논문 2020-2-18 http://dx.doi.org/10.29056/jsav.2020.12.18

오픈소스 SW 개발 방법론 소개 및 분석

손경아*, 윤영선**†

Introduction and Analysis of Open Source Software Development Methodology

Kyung A Son*, Young-Sun Yun**†

요 약

최근 인공지능과 빅데이터, 자연어처리 기술을 포함한 4차 산업혁명 관련 기술이 발전하면서 개인이나 팀 단독의 연구 및 개발 작업 방식의 한계를 가져오고 있다. 이런 한계를 극복하기 위하여 자신의 기술을 공개하고 협업하는 오픈소스 소프트웨어 개발 방식이 활발해지고 있으며, 회사 내부에서도 유사 개발 방법을 적용하여 회사 구성원들의 참여를 독려하고 품질을 개선하는 노력을 진행하고 있다. 이런 경향을 반영하여 IT 기술을 주도하는 선도 회사들은 적극적으로 오픈소스와 이너소스로 대변되는 공개 개발방식에 대한 지원 정책들을 제시하고 있다. 본 논문에서는 최근 활발히 논의되고 있는 오픈소스 모델, 이너소스 모델 및 그와 유사한 DevOps 모델을 소개하고 각각의 특징 및 구성 요소를 비교하였다. 비교 결과를 바탕으로 어떤 특정 모델의 우수성을 주장하는 것보다는 고객의 요구사항을 만족하면서 질적 향상을 도모할 수 있도록 각 장점에 따라 개인 또는 소속 기관의 소프트웨어 개발정책을 수립할 수 있을 것으로 판단한다.

Abstract

Recently, concepts of the Fourth Industrial Revolution technologies such as artificial intelligence, big data, and cloud computing have been introduced and the limits of individual or team development policies are being reviewed. Also, a lot of latest technology source codes have been opened to the public, and related studies are being conducted based on them. Meanwhile, the company is applying the strengths of the open source software development methodology to proprietary software development, and publicly announcing support for open source development methodology. In this paper, we introduced several software development methodology such as open source model, inner source model, and the similar DevOps model, which have been actively discussed recently, and compared their characteristics and components. Rather than claiming the excellence of a specific model, we argue that if the software development policy of an individual or affiliated organization is established according to each benefit, they will be able to achieve software quality improvement while satisfying customer requirements.

한글키워드 : 오픈소스, 이너소스, DevOps, 애자일, SW개발 방법론 비교

keywords : Open Source, Inner Source, DevOps, Agile, Comparison of SW Development Methodology

* 울산과학기술원(UNIST) U교육혁신센터
** 한남대학교 정보통신공학과
† 교신저자: 윤영선 (email: ysyun@hnu.kr)
접수일자: 2020.11.08. 심사완료: 2020.12.03.
게재확정: 2020.12.21.

1. 서 론

최근 인공지능과 클라우드, 연결성 등으로 대

표되는 제4차 산업혁명에 관한 관심이 높아지면서, 오픈소스(Open Source)라는 용어가 많이 회자되고 있다. 오픈소스는 정확히 오픈소스SW (Open Source Software, OSS)를 의미하며, 소스코드를 누구나 자유롭게 접근 및 확인, 수정, 배포할 수 있게 특정 저장소 (repository)나 공개된 장소 (homepage 등)에 공개한 소프트웨어를 말한다[1]. 초기의 오픈소스는 개발과정에서 공개된 소스코드를 의미하였으나, 점차 SW 개발 방법론으로 확장되어 사용되고 있다.

초기의 오픈소스는 “프리소프트웨어 (free software)”라고 불리었는데, 이 개념은 1983년 Richard Stallman이 GNU Project를 통해 소개한 개념으로 소스코드를 읽고, 수정하고, 배포하는 과정에서 사용자/개발자의 자유가 가장 중요하며 사용자가 필요한 방식으로 작업할 수 있게 해야 한다는 생각에서 출발하였다. 프리소프트웨어는 소유의 개념이 아니라 사용의 개념으로 시작하였지만, “프리”라는 용어로 인하여 “공짜(무료)”라는 의미로 오해하는 경우가 많이 발생하였다. 이에 Christine Peterson은 “프리소프트웨어”를 “오픈소스”라는 용어로 대체하여, 소프트웨어의 공유와 협력, 개방적인 개발 방법으로서의 개념을 제안하였다. 그 후, “오픈소스”는 프리소프트웨어의 방법론과 제작 및 비즈니스 측면을 옹호하는 용어로, “프리소프트웨어”는 사용자의 자유라는 철학적인 개념을 강조하기 위한 용어로 사용되기 시작하였다.

