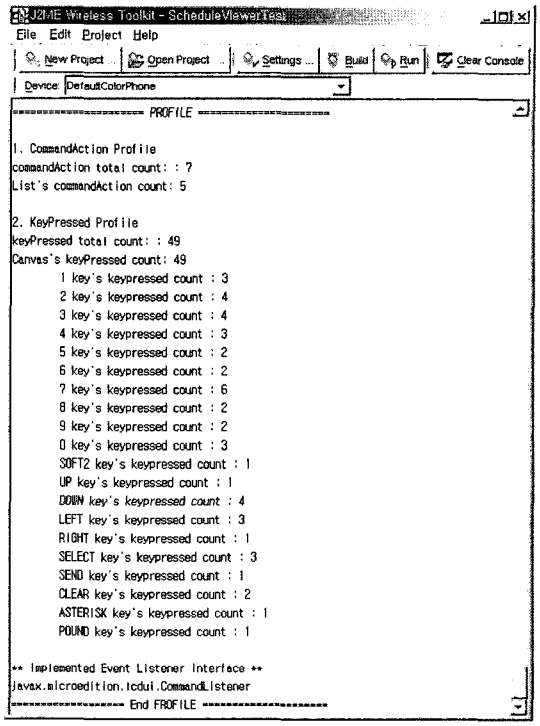

그림 9 프로파일 정보

각 프로그램을 변환하고 실행했을 때의 실험 결과를 그림 10, 그림 11에 정리하였다. 그림 10은 코드 삽입 전과 후의 줄 수를 비교해 놓았다. 이 실험을 통해서 코드 삽입으로 인한 라인 수 증가가 실용적으로 사용될 수 있을 정도임을 볼 수 있다.

본 연구에서는 사용한 프로그램들은 대부분 사용자가 종료시킬 때까지 수행되는 이벤트 구동 프로그램으로 전체실행시간은 측정할 수 없다. 따라서 본 실험에서는 변환 전과 후의 프로그램의 실행 시간을 조사하기 위하여 몇 개의 이벤트들에 대해서 이벤트 실행 시간 즉 발생부터 처리까지 걸리는 시간을 측정하였다. 그림 11은 코드 삽입 전과 후의 이벤트 실행 시간을 비교해서 보여주고 있으며 이 실험을 통해서 변환된 프로그램의 이

프로그램	줄 수	
	변환 전	변환 후
MonthlyScheduleViewer (MIDlet)	659	759
RMStress (MIDlet)	1337	1354
keycodes (MIDlet)	168	173
CalcMIDlet (MIDlet)	318	416
AddrBookMIDlet (MIDLet)	239	257
EventsLog (Applet)	136	289
GEditor (Application)	723	983

그림 10 벤치마크 프로그램과 예제 프로그램 변환 후 결과

프로그램 (MIDlet)	commandAction		paint		keyPressed		keyReleased	
	변환 전	변환 후	변환 전	변환 후	변환 전	변환 후	변환 전	변환 후
MonthlyScheduleViewer	0.01sec	0.10sec	0.09sec	0.14sec	0.01sec	0.08sec	·	·
RMStress (j2metestSuit)	0.04sec	0.18sec	·	·	·	·	·	·
keycodes (j2metestSuit)	·	·	0.02sec	0.09sec	0.01sec	0.11sec	·	·
CalcMIDlet	0.14sec	0.26sec	0.01sec	0.08sec	0.01sec	0.071sec	0.01sec	0.07sec
AddrBookMIDlet	0.06sec	0.15sec	·	·	·	·	·	·
프로그램 (J2SE Applet)	componentShown		mouseMoved		keyPressed		focusLost	
	변환 전	변환 후	변환 전	변환 후	변환 전	변환 후	변환 전	변환 후
EventsLog	0.00sec	0.01sec	0.01sec	0.03sec	0.01sec	0.01sec	0.01sec	0.04sec
프로그램 (J2SE Application)	mousePressed		mouseDragged		mouseReleased		mouseClick	
	변환 전	변환 후	변환 전	변환 후	변환 전	변환 후	변환 전	변환 후
GEditor	0.00sec	0.01sec	0.00sec	0.01sec	0.01sec	0.01sec	0.00sec	0.00sec

그림 11 벤치마크 프로그램과 예제 프로그램 코드 변환 결과

벤트 처리 시간이 현실적으로 사용될 수 있는 것을 확인할 수 있었다.

6. 관련 연구

본 연구는 기본적으로 프로그램의 동적 분석을 위한 도구를 개발한다. 이 동적 분석 방법은 현재 정적 분석 기법과 더불어 활발한 연구가 진행되고 있는 분야이다. 동적 분석은 프로그램 실행 중에 얻을 수 있는 실시간 정보를 사용자에게 제공할 수 있다. 이러한 요구에 맞추어 프로그램의 동적 분석을 위한 도구가 개발되고 있다. 현재 개발되어 있는 동적 분석 프로그램 도구로는 Sun 사에서 개발한 J2ME Wireless Toolkit과 McGill 대학에서 개발한 AdaptJ와 AdaptJ를 시각화한 EVolve가 대표적이다[6,7].

