ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 45, No. 9, pp. 904-910, 2018. 9
https://doi.org/10.5626/JOK.2018.45.9.904

육군정보체계 관리를 위한 소프트웨어 개발 프로세스 가시화 사례

(A Practice of Software Development Process Visualization for Army Information System Management)

장 우 성 † 손 현 승 †† 김 영 철 ††† 이 종 훈 ††††
(Woo Sung Jang) (Hyung Seung Son) (R.Young Chul Kim) (Jong Hoon Lee)

요 약 현대의 소프트웨어 프로젝트 성공률을 높이기 위해서는, 제조업 같은 공정 프로세스가 필요하다. 특히 육군의 정보시스템은 정보체계 개발 및 산출물 관리가 다소 미흡하다. 따라서 군 자체 소프트웨어 개발 프로세스의 고급화와 요구사항 추적 자동화가 필요하다. 또한 유지보수에서도 보수 지연, 비용 초과, 품질저하 등의 개선이 요구된다. 이런 문제 해결을 위해, 오픈소스 기반의 프로세스 가시화, 아키텍처 가시화, 문서 가시화 중에 프로세스 가시화를 적용했다. 이를 통해, 프로젝트 관련 모든 이해관계자들에게 소프트웨어 개발 모든 과정의 투명성과 추적성으로 소프트웨어의 품질을 확보했다. 이는 품질 지표 및 결과 가시화 그리고 스케줄 관리 및 통제가 가능하여 군 소프트웨어를 고품질화를 지속시킬 수 있다.

키워드: 소프트웨어 가시화, 소프트웨어 프로세스 가시화, 소프트웨어 품질, NIPA, 프로젝트 관리, 지속적 통합

Abstract To increase the chance of success of the current software project, we need a software process that is like a manufacturing process. Specially, the army information system should focus on developing an effective information system and managing byproducts as a whole process. Moreover, maintenance will require a way to avoid maintenance delays, which increase a costs, and degrade quality. To solve this problem, we apply a process visualization mechanism for the Army's system, one of process, architecture, and documentation visualization in NIPA. We can guarantee a high quality of software that will provide transparency and traceability for all stakeholders in the life cycle. We expect to maintain high quality for the army's software with quality indicators, result visualization, scheduling control, and management.

Keywords: software visualization, software process, software quality, NIPA, project management, continuous integration

· 이 논문은 2017년도 정부(교육부) 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(NRF-2017R1D1A3B03035421)과 2018년도 정보통신산업진흥원의 정보통신, 방송 연구개발사업(개방형OS 환경개발 및 보급, 확산 사업)의 지원을 받아 수행된 연구임(S1113-18-1001)
· 이 논문은 제44회 한국소프트웨어종합학술대회에서 '고품질 소프트웨어를 위한 군인력 자원관리 개발 프로세스 가시화 구축 사례'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 홍익대학교 소프트웨어공학 연구실
jang@selab.hongik.ac.kr
†† 정 회 원 : 모아소프트 소프트웨어 개발팀 개발팀장
hson@moasoftware.co.kr
††† 정 회 원 : 홍익대학교 소프트웨어융합학과 교수(Hongik Univ.)
bob@hongik.ac.kr
(Corresponding author임)
†††† 비 회 원 : 육군본부 정보체계관리단 단장
mizzouri@naver.com

논문접수 : 2018년 2월 22일
(Received 22 February 2018)
논문수정 : 2018년 7월 10일
(Revised 10 July 2018)
심사완료 : 2018년 7월 10일
(Accepted 10 July 2018)

정보과학회논문지 제45권 제9호(2018. 9)

1. 서 론

현재, 개발된 소프트웨어들도 내재된 버그와 같이 여러 가지 잠재적인 위험이 존재한다. 소프트웨어의 잠재적인 위험은 납기지연, 비용초과, 품질저하 등의 문제를 야기할 수 있다. 이러한 문제의 해결을 위해, 체계적이고 정량적인 접근 방법이 필요하다[1].

대표적인 기존 문제 해결 방법으로써, ALM(Application Lifecycle Management)이 존재한다. ALM은 요구사항 정의부터 배포 및 유지관리에 이르기까지의 개발 단계를 정의한 프로세스 집합으로써, 각 단계를 모니터링, 제어, 추적, 문서화 한다. 하지만 기존 ALM 도구의 경우 도구의 높은 성숙도로 인해 프로젝트에 적용 시 상당한 경험과 지식을 필요로 하기 때문에 실제 사용에 어려움을 겪는다[2].

