백로그 팩토링 : 스크럼 방법에서 재사용을 위한 태스크 팩토링의 확장

김지홍*

Backlog Factoring : Extension of Task Factoring for Reuse in Scrum Method

Ji-Hong Kim*

요 약 애자일 방법과 소프트웨어 재사용 기술은 서로 다른 접근방법이지만 통합을 통하여 상호 보완하려는 여러 연구가 나타나고 있다. 애자일 소프트웨어 개발 방법 가운데 가장 높은 인기와 사용에도 불구하고, 스크럼에서의 재사용 지원이 부족하다. 본 연구는 기존의 태스크 팩토링 기술을 스크럼 방법에 확장하여 스크럼 재사용 자산을 식별하고 이를 단순화하여 스크럼에서의 재사용이 지원되는 백로그 팩토링 기술을 제안하였다. 아울러, 제안된 기술을 전자 상거래 응용에 적용하여 백로그 재사용의 프로토타이핑을 보일 수 있었다.

주제어 : 스크럼, 애자일 소프트웨어 개발, 소프트웨어 재사용, 애자일 방법, 소프트웨어 공학

Abstract Despite agile methods and software reuse being distinct approaches, there has been a growing amount of research on combining these approaches. Although Scrum is one of the most popular agile methods, reuse has not been actively supported. In this paper, we identify a new type of reusable Scrum assets, simplify the backlog process and propose a backlog factoring technique by extending task factoring to support reuse in the Scrum method. In addition, we can apply the proposed technique and show prototyping of backlog reuse in e-business applications.

Key Words : Scrum, Agile Software Development, Software Reuse, Agile Method, Software Engineering

1. 서론

애자일 방법과 소프트웨어 재사용 기술은 서로 다른 접근방법이지만 상호 보완을 위한 통합 연구가 나타나고 있다[3][11][20].

소프트웨어 재사용은 이전에 개발한 산출물의 재사용을 통해 생산성을 향상시키는 소프트웨어 기술로[15], 이미 잘 알려진 입증된 기술이다. 이는 사전에 재사용 가능한 자산의 준비와 계획을 하여 장기간에 재사용의 혜택을 얻는 기술이다. 그러나 오늘날과 같이 시장 조건과 기술이 급변하는 상황에서 재사용의 투자가 쓸모없어질 수 있는 불확실성의 위험이 있다. 반면에, 스크럼을 포함하는 애자일 방법은 인터넷 시대의 환경에 적응하도록 잦은 요건 변경을 수용하고, 빠른 출고를 위해 장기 보다는 단기의 관점, 추후 필요할 것으로 보다는 지금 필요한 것에 집중하는 특징을 갖는다[6].

그동안의 2가지 기술 통합에 관한 연구는 체계적인 소프트웨어 재사용을 위한 소프트웨어 프로덕트 라인 기술에 애자일 방법을 도입하는 여러 방안이 제시되었으나, 구체적인 프로세스 수준에서의 접근은 부족하였다[6][9][11][13]. 아울러, 스크럼은 가장 인기 있는 애자일 방법이지만, 이 방법에서의 재사용 도입에 관한 연구는 상대적으로 부족하였다.

스크럼 방법에 재사용 기술의 도입으로 상호 보완될 수 있다. 본 논문은 스크럼 방법에서 재사용 가능한 자산과 유형을 식별하고, 이를 단순화하여 백로그 수준의 재

이 논문은 2012년도 가천대학교 교내연구비 지원에 의한 결과임.(GCU-2012-R246)
*정회원 : 가천대학교 컴퓨터공학과 교수
논문접수: 2012년 10월 27일, 1차 수정을 거쳐, 심사완료: 2012년 11월 20일

사용 방안을 제시한다.

본 논문의 구성은 2장에서 기초 연구에 속하는 애자일 방법, 스크럼 그리고 재사용에 관해 알아본다. 3장에서는 백로그 재사용과 백로그 팩토링 방안을 제안하고, 4장에서는 프로토타이핑을 보인다. 5장에서는 관련 연구를 비교하고, 6장에서 결론을 기술한다.

