

# 개인 소프트웨어 프로세스 지원을 위한 도구

## (A Tool to Support Personal Software Process)

신현일<sup>†</sup> 정경학<sup>\*\*</sup> 송일선<sup>\*\*</sup> 최호진<sup>\*\*\*</sup> 백종문<sup>\*\*\*</sup>  
 (Hyunil Shin) (Kyounghak Jung) (Ilsun Song) (Hojin Choi) (Jongmoon Baik)

**요약** 개발자 개인의 소프트웨어 개발 프로세스를 개선시켜 소프트웨어의 품질을 향상시킬 수 있도록 돕는 기법으로 PSP(Personal Software Process)가 널리 쓰이고 있다. PSP에 제시된 측정 및 분석 활동을 지속적으로 수행함으로써 개별 개발자는 자신의 개발 프로세스에 내재된 약점을 파악할 수 있고, 이렇게 수집된 과거 프로젝트의 데이터를 이용하여 공수와 품질에 대한 예측의 정확도를 높일 수 있다. 그러나 수동으로 행해지는 데이터 수집의 오버헤드와 개발작업-측정작업 간의 문맥전환에 따른 집중력 분산의 문제점으로 인해 신뢰도 높은 데이터를 수집하기가 쉽지 않은 것이 현실이다. 한편, PSP에 제시된 문서형태의 프로세스 가이드는 프로세스 정보 검색의 불편함과 추가적인 정보를 삽입하는 데 어려움을 가지고 있다. 본 논문에서는 이러한 문제점들을 해결하기 위해 개발된 PSP 지원도구를 소개한다. 개발된 도구는 데이터 수집의 신뢰성을 높이기 위해 데이터 자동 수집 기능을 제공하고, PSP 프로세스 정보의 효율적인 검색을 위한 EPG(Electronic Process Guide) 기능 및 추가적인 프로세스 정보의 저장을 위한 경험 저장소 기능을 제공한다.

**키워드** : PSP, PSP 지원도구, EPG, 경험 저장소, 데이터 자동 수집

**Abstract** The PSP (Personal Software Process) is developed to help developers make high-quality products through improving their personal process. With consistent measurement and analysis activity that the PSP suggests, developers can identify process deficiencies and make reliable estimates on effort and quality. However, due to the high-overhead and context-switching problem of manual data recording, developers have difficulties in collecting reliable data, which can lead wrong analysis results. On the other hand, the paper-based process guides of the PSP are inconvenient to navigate its process information and difficult to attach additional information. In this paper, we introduce a PSP supporting tool developed to handle these problems. The tool provides automated data collection facilities to help acquire reliable data, an EPG (Electronic Process Guide) for the PSP to provide easy access and navigation of the process information, and an experience repository to store development experience as additional information about the process.

**Key words** : Personal Software Process, PSP supporting tool, Electronic Process Guide, Experience Repository, Automated Data Collection

## 1. 서론

소프트웨어 개발 프로세스의 개선은 조직이나 프로젝트

차원에서뿐만 아니라 팀이나 개인 차원에서도 소프트웨어 품질 향상을 위한 효과적인 방안이다. 개발자 개인의 소프트웨어 개발 프로세스를 개선시켜 소프트웨어의 품질을 향상시킬 수 있도록 돕는 기법으로 PSP[1]가 최근에 널리 쓰이고 있다. PSP에서 제시되는 방법에는 정의되고 측정 가능한 프로세스, 과거 프로젝트 데이터에 기반한 소프트웨어 크기 및 개발시간 예측, 설계 및 코드 검토, 프로세스 품질 측정, 프로젝트 계획 및 획득가치(Earned Value)를 이용한 계획 추적 등이 포함된다.

PSP가 제시하는 다양한 방법들 중에서 측정 및 분석의 지속적인 수행을 통해 개별 개발자는 자신의 개발

· 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 지원 사업의 연구결과로 수행되었습니다(IITA-2006-(C1090-0603-0032)).

† 정 회 원 : 한국정보통신기술협회 시험인증연구소  
shi@tta.or.kr

\*\* 학생회원 : 한국정보통신대학교 공학부  
aimania@icu.ac.kr  
iseeyou@icu.ac.kr

\*\*\* 종신회원 : 한국정보통신대학교 공학부  
hjchoi@icu.ac.kr  
jbaik@icu.ac.kr

논문접수 : 2007년 4월 25일  
심사완료 : 2007년 6월 28일

프로세스에 내재된 약점을 파악할 수 있고, 파악된 약점을 해결할 수 있는 프로세스 개선 계획을 세울 수 있다. 또한 이렇게 수집된 과거 프로젝트의 데이터에 기반하여 소프트웨어의 품질과 개발 공수에 대한 예측의 정확도를 높일 수 있다. 이러한 이득을 얻기 위해서는 신뢰도 높은 데이터의 수집이 중요하지만, 수동으로 행해지는 데이터 수집의 오버헤드와 개발작업-측정작업 간의 문맥전환(Context Switching)에 따른 집중력 분산의 문제점으로 인해 신뢰도 높은 데이터를 수집하기가 쉽지 않고, 이로 인해 잘못된 데이터 분석 결과가 유도될 수 있다[2,3]. 이러한 문제점들은 데이터 자동 수집 도구를 이용하여 해결할 수 있다. 그러나 모든 데이터가 자동으로 수집될 수 없으므로 수동 수집도 함께 지원될 필요가 있다. 수동 데이터 수집은 여전히 문제가 될 수 있지만, 데이터 자동 수집을 통해 수동 입력을 필요로 하는 데이터의 양이 줄어들므로 데이터 품질 문제를 상당히 해결할 수 있다.

PSP가 제시하는 다양한 방법들을 개발자가 익히는 데 도움을 주기 위해 PSP는 점진적으로 발전하는 6개의 프로세스(PSP0, PSP0.1, PSP1, PSP1.1, PSP2, PSP2.1)를 제시한다. 각 프로세스는 이전 프로세스의 내용을 모두 포함하면서 새로운 프로세스 단계 또는 방법들을 제시한다. 각 프로세스의 수행을 도와주기 위해 프로세스 스크립트, 템플릿, 체크리스트 등의 자료들이 문서형태로 제공되는데 이 자료들은 문서형태의 프로세스 가이드로 보여질 수 있다. 이 문서형태의 프로세스 가이드는 일반적으로 필요한 프로세스 정보 검색의 불편함과 추가 프로세스 정보 삽입 또는 기존 정보 수정의 어려움을 가지고 있는데 이러한 문제점들은 웹 기술을 이용한 EPG를 통해 해결될 수 있다[4]. PSP 프로세스 정보의 효율적인 검색과 추가적인 정보를 삽입할 수 있게 하기 위해서 PSP 프로세스 가이드를 위한 EPG 개발이 요구된다.