소프트웨어 프로세스 가시화[3,4]는 소프트웨어 요구사항 정의에서부터 개발, 테스트에 이르는 전체 과정을 가시화한다. 회사 내 개발 프로세스에 최적화된 가시화 환경을 구축하기 위한 상세한 단계를 제시한다. 결과적으로, 1) 요구사항에서부터 개발 및 테스트까지의 상호 추적성의 장점이 있고, 2) 소프트웨어 개발 과정에 요구사항 단계부터 설계 및 코드의 연관성을 사전에 감지할 수 있으며, 3) 소프트웨어의 품질 점수를 측정할 수 있는 정량적인 환경을 제공하고, 4) 오픈 소스 도구를 활용하여 환경구축비용을 절감한다.

본 논문은 현재의 미성숙된 군 지원 자원관리 시스템 개발을 위한 소프트웨어 프로세스 가시화 기법 적용 방법을 제안한다. 소프트웨어 품질의 가시화를 위해 현재 군의 소프트웨어 개발 조직에 적합한 품질 측정 항목을 선정한다. 요구사항 및 개발 진행 상황의 추적을 위한 가시화 환경을 구축한다.

본 논문의 구성은 다음과 같다. 2장은 관련 연구를 언급한다. 3장은 군 자원관리 시스템의 프로세스 가시화를 언급한다. 4장은 프로세스 적용 및 개선 사례를 언급한다. 마지막으로 결론을 언급한다.

2. 관련 연구

2.1 소프트웨어 가시화 기법

좋은 소프트웨어의 개발은 소프트웨어 개발 과정 전반에 대한 관리가 중요하다. 즉 소프트웨어 개발 프로세스에 대한 관리가 필요하다. 하지만 국내 중소기업의 경우, 인력 및 비용의 부족 때문에 수행이 어렵다.

소프트웨어 가시화 기법[1]은 소프트웨어의 시각화와 자동 문서화에 초점을 두어 인력 및 비용의 부담을 최소화하면서 소프트웨어 개발 품질 관리를 수행하는 특징을 가진다. 이는 중소기업의 소프트웨어 품질 관리 비용에 대한 부담을 덜어줄 수 있다. 소프트웨어 가시화 기법은 개발 프로세스의 시각화, 소스코드의 시각화, 소스코드의 문서화, 개발 프로세스의 문서화의 방향으로 구성된다. 본 논문은 소프트웨어 가시화 기법 중 개발 프로세스의 시각화 방법을 군 지원 자원관리 시스템에 적용한 사례이다.

2.1.1 소프트웨어 프로세스 가시화

프로세스 가시화 절차는 그림 1과 같다. 계획 단계에서는 프로젝트 성공을 위한 목표를 수립한다. 목표의 성공을 위해 현 조직이 필요로 하고, 개선하고자 하는 지표를 설정한다.

수행 단계에서는 가시화 절차 수행을 위한 자동화 환경을 구축한다. 소프트웨어의 요구사항과 개발 진척도를 자동적으로 가시화하고, 구현된 기능과 실제 요구사항을 추적할 수 있는 환경을 조성한다.

검증 단계에서는 가시화된 개발 과정을 모니터링한다. 현재 개발된 소프트웨어의 품질 지표와 목표 수행 정도를 지속적으로 실시간으로 확인하여 개선한다.

프로세스 가시화의 흐름도는 그림 2와 같다. 이해관계자와의 회의를 통해 요구사항을 정의하고, 정의된 요구사항을 프로젝트 관리 도구에 입력한다. 프로젝트 관리 도구는 입력된 요구사항을 기반으로 기능 요구사항 및 상세 설계가 기록되고, 구현 및 테스트에 대한 추적성을

