제 25 회 한국정보처리학회 춘계학술발표대회 논문집 제 13 권 제 1 호 (2006. 5)

PSP 지원을 위한 개인 메트릭 자동 수집 및 분석 도구 개발

신현일*, 최호진*, 백종문*
*한국정보통신대학교 공학부
e-mail : {linugee, hjchoi, jbaik}@icu.ac.kr

An Automated Metrics Collection and Analysis Tool for PSP Support

Hyunil Shin*, Hojin Choi*, Jongmoon Baik*
*Information and Communications University, School of Engineering

요 약

소프트웨어 개발 프로젝트에서 메트릭 수집 및 분석 활동이 점차 중요하게 인식되고 있다. 메트릭 수집 및 분석 활동은 조직/프로젝트, 팀, 개인 모든 레벨에서 수행되어야 하는 중요한 활동으로 여겨져 오고 있다. Personal Software Process(PSP)[1]에서 개발자 개개인이 수행해야 되는 메트릭 수집 및 분석 활동이 제시된다. 이러한 메트릭 수집 및 분석 활동을 통해 개발자는 소프트웨어 품질 향상, 계획 단계에서 보다 정확한 예측 활동, 개인 프로세스의 정량적 관리 등의 이득을 얻을 수 있다. 이러한 이득을 얻기 위해서는 신뢰성 있는 메트릭 데이터의 수집이 무엇보다 중요하게 된다. 그러나 메트릭 수집의 오버헤드와 context switching 과 같은 문제로 인해 개발자가 신뢰성 있는 메트릭을 수집하는데 많은 어려움이 겪는다[2, 3]. 또한 PSP 가 제시하는 분석 기법만으로는 수집된 메트릭에 대하여 의미 있는 분석을 하기 어려운 문제점이 존재한다. 이러한 문제점들을 감소 시키기 위해 메트릭 수집 및 분석 도구를 개발하였고 본 논문에서는 이 개발된 도구를 설명한다. 이 도구의 핵심은 메트릭의 자동 수집과 다양한 분석 결과의 제공을 통해 신뢰성 있는 메트릭 데이터의 획득과 의미 있는 분석을 가능케 하는 데 있다.

1. 서론

체계적이고 지속적인 소프트웨어 개발 프로세스 측정 및 분석 활동은 프로젝트 관리, 프로세스 개선, 소프트웨어 품질 향상에 있어서 중요한 요소 중의 하나로 알려져 있다 [9, 10]. 측정 및 분석 활동은 조직/프로젝트, 팀, 개인 모든 레벨에서 수행 되어야 하는 핵심 활동으로 강조되고 있다.

소프트웨어 개발자 개개인의 소프트웨어 개발 역량을 향상시키기 위한 방법으로서 PSP 가 개발되었다[1]. 소프트웨어 개발 역량의 향상을 위해 계획, 계획 추적, 소프트웨어 크기와 개발 시간 예측, 소프트웨어 품질 관리, 설계/코드 리뷰, 소프트웨어 설계, 메트릭 수집 및 분석의 다양한 기법들이 PSP 에서 제시된다. 이러한 기법들을 통해 소프트웨어 품질의 향상, 보다 정확한 예측, 개인 프로세스의 정량적 관리, 자신 역량의 객관적인 평가 등의 이득을 개발자들이 얻을 수 있다. 이러한 이득을 얻기 위해서는 무엇보다 메트릭 수집 및 분석 활동이 중요하게 되고 이러한 메트릭 수집 및 분석 활동의 효과적인 수행을 위해서는 무엇보다 신뢰성 있는 메트릭 데이터의 수집이 중요해진다. 그러나 메트릭 수집의 오버헤드와 context switching 과 같은 문제로 인해 개발자가 신뢰성 있는 메트릭을 수집하는 데는 많은 어려움이 존재하는 걸로 알려져 있다[2, 3]. 또한 PSP 가 제시하는 분석 기법만으로는 수집된 메트릭에 대하여 의미 있는 분석을 하기 어려운 문제점이 존재한다. 이러한 문제점들을 해소하기 위해 메트릭 자동 수집 및 분석 도구를 개발하였고, 본 논문에서는 개발된 메트릭 자동 수집 및 분석 도구를 설명한다.

본 논문은 다음과 같이 구성된다. 2 장에서는 관련

연구가 기술되고 3 장에서는 메트릭 자동 수집 및 분석 도구의 아키텍처가 소개된다. 4 장과 5 장에서는 각각 이 도구에서 제공되는 메트릭 수집과 분석에 대한 설명이 주어진다. 6 장에서는 결론 및 향후 연구에 대해서 기술된다.

