韓國 컴퓨터情報學會 論文誌
第10卷 第3號, 2005. 7.

2005-10-3-1-4

프로그램 성능 평가 로그 정보를 이용한 레포트 뷰 생성기 구현

조 용 윤*, 유 재 우**

An Implementation of the Report View Generator using Program Performance Log Information

Yong-Yoon Cho*, Chae-Woo Yoo**

요 약

소프트웨어 개발자는 소프트웨어 개발 속도 향상과 품질 개선을 위해 성능 평가 도구를 이용 할 수 있다. 그러나 성능 평가 도구가 생성하는 텍스트 기반의 평가 결과는 이해가 어렵고 복잡하여 결과 분석에 많은 시간과 노력을 요구한다. 본 논문은 소프트웨어의 텍스트 기반 성능 평가 정보를 다양한 그래픽 뷰(views)를 통해 제공하기 위한 레포트 뷰(report view) 생성기를 제안한다. 제안하는 생성기는 복잡한 성능 평가 로그(log)를 분석하여 다루기 쉬운 자료 구조로 변환하고 이것을 클래스(class) 형태의 API를 통해 그래픽 기반 레포트 뷰로 출력한다. 이를 위해, 로그 분석기(log analyzer)는 복잡한 텍스트 기반 성능 평가 로그(log)를 평가 항목에 따라 구별된 XML 문서로 변환하고 추출하는 모듈을 제공한다. 또한, 화면 구성기(view composer)는 XML 로그 문서로부터 생성할 레포트 뷰의 내용 명세 정보를 추출하고 개발자로부터 화면 구성 형식 정보를 입력받는다. 입력된 레포트 뷰 구성 정보는 화면 구성기가 제공하는 클래스 API를 통해 레포트 뷰로 변환 생성된다. 따라서 개발자는 자신이 선택한 특정 성능 평가 항목에 대한 로그 정보와 화면 구성 정보를 이용해 다양한 그래픽 기반의 레포트 뷰를 생성할 수 있으며, 성능 평가 결과에 대한 직관적인 분석과 빠른 소프트웨어 품질 개선이 가능하여 소프트웨어 개발 효율성을 높일 수 있을 것으로 기대된다.

Abstract

A software developer can use a performance evaluation tool to elevate development speed and improve quality of softwares. But, evaluation results that most performance evaluation tools offer are complicated strings. Therefore, a developer cannot intuitively understand the meanings of the results and must make much times and efforts in analysing the result. In this paper, we propose a report view generator that can transform and provide the text-based performance evaluation results for softwares with various graphic-based views. Our proposed generator consists of a screen generator that creates a structural XML document about the text-based performance evaluation results and a log analyzer that makes various report view through the created XML evaluation document. Because the XML evaluation result document can express the result information structured according to performance evaluation items for resources of softwares, it can have flexibility in offering and integrating the result information for the items. Through the suggested report view generator, developers can intuitively understand and analysis performance evaluation results of embedded software. And they can easily and quickly improve software quality and improve development efficiency of softwares.

▸ Keyword : Report view generator, Log analysis , User interface, Program Performance evaluation

---

• 제1저자 : 조용윤 ※ 본 연구는 숭실대학교 교내 연구비 지원으로 이루어졌음.
• 접수일 : 2005.04.11, 심사완료일 : 2005.05.20
* 숭실대학교 컴퓨터학부 대학원 박사과정, ** 숭실대학교 컴퓨터학부 교수

Ⅰ. 서 론

소프트웨어 개발자는 효율적인 소프트웨어 개발을 위해 개발 소프트웨어의 성능 평가와 분석을 실시한다[1]. 개발자는 소프트웨어 성능 평가 도구(PET - Performance Evaluation Tool)를 통해 보다 편리한 소프트웨어 성능 평가와 분석이 가능하다[1]. 그러나 소프트웨어 성능 평가 도구가 제공하는 평가 결과는 복잡한 텍스트 형태의 로그 정보이다[2]. 텍스트 형태의 로그 정보는 개발자가 이해하기 힘들고 효율적인 이용이 어렵다. 따라서 소프트웨어 성능 분석 도구가 생성한 텍스트 기반 성능 평가 정보는 개발자의 직관적 이해와 분석을 위해 다양한 그래픽 형태의 레포트 뷰(view)로 제공되는 것이 타당하다[3]. 개발자는 하나의 성능 평가 결과에 대해 다양한 형태의 그래픽 뷰를 원할 수 있다. 즉, 어떤 개발자는 메모리 사용량에 대해 함수별 막대 그래프 형태로 표현되는 성능 평가 뷰를 원할 수 있다. 또한 다른 개발자는 실시간 시퀀스 다이어그램을 통해 함수 호출에 따른 메모리 사용량의 변화에 대한 성능 평가 뷰를 원할 수 있다. 일반적인으로 그래픽 기반 레포트 뷰는 화면 구성 정보와 내용 정보의 조합을 통해 생성된다[1][2]. 그러나, 레포트 뷰 생성을 위해 하나의 화면 구성 정보와 하나의 내용 정보를 고정적으로 조합하는 방식은 개별자가 원하는 다양한 뷰 생성을 위해 비효율적이다. 따라서 성능 평가 도구는 사용자의 기호에 따른 다양한 형태의 성능 평가 뷰를 생성할 수 있는 방법을 제공해야 한다. 본 논문은 하나의 성능 평가 로그에 대해 다양한 그래픽 뷰를 생성할 수 있는 소프트웨어 성능 평가 레포트 뷰 생성기를 제안한다. 이를 위해 제안하는 레포트 뷰 생성기는 로그 분석기(log analyzer)와 화면 구성기(view composer)로 구성된다. 로그 분석기는 성능 평가 정보의 효율적인 사용을 위해 저수준의 성능 평가 로그를 파싱(parsing)하여 각 평가 항목에 따라 XML 문서로 변환한다. 화면 구성기는 생성될 레포트 뷰의 내용 명세 정보로써 생성된 XML 성능 평가 정보를 선택 한다. 이때, 화면 구성기는 선택된 내용 명세 정보와 개발자가 선택한 화면 구성 정보를 클래스 형태의 API에 입력하여 레포트 뷰를 구성한다. 이를 위해, 본 논문에서는 개발자에게 편리한 화면 구성 정보 입력을 위한 폼(form) 스크립트(script)를 제공한다. 따라서 개발자는 제안하는 레포트 뷰 생성기를 통해 자신이 선택한 내용 명세 정보와 화면 구성 정보를 통해 다양한 조합의 레포트 뷰 생성이 가능하다. 본 논문에서 제안하는 성능 평가 레포트 뷰 생성기는 교차 개발 환경기반의 임베디드 소프트웨어를 대상으로 실험 하고 구현한다. 임베디드 시스템은 다른 기기 안에 내장(embedded)되어 해당 기기를 제어하기 위한 컴퓨터 시스템을 말한다[2]. 임베디드 소프트웨어는 임베디드 시스템에 포함되어 마이크로프로세서를 제어하는 소프트웨어이다[2]. 일반적으로 임베디드 소프트웨어 개발은 호스트/타겟(host/target)기반의 교차 개발(cross development) 환경에서 이루어진다[3]. (그림 1)은 임베디드 소프트웨어를 위한 교차 개발 환경 의 개념도 이다.