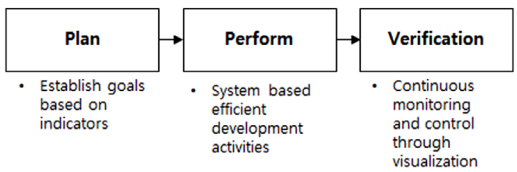


그림 1 프로세스 가시화 절차
Fig. 1 Procedure for Process Visualization

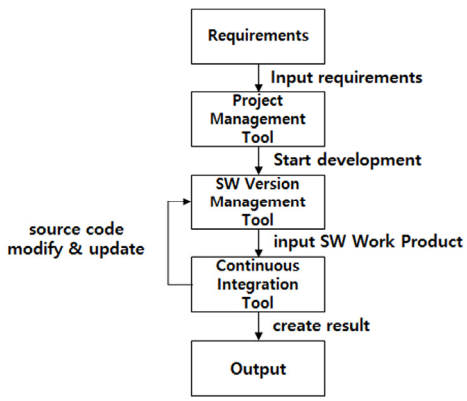


그림 2 프로세스 가시화 흐름도
Fig. 2 A Flow Chart of Process Visualization

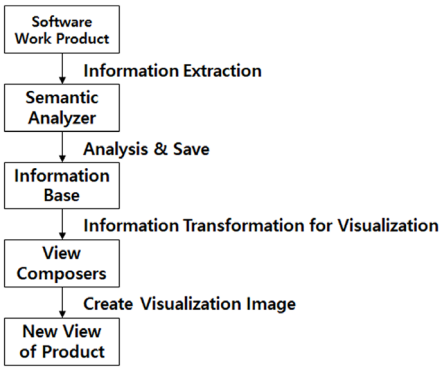


그림 3 아키텍처 가시화 흐름도

Fig. 3 A Flow Chart of Architecture Visualization

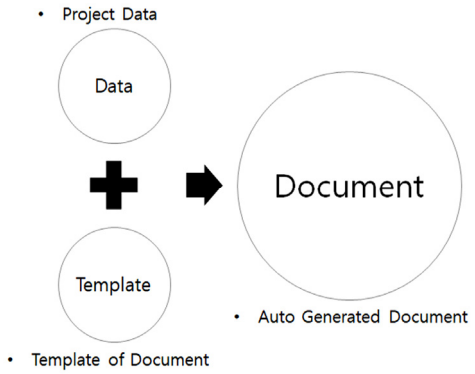


그림 4 문서 자동화 흐름

Fig. 4 A Flow Chart of Automatic Document Generation

가진다. 소프트웨어 구현은 소스코드 버전 관리 시스템을 사용하여 소스코드의 버전관리를 수행하며, 소스코드는 지속적 통합 도구를 통해 주기적으로 빌드된다. 지속적 통합 도구는 주기적으로 소스코드를 빌드함과 동시에 소스코드의 품질을 측정하고, 관련 문서를 자동 생성한다.

2.1.2 소프트웨어 아키텍처 가시화

아키텍처 가시화는 소프트웨어의 내부 구조를 가시화에 있다. 기본적으로 역공학 기법을 통해 개발 중인 소스코드의 구조를 가시화함으로써 개발 중인 소스코드가 요구사항을 올바르게 충족하는지 검사하고, 프로젝트 내에서 목표로 하는 품질을 만족하는지 검증한다.

아키텍처 가시화의 흐름도는 그림 3과 같다. 개발 중인 소프트웨어의 정보를 추출하여 분석하고, 저장한다. 저장된 데이터를 시각화 정보로 변환한다. 동시에 소프트웨어에 대한 품질을 검사한다. 변환된 시각화 정보 데이터는 이미지로 생성된다[5].

2.1.3 문서 가시화

문서 가시화는 소프트웨어 개발 사이클 내에 입력된 데이터들을 기반으로 산출물 생성의 자동화이다. 소프트웨어 개발 활동에서, 문서화 작업은 때로는 개발 과정에 부담으로 작용할 수 있다. 요구사항 기능 명세서, 요구사항 추적 매트릭스 등의 문서를 자동 생성하여 문서 작업 업무의 부담을 줄이고, 개발 편의성 확보를 지원한다.

문서 자동화의 흐름도는 그림 4와 같다. 업무 내에서 사용되는 문서 양식으로 기반으로 템플릿을 만든다. 문서화를 수행하기 위한 데이터를 기존 환경에서 가져와 템플릿에 대입하여 문서를 자동 생성한다[6].

프로젝트 요구사항 또는 개발 진행 상황이 변경될 때마다 문서에 자동 반영되어 출력이 가능하다.

2.2 프로젝트 관리

프로젝트 관리 기법은 개별 프로젝트를 관리하는 단일 프로젝트 관리 기술에서, 팀 간 인적/물적 자원을 공유하는 복수 프로젝트 관리 기술, 분산된 프로젝트 팀 간의 통합된 프로젝트 관리를 위한 클라이언트/서버 방식의 통합 프로젝트 관리 기술로 발전하고 있다. 일반적인 프로젝트 관리 시스템은 프로젝트 계획/일정관리, 자원관리, 프로젝트 진도관리, WBS(Work Breakdown Structure) 관리, 성과 관리 등의 기능을 포함한다[7].

레드마인(Redmine)은 프로젝트 관리 기능을 지원하는 오픈 소스 기반의 도구이다. 웹 기반의 프로젝트 관리와 버그 추적 기능을 제공한다. 화면 기반의 프로젝트 관리에 도움이 되도록 달력과 간트 차트를 제공하고, 일정관리 기능을 제공한다. 또한 통합된 프로젝트 관리 기능과 이슈 추적, 기타 형상 관리 기능을 제공한다[8].

2.3 지속적인 통합

소프트웨어 제품은 서로 간의 복잡한 종속적 관계를 가지는 수십, 수백 개의 구성 요소로 구성되어 있다. 이러한 경우, 한 구성 요소의 변경으로 인해 많은 구성 요소의 동작이 영향을 받을 수 있다. 여러 구성 요소를 업데이트 한 경우, 안정화 작업을 위해 인력을 배정하는 경우가 생길 수 있고, 소프트웨어의 통합 지옥(Integration Hell)이 발생할 수 있다. 이러한 문제를 해결하기 위한 방법 중 하나로써, 지속적인 통합 방법이 있다.