2. 관련 연구

1) Hackystat

Hackystat 는 PSP 메트릭(시간, 크기, 결함)을 자동으로 수집하고 수집된 데이터를 분석해 주는 도구[2]이다. 센서라는 프로그램이 개발 도구에 플러그인 형태로 삽입되어 PSP 메트릭을 자동으로 수집한다. 수집된 데이터는 서버를 통해 데이터베이스에 저장되고 개발자는 웹 브라우저를 통해서 수집된 데이터를 분석할 수 있다.

2) PROM (Pro Metrics)

PROM[7]은 소프트웨어 제품과 프로세스 메트릭을 자동으로 수집하는 데이터 수집 및 분석 도구이다. 수집된 데이터는 PSP 메트릭, 프로시저와 객체 지향 메트릭, 워드 프로세서를 이용한 문서 작업 등 코딩 작업 이외의 활동을 추적하기 위한 특별 메트릭을 포함한다. PROM 은 다음의 네 가지 컴포넌트로 이루어져 있다.

- PROM Probes: 개발 관련 도구에 플러그인 형태로 부착되어 관련 메트릭을 수집하는 PROM plug-in 과 운영체제의 시스템 호출을 감시하는 PROM Trace를 포함한다.
- PROM Transfer: 수집된 데이터를 플러그인으로부터 받고, 필요한 프리프로세싱(pre-processing)을 한 후, 프리프로세싱된 메트릭 데이터를 PROM Server로 보낸다.
- PROM Server: PROM Database에 데이터를 저장하는 역할을 하며 분석 및 프로젝트와 사용자 관리 등의 기능을 웹 서비스를 통해서 제공한다.
- PROM Database: 사용자와 프로젝트 관련 정보, 수집되는 메트릭 데이터와 분석 결과를 저장한다.

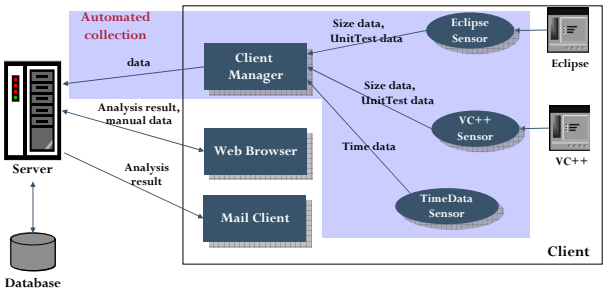
개발자와 프로젝트 관리자는 웹 페이지를 통해서 PROM Server 에 원하는 종류의 분석을 요청할 수 있다. 예를 들어, 코드의 길이, 복잡도, 커플링(coupling)을 나타내는 그래프, 한 프로젝트에 쓰여진 전체 시간 등 원하는 분석을 웹 인터페이스를 통해서 PROM Server 에 요청할 수 있다. 또한 제 3 자(third-parties)가 제작한 분석도구를 이용해서 원하는 분석을 할 수 있게 하는 메커니즘을 제공한다.

센서를 이용한 메트릭의 자동 수집과 서버를 통한 수집된 메트릭의 분석을 제공한다는 점에서 Hackystat 와 PROM 도구는 본 논문에서 설명하려는 도구와 비슷하다. 본 논문에서는 다음과 같은 점에서 위 연구들을 확장하였다.

- 식스 시그마 분석의 추가: 식스 시그마 분석을 통해 개발자에게 좀 더 다양하고 의미 있는 분석 결과를 전달해 준다.
- TimeData Sensor: 개발 시간을 수집하기 위해 Hackystat, PROM 과 다르게 모든 종류의 파일을 모니터링 할 수 있는 TimeData Sensor 을 개발하였다. Hackystat, PROM 에서는 파일(예, doc)과 연관되어 있는 도구 (예, MS word 등)의 플러그인(센서)을 개발함으로써 해당 파일의 수정 시간을 수집한다. 이러한 방법은 도구마다 플러그인의 개발을 필요로 하기에 플러그인이 개발되지 않은 도구와 관련된 파일의 수정을 모니터링 하지 못하는 단점을 가지고 있다.