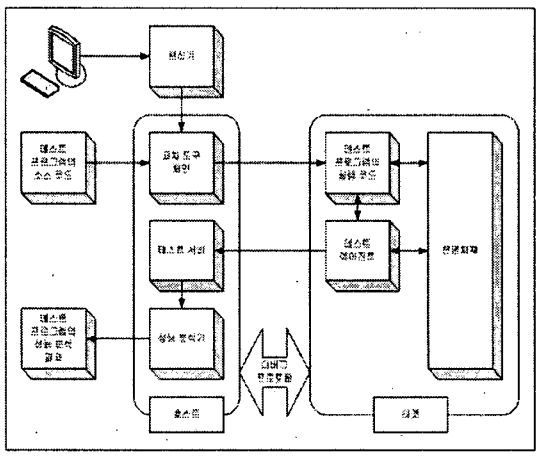

그림 1. 임베디드 소프트웨어의 교차 개발 환경
Fig 1. Cross-Development Environment in Developing Embedded Software

교차 개발 환경에서의 실제 프로그램 성능 평가는 타겟에서 성능 평가 모듈에 의해 실행되며, 개발자는 호스트에서 성능 평가 결과를 분석 한다. 범용 컴퓨터 시스템과는 달리 임베디드 시스템은 적은 컴퓨팅 자원만을 지원하기 때문에, 임베디드 소프트웨어는 제한된 자원에 대해 최적화된 성능을 발휘할 수 있도록 설계되어야한다[2]. 따라서 임베디드 소프트웨어 개발자는 임베디드 소프트웨어가 사용하는 프로세스와 쓰레드 정보뿐 아니라 각 프로세스 별 CPU 및 메모리(스택, 힙, 텍스트) 사용량과 같은 자원 사용 상태 정보(resource usage status information)를 분석하여야만 한다[2][3]. 본 논문은 리눅스 기반의 임베디드 시스템

에서 GNU[4] 툴을 사용한 저수준의 로그 분석을 통한 다양한 그래픽 뷰 생성을 위한 레포트 뷰어를 구현한다. 본 논문은 2장에서 관련 연구를 소개하고 3장에서 제안한 레포트 뷰어의 구조를 설명한다. 4장은 결론 및 향후 연구방향에 대해 설명하고 맺는다.

II. 관련 연구

2.1 임베디드 소프트웨어 성능 테스팅 방법

임베디드 시스템에서 저수준의 로그는 하드웨어적인 성능 측정 방법과 소프트웨어적인 성능 측정 방법을 통해 생성할 수 있다[3]. 하드웨어적인 방법은 빠른 실행과 타겟에 최적화된 테스팅이 가능하고 분석 결과를 저장하기 위한 공간을 별도로 가질 수 있는 장점을 가지고 있다. 하지만 추가적으로 고가의 장비를 필요로 하고 타겟과 호스트 사이의 하드웨어적 연결을 위한 추가적 노력이 요구되고 에뮬레이터(emulator)에 대한 별도의 학습이 필요하다. 소프트웨어적인 방법은 고가의 에뮬레이터를 필요하지 않는다. 그러므로 에뮬레이터에 대한 추가적인 학습이나 고 비용의 개발비가 없어도 성능 평가 방법을 사용할 수 있는 장점을 가지고 있다. 성능 테스팅을 위한 타겟 테스팅 에이전트를 개발 보드에서 실행시킴으로 호스트에서 서비스를 요청하면 시리얼이나 이더넷을 통해 호스트로 성능 평가 정보를 보내준다. 또한, 성능 평가는 일반적인 디버깅 기능을 포함하여 정지점 관리와 레지스터 수정/조회, 함수 호출 관리, 이벤트 관리, 예외 통보 관리 등의 기능을 제공 할 수 있다. 그러나 발생된 모든 성능 테스트 결과는 저수준의 텍스트 기반 정보이기 때문에 개발자가 직관적으로 성능 테스트 결과 정보를 파악하기는 어렵다. 즉, 실시간 에이전트를 통해 발생한 테스트 결과에 대해 개발자가 직접 대화형 쉘과 자원 모니터를 통해 분석해야 한다. 개발자가 직접 분석하는 방법은 응용프로그램 개발 이외의 수고를 들게 하여 응용프로그램의 개발을 비효율적으로 만드는 약점이 있다. 일반적으로 임베디드 소프트웨어는 다음과 같은 요구사항을 만족해야 한다[2][3].