지속적인 통합 방법은 개발자가 짧은 시간 간격으로 자주 반복하여 릴리즈 할 수 있도록 유도한다. 이 방법을 통해 통합에서 발생할 수 있는 이슈를 미연에 제거하고 방지한다. 또한 소프트웨어의 질적 향상과 소프트웨어를 배포하는데 걸리는 시간을 단축한다[9].

젠킨스(Jenkins)는 지속적인 통합 기능을 지원하는 오픈 소스 기반의 도구이다. 소프트웨어 개발 시 지속적 통합 서비스를 제공한다. 다수의 개발자들이 하나의 프로그램을 개발할 때 버전 충돌을 방지하기 위해 각자 작업한 내용을 공유 영역에 있는 저장소에 빈번히 업로드 함으로써, 지속적 통합이 가능하도록 지원한다[10].

2.4 소스코드 버전 관리 시스템

소스코드 버전 관리 시스템은 사용자가 정의한 시간에 따라 모든 소스코드 혹은 파일을 저장하고, 변경된 코드에 대한 로그를 기록한다. 작성자, 시간, 추가/수정/삭제 사유 등에 대한 자세한 사항을 기록하며, 만약 실수로 파일을 수정하였다면, 이전 버전으로 회귀가 가능하다. 소스코드가 저장되는 공간은 소스 저장소라고 불린다. 소스코드 버전 관리 시스템은 일반적으로 전체적인 소스코드를 관리하는 서버, 개발자들이 개발하는 소스코드들을 서버에 업로드해주는 클라이언트로 구분된다[11].

TortoiseSVN은 소스코드 버전 관리 기능을 지원하는 도구이다. 소스코드 버전 관리 시스템의 Client 기능을 수행하는 도구로써, 개발자들이 개발한 코드를 서버에 업로드하거나, 서버에 업로드 된 코드를 다운로드 받으며, 사용자의 편의성을 위한 UI를 제공한다[12].

3. 군 자원관리 시스템 개발의 프로세스 가시화

3.1 프로세스 가시화 계획

계획 단계에서는 지표 설정 및 목표를 정의한다. 이 단계에서는 이해 관계자들 간의 토의를 통해 현재 개발 환경에서 가장 필요한 요소를 추출한다. 군 자원관리 시스템 개발에서 가장 필요한 목표는 요구사항 달성 여부와 코딩률 준수이다. 그리고 관리자가 모든 프로젝트 진행상황을 한눈에 파악하기 위한 대시보드가 필요하다. 목표의 달성을 위해 다음과 같은 지표를 수행한다.

- 프로젝트의 진행상태
- 코딩률 준수상태
- 사업 별 사업기간
- 테스트 결과
- 사업추진율
- 투입인원
- 요구사항 완료율
- 요구사항 지연율

3.2 프로세스 가시화 수행

수행 단계에서는 계획 단계에서 설정한 목표 및 지표를 가시화하고, 요구사항 추적 매트릭스를 만들기 위한 자동화 환경을 구축한다.

요구사항 및 진척도 입력 및 관리를 위해 레드마인을 사용한다. 주기적으로 프로젝트 빌드, 진척도 확인, 목표/지표 계산을 위해 젠킨스를 사용한다. 젠킨스를 통해 주기적으로 갱신된 결과를 가시화하기 위해 대시보드를 구축한다. 각 도구는 웹 기반 환경으로 구현된다. 도구별 기능은 표 1과 같다.

도구를 수행시키기 위해 그림 5와 같은 환경을 구성한다. 젠킨스는 아파치 톰캣 환경에서 기능이 수행된다. 레드마인, 대시보드는 PHP 모듈이 적용된 아파치 환경에서 기능이 수행된다. 각 도구는 데이터를 MySQL 데이터베이스에 저장한다.

각 도구들을 이용한 프로세스 가시화 수행 절차는 그림 6과 같다. 레드마인에 요구사항을 입력하고, 설계, 구현, 테스트에 대한 관리 및 추적을 수행한다. TortoiseSVN을 사용하여 소스코드를 관리한다. Jenkins를 사용하여 소스코드의 주기적 빌드를 수행하고 소프트웨어의 품질을 측정한다. 측정된 품질을 다이어그램 혹은 대시보드를 통해 가시화된다.