2. 기초 연구

2.1 애자일 방법과 스크럼

애자일 방법은 종전의 개발단계를 단순화하고 문서화 등의 개발 오버헤드를 줄여 단기간의 중소형 시스템 개발에 적합하도록 제안되었다[17][21]. 이는 애자일 원칙을 준수하는 여러 방법들을 총칭하는 이름이며 대표적인 방법으로 익스트림 프로그래밍(XP), 스크럼(Scrum), 피처 중심의 개발(Feature Driven Development : FDD) 등이 있다. 애자일 방법은 짧은 주기의 반복 점증적 방식을 바탕으로 빠른 개발과 요구사항 변경을 수용하는 특징을 갖는다. 사전 작업보다는 변화에 대응이 용이한 적응적(adaptive) 또는 후대응적(reactive) 개발 접근방법을 갖는다[7].

스크럼은 복잡한 제품 개발을 관리하는데 사용되는 프로세스 프레임워크이다. 이는 제품 개발을 위한 프로세스나 기술이 아니고 다양한 프로세스나 테크닉을 적용할 수 있는 반복 점증적 개발 프레임워크이다[22]. 스크럼에서는 스프린트라고 부르는 짧은 반복을 사용하여 팀원이 해당 기능을 구현한다. 스크럼은 제품 소유자(PO), 팀, 스크럼 마스터(SM)의 3가지 역할을 통해 반복적이고 점증적으로 수행한다. 스크럼에서의 산출물은 제품 백로그, 스프린트 백로그, 제품 증분이 있다.

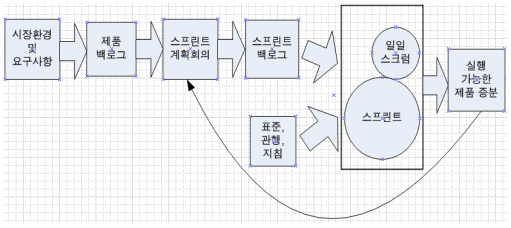
2.2 소프트웨어 재사용과 유형

소프트웨어 재사용이란 이전 소프트웨어 개발에 생성된 각종 산출물을 새로운 개발에 다시 사용을 말하며[10], 크게 기회적 재사용과 체계적 재사용으로 구분된다[19].

첫 번째 유형은 사전에 별도의 재사용을 위한 투자없이, 새로운 응용의 개발에 기회가 되면 재사용하는 단기 저비용 방식으로 우연적 재사용이라 부르기도 한다[19]. 두 번째 유형은 재사용의 의도적인 계획을 갖고 장기적 재사용 이익을 위해 적절한 자산의 생성, 관리 및 응용을 말한다[8]. 대표적인 체계적 재사용으로 소프트웨어 프로덕트라인 공학이 있다. 이는 제품군에 속하는 제품들의 공통점과 가변점을 사전에 식별하고 준비하는 도메인 공학과 새로운 응용에 재사용하는 어플리케이션 공학으로 구성되는 대규모의 재사용을 지원하는 방법이다[16].

2.3 스크럼 프로세스와 산출물

스크럼 프로세스는 그림 1과 같다[2]. PO는 제품에 필요하다고 요청되는 요구사항을 제품 백로그 항목(PBI)으로 제품 백로그(PB)라는 목록에 등록한다. 이때 SM 또는 팀의 협조를 받을 수 있으나 PBI의 결정 및 변경에 대한 최종 책임은 PO에게 있다.



[그림 1] 스크럼 프로세스

스프린트 계획회의 전반부에서 PO는 팀에게 우선순위가 가장 높은 PBI을 제시하고 또 상세히 설명을 한다. 후반부에서는 PO가 요구한 항목에 대해, 팀과 SM은 이를 태스크(task)라고 부르는 작업 단위로 분해하여 스프린트 백로그(SB)라는 목록에 저장한다. 팀은 스프린트(sprint)라고 하는 2-4주의 기간 동안, SB에 있는 스프린트 백로그 항목(SBI)을 구현한다. 이외에, 스크럼 가이드 개정판에서는 제품 백로그 손질(grooming)이라는 활동을 추가로 명시하고 있다[22]. 이는 한 스프린트의 수행 후 PB의 갱신을 위한 항목의 추가, 삭제, 수정, 분할의 활동을 말한다. 이 활동은 스프린트 중, 스프린트 종료 후, 또는 필요할 때 언제든지 수행될 수 있다[18].