3. 메트릭 자동 수집 및 분석 도구 아키텍처

메트릭 자동 수집 및 분석 도구에는 크게 메트릭 자동 수집, 매뉴얼 입력, 수집된 메트릭에 대한 다양한 분석 결과 제공 기능을 포함한다. 이러한 기능을 수행하기 위해 이 도구는 메트릭의 자동 수집을 담당하는 센서, 수집된 데이터를 서버로 보내는 클라이언트 매니저, 수집된 메트릭의 분석과 매뉴얼 입력을 담당하는 서버, 그리고 데이터베이스로 구성된다. 이 도구에서 지원하는 메트릭 자동 수집과 매뉴얼 입력은 4 장에서, 제공되는 다양한 분석 결과는 5 장에서 자세히 설명된다.



(그림 1) 도구 아키텍처

1) 센서

센서는 메트릭을 자동으로 수집해 주는 소프트웨어 도구이다. 이클립스[4] 플러그인으로 개발된 Eclipse Sensor 와 VC++ 플러그인으로 개발된 VC++ Sensor 는 각각 이클립스와 VC++에서 실행되는 유닛 테스트(예, JUnit[5] 테스트)의 실행 결과와 LOCC[6]와 같은 코드 카운팅 도구의 실행 결과를 자동으로 수집한다. TimeData Sensor 는 개발자가 지정한 디렉토리 내의 파일들의 크기를 주기적으로 모니터링 함으로써 개발 시간 정보를 수집한다. 각 센서에서 수집된 데이터는 XML 문서로 특정 디렉토리에 저장된다.

2) 클라이언트 매니저

클라이언트 매니저는 다양한 센서들이 생성한 XML 문서를 서버로 보내는 기능을 담당한다.

3) 서버

서버는 웹 애플리케이션으로 개발되었다. 클라이언트 매니저에서 전달되는 XML 문서를 파싱하여 데이터베이스에 메트릭을 저장하고, 개발자의 요청에 따

라 다양한 분석 결과를 웹브라우져를 통해 개발자에게 전달한다. 분석 결과는 메일 클라이언트를 통해서 개발자에게 주기적으로(예, 매일) 전달될 수도 있다. 또한 메트릭의 매뉴얼 입력 및 수정, 프로젝트 등록 기능을 포함한다.

4) 데이터베이스

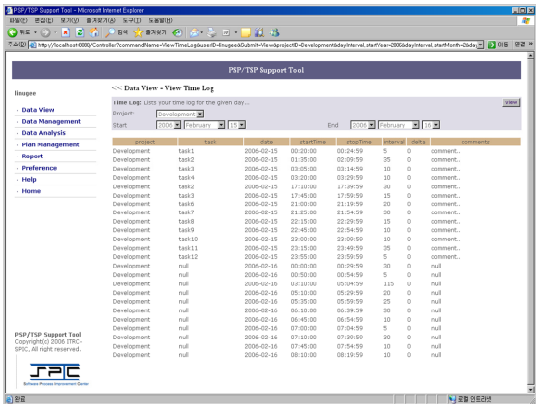
사용자, 프로젝트 정보 그리고 수집된 메트릭을 저장한다.

4. 메트릭 수집

본 논문이 소개하는 도구는 메트릭의 자동 수집, 매뉴얼 수집 모두를 지원한다. 본 장에서는 개발 시간, 결함, 소프트웨어 크기를 자동으로 수집하는 방법과 매뉴얼 수집 방법이 기술된다.

1) 개발 시간

개발자는 프로젝트 전체 기간 내에서 프로젝트의 다양한 활동에 사용한 시간을 계속적으로 기록해야 한다. 설계, 코딩, 테스트, 회의, 코드/설계 리뷰 등의 다양한 활동이 프로젝트에 포함된다. 다양한 활동 중에서 자동으로 수집되는 개발 시간은 개발자가 개발 관련 파일(예, 소스 파일, 디자인 문서) 작성 및 수정에 사용한 시간이다. 개발 시간을 자동으로 수집하기 위해 개발자가 수정, 작성 중인 개발 관련 파일을 주기적으로 모니터링 한다. 개발자가 지정한 디렉토리 안에 있는 파일들의 크기를 30 초 간격으로 체크함으로써 개발자가 어떤 파일을 수정 중인지 알 수 있다. 크기가 변경된 파일이 있으면 해당 파일의 이름과 모니터링 된 시간을 데이터베이스에 저장한다. 이렇게 저장된 정보들은 5 분 단위로 다시 분석된다. 예를 들어 21 시 10 분 11 초에 파일 크기 변경이 있었으면 21 시 10 분 00 초부터 21 시 15 분 00 초까지가 개발 시간으로 인식된다. 즉, 5 분 단위에 최소한 한 파일의 변경이 있었으면 해당 5 분이 개발자가 사용한 개발 시간으로 인식되는 것이다. 이러한 방법은 [2]에서 처음으로 소개되었다. 이렇게 생성된 시간 정보는 시간 기록 로그(그림 2)의 입력으로 들어가게 된다. 시간 기록 로그에는 프로젝트, 업무, 날짜, 시작 시간, 마침 시간, 인터럽트 시간, 주석(comments)이 포함된다. 개발자는 시간 기록 로그를 수정하거나 자동으로 수집될 수 없는 회의 시간 등을 추가로 입력할 수 있다.