본 논문에서는 앞서 언급된 PSP 수행의 어려움을 감소시키기 위해 개발된 자동 데이터 수집 도구를 설명한다. Jasmine은 데이터 자동 수집을 제공하면서 개인 프로세스 및 품질 관리 지원하고 PSP 활동과 산출물에 대한 가이드를 위한 EPG와 개발경험 저장소를 위한 경험 저장소를 제공한다.

본 논문은 다음과 같이 구성된다. 2절에서는 센서 기반 데이터 자동 수집, EPG와 경험 저장소에 대한 배경을 간략히 소개하고 3절에서 Jasmine 도구의 아키텍처와 주요 기능을 기술한다. 4절에서는 기존 PSP 지원 도구와 Jasmine의 비교 설명하고 5절에서는 결론 및 향후 연구에 대해서 기술한다.

## 2. 연구 배경

### 2.1 센서 기반 데이터 자동 수집

높은 오버헤드나 문맥전환과 같은 수동 데이터 수집의 어려움을 해결하기 위해, 센서 기반으로 PSP 데이터를 자동으로 수집하고 수집된 데이터에 대해서 다양한 분석을 제공하는 Hackstat[2,5]와 같은 도구들이 개발되어 왔다. 이러한 도구들은 센서의 설치 및 구성에만 개발자의 노력을 필요로 하며 데이터 수집은 전적으로 센서에 의해서 수행된다. Eclipse, Microsoft Office, JBuilder 등의 개발 관련 도구에 센서들이 플러그인 형태로 부착되어 작업시간, 컴파일 에러와 같은 데이터 자동 수집을 담당한다. 센서는 개발 관련 도구에서 발생하는 이벤트(예, 파일 크기 변화, 마우스 및 키보드 이벤트, 단위 테스트의 실행)를 모니터링 함으로써 필요한 정보를 수집하고 수집된 데이터를 서버로 전송한다. 서버는 수집된 데이터를 데이터베이스에 저장하고 차트나 표를 이용한 다양한 분석을 개발자에게 제공한다.

Hackstat에서 자동으로 수집되는 시간, 결함 데이터는 PSP가 요구하는 데이터와 차이가 있다. 이 차이점은 모든 데이터가 자동으로 수집될 수 없다는 점과 자동으로 수집된 데이터가 필요한 모든 정보를 가질 수 없다는 점에서 비롯된다. 예를 들어, 소스코드, 설계 문서 등 소프트웨어 산출물의 작성 및 수정 활동에 사용된 시간을 자동으로 수집할 수는 있지만 회의, 설계 또는 코드 검토에 사용된 시간을 자동으로 수집할 수는 없다. 또한, 자동으로 수집된 시간 데이터가 어느 개발 단계에 포함된 것인지도 자동으로 판단하기가 어렵다. 결함 데이터는 JUnit과 같은 단위 테스트 메커니즘 또는 Bugzilla와 같은 버그 리포팅 시스템에 부착된 센서에 의해서 자동 수집된다. 그러나 이러한 방법으로는 설계/코드 검토에서 발견된 결함을 자동으로 수집할 수 없고, 자동 수집된 결함 데이터에는 결함 제거에 사용된 시간, 이 결함이 삽입된 개발 단계, 결함 종류를 포함하지 못한다.

### 2.2 EPG와 경험 저장소

프로세스 가이드는 프로세스 수행자들에게 주어진 프로세스의 이해와 수행을 돕기 위해 프로세스 정보와 기타 유용한 정보를 제공해 주는 참고 자료이다[4]. 프로세스 가이드가 제공하는 기본적인 정보에는 프로세스 활동, 산출물, 역할과 이러한 것들의 관계에 대한 상세한 설명이 포함된다. 프로세스 가이드는 프로세스 지식의 전파가 중요한 프로세스 개선 활동에서 매우 필요시 된다. 과거에는 프로세스 가이드가 문서형태로 제공되었는데 문서형태의 프로세스 가이드는 사용성에서 많

은 문제점을 가지고 있다[4]. 문서형태의 특성상 프로세스 정보가 특정 기준에 의해 차례대로 나열되어 있기 때문에 필요한 정보를 검색하는데 불편하고, 관련 있는 정보(예, 활동과 이 활동의 입력 및 출력 산출물)를 함께 묶을 수 없는 어려움을 가지고 있다. 또한 새로운 프로세스 정보의 추가 또는 기존 정보의 수정은 새로운 프로세스 가이드의 출판을 요구하므로 문서형태 프로세스 가이드의 유지보수에 높은 비용이 든다.

문서형태의 프로세스 가이드가 가지는 이러한 문제점들은 웹 기술을 이용하여 프로세스 정보를 제공하는 EPG에 의해서 해결될 수 있다[4,6]. 그러나 프로세스 가이드를 웹에서 다운로드 할 수 있게 PDF, Microsoft Word, 또는 다른 전자 파일로 제공하거나 프로세스 정보를 웹 브라우저에서 볼 수 있도록 단순히 HTML로 제공하는 것은 EPG로 취급되지 않는다. [4]에서 EPG가 만족시켜야 되는 기본 요구사항이 제시된다.

- EPG는 잘 만들어진 문서형태 프로세스 가이드가 제공하는 모든 정보를 포함해야 한다.
  - 각각의 웹 페이지는 프로세스 수행자가 한 눈에 쉽게 이해할 수 있을 만큼의 프로세스 정보를 포함하는 것이 요구된다.
  - 프로세스 정보의 효율적인 검색을 위해 하이퍼링크(Hyper-links)의 적절한 사용과 프로세스 흐름을 보여주는 다이어그램이 요구된다. 또한 관련 있는 정보들(예, 활동과 이 활동과 관련된 산출물)은 하이퍼링크를 통해 서로 연결되어야 한다.
  - 개발자가 EPG를 쉽고 간단히 사용할 수 있도록 하기 위해 모든 페이지는 비슷한 구조로 되어 있어야 한다.
- 제시된 기본적인 요구사항 외에 EPG는 산출물의 예제, 주석과 같은 추가적인 프로세스 정보를 포함할 수 있는데, 이러한 정보의 관리를 위해 EPG는 경험 저장소와의 통합을 이끌었다. 경험 저장소는 개발 관련 경험 및 지식의 저장, 구조화, 재사용 및 공유에 사용되는 소프트웨어 도구로써 과거 프로젝트의 경험 및 지식인 현재 프로젝트 문제점의 해결에 중요한 자원이 된다는 지식 및 경험 관리 관점에서 중요한 역할을 담당한다 [7,8].