하나의 스프린트는 여러 일일 스크럼(daily scrum)으로 수행되며, 스프린트가 끝나면 하나의 동작되는 제품 증분을 갖게 된다.

2.4. 태스크 팩토링

태스크 팩토링이란 각 이터레이션(iteration)에 수행될 새로운 작업을 결정할 때, 이전 태스크와 동일 또는 유사성 유무를 간단하게 발견하여 중복 작업의 단축이

가능한 태스크를 결정하는 활동 및 기술이다[1].

재사용을 지원하는 태스크 팩토링 절차는 제시-발견-결정의 순환적인 스텝으로 이루어지며 개발자 모두가 참가하는 팀원 회의에서 수행된다. 태스크 팩토링은 이터레이션에, 취급되는 산출물은 태스크에 국한되어 있다.

태스크 팩토링 개념은 스크럼에서 PB와 SB에도 확장 적용 가능하다.

3. 재사용을 지원하는 백로그 팩토링

3.1 백로그 재사용을 지원하는 개발

애자일 방법의 장점을 유지하며 새로운 형태의 재사용을 지원하는 개발 방안을 제시한다.

3.1.1 백로그 재사용과 자산

백로그는 사용자 요구사항과 이들 개발에 필요한 세부작업의 목록과 항목으로서 개발 초기에 결정되어 작업이 완료될 때 까지 각종 개발 작업에 사용되고 참고하는 중요한 재사용 자산이다. 백로그 재사용이란 이전 스프린트에서 생성한 백로그와 이에 관련된 후속 산출물을 동일 또는 유사한 새로운 개발에 다시 사용하여 빠른 개발과 비용 절감을 이끄는 소프트웨어 기술이다. 부수적으로는 제품 내 동일한 자산의 사용을 촉진하여 제품의 사용과 유지보수를 용이하게 한다.

한편, 스크럼 개발에서 재사용 가능한 백로그 자산은 크게 표 1과 같이 3가지 종류가 있다.

〈표 1〉 재사용 가능한 백로그 자산

종류	설명
백로그 목록	제품 백로그, 스프린트 백로그
백로그 항목	스토리, 태스크
백로그 산출물	산출물(테스트, 코드, 설계)

백로그 목록(BC)은 하나의 프로젝트 개발에 요구된 백로그 항목(BI)을 모아 놓은 리스트이다. 이는 전체적인 백로그 현황 파악과 재사용성 있는 항목을 빠르게 찾을 수 있는 자산이다. 스크럼에서 BC에는 PB와 SB가 있으며 특정조직에 따라 프로젝트 백로그, 릴리즈 백로그도 사용된다.

BI는 제품 개발에 필요한 개별적 요구사항이나 작업을 나타낸다. 이는 BC보다 더 구체적이고 상세한 표현으로 빠른 이해와 재사용이 가능한 자산이다. BI에는 PBI와 SBI가 있으며 스토리나 태스크 카드로 표현된다.

백로그 산출물(BD)은 특정 BI를 스프린트에서 수행한 결과물이다. 이는 산출물의 종류에 따라 코드의 재사용, 설계의 재사용 또는 테스트의 재사용이 가능한 자산이다.

3.1.2 백로그 팩토링

팀원들이 갖는 현 프로젝트에 대한 문제, 자원에 대한 인식, 그리고 선호하는 해결 방식은 비슷하기 때문에, 개발기간중 동일 또는 유사한 작업들이 반복될 수 있다. 백로그 팩토링이란 각 스프린트에 수행할 요구사항이나 새로운 작업을 결정할 때, 이전 BI와 동일 또는 유사 유무를 간단하게 발견하여 중복 작업의 단축이 고려된 BI를 제시하는 활동 및 기술이다.

3.2 백로그 재사용을 위한 고객화 활동

프로젝트의 제약과 개발 조직의 특성을 반영하여 백로그 팩토링 절차를 효과적으로 수행하도록, 고객화(customizing) 가능한 민첩한 활동들이 지원된다.

3.2.1 단순 백로그 이름 규칙

고객화 가능한 이름 규칙은 백로그의 일차적 식별을 단순화 시켜준다. 스크럼팀은 개발 초기에 프로젝트 특성에 맞는 키워드를 사용하여 표 2와 같은 이름의 규칙을 갖는다.