- 임베디드 시스템에 사용되는 프로세서는 일반적으로 전력 소모와 크기를 줄이기 위하여 낮은 성능을 가지므로, 불필요한 코드들이 가능한 적어야 하며 프로세서 자원을 적게 사용해야 한다.
- 임베디드 시스템은 범용 시스템에 비하여 매우 적은 용량의 메모리를 가지고 있고, 임베디드 운영체제는 프로세스의 메모리 영역을 보호하는 기능이 없거나 미약하여, 프로세스가 종료되더라도 명시적으로 해제되지 않은 메모리 블록들에 대한 완전한 회수가 불가능할 수 있다. 따라서 메모리 자원을 적게 사용해야 하며 메모리 누수(memory leak)를 방지해야 한다. 따라서 임베디드 소프트웨어 개발자는 개발 소프트웨어가 이와 같은 요구사항을 만족하는지에 대한 테스팅이 반드시 필요하다. 다음 〈표 1〉 은 임베디드 소프트웨어에 대해 요구되는 가장 기본적이며, 중요한 성능 측정 항목이다[3].

표 1. 임베디드 소프트웨어의 성능 평가항목
Table 1. Performace evaluation items for embedded software

테스팅 범위	내용
성능 테스팅	- 응용 프로그램 전체 또는 원하는 일부분의 실행에 걸리는 시간을 측정 - 특정 함수별 실행 시간 측정 - CPU 할당 및 처리 시간 측정
코드 범위 테스팅	- 응용 프로그램을 실행 시 실제로 사용되는 부분과 사용되지 않는 부분, 자주 사용되는 부분, 거의 사용되지 않는 부분을 측정
메모리 테스팅	- 메모리의 할당과 해제 정보를 출력 - 전체 메모리 사용량 측정. 또한, - 할당 되었는데 해제되지 않은 메모리를 검사함으로써 메모리 누수를 측정 - 중복 해제되어 버그를 유발시킬 수 있는 코드를 측정
트레이스 테스팅	- 응용 프로그램의 실행과 함께 어떤 순서로 함수 호출이 일어나는지를 출력 - 응용 프로그램이 정상적으로 진행되고 있는지 여부 측정 - 불필요하게 함수 호출이 많이 일어나지는 않는지를 측정

〈표 1〉에서 메모리에 관한 성능 테스팅 정보는 메모리의 할당과 해제에 관한 로그를 분석하여 메모리 누수와 메모리 사용량에 관한 정보를 알려준다. 메모리의 누수는 소프트웨어의 메모리 자원을 불필요하게 소모하는 것으로 메모리 자원이 적은 임베디드 시스템에서 치명적이 오류가 된다. 함수의 자취에 관한 정보는 재귀적으로 함수가 호출되는 응용

프로그램에서 외부 입력에 따른 함수 호출 경로를 파악할 수 있는 정보가 된다. 그리고 블록의 실행 범위에 관한 정보는 실행되지 않은 블록이 소스 코드에 삽입되어 실행되지 않는 기계어 코드를 제거하는 정보가 된다. 마지막으로 함수의 시간적 성능에 관한 정보는 전체 소프트웨어의 시간적 성능을 모듈별로 측정할 수 있는 정보가 된다.

2.2 임베디드 소프트웨어 성능 테스팅 도구

gprof[4]는 유닉스와 리눅스의 개발 도구 패키지에 기본으로 포함되어 있으며, 가장 널리 사용되고 있는 소프트웨어 성능 평가 도구이다. 유닉스 계열 표준 컴파일러인 cc 명령에 '-pg' 옵션을 전달하면, 코드 생성 단계에서 성능 평가를 위한 코드들을 삽입하게 된다. 이렇게 만들어진 실행 프로그램을 일반적인 응용 프로그램과 동일한 방법으로 실행시키면, 프로그램 종료 직전에 텍스트 형태의 로그 파일을 남긴다. 이 파일을 gprof를 통해 분석하면 성능 평가 결과를 얻을 수 있다. 그러나 출력되는 결과가 각 함수의 실행 통계를 나타내는 함수들 간의 호출 관계를 나타내는 Call Graph만으로 제한적이다. 따라서 개발자가 원하는 각 자원 활용 성능 평가 결과에 대한 다양한 형태의 뷰를 얻을 수 없다. 또한, 로그 수집 과정에서 표준 C 라이브러리 이외에 GNU C 라이브러리(glibc)에 포함된 함수들을 다수 호출하기 때문에, GNU C 라이브러리가 포팅 되지 않은 임베디드 운영체제에서는 사용이 불가능하다.

FunctionCheck[5]는 gprof의 부족한 기능 몇 가지를 보완한 공개 도구이다. 구체적으로 성능 평가 데이터 파일의 이름 변경 가능, 멀티 쓰레드와 멀티 프로세스의 지원, 메모리 성능 평가 출력 기능 등이 추가되었다. 그러나 기본적으로 gprof와 같은 방법으로 성능 평가 정보를 수집하므로 동일한 환경 제한과 오차를 가질 수밖에 없다.

Soft4Soft 사의 RESORT[6]는 소프트웨어 매트릭스(metrics) 기반으로 패키지 소프트웨어를 분석, 테스트하고 성능 평가 결과에 대해 다양한 그래픽 뷰를 제공하여 품질 관리를 지원하는 도구이다. 그러나 이 도구는 범용 패키지 소프트웨어의 개발을 지원하는 도구로, 적은 자원을 갖고 교차 개발을 지원하는 임베디드 소프트웨어 개발 환경에서는 사용이 불가능하다.