EPG와 경험 저장소를 통합한 몇몇 연구가 있어 왔다. [8,9]에서 EPG와 경험 저장소의 결합된 시스템의 구현 및 중소기업에서의 성공적인 적용 사례가 제시된다. 수집된 다수의 경험 데이터에 사용자들이 쉽게 접근할 수 있도록 하기 위해 이 결합된 시스템에서 각각의 경험 데이터는 관련 프로세스 요소에 첨부된다. EPG와 경험 저장소의 결합은 [7]에서도 언급되는데, [7]은 경험 저장소가 갖추어야 하는 품질 요구사항을 제시하고 있

다. 이 중 하나가 점점 늘어나는 경험 데이터에 사용자가 쉽게 접근할 수 있도록 경험 저장소는 관련 프로세스에 따라서 조직되어야 한다는 것이다.

### 3. Jasmine 아키텍처 및 주요 기능

이 절에서는 PSP 프로세스 수행을 도와주기 위해 개발된 Jasmine 도구를 설명한다. 그림 1과 같이 Jasmine은 PPMT(Personal Process Management Tool)와 PSPG/ER(PSP Guide/Experience Repository)로 이루어져 있다. PPMT는 데이터 수집의 오버헤드와 문맥 전환을 감소시키기 위한 자동 데이터 수집과 차트, 그래프 형태로 다양한 데이터 분석 결과를 제공하고 프로젝트 계획, 획득가치를 이용한 프로젝트 추적을 지원한다. 또한, PSPG/ER는 PSP 프로세스 정보의 쉬운 접근을 위해 PSP 프로세스 가이드의 EPG 버전과 추가적인 프로세스 관련 정보를 수집할 수 있도록 경험 저장소를 제공한다. 경험 저장소는 개발자들이 산출물의 예, 교환, 소스코드 예제, 기타 유용한 정보 등의 개발경험을 저장할 수 있게 하고 저장된 개발경험을 공유할 수 있게 한다.

#### 3.1 PPMT

PPMT는 그림 1과 같이 서버-클라이언트 구조로 되어 있다. 클라이언트는 데이터 자동 수집을 담당하는 센서들로 이루어져 있고, 데이터 자동 수집을 제외한 모든 기능을 수행하는 서버는 웹 브라우저를 통해 사용자와 상호작용하는 웹 애플리케이션으로 개발되었다. PPMT의 주요 컴포넌트는 다음과 같다.

- 센서: 개발 관련 도구에 플러그인 형태로 부착되어서 필요한 데이터를 자동으로 수집하고 수집된 데이터를 PPMT Client로 전송한다.
- PPMT Client: PPMT Client의 주요 기능은 센서에서 받은 데이터를 PPMT Server로 전송하는 것이다. 또한 PPMT Client는 서버와의 연결이 끊졌을 경우 센서 데이터를 임시 보관하고, 필요한 경우 센서 데이터를 전처리(pre-processing) 하는 역할을 담당한다. 모든 센서들이 공통적으로 수행해야 하는 이러한 기능을 PPMT Client가 처리함으로써 센서 개발을 용이하게 한다.
- PPMT Server: PPMT Server는 데이터 자동 수집을 제외한 기능들-데이터 저장 및 분석, 데이터 수동 입력, 계획 및 획득가치 추적 지원, 사용자/프로젝트 관리-을 수행한다. 서버는 자바 기반(자바 서블릿, JSP, 자바빈즈)으로 구현되었으며 서블릿과 JSP 실행을 위해 Apache Tomcat을 이용한다.
- Database: 데이터베이스는 센서 데이터, 시간 및 결합, 작업 및 스케줄 계획 등의 수동으로 수집된 데이터, 사용자/프로젝트 관련 정보를 저장한다. 현재 구현

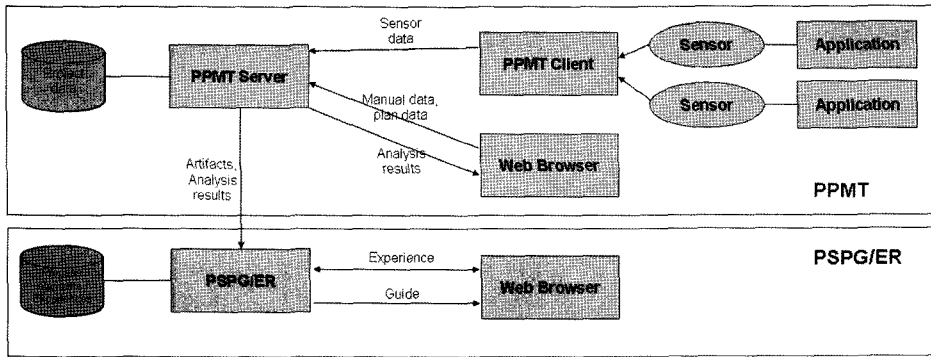


그림 1 Jasmine 아키텍처

에서는 MySQL이 데이터베이스로 사용되고 있다.

센서, PPMT Client, PPMT Server 간에 센서 데이터 전송을 위해 XML 포맷을 이용하고 있다. 센서는 하나의 프로그래밍 언어가 아닌 다양한 언어로도 개발될 수 있으므로, 센서 데이터는 특정 언어에 종속적이지 않는 XML의 포맷으로 작성될 필요가 있다. PPMT에서 제공되는 주요 기능들은 다음 절에서 자세히 설명한다.

3.1.1 센서 기반 데이터 자동 수집

시간, 결합, 소프트웨어 크기 데이터 기록을 돕기 위해 PPMT는 센서 기반 데이터 자동 수집을 제공한다. 자동 수집된 시간, 결합 데이터는 각각 시간, 결합 일지의 항목으로 기록되는데 이는 앞서 언급한 것처럼 자동 수집된 시간, 결합 데이터는 PSP가 요구하는 모든 정보를 포함하지 않기 때문에 개발자가 필요한 정보, 이를테면 작업시간이나 결합일지를 추가하거나 기존 데이터를 수정할 수 있게 하기 위함이다.

개발자가 작성 및 수정하는 소프트웨어 산출물 또는 개발자가 사용하는 소프트웨어 도구를 계속적으로 모니터링 함으로써 소스코드 작성, 수동 테스트, 설계문서 작성, 코드 검토 등에 사용된 시간을 자동 수집할 수 있다. 현재 구현에서는 소스코드 작성, 윈도우즈 애플리케이션과 웹 애플리케이션의 수동 테스트에 사용된 시간의 자동 수집이 지원된다. 소스코드 작성에 사용된 시간은 소스코드 파일의 크기를 계속적으로 모니터링 함으로써, 수동 테스트에 사용된 시간은 해당 애플리케이션에서 발생하는 마우스 또는 키보드 이벤트를 모니터링 함으로써 자동 수집된다. 현재 구현에 포함된 센서는 다음과 같다.