J2ME Wireless Toolkit은 자바 가상 기계에서 제공하는 JVMPI[9]를 이용하여 메모리 사용 및 쓰레기 수집 등을 트레이스 할 수 있다. 또한 메쏘드 호출, 발생된 예외, 클래스 로딩 등을 트레이스 할 수 있다. 프로그램 실행이 끝난 후에는 메쏘드 사용 등에 관한 프로파일 정보를 제공한다. 그러나 이 도구는 트레이스가 실행에 너무 큰 부담을 주어 실행 속도가 매우 느려질 뿐만 아니라 관련 라이브러리 등을 모두 트레이스 함으로써 실제 프로그래머가 작성한 부분을 효과적으로 트레이스 하기 힘들다.

다음 실험에서 JVMPI의 오버헤드를 보여준다. 이 실험은 JVMPI(Java Virtual Machine Profiler Interface)를 이용할 경우 방대한 양의 트레이스로 인하여 실행시간의 지연이 어느 정도인지를 보여 준다. 그림 12 JVMPI를 사용하여 실행시킨 결과이다. J2ME Wireless Toolkit의 JVMPI를 이용한 트레이스 기능을 선택하고

	트레이스 기능 선택하지 않은 경우	JVMPI를 이용한 트레이스 기능 선택한 경우
로딩 시간	1초	6분 13초
실행 시간	1초	37초

그림 12 JVMPI를 이용한 트레이스의 시간 측정

프로그램을 실행 시켰을 경우 실행 시간을 측정했다.

본 논문에서는 이러한 문제점 해결을 위해 실행 중에 실시간으로 이벤트 발생 및 처리 과정을 보여줄 수 있고, 사용자가 원하는 프로파일을 제공할 수 있는 동적 *이벤트 분석 시스템*을 개발했다. 특히 이 시스템은 실행 시간에 부담을 주는 JVMPI를 사용하지 않고 입력 프로그램에 이벤트 실행 정보를 출력할 수 있도록 *코드를 삽입*하여 실행한다.

AdaptJ는 자바프로그램의 동적 분석을 위한 도구이다[7]. 이 도구는 기본적으로 JVMPI를 사용하여 사용자가 원하는 동적 실행 정보를 파일의 형태로 저장하여 프로그램 종료 시에 이 정보를 제공한다. 사용자는 동적 실행 정보 중 옵션을 선택하여 필요한 정보만을 추출할 수 있다. 이 도구의 주된 목적은 첫째, 자바 프로그램의 실행에 관련된 동적 실행 정보의 수집을 쉽게 해주는 것이고 둘째, 수집된 정보를 빠르고 쉽게 분석하는 프로그램을 구현하는 프레임워크를 제공하는 것이다. 프로그램 종료 후 생성된 트레이스 정보를 파일 형태로 사용자에게 제공한다. 또한 McGill 대학에서 개발한 EVolve는 AdaptJ가 생성한 트레이스 정보를 시각화하여 사용자에게 보다 편리하게 제공한다[8].

이 도구의 단점은 프로그램 실행 중 실시간으로 트레이스 정보를 제공하지 않는다는 점이다. 프로그램 종료 후 사용자가 원하는 이벤트에 대한 정보를 파일의 형태

로 제공하므로 사용자가 동적 분석 정보를 제공 받기 위해서는 프로그램을 실행하고 종료해야하는 번거로움이 있다. 본 연구는 위와 같은 단점을 해결하기 위해 개발 되어졌다.

7. 결론 및 향후 과제

J2ME 프로그램의 이벤트 처리에 대한 동적 분석은 프로그램 안정성과 신뢰성 향상에 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 모바일 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.

참 고 문 헌

[1] B. Bokowski, A. Spiegel. Barat - A Front-End for Java. Technical Report B-98-09 Dec. 1998.

[2] http://www.sharemation.com/bokowski/barat/ index.html

[3] J. White. An Introduction to Java 2 Micro Edition (J2ME); Java in Small Things. 23rd International Conference on Software Engineering (ICSE'01) May 12-19, 2001, Toronto, Canada.

[4] http://java.sun.com/j2me/

[5] http://java.sun.com/products/j2mewtoolkit/

[6] Bruno Dufour, Karel Driesen, Laurie Hendren and Clark Verbrugge. Dynamic Metrics for Java. ACM OOPSLA '03, October, 2003, Anaheim, CA.

[7] McGill Univ. *J: A Tool for Dynamic Analysis of Java Programs, ACM OOPSLA'03, October, 2003, Anaheim, CA.

[8] Qin Wang, Wei Wang, Rhodes Brown, Karel Driesen, Bruno Dufour, Laurie Hendren and Clark Verbrugge, EVolve: An open extensible software visualization framework, ACM Symposium on Software Visualization 2003.

[9] http://java.sun.com/j2se/1.4.2/docs/guide/jvmpi/index.html

최 윤 정

2003년 2월 숙명여자대학교 컴퓨터과학전공 이학사. 2005년 2월 숙명여자대학교 컴퓨터과학전공 이학석사. 2005년 3월 삼성전자 연구원. 관심분야는 모바일 소프트웨어, 프로그램 모니터링

창 병 모

1988년 서울대학교 컴퓨터공학과 졸업(공학사). 1990년 한국과학기술원 전산학과(공학석사). 1994년 한국과학기술원 전산학과(공학박사). 1994년~1995년 한국전자통신연구소 박사후 연수 연구원 1995년~현재 숙명여자대학교 컴퓨터과학과 부교수. 관심분야는 컴파일러 구성론(정적분석, 코드최적화), 논리 프로그래밍, 모바일 프로그래밍