AstonLinux사의 CodeMaker[7], 미지리서치 사의 Linu@[8], MontaVista 사의 CDK[9], Lineo 사의 Embedix SDK[10] 등은 모두 윈도우 또는 리눅스 호스트 환경에서 임베디드 리눅스를 타겟으로 하는 임베디드 소프트웨어를 개발하는 통합 개발 환경이다. 그러나 이들 도구는 응용 소프트웨어의 개발 보조와 디버깅(debugging)이 주된 기능으로써, 소프트웨어 실행 전반에 걸친 성능 평가 결과를 산출해서 그래픽 형태의 레포트 뷰를 보여주는 성능 평가 도구로써의 기능은 적다.

III. 본론

3.1 임베디드 소프트웨어 성능 분석을 위한 레포트 뷰 생성기

본 논문에서 제안하는 그래픽 기반 레포트 뷰 생성기의 전체적인 구성은 (그림 2)와 같다.

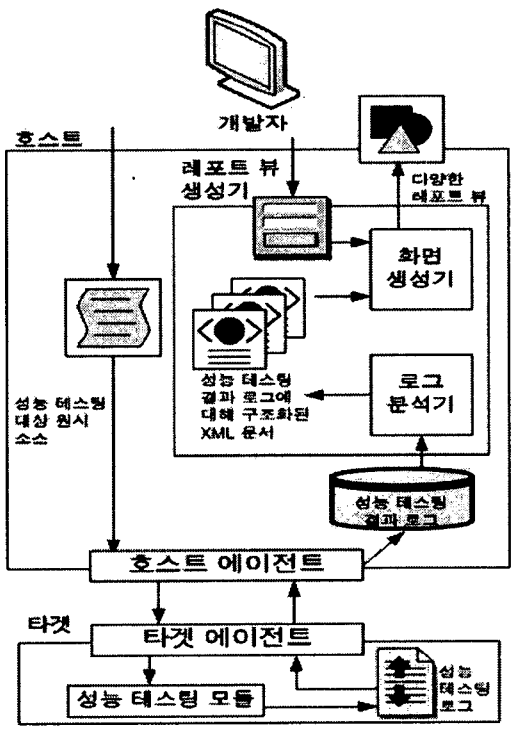

그림 2. 임베디드 소프트웨어 교차 개발 환경
Fig 2. Cross-Development Environment in Developing Embedded Software

(그림 2)에서 교차 개발 환경의 타겟 성능 테스팅 모듈은 호스트 에이전트로부터 전달받은 성능 테스팅 대상 원시 소스의 타겟 임베디드 시스템 자원에 대한 성능 실험을 실행한다. 이때, 생성되는 테스팅 결과 값은 스트링 형태의 로그이다. 따라서, 타겟 에이전트는 호스트 에이전트에게 테스팅 결과 로그를 전달하고 호스트 에이전트는 이를 결과 로그 저장소에 저장한다. 저장된 텍스트 형태의 로그 정보는 레포트 뷰 생성기에 포함된 로그 분석기에 의해 XML 형태의 구조화된 문서로 변환된다. 화면 생성기는 생성된 XML 문서로부터 생성할 레포트 뷰의 내용 명세 정보를 추출한다. 화면 생성기는 추출된 내용 명세 정보와 개발자가 입력하는 레포트 뷰 화면 구성 정보를 이용해 그래픽 형태의 결과 뷰를 생성한다.

3.2 로그 분석기

로그 분석기는 텍스트 형태의 성능 평가 로그 정보를 그래픽 뷰의 GUI 구성 요소로 쉽게 변환 생성할 수 있도록 구조화된 XML 문서로 변환하는 모듈이다. 제안하는 도구의 로그 분석기는 두 단계를 통해 저수준의 로그를 분석한다. 첫 번째 단계는 파싱(parsing) 단계이다. 파싱 단계를 통해 타겟에서 생성되어 구별 없이 통합된 성능 평가 정보의 로그 파일은 의미 있는 토큰 단위 정보로 나뉘고 평가 항목에 따라 분할된다. 두 번째 단계는 정보 변환(information translation) 단계이다. 파싱 단계를 통해 항목별로 분류된 로그는 정보 변환 단계를 통해 평가 항목별 엘리먼트(element)로 구조화된 XML 문서로 변환된다. (그림 3)은 로그 분석기의 구조를 나타낸다.

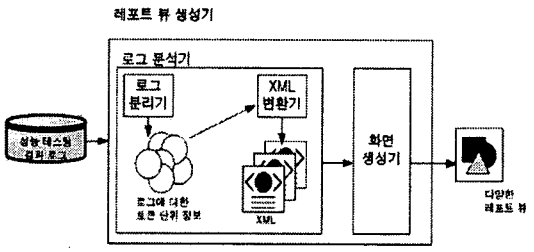

그림 3. 레포트 뷰 생성기의 로그 분석기 구성도
Fig 3. The log separator in the suggested report view generator

(그림 3)의 로그 분리기는 연속적인 텍스트 스트링 형태의 로그 정보를 어휘 분석(token analysis)을 통해 토큰(token)단위로 나누고 성능 평가 항목에 따라 분류 한다. 분류된 토큰 단위의 로그 정보는 XML 변환기에 의해 미리 정의된 DTD에 유효한(valid) XML 문서로 변환 생성된다. XML 변환기가 이용하는 DTD는 개발자의 요구에 따라 더욱 다양한 성능 평가 결과를 표현할 수 있기 때문에 확장성이 높다. 본 논문에서는 〈표 1〉에 기술된 성능 평가 항목에 대해 리눅스 기반의 임베디드 소프트웨어에 대한 성능 평가 로그 발생을 위해 GNU 툴과 메모리 트레이서(tracer)를 개선하였다. 생성된 저수준의 성능 평가 로그는 로그 분리기를 통해 XML 문서로 변환된다. 〈표 2〉는 로그 분석기가 〈표 1〉의 테스팅 항목에 따라 타겟에서 생성된 성능 평가 로그를 XML 문서로 변환하기 위한 DTD이다.