수행 단계에서 도구 간의 상호작용 방법은 그림 7과 같다. 레드마인은 소프트웨어 요구사항과 프로젝트 진척도 변경사항을 레드마인 DB에 입력한다. 젠킨스는 주기

표 1 도구 별 기능

Table 1 Specific Features of Open Source Tools

Tool	Functions
Redmine	• Requirement management • Requirement tracing • Human resource management • Schedule management
Jenkins	• Periodic sourcecode build • Periodic progress check • Periodic SW quality calculation
Dashboard	• Development progress Visualization • SW quality score visualization

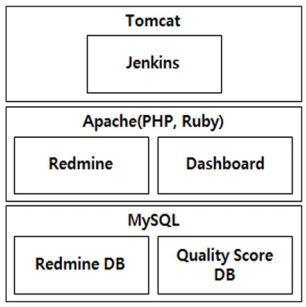


그림 5 전체 프레임워크

Fig. 5 Complete Framework

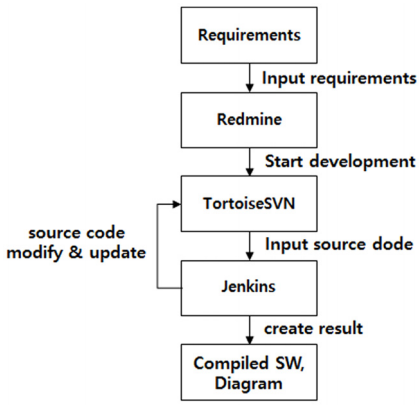


그림 6 프로세스 가시화의 수행 절차

Fig. 6 An Executing Procedure for Process Visualization

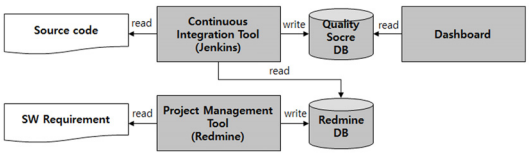


그림 7 오픈 소스 도구 간 상호작용
Fig. 7 An Interaction between Toolchains

적으로 소스코드를 빌드하고, 레드마인 DB를 읽어 품질 점수를 계산하여 품질점수 DB에 입력한다. 대시보드는 품질점수 DB를 읽어 품질점수를 테이블화 및 그래프화 한다. 사용자는 웹브라우저를 통해 대시보드에 접속하여 프로젝트 가시화 환경을 사용할 수 있다.

3.3 프로세스 가시화 검증

검증 단계에서는 그림 8과 같이 구현된 시스템을 통해 현재 상황을 모니터링하고, 문제점을 개선한다.

레드마인은 프로젝트 개요를 작성 및 관리하고, 이 외에도 인원 관리, 일감 관리, 일정 관리 기능을 제공한다. 일감 관리 기능을 통해 요구사항 추적성을 사용 가능하다.

젠킨스는 진행 중인 모든 개발 프로젝트를 관리하고, 프로젝트의 소스코드를 주기적으로 빌드하면서 툴체인을 설정 및 실행한다.

툴체인은 소스코드를 분석하여 소스코드의 품질 점수를 계산하고, 소스코드 관계 그래프를 그린다. 프로젝트 진행상태, 코딩룰 준수상태, 사업별 사업기간, 테스트 결과, 사업 추진율, 투입인원, 요구사항 완료율, 요구사항 지연

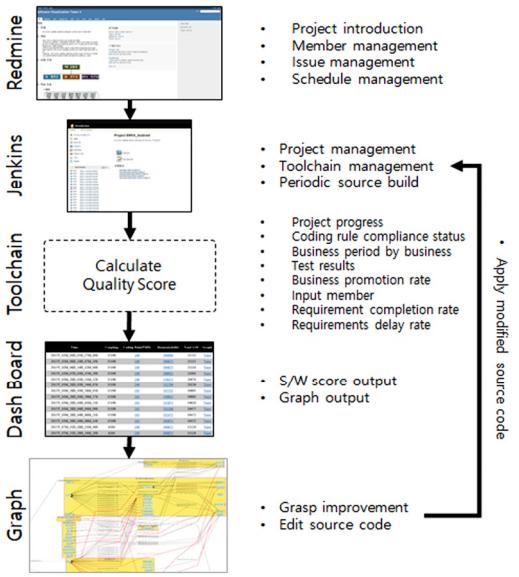


그림 8 프로세스 가시화 검증 절차
Fig. 8 A Validation Procedure for Process Visualization

율을 계산한다.

대시보드는 분석된 기록된 품질 점수와 그래프의 리스트를 출력한다. 젠킨스에서 한번 빌드 시 하나의 기록이 생성된다.

그래프는 소스코드 모듈들의 관계를 나타낸다. 한 번의 품질 점수 계산 당 하나의 그래프가 생성되어 대시보드에 표시된다.

개발자 혹은 관리자는 주기적으로 기록되는 대시보드를 조회하여 현 소프트웨어의 품질 점수와 소프트웨어 모듈 구조에 대한 그래프를 확인한다. 그리고 미비한 모듈에 대한 소스코드를 개선하고, 다시 대시보드를 통해 수정 결과를 확인한다. 소프트웨어 개선 결과는 누적되는 기록들을 통해 검증 가능하다.