(그림 2) 시간 기록 로그

2) 결함

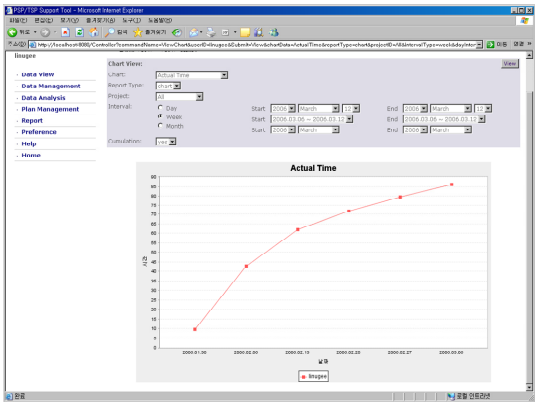
개발자는 요구사항, 설계, 구현, 테스트의 각 단계에서 발생한 결함을 계속적으로 수집해야 한다. 결함 기록 로그에는 결함 유형, 결함이 발생한 단계, 결함을 제거한 단계, 결함 제거 시간 등이 포함된다. 결함의 자동 수집을 위하여 유닛 테스트의 실행 결과를 자동으로 수집한다. 예를 들어 이클립스에서 개발자가 JUnit 테스트를 실행할 때마다 유닛 테스트의 성공 여부, 테스트 시간, 테스트 클래스/메소드 이름, 에러 메시지 등의 실행 결과가 데이터베이스에 저장된다. 수집된 유닛 테스트 결과는 중복된 유닛 테스트의 제거 등의 프리프로세싱을 거쳐 결함 기록 로그로 만들어진다. 시간 기록 로그와 마찬가지로 개발자는 결함 기록 로그를 수정하거나 코드/설계 리뷰 등을 통하여 발견된 결함을 추가할 수 있다.

3) 소프트웨어 크기

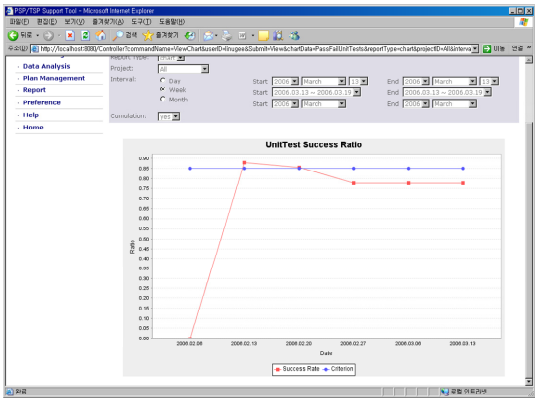
소프트웨어 크기를 자동으로 수집하기 위해서는 LOCC 와 같은 코드 카운팅 도구의 실행 결과를 자동으로 수집한다. 개발자는 직접 코드 카운팅 도구를 실행하거나 주기적으로(예, 날마다) 실행되도록 설정할 수 있다. LOCC 는 두 가지 모드로 실행될 수 있는데 하나는 소프트웨어 전체 크기를 수집하는 것이고, 다른 하나는 추가, 변경된 코드의 크기를 수집하는 것이다. 추가, 변경된 코드의 크기를 얻기 위해서는 소스 코드의 형상관리를 필요로 한다.

5. 메트릭 분석

수집된 메트릭의 분석을 위해 본 도구는 다양한 차트 분석과 식스 시그마 분석을 제공한다. 제공되는 차트에는 누적된 개발 시간(그림 3), 유닛 테스트 성공과 실패 개수, 유닛 테스트 성공률(그림 4), 소프트웨어 크기 변화 등이 있다. 그림 3 는 시간 기록 로그를 분석하여 총 사용 시간을 보여주는 차트이고 그림 4 은 개발자의 유닛 테스트 성공률과 조직 또는 프로젝트 내에서 정해진 표준 유닛 테스트 성공률과의 비교를 보여주는 차트이다.