- Eclipse 센서: 자바 소스 파일의 크기를 주기적으로 (예 30초) 모니터링 함으로써 소스코드 작성에 사용된 시간을 자동으로 수집한다. 또한 Eclipse에서 개발자가 실행한 윈도우즈 애플리케이션에 마우스 또는 키

보드 이벤트를 감지하는 윈도우즈 후킹(Windows Hooking)[10] 모듈을 삽입하여 실행된 애플리케이션의 수동 테스트에 사용된 시간을 자동으로 수집한다.

- IE 센서: Internet Explorer에서 발생하는 마우스, 키보드 이벤트를 감지하여 웹 애플리케이션 수동 테스트에 사용된 시간을 자동으로 수집한다. 웹 애플리케이션은 "http://localhost:8080/"과 같은 특정 사이트 내에서 실행되므로 사용자가 지정한 사이트를 가진 Internet Explorer에서 발생하는 이벤트만을 자동으로 수집한다.

이렇게 수집된 시간 센서 데이터는 다음과 같은 방법으로 시간 일지에 기록된다. 사용자가 설정 가능한 간격(예, 5분) 동안에 최소한 하나의 시간 데이터(파일 크기 변경, 마우스 또는 키보드 이벤트)가 있으면 이 간격 동안 개발자가 작업을 했던 것으로 간주한다. 이러한 간격들이 연속적이면 이 연속된 시간들을 하나의 항목으로 합쳐서 시간 일지에 기록한다. 시간 일지에 기록된 시간 데이터가 어떠한 작업과 관련된 것인지를 개발자가 파악하는데 도움을 주기 위해 수정된 파일 또는 사용된 도구에 대한 정보를 함께 기록한다.

결합은 단위 테스트, 버그, 컴파일 에러 등을 수집함으로써 자동 수집 된다. 현재 버전에서는 실패한 단위 테스트, 컴파일 에러, 런타임 에러를 자동으로 수집하여 결합 일지의 항목으로 기록한다. 표 1에 나타나 있듯이 수집된 결합 데이터에 대해서 제거 단계, 상세 설명, 결합 종류, 발견된 날짜 정보가 자동으로 결합 일지에 기록된다. 또한 보다 다양한 결합 분석을 할 수 있도록 결합이 발생된 소스 파일 정보를 추가로 기록한다. 자동으로 수집된 결합 데이터를 보여주는 결합 일지 화면의 일부가 그림 2에 나타나 있다. 현재 버전에서는 JUnit을 이용한 단위 테스트의 결과, 자바 컴파일 에러, 자바 프로그램 실행 중에 발생하는 자바 예외(exception)의 수집을 Eclipse 센서가 담당하고 있다.

표 1 실패한 단위 테스트, 컴파일 에러, 런타임 에러의 결함 정보

	Failed unit tests	Compile errors	Runtime errors
Remove phase	Test	Compile	Test
Description	The stack trace of the exception	The description of the syntax error	The stack track of the exception
Defect type	The exception type	Syntax	The exception type
Found date	(automatic)	(automatic)	(automatic)
Inject phase	(manual)	(manual)	(manual)
Fix time	(manual)	(manual)	(manual)
Resource	The test source file, the failed unit test method	The source file	The source file

<< Data View - View DefectLog

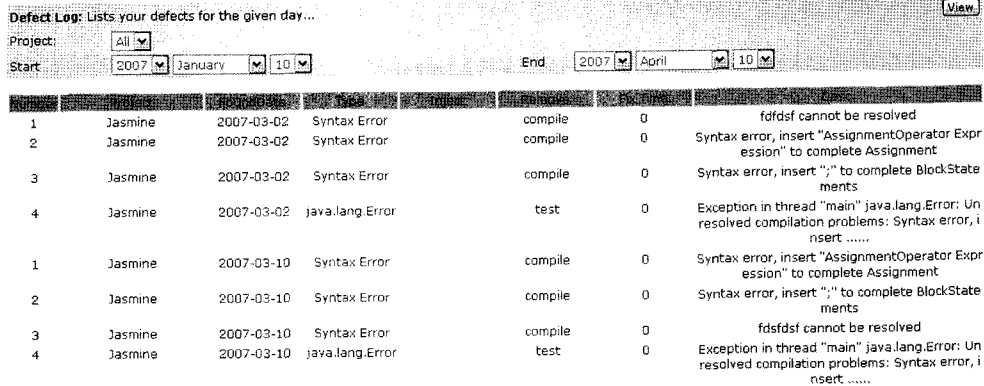


그림 2 결함 일지

소프트웨어 크기는 라인 카운팅 도구에 의해 LOC (Lines Of Codes)로 측정되어 자동 수집될 수 있다. 현재 버전은 LOCC[11]로 측정된 LOC결과를 자동으로 수집한다. 전적으로 센서에 의해서 수집되는 시간, 결함과 달리 소프트웨어 크기 수집을 위해서는 개발자가 라인 카운팅 도구를 실행해야 하는데 이를 위해 Eclipse 센서에서 그림 3과 같이 Eclipse 프로젝트에 대해서 라인 카운팅 도구를 실행할 수 있게 하는 메뉴를 제공하고 있다.

3.1.2 데이터 수동 입력 지원

PPMT는 자동으로 수집되지 못하는 설계/코드 검토 시간, 설계 검토에서 발견된 결함 등의 데이터를 기록하거나 기존 데이터를 수정할 수 있게 하는 기능을 제공한다.

3.1.3 계획 및 획득가치 추적 지원

PSP에서 개발자들은 계획단계에서 작업 및 스케줄 계획을 세우고 획득가치를 이용하여 프로젝트 진행상황을 추적하는 것이 요구된다. 이러한 활동을 도와주기 위해 PPMT는 작업 및 스케줄 계획 템플릿 작성을 위한 품을 제공하고, 수집된 시간 일지의 데이터와 계획 데이터를 바탕으로 획득가치를 자동으로 계산한다. 그림 4에 작성된 획득 가치 추적의 예제가 나타나 있다.

3.1.4 데이터 분석 및 리포트 생성

PPMT는 개발자의 요청에 따라 수집된 데이터에 대해 차트 또는 표 형태로 다양한 데이터 분석과 분석 결과를 요약해서 보여주는 리포트를 제공한다. 제공되는 분석에는 시간 경과에 따른 데이터의 변화를 보여주는 경향차트(Trend Chart), 계획가치(Planned Value)와 획득가치를 보여주는 획득가치차트, 결함 분석을 위한 파레토 차트(Pareto Chart)가 있다. 또한 단계 결함률(Phase Yield), A/FR(Appraisal to Failure Ratio) 등의 품질 측정변수(Quality Measure) 분석이 제공된다.