〈표 2〉 백로그 이름 규칙

```
<backlog_name> := <action> <object>
<action>       := /* registered verb   */
<object>       := /* registered object */
```

백로그의 이름은 단순하게 <action>과 <object>로 구성되며, <action>은 행동 또는 명령을, <object>는 대상을 나타내는 예약어 또는 등록어가 사용된다.

3.2.2 단순 백로그형(type) 지원

BI의 빠른 분류와 간단한 재사용 처리를 위해 백로그형이 지원된다. BI가 다루는 영역을 직접, 간접, 그리고 특화의 관점으로 나누고 표 3과 같은 형을 지원한다.

- 직접형 : 소스 코드의 생성, 수정, 검사 및 업그레이드에 직접 관련되는 각종 요구사항이나 작업을 직

접형이라 한다.

〈표 3〉 백로그 유형

종류	설명
직접형	구현 또는 코딩 관련 백로그
간접형	분석/설계 또는 코딩 비관련 백로그
특화형	비즈니스 특화 백로그

- 간접형 : 소스코드에 직접 관련되지 않은 분석이나 설계에 관련된 요구사항이나 작업들을 간접형이라 한다.
- 특화형 : 개발 조직이 갖는 특화업무의 효과적 재사용을 지원하도록 사용자 정의 가능한 형이다. 특화형의 미정의시, 데이터베이스 또는 저장소에 관련된 요구사항이나 작업을 위한 형이 지원된다. 데이터의 읽기, 쓰기, 갱신 등에 관련된 요구사항이나 작업이 해당된다.

3.2.3 단순 공통성 검사

동등 및 유사성의 빠른 초기 식별을 위해 BI 이름에 나타난 행동과 대상을 중심으로 다음과 같은 검사를 갖는다.

- 동일 대상 : 이전 스프린트에 수행한 BI와 새로운 스프린트에 처리할 후보 BI 이름에서 대상(object)의 동일 여부를 검사한다.
- 동일 행동 : 이전 BI와 후보 BI 이름에서 행동(action)의 동일 여부를 검사한다. 백로그가 다루는 대상은 서로 다르지만 행위가 같은 경우들이 해당된다.
- 동일 작업 : 이전의 스프린트를 위해 개발된 BI와 후보 BI 이름의 행동과 대상이 모두 동일한 지 여부를 검사한다.

3.2.4 적용 백로그 메소드

후보 BI가 기존 BI로 재사용될 수 있는지 유무와 정도를 고려하여 최종 BI를 결정하는 백로그 할당 메소드가 지원된다.

- Same() 메소드 : Same(a,b) 형식으로 a와 동일한 재사용 가능한 b가 있는 경우, a 대신 b를 사용하여 가장 높은 재사용을 촉진한다.
- New() 메소드 : New(a,b) 형식으로 a 대신 재사용될 BI가 없을 경우, 임시적인 a를 새로운 이름의 b로 할당하는 메소드이다. 즉, b가 최종적으로 제시되는 항목이다.
- Similar() 메소드 : Similar(a,b,c) 형식으로 a와 기존의 b가 유사한 경우, 공유 가능하도록 진화시킨 a를 c의 이름으로 최종 제시한다. Same()보다는 재사용성은 작지만 new() 보다는 큰 크기의 재사용성을 갖는다.

3.3 백로그 팩토링 절차

재사용을 지원하는 백로그 팩토링 절차는 그림 2와 같이 착수-발견-결정의 순환적인 스텝으로 이루어지며 스크럼 팀 모두가 참가하는 스프린트 계획회의와 스프린트 손질하기와 연계되어 수행된다.

[그림 2] 백로그 팩토링 스텝

백로그 착수(Initiate)에서는 백로그 팩토링의 초기적 작업을 준비하는 스텝이다. BI 항목을 충분히 이해하고 스프린트 기간에 작업이 가능하도록 여러 BI로 분해하고 간단한 규칙의 이름을 부여한다. 백로그 발견(Discover)은 새로운 스프린트를 위해 제시된 후보 BI와 동일하거나 비슷한 기존의 BI가 있는지를 간단하게 발견하는 스텝이다. 백로그 결정(Decide)은 제시한 후보 BI에 대하여 재사용 가능한 BI가 있는지 여부를 검토하고 최종 BI를 결정하고 작성하는 스텝이다.