본 논문에서는 최근 주목을 받는 오픈소스 개발 방법론과 그 방법론을 회사 내부에 적용하는 이너소스 개발 방법론, DevOps 개발 방법론을 소개하고 차이점과 유사점을 비교 분석하고자 한다. 2장에서는 SW 개발 방법론으로써 전통적인 방법과 3가지 개발 방법을 소개하고, 3장에서는 소개한 방법들을 비교 분석하고, 4장에서 요약 및 결론을 맺고자 한다.

2. SW 개발 방법론

2.1 전통적인 SW 개발 모델

SW 개발 방법론은 SW 개발에 필요한 반복적인 개발 과정을 정리하고 표준화하여, 개발 과정에서의 일관성을 유지하고 프로그래머들 간의 효과적인 협업을 이루어질 수 있도록 돕기 위한 방법론이다. 이들 방법론은 구성요소와 전개 방식에 따라 여러 가지로 구분할 수 있다. 먼저 절차 중심의 소프트웨어 개발 방법론으로 코드를 제한된 구조에서 생성하여 순차적으로 실행시키는 구조적 방법론이 있으며, 설계와 구현 단계에서 데이터를 우선적으로 개발하고, 문제 영역을 세분화한 후 하향식으로 전개하는 정보 공학적 방법론, 분석과 설계 과정의 전 단계를 데이터 중심으로 개발하는 객체 지향 방법론, 문제를 조각으로 나누어 각각 컴포넌트를 생성한 후 다시 조합하여 재사용하는 CBD (Component Based Development) 분석 방법론, 고객과의 협력을 중시하고 프로세스나 도구에 국한되지 않는 자기 적응적 방식을 사용하고 일정한 주기마다 프로토타입을 만들어내는 애자일 방법론 등이 소개되고 있다. 여러 SW개발 방법론을 모두 소개하는 것은 어려우므로 구성이나 개발 방식에서 명확히 대비되는 폭포수모델과 애자일 모델을 설명하고자 한다.

- 폭포수 모델

폭포수 모델은 SW를 개발하는 단계가 위에서 아래로 물이 떨어지는 것처럼 순차적으로 진행된다고 명명된 방식으로 요구사항 분석, 설계, 구현, 테스트, 유지 보수 단계로 구성된다.

폭포수 모델의 특징은 순차적으로 각 단계가 진행되기 때문에 여러 단계가 병행적으로 진행되거나 거꾸로 진행되는 경우가 거의 없다는 것이

다. 따라서 개발 분야가 단순하거나 잘 알고 있는 경우에 적합하며 각 단계를 진행할 때마다 체계화된 문서가 작성된다. 일반적으로 개발과정을 각 단계별로 명확히 구분할 수 있기 때문에 SW 개발 노임 등을 산정할 때에 주로 인용되고 있다.

● 애자일 모델

애자일 모델은 폭포수 모델과 대비되어 실용적인 측면을 강조한 개발 방법론이다. 폭포수 모델과 달리 문서 작성을 축소하고 실제 개발을 권장하는 방법이다. SW 개발단계가 명확하게 구분되지 않아 개발과정을 체계적으로 관리하기 어렵다는 단점이 있으나, 개발단계가 반복적으로 수행되면서 요구사항 변경이나 설계 변경 등이 반영될 수 있으므로 대규모 SW 개발이나 완벽하지 않은 요구사항 분석 및 설계에 적합한 개발 방법론이라 할 수 있다. 또한, 문서 작성에 따른 부담이 적고 실제 개발을 지향하는 방법론이기 때문에 실제 작동하는 SW를 빠르게 확인할 수 있다는 장점으로 현대에 각광을 받는 방법론이다.