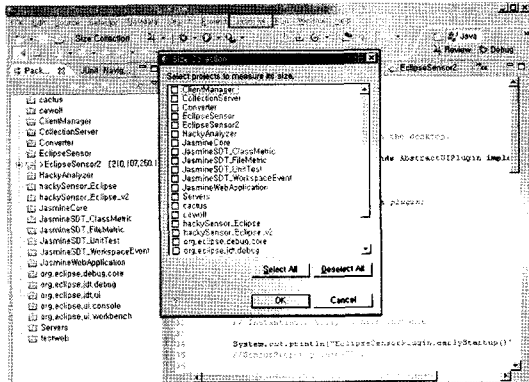


그림 3 크기 수집 메뉴

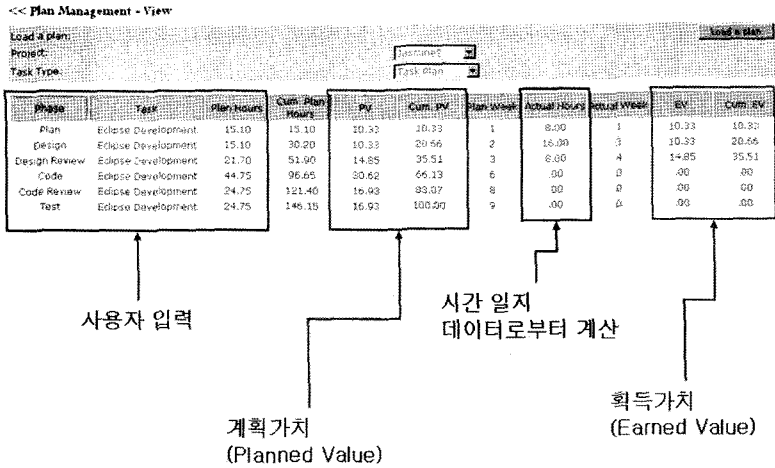


그림 4 획득 가치 추적

<< Data Analysis - Pareto Chart for Defects

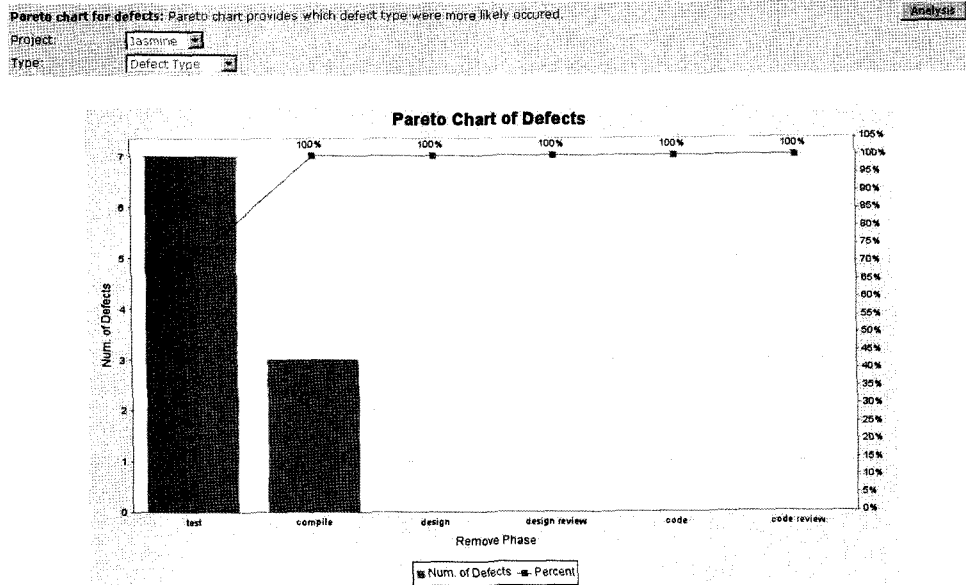


그림 5 결함 분석 파레토 차트

그림 5에 결함 단계별 결함 개수를 보여주는 파레토 차트가 나타나 있다. 생성되는 리포트에는 프로젝트의 주간 데이터를 요약한 주간 리포트, 프로젝트 전체 기간 동안의 프로젝트 데이터를 요약한 프로젝트 리포트가 있다.

3.2 PSPG/ER

PSPG/ER은 개인 개발자들이 개발 과정에 있어서 필요한 자료들(프로세스 스크립트, 템플릿, 체크리스트 등)을 보다 쉽게 검색하고, 추가적인 정보의 삽입이나 변경을 용이하게 하기 위해, PSP활동에 대한 가이드와 개발

경험을 저장하는 경험 저장소를 제공하는 도구이다. PSPG/ER이 제공하는 주요 PSP요소에는 PSP활동(계획, 설계, 설계 검토 등), 산출물(작업 및 스케줄 계획, 프로젝트 계획 요약 등), PSP 프로세스(PSP0, PSP0.1, PSP1 등)가 포함된다.

그림 6은 PSPG/ER의 전체적인 구조를 도식화한 다이어그램이다. 메인 페이지에서는 PSP프로세스 목록이 나열되고, 이 중 특정 프로세스를 선택하면 선택된 프로세스와 관련한 활동 페이지가 열린다. 활동 페이지에서는 해당 활동의 명세서 및 개발과정에서 얻은 경험과

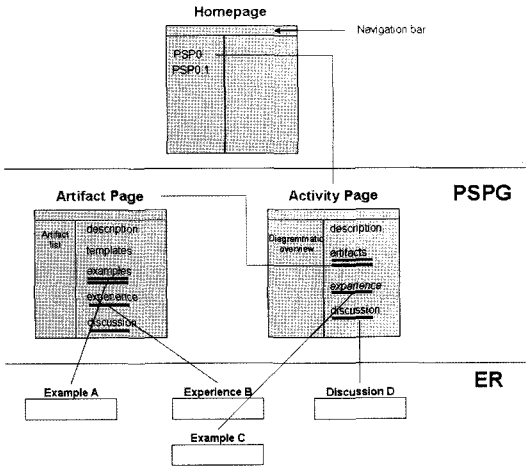


그림 6 PSPG/ER의 전체 구조

토의사항을 저장하거나, 산출물 목록 가운데 하나를 선택하여 산출물 페이지로 이동할 수 있다. 산출물 페이지에서는 전체 산출물의 리스트 및 템플릿, 예제 등을 저장하여 다른 개발자들과 산출물을 공유할 수 있다.