순환적인 스텝을 바탕으로 하는 기본적인 백로그 팩토링 절차는 그림 3과 같다.

```
while (more item to process) {
  item = current_top_of_item_from_backlog();
  perform backlog_factoring(item);
}
backlog_factoring(item) {
  perform initiate_backlog();
  perform discover_backlog();
  perform decide_backlog();
}
```

[그림 3] 백로그 팩토링 절차

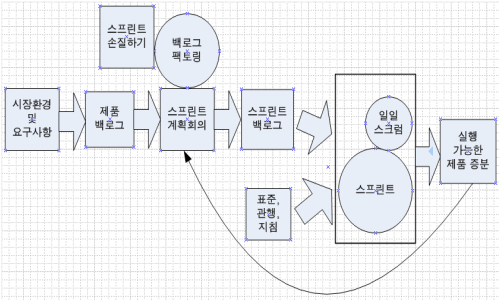
매번의 팩토링에는 initiate_backlog(), discover_backlog(), decide_backlog()가 수행된다. 먼저, current_top_of_item_from_backlog()를 통해 현 스프린트에서 처리할 항목(item)을 선택하고 백로그 팩토링을 수행한다. Initiate_backlog()에서는 크기가 큰 PBI나 SBI가 있으면 분해(split)한 후 <action>과 <object> 규칙의 BI 이름을 부여한다. 이어서, discover_backlog()에서는 백로그 형과 공통성 검사를 바탕으로 적용 메소드를 판단하고 최종 BI를 결정하여 백로그에 등록한다.

세부적으로, 백로그 팩토링 기본 절차에는 표 4와 같이 3.2절에서 소개된 고객화 활동들이 지원된다.

〈표 4〉 백로그 팩토링 스텝의 세부 활동

주요 스텝	세부 활동
1. initiate_backlog() 백로그 착수	1. 백로그 설명 및 세분화 2. 단순 규칙 백로그 이름 파악
2. discover_backlog() 백로그 발견	1. 백로그 형 검사 2. 공통성 검사
3. decide_backlog() 백로그 결정	1. 적용 백로그 메소드 결정 2. 백로그 카드 설명문 결정

본 논문에서 제안한 백로그 팩토링을 지원하는 확장 스크럼 프로세스는 그림 4와 같다. 백로그 팩토링은 스프린트 손질하기와 스프린트 계획회의 활동에서 수행된다. 스크럼에서 스프린트 손질하기는 백로그의 업데이트가 필요할 때 언제든지 수행될 수 있다.



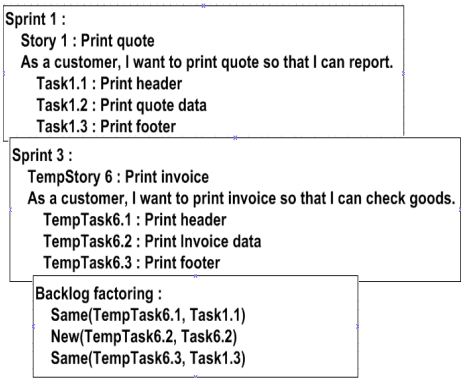
[그림 4] 스크럼 프로세스와 백로그 팩토링

4. 적용과 프로토타이핑

본 장은 3장에서 제안한 백로그 팩토링 기술을 전자상거래 응용에 적용하는 프로토타이핑을 보인다.

• 백로그 팩토링과 메소드 적용

백로그 재사용을 위하여, 한 sprint를 마치고 다음 sprint에서의 백로그 팩토링과 메소드를 적용한다.