표 2. 로그 정보에 대한 XML DTD
Table 2. XML DTD for log informations

```
〈?xml version="1.0" encoding="euc-kr"?〉
〈!ELEMENT profile (memoryProfile,
performanceProfile, codeCoverageProfile,
traceProfile)〉
〈!ATTLIST profile
  name CDATA #REQUIRED
〉
〈!ELEMENT memoryProfile〉
〈!ATTLIST memoryProfile
  allocatedBlockCount CDATA #REQUIRED
  unfreedBlockCount CDATA #REQUIRED
  maximumBlockCount CDATA #REQUIRED
  allocatedByteSize CDATA #REQUIRED
  unfreedByteSize CDATA #REQUIRED
  maximumByteSize CDATA #REQUIRED
〉
〈!ELEMENT performanceProfile〉
〈!ATTLIST performanceProfile
  name CDATA #REQUIRED
  path CDATA #REQUIRED
  functionAndDescendantsTime CDATA #REQUIRED
  functionTime CDATA #REQUIRED
  functionAverageTime CDATA #REQUIRED
  functionMaximumTime CDATA
#REQUIRED
  functionMinimumTime CDATA
#REQUIRED
  functionCalls CDATA #REQUIRED
〉
〈!ELEMENT codeCoverageProfile
(codeCoverageElement)*〉
〈!ELEMENT codeCoverageElement〉
〈!ATTLIST codeCoverageElement
  name CDATA #REQUIRED
  path CDATA #REQUIRED
  blocksCount CDATA #REQUIRED
  blocksCovered CDATA #REQUIRED
  functionsAndExitsCount CDATA
#REQUIRED
  functionsAndExitsCovered CDATA
#REQUIRED
  functionsCount CDATA #REQUIRED
  functionsCovered CDATA #REQUIRED
〉
```

생성된 XML 문서는 테스팅 결과에 대해 항목별로 잘 구조화되어 있기 때문에 그래픽 뷰를 생성하기 위한 정보로써 재가공 되기 매우 쉽다. 또한, 개발자는 생성된 XML 문서의 각 엘리먼트(element) 단위의 개별 정보를 서로 조합하여 다양하고 효율적인 레포트 뷰 구성이 가능하다. 다음은 로그 분석기가 〈표 1〉의 DTD를 이용해 입력되는 성능 평가 결과에 대한 저수준 로그를 구조적인 XML 문서로 변환하기 위한 알고리즘 이다.

3.2.1 Repeat

// 입력되는 저수준 로그 파일을 줄 단위로 읽어 들여 각 토큰을 성능 항목별로 구분된 배열에 저장한다.

```
if (log_analyzer(string) == Memory)
   // 메모리 관련 결과를 배열 A1에 저장
or (log_analyzer(string) == Trace)
   // 트레이스 관련 결과를 배열 A2에 저장
or (log_analyzer(string) == Coverage)
   // 커버리지 관련 결과를 배열 A3에 저장
or (log_analyzer(string) == Performance)
   // 퍼포먼스 관련 결과를 배열 A4에 저장
Until end-of 저수준 로그 화일
```

3.2.2 Repeat Ai

// 성능 항목별 구분 저장된 각 배열(Ai)의 성능평가 결과에 대해 미리 정의된 관련 태그를 태깅(tagging)한다.

```
Repeat Ai(j)
```

// Ai 배열의 j번째 항목의 성능 평가 결과 내용을 〈표 2〉에 나타난 각 항목별 엘리먼트의 해당 속성 값과 연결한다.

```
   Until end-of Ai
Until end-of A1, A2, A3, A4
```

3.2.3 Repeat Ai

// 태깅된 각 결과의 요소들(elements)을 조합해 DTD에 유효한 (valid) XML 문서를 생성한다.

```
Until end-of A1, A2, A3, A4
```

3.3 화면 생성기

화면 생성기는 생성된 XML 문서의 정보를 이용해 그래픽 뷰를 생성하기 위한 모듈이다. 이때, 레포트 뷰 생성기는 개발자에게 그래픽 뷰의 화면을 구성하는 각 그래픽 요소들의 모양과 위치 정보를 입력할 수 있도록 폼(form) 형태의 GUI를 제공한다. 개발자는 레포트 뷰 생성기가 제공하는 GUI를 통해 자신이 원하는 형태의 뷰를 구성할 수 있다. (그림 4)는 화면 생성기의 구조를 나타낸다.

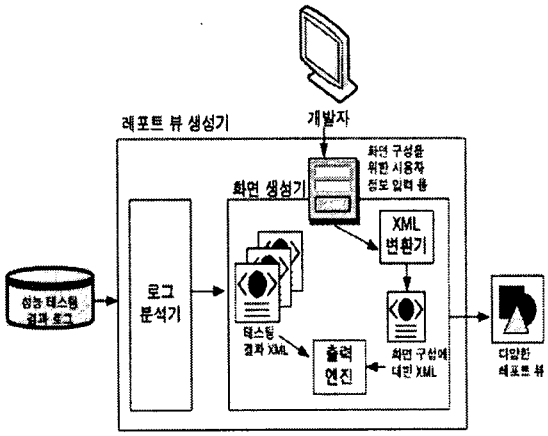

그림 4. 레포트 뷰 생성기의 화면 생성기 구성도
Fig 4. The view maker in the report view generator

(그림 4)에서 화면 생성기는 로그 분석기가 생성한 성능 테스팅 결과 XML 문서를 개발자가 입력한 화면 구성 정보와 함께 레포트 뷰 생성을 위해 사용한다. 이때, 개발자는 자신의 개발 시점에서 요구되는 성능 평가 항목에 대해 원하는 그래픽 형태로 보기를 원할 수 있다. 즉, 어떤 개발자는 성능 평가 대상 소프트웨어의 메모리 누수에 관련된 정보를 실행되는 함수의 순서에 맞추어 막대그래프나 원형그래프 형태로 비교 출력되는 레포트 뷰를 원할 수 있다. 또한 그 개발자는 어떤 시점에서는 메모리 누수에 관련된 정보를 독립적인 형태의 챠트 그래프로 보기를 원할 수 있을 것이다. 본 논문에서 제안하는 화면 생성기는 개발자가 원하는 성능 평가 정보는 로그 분석기가 생성한 XML 문서를 이용하고 그 정보를 어떤 형태의 레포트 뷰로 보여줄 것인