3.2.1 PSPG

그림 7에서와 같이 활동 페이지는 3개의 프레임(네비게이션 바, 다이어그램, 설명 영역)으로 나누어져 있다. 상단부의 네비게이션 바(Navigation Bar)는 현재 페이지의 위치를 계층적 구조로 알려주고, 자신이 원하는 상위 페이지로 빠르게 이동할 수 있게 해준다. 예를 들어 그림 7의 네비게이션 바에서 보여지는 HOME|PSP2.1|Development의 경우, 현재 페이지는 Development Script를 담고 있으며, 상위 페이지로 PSP2.1과 HOME이 위치하고 있다는 의미를 갖는다. 좌측 프레임은 프로세스 다이어그램을 이용하여 프로세스의 전체적인 흐름

**PSP2.1 Development Process**

Description

Step	Activities	Description
1	Design	- Review the requirements and produce an external specification to meet them. - Complete Functional and Operational Specification templates to record this specification. - Produce a design to meet this specification. - Record the design in Functional, Operational, State, and Logic Specification templates. - Record in the Defect Recording log any requirements defects found. - Record time in the Time Recording log.
2	Design Review	- Follow the Design Review checklist and review the design. - Fix all defects found. - Record defects in the Defect Recording log. - Record time in the Time Recording log.
3	Code	- Implement the design following the Coding standard. - Record in the Defect Recording log any requirements or design defects found. - Record time in the Time Recording log.
4	Code Review	- Follow the Code Review checklist and review the code. - Fix all defects found. - Record defects in the Defect Recording log. - Record time in the Time Recording log.
5	Compile	- Compile the program until error-free. - Fix all defects found. - Record defects in the Defect Recording log. - Record time in the Time Recording log.
4	Test	- Test until all tests run without error. - Fix all defects found. - Record defects in the Defect Recording log. - Record time in the Time Recording log. - Complete a Test Report template on the tests conducted and the results obtained.

Artifacts

- Design Review Checklist
- Code Review Checklist
- Test Report Template
- Time Recording Log
- Defect Recording Log
- Design Templates

← 산출물 페이지로의 링크

Experiences

Title	Author	Date
<a href="#">Worked into...</a>	Francisco	07.02.19
<a href="#">Status...</a>	Prakash	07.01.12

그림 7 활동 페이지

을 보여준다. 또한 현재 선택된 활동의 위치를 비롯하여 하이퍼링크를 이용해 관련 활동의 정보를 쉽게 참조할 수 있도록 하였다.

설명 영역에 해당하는 우측 프레임은 선택된 활동의 상세 설명, 관련 산출물 페이지로의 링크, 이 활동과 관련된 개발 경험으로의 링크 등을 포함하고 있다. 예를 들어, 그림 7에서는 PSP 2.1 Development Process의 Activity에 대한 내용(Design, Design Review, Code, Code Review, Compile, Test)이 상세 설명이 기술되어 있고, 테이블에 기재되어 있는 활동의 이름을 클릭하면 선택한 항목의 페이지로 바로 이동할 수 있다. 또한 산출물 페이지로 이동하려면 테이블 아래에 있는 산출물 목록 중 하나를 선택하면 된다.

그림 8은 그림 7의 산출물 목록 중 Design Review Checklist를 선택했을 때 나타나는 산출물 페이지의 일부이다. 활동 페이지와 같이 산출물 페이지도 3개의 프레임 구조로 이루어져 있다. 산출물 페이지는 활동페이지의 다이어그램과는 다르게, 좌측 프레임이 산출물의 리스트로 구성되어 있고, 우측 프레임은 개발자들의 경험을 저장할 수 있는 경험 저장소가 포함되어 있다.

3.2.2 ER

경험 저장소(Experience Repository: ER)는 각 사용

자가 개발 과정에서 얻은 개발경험을 저장하고, 저장된 경험들이 다른 개발자들과 공유될 수 있도록 해주는 기능을 포함한다. ER은 총 4개의 항목(템플릿, 예제, 경험, 토의)으로 이루어져 있다. 템플릿(Templates)은 각 활동에서 자주 사용되는 서식 및 도표의 기본 골격에 대한 자료를 저장하고 예제(Examples)는 해당 활동에 관련해 본보기가 될 만한 자료를 포함한다. 경험(Experiences)은 각 개발자가 개발과정에서 얻은 경험적 지식을 저장하고, 토의(Discussions)는 개발과정에서 직면한 문제를 해결하거나 더 나은 해답을 찾기 위해 혹은 다른 개발자들의 의견을 얻고자 할 때 이용한다.

그림 9는 개발자가 개발과정에서 얻은 새로운 산출물을 페이지에 저장하는 팝업 윈도우이다. 개발자가 산출물을 추가하고자 할 때에는 상세 설명 페이지에 위치한 add버튼을 클릭하면 된다. 버튼을 클릭하면 그림과 같이 필요한 항목을 입력할 수 있는 팝업 창이 나타난다. 산출물을 저장하기 위해 필요한 입력으로는 산출물의 종류, 제목, 내용, 첨부파일, 그리고 해당 산출물의 공개 여부이다. 산출물의 종류는 앞에서 설명한 것과 같이 4개의 항목(템플릿, 예제, 경험, 토의)중 현재 작성자가 추가하려고 하는 데이터의 종류를 선택하는 것이고, 산출물의 공개여부는 다른 개발자들과 해당 산출물을 공