4. 프로세스 적용 및 개선 사례

프로세스 적용 결과는 보안상의 이유로 인해 외부 공개가 불가능하다. 그렇기 때문에 유사한 형식을 가진 그림으로 대체한다.

그림 9는 레드마인의 일감 페이지이다. 일감 페이지는 프로젝트의 진행 상태를 실시간으로 저장하고, 보여준다. 한 행 당 하나의 일감을 나타낸다. 일감에 대한 속성은 레드마인 설정에서 변경 또는 추가할 수 있다.

본 논문은 레드마인에 요구사항 추적성을 부여하기 위해 일감들의 종류를 요구사항, 설계, 구현, 코드리뷰, 테스트로 나누었다. 그리고 각 일감 간의 연관성을 부여하여 일감의 상위 일감, 하위 일감을 지정할 수 있도록 구성하였다.

#은 레드마인에서 관리되는 일감들의 고유번호이다. ID는 프로젝트에서 관리되는 일감들의 고유문서번호이다. Tracker는 일감의 타입을 의미하며, Requirement, Design, Implementation, Code Review, Test case의 속성을 가진다. 일감을 하나의 Tracker 속성을 가질 수 있다. Status는 일감의 진행 상태를 의미하며, New, In Progress, End 속성을 가진다. Priority는 일감의 우선순위를 의미하며, Normal, Urgent 속성을 가진다. Subject는 일감의 제목을 의미한다. Assignee는 권한을 가진 사용자를 의미한다. Updated는 작성 및 수정 일자를 의미한다.

레드마인의 일감은 현재 프로젝트 내에서 진행되는 작업들의 정보를 포함한다. 일감의 종류는 Requirement, Design, Implementation, Testcase, Code review이다. 하나의 Requirement는 여러 개의 Design으로 나뉠 수 있다. 하나의 Design은 여러 개의 Implementation으로 나뉠 수 있다. 하나의 Implementation은 여러 개의 Testcase와 Code review로 나뉠 수 있다. 각 일감은 나뉘어진 일감들에 대한 ID를 저장한다. 나뉘어진 일감들은 나뉘기 전의 일감에 대한 ID를 저장한다. 결과적으로, 각 일감들은 자신과 연관된 일감들에 대한 추적성

#	ID	Tracker	Status	Priority	Subject	Assignee	Updated
63		Code review	New	Normal	The comment is not enough	Admin, son	09/19/2017 06:16 PM
61	IMT02	Implementation	New	Normal	Controller		09/19/2017 05:30 PM
60	IMT01	Implementation	New	Normal	Communicator		09/19/2017 05:28 PM
59	TC01	Test case	In Progress	Normal	Test case for the Referee class		09/19/2017 06:05 PM
58	TC	Test case	New	Normal	Events and State Transitions		09/19/2017 03:45 PM
56	DST02	Design	New	Normal	Communicator state diagram		09/19/2017 03:28 PM
55	DST01	Design	New	Normal	Controller state diagram		09/19/2017 03:25 PM
54	DC02	Design	New	Normal	Controller class		09/19/2017 03:08 PM
53	DC01	Design	New	Normal	Communicator class		09/19/2017 03:08 PM
52	DS05	Design	New	Normal	Sequence diagram for remote player's move in the game of tic-tac-toe.		09/19/2017 03:07 PM
51	DS04	Design	New	Normal	Sequence diagram for local player's move in the game of tic-tac-toe.		09/19/2017 03:07 PM
50	DS03	Design	New	Normal	Sequence diagram for the local player to challenge an opponent to play a match of the game of tic-tac-toe.		09/19/2017 03:07 PM
49	DS02	Design	New	Normal	Sequence diagram for the gameroom setup in the game of tic-tac-toe		09/19/2017 03:07 PM

그림 9 구현된 레드마인의 예
Fig. 9 An Example of Redmine

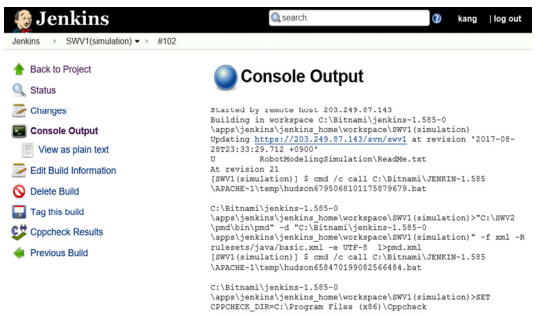


그림 10 구현된 젠킨스의 예
Fig. 10 An Example of Jenkins

을 가질 수 있다.

젠킨스는 주기적으로 소스코드를 빌드하고, 소스코드 품질 점수를 계산한다. 그림 10은 젠킨스의 빌드 결과를 표시한다. 사용자는 빌드 결과를 통해 에러의 유무를 확인할 수 있다.