지를 결정하기 위한 정보로 개발자가 입력하는 화면 구성 정보를 이용한다. 때문에, 기존 성능 평가 분석 도구가 수동적이고 고정적인 그래픽 뷰만을 제공하여 성능 평가 결과를 레포팅하는 것과 비교해 제안하는 레포트 뷰 생성기는 개발자에게 자신이 원하는 형태의 자료와 GUI 구성을 통해 능동적이고 다양한 뷰를 조합하고 생성할 수 있는 기회를 제공할 수 있다. 이때, 사용자가 더욱 편리하게 화면 구성 정보를 입력할 수 있도록 폼 형태의 인터페이스가 제공된다. 화면 생성기를 구성하는 XML 변환기는 폼 인터페이스를 통해 입력된 화면 구성 정보를 구조화된 XML 문서로 변환한다.

Ⅳ. 실험 및 평가

제안하는 성능 평가 레포트 뷰 생성기는 임베디드 소프트웨어를 개발하기 위한 일반적인 시스템 소프트웨어 언어인 ANSI C 소스를 성능 평가 대상 소스로 실험하였다. 또한 성능 평가 레포트 뷰가 이용하는 성능 평가 결과 로그를 위하여 〈표 3〉과 같은 교차 개발 임베디드 시스템 환경을 사용했다.

표 3. 시스템 개발 환경
Table 3. System development environment

CPU		ARM9
OS		Velos
Board		HRP-SC2410 (Ami)
Development Tool	Host	Java2TM SDK v1.4.2
	Target	GNU Cross Tool Chain for Cygwin - GNU C Compiler (gcc) v2.95 for ARM - ld, as, ar, objdump, objcopy - C Library (libc)
Hardware		Lauterbach TRACE-32 ICD (In-Circuit-Debugger)

개발 환경은 교차 개발 환경을 지원하기 위한 호스트/타겟 구조이며, 운영체제는 한국 MDS사와 서울대가 개발하고 상용화 한 국산 임베디드 운영체제 솔루션인 Velos를 사용한다[13]. Velo는, POSIX 표준 API, 선점형 스케쥴링에 의한 다중 쓰레드와 함께 임베디드 시스템에 특성화 된 빠르고 유연한 인터럽트 처리와 고유의 실시간 주기 쓰레드를 지원하며, 동적 메모리 관리, 타이머, 시그널 처리 기능이 포함되어 있다. 현재 성능 평가 엔진은 국산 임베디드 운영체제인 Velos 환경에 맞게 작성되어 있으며, 다른 운영체제를 위해 개발하려는 경우 타겟 성능 평가 엔진과 타겟 성능 평가 라이브러리만 수정하면 되기 때문에 확장성이 용이하다. (그림 5)는 제안한 레포트 뷰 생성기를 위한 도구창(tool window)이다.

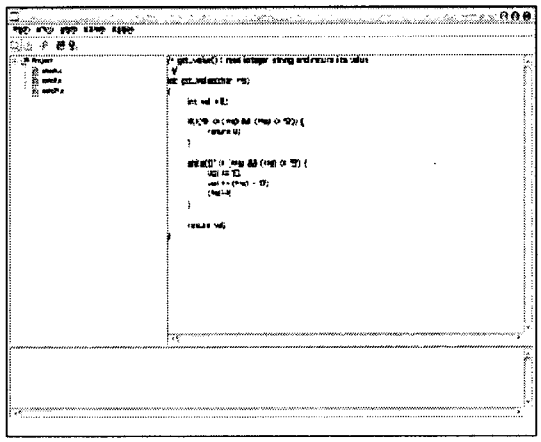

그림 5. 레포트 뷰 생성기 도구 창
Fig 5. The tool window for report view generator

성능 평가 대상 원시 코드는 3개의 함수로 이루어진 간단 C 코드이다. (그림 5)에서 프로젝트(project)로 원시 코드를 빌드(build)하면, 포함된 GNU 교차 개발 툴 체인(cross-development tool chain)[14]이 소스 파일에 대하여 순서대로 어휘 분석, 구문 분석, 코드 수정, 언파싱(unparsing), 컴파일을 수행하며[15], 모든 소스 파일이 컴파일 과정까지 완료되면 각 목적 코드들을 함께 링크하여 타겟에서 실행될 수 있는 실행 코드를 생성한다. 생성된 실행 코드에 대해 타겟의 성능 평가 엔진은 성능 평가를 실행하여 그 결과를 로그 형태로 생성한다. 〈표 4〉는 GNU 툴의 gprof 와 mtrace[14]가 생성한 가공되지 않은 저수준의 로그 중 일부이다.