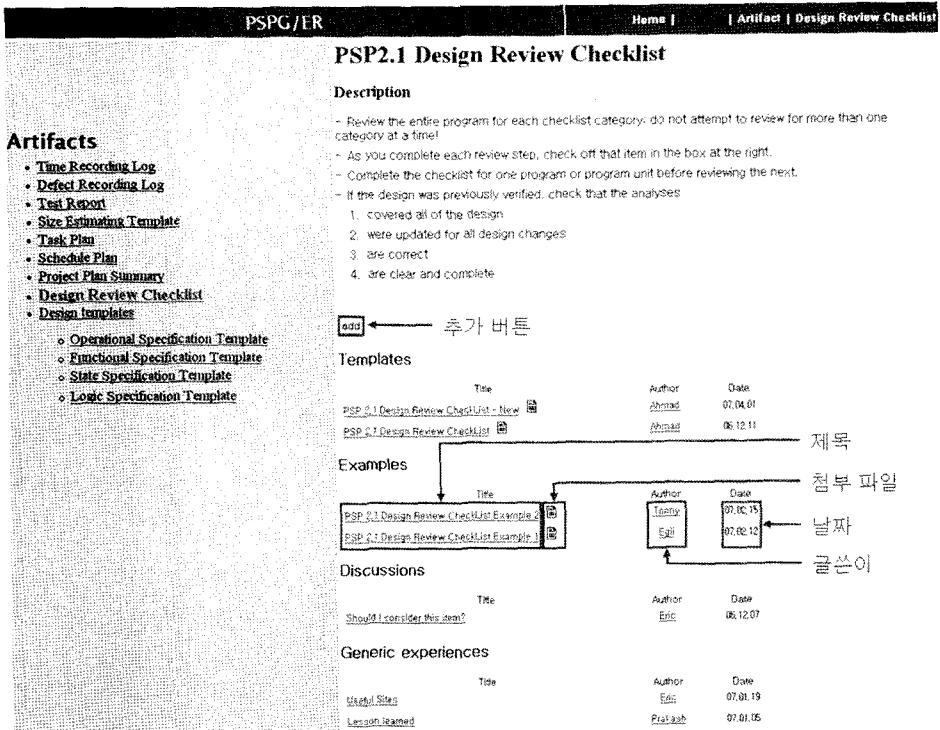


그림 8 산출물 페이지



그림 9 경험입력

유할 것인지를 선택하는 항목이다. 만일 작성자가 자신의 경험 데이터를 공유하고 싶지 않다면, 접근권한을 설정하는 Private 체크박스를 선택 표시하면 된다.

산출물의 저장이 끝나면 그림 8과 같이 상세 정보 페이지 하단부에 저장된 경험들이 테이블 형식으로 보여진다. 페이지에서 보여지는 산출물의 항목으로는 제목과 글쓴이, 그리고 산출물이 저장된 시간이다. 제목에는 저장된 산출물의 상세 정보를 확인할 수 있도록 해주는 링크가 연결되어 있고, 글쓴이의 이름에는 직접 메일을 보낼 수 있도록 설정되어 있다. 만일 글쓴이가 산출물을 저장하면서 파일을 첨부했다면 제목의 우측에 문서모양의 아이콘이 나타난다. 사용자는 이 아이콘을 선택함으로써 첨부파일을 다운로드 할 수 있다.

저장된 경험 데이터를 수정하고자 할 때에는 산출물의 제목을 선택하면 된다. 이 후, 팝업 윈도우의 각 폼에 입력되어 있는 기존의 내용 수정입력하고 확인버튼을 선택하면 데이터를 수정할 수 있다.

### 3.3 PPMT와 PSPG/ER의 상호작용

PPMT와 PSPG/ER의 주요 상호작용 기능 중의 하나는 PPMT에서 얻은 각종 산출물을 경험 저장소에 저장하는 것이다. PPMT에서 얻을 수 있는 산출물로는 시간, 결함 일지와 히스토리(history) 데이터 뿐만 아니라, 주간, 프로젝트 보고, 그리고 차트, 표와 같은 분석 결과가 될 수 있다. PPMT는 개발자가 선택한 자료를 파일로 만들어 경험 저장소에 저장할 수 있다. 이를 통하여 각 개발자는 좀 더 편리하게 개발경험을 저장하고, 다른 개발자들과 공유할 수 있게 된다.

또 다른 상호작용은 관련 페이지로의 링크를 제공하

는 것이다. 예를 들어 PSPG/ER의 시간 일지 산출물 페이지는 PPMT의 시간 일지 폼으로의 링크를 포함하고 있고 반대로 PPMT의 시간 일지 폼은 해당 산출물 페이지로의 링크를 포함한다. 이를 통해 개발자들은 필요한 프로세스 정보를 더욱 편리하게 찾을 수 있을 것이다.

## 4. 관련 연구와의 비교

PSP 데이터 수집과 분석을 원활하게 하도록 지원하기 위해 Process Dashboard [12], Hackstat, PSPA [13]와 같은 PSP 지원도구들이 지금까지 개발되어 왔다. 위의 도구들 가운데 데이터 수집 기능의 측면에서 Process Dashboard는 자동으로 PSP 데이터를 수집하는 기능이 없고, PSPA는 컴파일러 에러만을 자동으로 수집해주는 반면, Hackstat은 센서 기반으로 다양한 데이터의 자동 수집을 제공해준다. 따라서 기능적인 면에서 Hackstat이 Jasmine과 가장 유사하다. 하지만 Hackstat과 Jasmine의 근본적인 차이점은 각 도구가 지향하는 목적이 있다. Hackstat은 데이터 자동 수집 및 분석에만 초점을 맞추고 있기에 PSP 지원도구라기 보다는 데이터 자동 수집 및 분석 도구라고 볼 수 있다. 그러므로 Hackstat은 계획, 계획 추적, 추정(estimation)과 같은 PSP 활동을 지원하지 않고 데이터 분석에도 한계점이 있다. 이러한 단점은 Hackstat이 데이터 수동 입력을 지원하지 않는다는 점과 자동 수집된 데이터가 PSP가 요구하는 모든 정보를 가지고 있지 않다는 점에서 기인한다.

반면, Jasmine은 PSP의 전반적인 활동 지원을 목표로 한다. 첫째로 데이터 자동 수집과 함께 수동 입력을 모두 지원하고 있다. 자동 수집된 시간, 결함 데이터에 PSP가 요구하는 모든 정보를 개발자가 추가로 입력할 수 있도록 시간, 결함 센서 데이터를 시간, 결함 일지의 항목으로 변환한다. 이는 Hackstat에서 가능하지 않는 다양한 분석을 가능케 하는데 그 예로 결함 데이터의 파레토 분석(이 있다. 또한, 표 2에 나타나 있는 것과 같이 Jasmine은 수동 테스트에 사용된 시간, 컴파일 에러, 런타임 에러를 추가적으로 자동 수집한다. 마지막으로 Jasmine은 PSP 프로세스 가이드를 위해 EPG와 경험 저장소를 결합하여 제공한다.