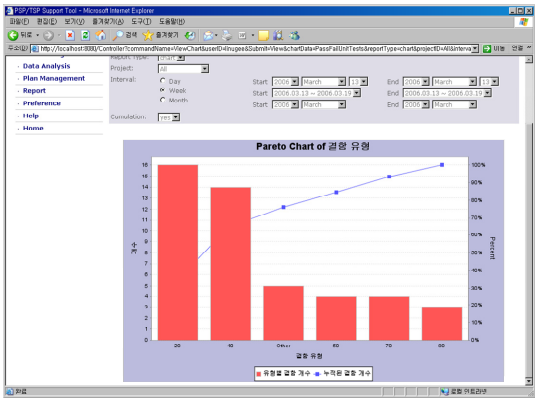


(그림 3) 누적된 개발 시간 차트



(그림 4) 유닛 테스트 성공률 차트

식스 시그마 분석[8]에는 그림 3 과 같은 파레토 차트(Pareto Chart), 관리도(Control Chart) 등이 포함된다. 그림 5 은 결함 개수와 결함 유형 데이터를 파레토 차트에 적용한 결과로써 자주 발생하는 결함 유형들을 분석하는 데 도움을 준다.



(그림 5) 파레토 차트

6. 결론

개인 레벨의 메트릭 수집 및 분석 활동에 있어서 신뢰성 있는 메트릭 데이터의 획득의 어려움과 분석 기법의 결여로 인한 의미 있는 분석의 어려움이 주요 문제점들로 인식되어 왔다. 본 논문에서는 이러한 문제점들을 해소하기 위해 개발된 메트릭 자동 수집 및 분석 도구를 설명하였다. 제안된 도구의 사용을 통해 신뢰성 있는 메트릭 데이터의 획득과 이러한 신뢰성 높은 메트릭 데이터를 바탕으로 의미 있는 분석을 할 수 있고, 더 나아가서는 메트릭 수집에 드는 비용의 감소와 위험 요소의 조기 파악을 가능케 할 수 있을 것이다.

향후 연구로서 메트릭 자동 수집 및 분석 기법의 확장이 요구된다. 메트릭 자동 수집 기법의 확장에는 코드 리뷰 도구, 웹브라우저 사용 등 파일 수정을 수반하지 않는 활동에 사용되는 시간을 자동으로 수집하는 연구가 포함될 것이다. 분석 기능의 확장에는 수집된 데이터에 대해 보다 다양한 식스 시그마 분석 기법의 적용과 두 개 이상의 메트릭을 함께 분석하는 기능이 포함될 것이다. 또한, 개인 레벨에서 수집된 데이터의 분석을 바탕으로 팀 레벨에서의 메트릭 수집 및 분석 활동을 지원해 주는 도구로의 확장도 중요한 연구 분야가 될 것이다.

참고문헌

[1] W. S. Humphrey, PSPSM: A Self-Improvement Process for Software Engineers, Addison-Wesley, 2005.
[2] P. M. Johnson, H. B. Kou, J. M. Agustin, C. Chan, C. A. Moore, J. Miglani, S. Zhen, and W. E. Doane. Beyond the personal software process: Metrics collection and analysis for the differently disciplined. In Proceedings of the 2003 International Conference on Software Engineering, Portland, Oregon, May 2003.
[3] P.M. Johnson, A.M. Disney, “A critical analysis of PSP data quality: Results from a case study”, Journal of Empirical Software Engineering, December 1999.
[4] Eclipse, http://www.eclipse.org/
[5] JUnit, http://www.junit.org/index.htm
[6] LOCC, http://csdl.ics.hawaii.edu/Tools/LOCC
[7] Sillitti A., Janes A., Succi G., Vernazza T., “Collecting, Integrating and Analyzing Software Metrics and Personal Software Process Data”, EUROMICRO 2003, Belek-Antalya, Turkey, 1 - 6 September 2003.
[8] Pete Pande, Larry Holpp, What is Six Sigma?, McGraw-Hill, 2002.
[9] J. McGarry, D. Card, et al., Practical Software Measurement, Addison Wesley, 2002.
[10] M. B. Chrissis, M. Konrad and S. Shrum, CMMI - Guidelines for Process Integration and Product Improvement, Addison-Wesley, 2003.