표 4. 입력 C 코드에 대한 로그 화일
Table 4. Log file for inputted C codes

```
index % time self  children   called     name
                                95000     calc2 <cycle 1> [3]
[2]     0.0  0.00   0.00       95000     calc21 <cycle 1> [2]
                                900       calc2 <cycle 1> [3]
-----------------------------------------------
                                900       calc21 <cycle 1> [2]
        0.00 0.00   1000/1000             stack [13]
[3]     0.0  0.00   0.00       1900      calc2 <cycle 1> [3]
                                95000     calc21 <cycle 1> [2]
-----------------------------------------------
Index by function name
   [3] calc2       [2] calc21       [1] <cycle 1>

@ <Location>:(Called function + Called Location)[Instruction
    Location] +/- Address Size
= Start
@ ./ex-n:(mtrace+0x169)[0x80484e9] + 0x804a378 0x12
@ ./ex-n:[0x8048596] - 0x804a378
```

레포트 뷰 생성기에 있는 로그 분리기는 <표 4>의 텍스트 형태의 로그를 각 의미에 따라 분리한다. 예를 들어 표 4에서 첫 번째 라인의 정보는 불필요한 정보이고, "[1]"과 같은 토큰은 함수의 인덱스정보이다. 그러므로 인덱스 정보를 기준으로 한 함수의 정보를 추출해낸다. 추출된 정보는 XML 변환기를 통해 XML 문서로 변환된다. <표 5>는 XML 변환기가 <표 4>의 로그 정보에 대해 생성한 XML 문서의 일부분이다.

표 5. 로그 정보에 대해 생성된 XML 문서
Table 5. XML Document for log informations

```
<?xml version="1.0" encoding="UTF-8"?>
<profile>
<memoryProfile>
   ...
</memoryProfile>
<coverageProfile>
  <coverageElement   blocksCount="3"
                     blocksCovered="1"
                     functionsAndExitsCount="1"
                     functionsAndExitsCovered="1"
                     functionsCount="1"
                     functionsCovered="1"
                     name="push_stack"
    path="C:\test\stack.c"/>
   ...
</coverageProfile>
<performanceProfile rootTime="67">
  <performanceElement name="main"
                      path="C:\test\calc2.c"
                functionAndDescendantsTime="67"
                      functionAverageTime="6"
                      functionCalls="1"
                      functionMaximumTime="6"
                      functionMinimumTime="6"
    functionTime="6"/>
   ...
</performanceProfile>
<traceProfile>
   ...
</traceProfile>
</profile>
```

로그 분석기에 의해 생성된 성능 평가 결과 XML 문서는 사용자가 입력하는 그래픽 구성 정보와 함께 화면 생성기에 의해 다양한 형태의 그래픽 뷰로 변환된다. (그림 6)은 화면 생성기가 <표 5>의 XML 문서와 개발자의 정보를 입력받아 메모리에 관련된 정보를 막대그래프 형태로 생성한 레포트 뷰이다.

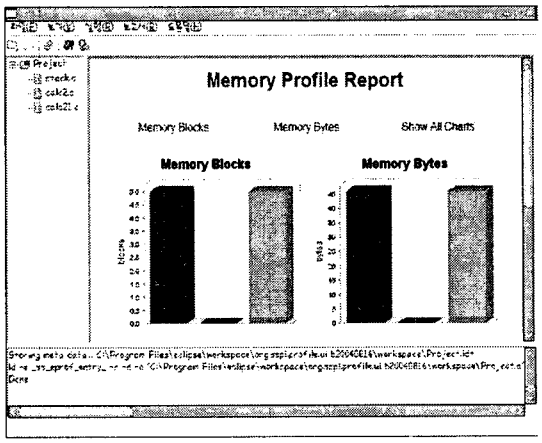

그림 6. 메모리 성능 평가 그래픽 뷰
Fig 6. Graphic view for memory performance evaluation

(그림 6)에서 레포트 뷰는 성능 평가 대상 소프트웨어가 어느 정도의 메모리를 사용하며, 메모리 누수 발생의 가능

성은 없는지, 그리고 메모리 관련 코드에서 발생한 에러의 정확한 원인이 무엇인지 등을 판단하여 보고한다. 개발자는 메모리 프로파일 결과를 참고로 하여, 메모리 사용에 문제가 있는 코드들을 쉽게 찾아내 수정할 수 있다. (그림 7)은 함수의 호출에 관련된 성능 평가 결과를 사용자의 그래픽 화면 요구에 따라 시퀀스 다이어그램(sequence diagram) 형식과 파이 그래프(pie graph) 형식으로 변경하여 나타내고 있다.

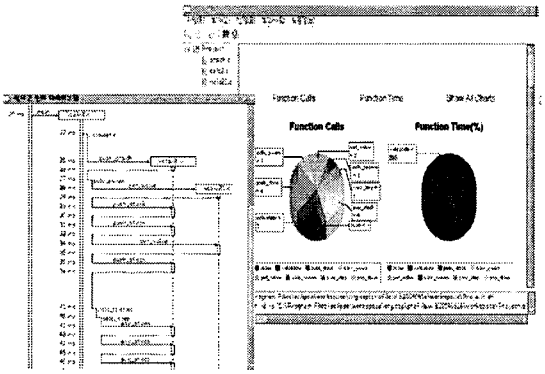

그림 7. 함수 관련 성능 평가 레포트 뷰
Fig 7. Graphic view for performance evaluation related in functions