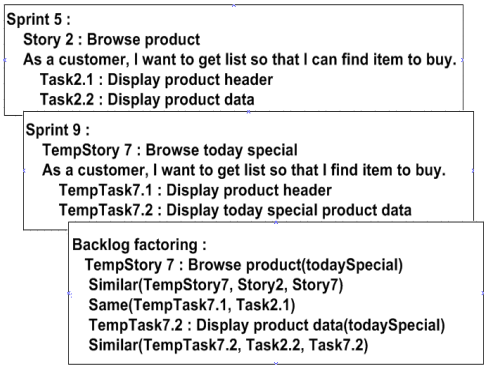
[그림 5] 백로그 팩토링(Same(), New() 메소드)

그림 5의 상위 부분은 기 수행된 여러 SBI 가운데 재사용 대상이 되는 Story1(Print quote)과 이를 위해 세분화된 3개의 태스크(Task1.x)를 보이고 있다. 중간 화면은 sprint 3에서 story6에 대한 백로그 팩토링 착수 스텝 과정의 일부를 보여준다. 처리과정의 임시적 표현인 TempStory6(Print invoice)는 <action>과 <object> 이름 규칙을 갖는 3개의 태스크(TempTask6.x)로 분해되었다. 3번째 화면은 백로그 팩토링의 나머지 스텝을 보인다. 발견 스텝에서 TempTask6.1과 Task1.1에 대한 백로그형과 공통성 검사가 수행되어 동일 작업(Same(TempTask6.1, Task1.1))으로 식별되었다. 이어지는 결정 스텝에서, TempTask6.2는 새로운 작업(New(TempTask6.2, Task6.2))으로 판단되어 새로운 태스크인 Task6.2의 할당을 보인다.

• Similar() 메소드의 적용

Sprint 5를 끝내고 sprint 9를 위한 백로그 팩토링을 보인다. 그림 6의 상단은 sprint 5에 수행된 story와 task를 보여준다. 백로그 착수 스텝에서 스토리 7(TempStory7)에 대한 임시적 중간과정 표현인 TempTask7.1과 TempTask7.2가 제시되었다(중앙 화면). 발견 스텝에서 TempStory7과 Story2 그리고 TempTask7.2와 Task2.2간의 유사성이 식별되었다. 마지막 스텝에서 Similar() 메소드가 적용되고, 수정 진화되어 제시된 백로그 Story7과 Task7.2의 결정을 보이고

있다(마지막 화면).



[그림 6] 백로그 팩토링(Similar() 메소드)

5. 관련 연구 비교

소프트웨어 재사용과 애자일 방법은 각각 독자적으로 연구되어왔으나 두 가지 기술의 장점을 통합하는 연구가 늘어나고 있다[5][11][12].

이 가운데 기존의 특정 애자일 방법과의 통합을 제안한 것으로, Hanssen[7]은 전략, 전술, 운영 수준에서 Evo라는 애자일 방법과 소프트웨어 프로덕트 라인 공학의 통합을 통해 성공적인 사례 연구를 발표하였다. 그러나 이 연구에서는 어떻게 프로세스가 애자일 환경에서 통합되는지는 강조하지 않았다. 아울러, Paige[14]는 FDD를 이용하여 소프트웨어 프로덕트 라인의 구축 방안을 제안했다. 이 연구는 통합의 가능성을 약속하고 있지만, 구체적인 프로세스와 사례가 기술되지 않았다.

최근에는, 스크럼을 기반으로 하는 애자일 방법과의 통합에 관한 연구가 점차 나타나고 있다. Diaz[4]는 스크럼 방법을 테일러링한 방안을 제시하였다. 구체적으로, 스프린트를 2개의 서브 프로세스로 나누어 도메인공학과 어플리케이션 공학을 각각 지원하는 프로세스를 제안하였다. Kircher[9]는 FDD, 피처중심의 개발(feature oriented development), 피처 모델링(feature modeling) 기술에서의 공통적인 피처를 중심으로 스크럼을 사용하는 애자일 방법과 프로덕트라인 공학을 통합하는 프로젝트 성공 사례를 보고하였다. 모두 스크럼을 이용하였으나 이들 연구는 프로덕트라인 공학을 위한 접근 제시인데 반해, 본 연구는 스크럼 핵심 자산인 백로그의 재사용을 목표로 하는 차이가 있다.

6. 결론

본 논문은 스크럼 방법에서의 재사용을 지원하기 위하여 기존의 태스크 팩토링 기술을 확장한 백로그 팩토링 방법을 연구하였다. 스프린트 특성을 활용 할 수 있도록 스크럼 핵심 자산과 처리를 단순화할 수 있었다. 그리고 백로그 재사용을 지원하는 고객화 가능한 기본 활동과 착수-발견-결정의 스텝을 기반으로 하는 백로그 팩토링 기술을 제안할 수 있었다.