## 5. 결론

본 논문에서 개발자들의 PSP수행 지원을 목적으로 개발된 Jasmine도구를 설명하였다. Jasmine은 수동 데이터 입력의 문제점들을 완화하기 위해 데이터 자동 수집을 제공할 뿐만 아니라 프로젝트 계획 및 추적을 지원하고 다양한 데이터 분석을 제공한다. 이러한 기능을

표 2 센서 기반 데이터 자동 수집 기능 비교

데이터		Jasmine	Hackstat
시간	소스코드 수정에 사용된 시간	Eclipse	Eclipse, Visual Studio, JBuilder, IntelliJ Idea
	문서 작성에 사용된 시간	X	Microsoft Office, OpenOffice, Emacs
	코드 검토에 사용된 시간	X	Jupiter
	수동 테스트에 사용된 시간	Internet Explorer (웹 어플리케이션), Eclipse (윈도우즈 어플리케이션)	X
결합	실패한 단위 테스트	JUnit	JUnit, CPPUNIT
	컴파일 에러	Eclipse	X
	런타임 에러	Eclipse	X
	배포 후 발생하는 버그	X	Bugzilla, Jira

통해 개발자들은 개인 프로세스의 약점을 파악하여 프로세스 개선 계획을 세울 수 있고 실천된 개선 계획이 올바르게 수행되었는지 또는 어떠한 효과를 가져왔는지를 파악할 수 있을 것이다. 또한 Jasmine은 PSP 프로세스 정보에 개발자들이 쉽게 접근할 수 있도록 PSP 프로세스 가이드의 EPG버전과 추가적인 프로세스 관련 정보를 저장할 수 있도록 경험 저장소를 제공한다. 경험 저장소와 통합된 EPG는 개발자들이 PSP 프로세스를 이해하고 수행하는데 도움을 줄 것이고 이 도구를 이용해 축적한 경험 데이터는 향후 프로젝트에서 발생하는 문제점을 해결하는 데 커다란 도움을 줄 것이다.

이 연구는 PSP/TSP 지원도구 개발을 목표로 하고 있는 프로젝트의 첫 번째 단계이다. TSP(Team Software Process) 지원이 향후 연구로써 계획되어 있다. 개발자뿐만 아니라 팀 관리자가 TSP를 원활하게 수행할 수 있도록 팀 데이터의 자동 수집 및 분석, 팀 계획 프로세스 및 팀 프로젝트 추적 지원, TSP를 위한 EPG를 향후에 개발하여 추가할 예정이다. 또한 다양한 개발 관련 도구(예, Visual Studio, Microsoft Office)를 위한 센서와 테스트 커버리지(test coverage), 소프트웨어 배포 후 발생한 버그, 코드 품질 메트릭 등을 수집하는 센서 개발이 향후 연구에 포함되어 있다. 또한, 보다 체계적인 데이터 분석 지원을 위해 관리도(control chart), 회귀분석(regression analysis) 등의 SixSigma 분석 기법을 통합할 예정이다. 마지막으로 실제 사용에 기반하여 도구를 개선하기 위해 학생 수업 또는 산업체 프로젝트에 적용을 계획 중에 있다.

참고 문헌

[1] W. S. Humphrey, "PSP(sm) : A Self-Improvement Process for Software Engineers," SEI Series in Software Engineering, Addison-Wesley Professional, 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] Disney, A. & Johnson, P., "Investigating Data Quality Problems in the PSP," Sixth International Symposium on the Foundations of Software Engineering (SIGSOFT'98), Orlando, FL., November, 1998.  
 [4] M. Kellner, U. Becker-Kornstaedt, W. Riddle, J. Tomal, M. Verlage," Process guides: effective guidance for process participants," in: Proc. of the Fifth International Conference on the Software Process, Chicago, IL, USA, June 1998, ISPA Press, 1998, pp. 11-25.  
 [5] Johnson, P.M.; Hongbing Kou; Agustin, J.M.; Qin Zhang; Kagawa, A.; Yamashita, T., "Practical automated process and product metric collection and analysis in a classroom setting: lessons learned from Hackstat-UH," International Symposium on Empirical Software Engineering, 2004.  
 [6] L. Scott, L. Carvalho, R. Jeffery, J. D'Ambra and U. Becker-Kornstaedt, "Understanding the use of an Electronic. Process Guide," Information and Software Technology 44. (10), 2002, pp. 601-616.  
 [7] Kurt Schneider, Jan-Peter von Hunnius, "Effective Experience Repositories for Software Engineering," icse, p. 534, 25th International Conference on Software Engineering (ICSE'03), 2003.  
 [8] Louise Scott, Lucila Carvalho, Ross Jeffery, "A Process-Centred Experience Repository for a Small Software Organisation," apsec, p. 603, Ninth Asia-Pacific Software Engineering Conference (APSEC'02), 2002.  
 [9] Felicia Kurniawati, Ross Jeffery, "The Long-term Effects of an EPG/ER in a Small Software Organisation," 2004 Australian Software Engineering Conference.  
 [10] Windows Hooking Mechanism, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/wiui/windowsuserinterface/windowing/hooks.asp>

- [11] LOCC, <http://csdl.ics.hawaii.edu/Tools/LOCC/>
- [12] Process Dashboard, <http://processdash.sourceforge.net/>
- [13] Raymund Sison, David Diaz, Eliska Lam, Dennis Navarro, Jessica Navarro, "Personal Software Process (PSP) Assistant," apsec, pp. 687-696, 12th Asia-Pacific Software Engineering Conference (APSEC'05), 2005.



백 종 문

1993년 조선대학교 컴퓨터과학 및 통계학과 학사. 1996년 미국 University of Southern California Computer Science 석사. 2000년 미국 University of Southern California Computer Science 박사. 2001년~2005년 미국 모토로라 SSERL (Software and System Engineering Research Lab) 수석연구원. 2005년~현재 한국정보통신대학교 공학부 조교수. 관심분야는 소프트웨어 메트릭스, 소프트웨어 비용추정, 소프트웨어 동적 모델링, 소프트웨어 6시그마



신 현 일

2001년 금오공과대학교 컴퓨터공학과(학사). 2007년 한국정보통신대학교 공학부(석사). 2007년~현재 한국정보통신기술협회 재직 중. 관심분야는 소프트웨어 프로세스, 테스트



정 경 학

2007년 단국대학교 전기전자컴퓨터공학부(학사). 2007년~현재 한국정보통신대학교 공학부 석사과정. 관심분야는 소프트웨어공학, 인공지능



송 일 선

2007년 한국기술교육대학교 메카트로닉스공학부(학사). 2007년~현재 한국정보통신대학교 공학부 석사과정. 관심분야는 소프트웨어공학, 인공지능



최 호 진

1982년 서울대학교 컴퓨터공학과 학사  
 1985년 영국 뉴캐슬대학교 컴퓨팅학과 석사. 1982년~1989년 ㈜메이콤 중앙연구소 선임연구원. 1995년 영국 임페리얼 공대 컴퓨팅학과 박사. 1995년~1996년 임페리얼공대 박사후연구원. 1997년~2002년 한국항공대학교 전자공학부 교수. 2003년~2003년 미국 카네기멜론대학교 방문교수. 2002년~현재 한국정보통신대학교 공학부 교수. 2003년~현재 카네기멜론대학교 겸임교수. 관심분야는 유비쿼터스컴퓨팅, 인공지능, 소프트웨어공학, 그리드컴퓨팅, 데이터마이닝