(그림 7)에서의 두 가지 그래픽 뷰는 어떤 함수가 자주 실행되거나 실행 시간을 많이 소요하여 병목 현상을 발생시키는지 확인할 수 있는 정보를 제공한다. 그러나 파이 그래프 형식의 그래픽 뷰는 대상 소프트웨어에 포함된 각 함수의 실행 횟수와 실행 시간에 대해 시퀀스 다이어그램 보다 더욱 보기 편리한 통계 형태의 테이블 자료를 함께 제공한다. 또한, 전체 프로그램 실행시간 동안에 각 함수의 실행 여부에 대한 통계 정보를 제공하여 전체적인 함수 관련 정보를 한눈에 비교 분석할 수 있는 기회를 제공한다. 그러나 호출 순서를 직관적으로 파악하기에는 수치적인 통계표는 부적절할 수 있다. 따라서 개발자는 시퀀스 다이어그램 형태의 레포트 뷰를 통해 소프트웨어가 실행된 후 어떤 시간에 어떤 순서로 각 함수의 호출이 일어났는지를 직관적으로 알 수 있는 기회를 가질 수 있다. 이로부터 각 함수가 언제 어떤 경로를 거쳐 몇 번이나 실행되었으며, 비정상적으로 많이 호출되거나 시간이 많이 소요되지는 않는지 확인이 가능하다. 그러나 시퀀스 다이어그램은 일반적으로 프로그램의 실행에 대한 가장 자세한 정보를 제공하지만, 프로그램의 실행 시간이 길어지면 정보의 양이 지나치게 많아져서 개발자가 분석하기 어려울 수 있다. 따라서 개발자는 정보의 양과 자신의 기호에 따라 함수 실행과 호출에 관련된 성능 평가 정보를 다른 형태의 그래픽 뷰를 통해 분석할 수 있기 때문에 한 번도 실행되지 않은 함수나 블록 코드를 제거하고, 프로그램 전반에 걸친 코드의 유용성을 확인할 수 있다.

V. 결론 및 향후 과제

소프트웨어 개발 과정에서의 제품 성능 향상을 위해 성능 평가와 분석은 매우 중요하다. 본 논문은 개발자가 그래픽 뷰를 통해 소프트웨어 의 성능 평가 테스트 결과를 직관적으로 이해하고 분석할 수 있도록 지원하는 성능 평가 레포트 뷰 생성기를 제안하였다. 제안한 생성기는 레포트 뷰 생성을 위해 하나의 내용 명세 정보와 화면 구성 정보를 고정적으로 연결하는 방식을 탈피하여 개발자가 레포트 뷰 구성을 위한 정보를 선택할 수 있는 방법을 제공하였다. 이를 위해 로그 분석기는 성능 평가 로그를 정보 이용 효율이 높은 XML로 변환 저장하였다. 이것은 내용 명세 정보로써의 성능 평가 결과를 로그의 형태보다 훨씬 효율적으로 접근하고 추출할 수 있는 방법을 제공한다[16][17]. 또한, XML 형태의 성능 평가 로그는 웹(Web) 기술과의 연동이 용이하여 다양한 응용이 가능하다. 또한, 화면 구성기는 명세 정보와 화면 구성 정보를 입력으로 하는 클래스 형태의 API를 제공하였다. 따라서 개발자는 레포트 뷰 구성을 위한 편리한 클래스 API를 통해 자신이 원하는 다양한 레포트 뷰 구성이 가능하다. 본 논문은 클라이언트/서버 구조의 레포팅 뷰를 제공하는 시스템을 제안하였으나, 현재의 컴퓨팅 환경은 웹 기반으로 발전하고 있으므로 앞으로 웹 기반 레포팅 뷰 시스템으로 연구 발전 시켜야 할 것이다.

참고문헌

[1] Mustafa M. Tikir, JeffreyK. Hollingsworth, "Efficient Instrumentation for Code Coverage Testing", International Symposium on Software Testing and Analysis Proceedings of the ACM SIGSOFT, pp.86-96, 2002.

[2] Bart Broekman, Testing Embedded Software, Addison-Wesley, pp.128-129 Dec. 2002.

[3] L. Hatton, "Embedded Software Testing", Software Testing Congress, 2000.

[4] GNU Profiler (GPROF) and reference page, available at http://www.gnu.org/software/binutils/manual/gprof-2.9.1/html_mono/gprof.html.

[5] FunctionCheck, available at http://www710.univ-lyon1.fr/%7Eyperret/fnccheck/doc.html

[6] RESORT, http://www.soft4soft.com/

[7] CodeMaker, http://www.astonlinux.com/

[8] Linu@, http://www.mizi.com/ko/

[9] CDK, http://www.mvista.com/

[10] Embedix SDK, http://www.lineo.com/

[11] Marn van den Brand, Paul Klint, Chris Verhoef, "Re-engineering needs Generic Programming Language Technology", SIGPLAN Notices, Volume 32, pp. 128-135 1997.

[12] Chris Verhoef, "Towards Automated Modification of Legacy Assets", Annals of Software Engineering, Volume 9, pp. 221-228, Baltzer Science Publishers, 2000.

[13] Velos, http://www.velos.co.kr/

[14] GNU Tool Chain and C Compiler (GCC) web page, available at http://gcc.gnu.org.

[15] Alfred V. Aho, Ravi Sethi, Jeffery D. Ullman, Compilers: principles, techniques, and tools, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1986.

[16] 정강용, 박나연, "웹서버의 로그파일 분석에 의한 웹 서비스 활용에 관한 연구", 한국컴퓨터 정보학회 ,5권 1호, pp.32-38, 2000.3

[17] 정우식, 도경화, 전문석, "IDS/Firewall/Router 통합 로그 분석기 설계", 한국 컴퓨터 정보학회, 8권 1호, pp. 48-56, 2003.3

저자소개

조 용 윤

1995년 시립 인천대학교 전자계산과 졸업(학사)

1998년 숭실대학교 대학원 전자계산학과 졸업 (공학석사)

1999~현재 숭실대학교 박사과정

〈관심 분야〉 컴파일러, 프로그래밍 언어, 프로그래밍 환경, XML, 임베디드 시스템

유 재 우

1976년 숭실대학교 전자계산학과 (공학사)

1978년~1987년 한국과학기술원 전산학과(공학석사,공학박사)

1986년~87년, 96년~97년 Cornell Univ.및 Univ. of Pittsburgh Visiting Scientist

1983년~현재 숭실 대학교 컴퓨터학부 교수

〈관심 분야〉 컴파일러, 프로그래밍 환경, HCI, XML