하루에 한번의 빌드를 주기적으로 수행되도록 지정하였다. 빌드가 수행될 때 프로세스 가시화 계획에서 계획된 지표들을 소스코드에서 추출하는 툴체인을 함께 실행한다.

대시보드의 구현 결과는 보안상의 이유로 인해 외부 반출이 불가능하기 때문에 비슷한 형식을 가진 그림 11과 같은 대시보드를 예로 든다. 대시보드는 소스코드의 품질 점수를 출력하고, 실제 소스코드와의 추적성을 가진다.

Time은 젠킨스의 빌드 시간을 의미한다. Coupling은 소스코드의 결합도 점수를 의미한다. Coding Rule은 PMD의 코딩룰을 통해 추출된 점수를 의미한다. 링크를 클릭하면 상세한 룰의 점수를 확인할 수 있다. Maintainability는 소스코드의 유지보수성 점수를 의미한다. Total LOC는 소스코드의 라인수를 의미한다. Graph는 소스코드의 구조를 그래프로 그린 그림이다.

Time	Coupling	Coding Rule(PMD)	Maintainability	Total LOC	Graph
2015Y_02M_04D_05H_27M_00S	35198	249	594900	21532	View
2015Y_02M_06D_14H_07M_20S	35198	249	594675	21325	View
2015Y_02M_06D_14H_12M_00S	35198	249	594675	21110	View
2015Y_02M_07D_03H_47M_59S	35198	249	588625	21003	View
2015Y_03M_28D_03H_51M_22S	35198	249	576475	20970	View
2015Y_03M_28D_03H_54M_07S	35198	249	552700	20136	View
2015Y_03M_28D_03H_58M_05S	35198	243	545525	19895	View
2015Y_03M_28D_03H_59M_57S	35198	243	539825	19692	View
2015Y_03M_28D_04H_01M_51S	35198	243	535875	19619	View
2015Y_03M_28D_04H_04M_09S	35198	243	531500	19477	View
2015Y_03M_28D_04H_06M_21S	35198	243	532475	19472	View
2015Y_03M_28D_04H_08M_14S	35198	243	594675	19472	View
2015Y_07M_31D_18H_21M_48S	6264	249	594675	11126	View
2015Y_07M_31D_18H_34M_20S	6264	249	594675	11126	View

그림 11 구현된 대시보드의 예
Fig. 11 An Example of Dashboard

표 2는 실제로 구현된 대시보드에서 표현되는 지표이다. 기존의 경우, 지표 항목들이 대부분 수작업 또는 구두 보고로 진행되었기 때문에 누락 또는 잘못된 계산을 할 가능성이 존재하였다. 이러한 이유로 현재 프로젝트 진행에 대한 명확한 상황 파악이 어려웠다.

프로젝트 관리 도구와 지속적 통합 도구를 활용한 자동 가시화 기법 적용 결과, 도구를 통해 저장한 정보들을 주기적으로 읽어 들여 프로세스 가시화 계획에서 설정한 지표로 자동 계산하여 대시보드에 표시한다.

이 외에도 자체 제작 툴체인과 SPARROW를 사용하여 소프트웨어 품질 점수를 대시보드에 표시한다[13]. 레드마인에 입력된 요구사항과 설계 데이터를 자동 문서화하여 대시보드에 표시한다. 레드마인에 입력된 데이터들은 일감 ID와 소스코드 파일 링크를 사용하여 추적성을 가진다.

기존 프로젝트에서 책임자인 군장성 관점에서는 스케줄 제어, 개발자들의 개발 진척도 및 오류률에 대한 가시성이 확보되어 있지 못해서 스케줄 통제와 진척도를 알 수 없었다. 또한 프로젝트의 리스크 관리도 어려운 실정이었다. 이번 프로세스 가시화를 통해, 스케줄 통제,

표 2 실제 대시보드에 표시되는 지표
Table 2 Indicators displayed in the actual dashboard

Indicators	Before improvement	After improvement
Project progress	Verbal verification	Automatic
Business period by business	handwork	Automatic
Business promotion rate	handwork	Automatic
Requirement completion rate	Verbal verification	Automatic
Coding rule compliance status	possible	Automatic
Test results	possible	Automatic
Member management	handwork	Automatic
Requirements delay rate	handwork	Automatic

진척도 및 부산물의 내재화로 전체 개발 공정의 추적성이 가능해졌다.

결과적으로, 육군은 프로세스 가시화 적용으로 소프트웨어 개발 품질을 향상시켜 SP인증 2등급을 획득하였다.