2.2 오픈소스 개발모델

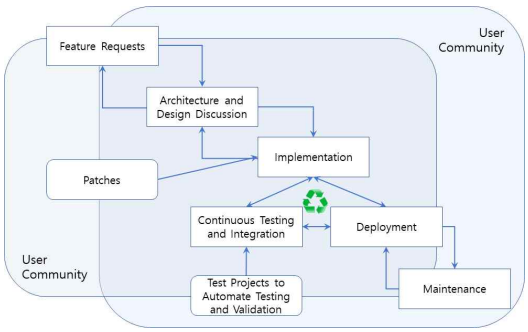

그림 1. 오픈소스 개발 모델[2][3]
Fig. 1. Open Source Development Model[2][3]

오픈소스 개발 방법론은 전통적인 순차적 소프트웨어 개발 방법론(예; 폭포수 모델)과 다른 특징을 보인다. 오픈소스 개발모델은 그림 1과 같이 사용자나 개발자로부터 요구되는 여러 특징에 기반한 새로운 프로젝트, 기존 오픈소스 SW나 제품의 기능 추가나 수정의 필요성으로부터 시작한다. 이러한 요구사항으로부터 구현에 대한 설계를 통하여, 동작하는 과정을 보이기 위한 프로토타입이 구현된다. SW가 동작하는 단계에서부터 여러 가지 버그가 포함된 상태라 할지라도 개발 버전이 배포되며, 검토 및 테스트단계를 거친다. 이 과정은 최근에 각광을 받는 소프트웨어 개발 방법론인 애자일 방식과 비슷한 특성을 가진 「조기 배포 및 반복적인 배포」 과정과 흡사하다. 애자일 방법론은 신속한 반복 작업을 통해 실제 작동 가능한 소프트웨어를 개발하여 지속적으로 제공하는 소프트웨어 개발 방법이다[4].

개발 버전으로 배포된 SW는 메일링 리스트나, 포럼, 게시판 등을 통하여 SW의 기능과 오류 등에 대하여 검토되고 피드백을 받게 된다. 이러한 피드백을 통하여 프로젝트 참가자는 각 기능과 오류를 검토하고 기능 개선을 진행하며, 다시 새로운 개발자 버전을 배포한다. 이 사이클은 프로젝트 참가자들이 SW 구현이 충분히 안정화(stable) 되었다고 판단할 때까지 반복되며, 안정화 판단이 결정되면 안정화 버전을 배포한다. 사용자나 커뮤니티로부터 요구된 새로운 기능이나 특징은 다시 수렴되며 새로운 개발자 버전을 개발하고 배포하는 과정과 안정화 과정을 반복적으로 거치게 된다.

이러한 과정으로 개발되는 오픈소스 모델은 표 1과 같은 독특한 특징을 가지며, 이들 특징에 기반하여 오픈소스의 가치를 판단할 수 있다. 오픈소스의 가치 판단기준은 사람마다 다를 수 있으나, 공통으로 중요성을 인정받는 가치 판단 기준은 표 2와 같이 정리할 수 있다.

표 1. 오픈소스 특징
Table 1. Open Source SW Characteristics

특징	설명
상향식 개발 (Bottom-up development)	● 프로그램 설계와 구축단계의 결정 단계에서 많이 기여한 사람이 중요한 역할을 담당함 ● 특정 목적이나 전략에 의하여 전체 프로그램을 설계하는 것이 아니라, 가장 많이 기여한 개발자의 뜻에 따라 프로그램의 설계와 구축 방향이 결정된다는 것을 의미 ● 코드개발이 프로그램의 정책이나 결정과정에 영향을 주기 때문에 상향식 개발로 간주할 수 있음
조기 배포, 빈번한 배포 (Release early, release often)	● SW를 공개하거나 배포하기 위하여 완전하게 동작하는 버전을 기다리지 않고, 어느 정도 기능이 구현되면 공개하거나 배포하여 동료 검토와 기능 제안, 버그 수정 등을 통하여 점진적 성능향상을 꾀함 ● 애자일 과정은 작동 가능한 SW 배포 후 비즈니스 요구사항이나 사용자 관점의 검토(오픈소스의 경우, 사용자 커뮤니티 검토)를 반영하기 때문에, 조기 배포 및 빈번한 배포는 애자일 과정과 유사
동료 검토	● 오픈 소스 프로젝트의 목적 및 소스코드를 이해할 수 있는 참가자들이 코드 검토를 하고, 기능 및 코드에 대한 주석과 피드백을 제공하기 때문에 버그를 조기에 수정할 수 있고, 기능을 쉽게 추가하고, 성능을 향상시켜 코드의 품질을 높일 수 있음
소규모, 점진적 개선	● 오픈소스 개발 모델에서는 종종 추가 기능이 아주 사소하며, 기존 기능에서 크게 변경되지 않음 ● 전체 코드를 대폭적으로 수정하거나 전체 구조를 재설계하지 않기 때문에 추가되는 코드를 쉽게 이해할 수 있음
보안 취약점 표시	● 오픈소스 커뮤니티는 보안을 매우 중요하게 여기기 때문에, 보안에 취약하거나 약점을 가지는 코드, 기능들은 면밀히 검토되고 있으며, 관련 보안 문제가 해결되지 않는다면 SW를 배포하지 않기 때문에, 보안성 높은 SW를 개발할 수 있음
지속적 품질 향상	● 광범위하고 심도 있는 동료 검토는 버그를 빠르게 수정할 수 있어서 지속적인 품질 향상 가능
테스트 자동화	● 대규모 오픈 소스 프로젝트를 위한 테스트 프로젝트가 종종 오픈 소스로 개발되고 있음 ● 자동화된 테스트와 테스트를 위한 환경을 조성 가능
전 과정 사용자 참여	● 개발자를 포함하여 프로그램 사용자가 전체 개발단계에 참여하여 요구사항을 추가하거나, 기능 개선 및 버그 수정 등에 기여할 수 있음