프로토타이핑을 통해, 전자 상거래 응용에 백로그 팩토링 기술을 적용하고 스크럼에서의 백로그 재사용을 보일 수 있었다. 본 기술은 시장 및 개발환경이 급변하고 재사용 준비가 부담되는 스크럼 개발에 재사용의 혜택을 제공할 수 있다.

앞으로의 연구 과제로, 백로그 팩토링 기술을 지원하는 소프트웨어 도구의 개발과 스크럼 개발에서 백로그 팩토링 기술의 이용을 다룰 것이다.

참 고 문 헌

[1] 김지홍 (2011), 애자일 방법에서 재사용을 지원하는 태스크 팩토링, *한국컴퓨터정보학회지* 16(11), 57-65.

[2] 켄 슈와버, 마이클 비틀, & 김기웅(역) (2008), 스크럼, 인사이트, 13.

[3] Diaz, J., Perez, J., Alarcon, P., & Garbajosa, J. (2011), Agile product line engineering : a systematic literature review, *SP&E*, May, 921-941.

[4] Diaz, J., Perez, J., Yague, A., & Garbajosaz, J. (2011), Tailoring the Scrum Development Process to Address Agile Product Line Engineering, JISBD.

[5] Ghanam, Y., Park, S., Maurer, F. (2008), A test-driven approach to establishing and managing agile product lines. *In Proceedings of SPLC (2)*, 151-156.

[6] Hanssen, G. K. (2011), Agile software product line engineering: enabling factors, *SP&E*, April, 883-897.

[7] Hanssen, G. K. & Figri, T. E. (2008), Process fusion: An industrial case study on agile software product line engineering, *Journal of Systems and Software*, June.

[8] Jacobson, I., Griss, M., & Jonsson, P. (1997), Software Reuse, Addison-Wesley, 439.

[9] Kircher, M. (2012), Combining Systematic Reuse with Agile Development Experience Report, *Proceedings of the 16th International SPLC*, 215.

[10] Krueger, C. W. (1992). Software reuse. *ACM Computing surveys*, 24 (2), 131-183.

[11] Martini, A., Pareto, L., & Bosch, J. (2012), Enablers and Inhibitors for speed with Reuse, *Proceedings of the 16th International SPLC*, 116-124.

[12] McGregor, J. D. (2008), Agile software product lines, deconstructed, *Journal of Object Technology*, 7(8), November-December.

[13] Mohan, K., Ramesh, B., Sugumaran, V. & Baruch C. (2010), Integrating Software Product Line Engineering and Agile Development, *IEEE Software*, May/June, 48-55.

[14] Paige, R., Wang, X., Stephenson, Z., & Brooke, P. (2006), Towards an Agile Process for Building Software Product Lines, *LNCS 4044*, 198-199.

[15] Pfleeger, S. L.(2010), Software Engineering, 4th ED., Pearson, 627.

[16] Pohl, K., & Linden, F. V. (2005), Software Product Line Engineering, Springer.

[17] Roth, R. M., Dennis, A., & Wixom, B. H. (2012), Systems Analysis and Design 5ed, Wiley, 57.

[18] Rubin, K. S. (2012), Essential Scrum: A Practical Guide to the Most Popular Agile Process, Addison-Wesley, 104-106.

[19] Schach, S. R. (2011), Object Oriented and Classical Software Engineering, 8th Ed., Pearson, 226.

[20] Silva, I., Neto, S., Almeida, D., & Meira, L. (2011), Agile software product lines: a systematic mapping study, *SP&E*, July, 899-920.

[21] Sommerville (2011), Software Engineering, 9th ED., Pearson, 58-72.

[22] www.scrum.org/Scrum_Guide.pdf

김 지 홍

- 1974: 경희대학교 전자공학과(공학사)
- 1982: California State University (Fullerton) Computer Science(이학석사)
- 1995: 경희대학교 전자계산공학과(공학박사)
- 1982~89: 미국 Interpac Software, 소프트웨어 엔지니어
- 98~현재: 가천대 컴퓨터공학과 교수
- 관심분야: 소프트웨어공학, UML, 소프트웨어 재사용, 애자일 개발
- E-Mail: wiskjh@gachon.ac.kr