5. 결 론

프로세스 가시화는 소프트웨어 개발 과정의 투명화와 소프트웨어의 품질을 확보하고, 소프트웨어 개발 문제를 조기 검출을 지원한다. 프로세스 가시화는 달성 목표, 품질 지표, 프로세스 가시화를 위한 도구 선정 과정에서 현 조직의 문화와 구조에 이해가 깊은 이해관계자 간의 토의가 필요하다. 그렇기 때문에 조직의 구조에 따라 상이한 품질 지표가 만들어질 수 있다. 하지만 이해관계자들이 조직 구조에 대한 이해가 부족하다면, 맞지 않는 품질 지표를 설정할 수 있다.

본 논문은 군 자원관리 시스템의 개발 사례를 통해 달성 목표와 품질 지표를 설정하고, 프로세스 가시화 환경을 구축하였다. 군 자원관리 시스템 개발 조직과 유사한 환경의 조직이라면, 본 논문이 해당 조직에 적합한 프로세스 가시화 환경 구축을 위한 좋은 사례가 될 것으로 기대한다.

References

[1] NIPA, "SW Development Quality Management Manual," 2013.

[2] Steve Wright, Corey Erkes, "Application Lifecycle Management," *Pro SharePoint 2010 Governence*, pp. 263-278, 2010.

[3] Geon Hee Kang, Bo Kuyng Park, Woo Sung Jang, Jun Sun Hwang, Ha Eun Kwon, Han Sol Lee, Hyun Joon Lee and R. Young Chul Kim, "Development Toolchain for Software Performance Visualization," *KCSE 2016*, Vol. 18, No. 1, pp. 395-398, 2016.

[4] Bo Kuyng Park, Ha Eun Kwon, Young Soo Kim, Sang Eun Lee and R.Young Chul Kim, "A Case Study on Improving SW Quality through Software Visualization," *Journal of The Korean Institute of Information Scientists and Engineers*, Vol. 41, No. 11, pp. 935-942, 2014.

[5] Woo Sung Jang, R. Young Chul Kim and Jae Hyup Lee, "Extending Reverse Engineering for Software Maintenance," *Research India Publications*, Vol. 10, No. 90, pp. 474-477, 2015.

[6] Woo Sung Jang, Jun Sun Hwang, Dong Ho Kim, Chae Yun Seo, R. Young Chul Kim, Byung Ho Park, Sang Eun Lee and Young Soo Kim, "Constructing with XML Data and XSLT for Process Asset Library (PAL)," *Korea Information Processing Society*, Vol. 22, No. 2, pp. 956-958, 2015.

[7] Jae Yong Baek, So Young Jung, Bo Hyun Kim, Seock Kyu Yoo, Seok Woo Lee and Hon Zong Choi, "Design of Architecture for Collaborative Project Management System based on Business Process," *International Journal of CAD/CAM*, Vol. 14, No. 5, pp. 338-345, 2009.

[8] Redmine Hompage, http://www.redmine.org.

[9] S. J. Kim, et al., "Automated continuous integration of component-based software: An industrial experience," *Proc. of the 23rd IEEE/ACM International Conference on Automated Software Engineering*, pp. 423-426, 2008.

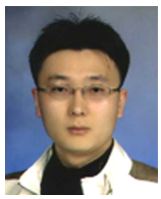
[10] Jenkins Homepage [Online]. Available: https://kenkins.io (downloaded 2018. Feb. 22)

[11] Subversion Homepage [Online]. https://subversion.apache.org (downloaded 2018. Feb. 22)

[12] TortoiseSVN Homepage [Online]. https://tortoisesvn.net (downloaded 2018. Feb. 22)

[13] Bo Kyung Park, So Young Moon, Chae Yun Seo, R. Young Chul Kim, Kwang Nam Kim, Young Sik Choi, Sang Hoon Shin, "An Applied Practice through developing Jana Parser for Code Quality of the National Defence Software Resource Management System," Vol. 20, No. 1, pp. 222-223, 2018.

장 우 성
정보과학회논문지
제 45 권 제 6 호 참조



손 현 승
2007년 홍익대학교 소프트웨어공학(학사). 2009년 홍익대학교 소프트웨어공학(석사). 2015년 홍익대학교 소프트웨어공학(박사). 2015년~2017년 홍익대학교 메카트로닉스 연구센터 박사후 연구원 2017년~현재 모아소프트 개발팀장. 관심분야는 임베디드 소프트웨어 자동화 도구 개발, 소프트웨어 프로세스 및 가시화, 메타모델 설계 및 모델 변환, 모델 검증 기법 연구, 한국형 테스트성숙도모델, 시큐어 코딩

김 영 철
정보과학회논문지
제 45 권 제 6 호 참조

이 종 훈
1990년 미국 미주리대학교 전산공학(석사). 1994년 미국 미주리대학교 전산공학(박사). 육군정보통신학교 체계운영학처장, 현재 육군본부 정보체계관리단장, 관심분야는 소프트웨어공학, 정보보호, 소프트웨어 가시화, 시큐어 코딩, NCW, 빅데이터 등