표 2. 오픈소스 가치 기준 및 판단 이유
Table 2. Value Criterions of Open Source SW

가치	판단 이유
동료평가	오픈소스는 누구나 접근할 수 있으며, 동일한 목적/목표를 달성하기 위해 여러 개발자가 적극적으로 검토 및 개선 작업에 참여하기 때문에, 소스코드를 공개한 개인/기관과 유사한 수준의 심사 및 검토가 가능
투명성	오픈소스를 사용하게 되면, 데이터의 흐름이나 변경사항 등이 공개되어, 개발자와 사용자가 확인 및 추적가능
안정성	여러 사용자에 의하여 다양한 환경에서 시험, 검토 및 확인되기 때문에 환경에 맞도록 지속적인 업데이트가 가능하며, 발생 가능한 다양한 경우의 위험 고려 가능
유연성	오픈소스는 다양한 환경에서 사용가능하기 때문에, 해당코드를 특정한 방식으로만 사용해야만 하는 제약이 존재하지 않으며, 새로운 환경에 쉽게 적용가능하며, 동료 개발자의 검토가 쉬움
경제성	오픈소스를 이용하여 개발하는 대부분의 경우 무료로 사용할 수 있기 때문에 비용을 절감가능
독립성	특정 업체의 제품을 사용하는 대신, 다양한 오픈소스로 대체할 수 있기 때문에 특정 업체에 대한 의존성이 감소
협력/협업	활발한 오픈소스 커뮤니티 덕분으로 폐쇄 그룹이나 기업에 의존하지 않고 다양한 지원, 자원, 관점을 접할 수 있음

2.3 이너소스 개발 방법론

이너소스는 오픈소스 개발 방법론을 회사 내부에 적용하여 자신들의 소프트웨어를 개발하고 배포하는 전략을 말한다. 이너소스는 2000년에 Tim O'Reilly에 의해 소개되었으며, CollabNet의 대표로 있으면서 오픈소스 스타일의 협업 개발을 소프트웨어 산업으로 가져가자고 주장하였다. CollabNet은 고객사가 오픈소스 소프트웨어를 전략적으로 이용하도록 도움을 준 최초의 회사이며, Hewlett-Packard와 Philips는 이너소스 프로그램을 적용한 최초의 고객이 되었다[5][6].

일반적으로 오픈소스를 말하면 공개와 투명성, 협업과 코드 재사용성, 동료 검토 등을 장점으로 말한다. 이들 오픈소스 개발 방법론의 장점과 모델을 회사 소유의 소프트웨어 개발에 적용하는 개발 방법론이 바로 이너소스이다. 이너소스는 두 가지 특성에 의하여 설명될 수 있다. (1) 개발 과정은 회사 내의 개발자 커뮤니티에 공개하고, (2) 개발 결과는 회사 소유이며, 소스코드에 대한 접근은 회사 내부로 제한한다. 이런 특성을 고려하여 그림 2와 같이 결과(상품)와 개발과정의 공개 여부에 따른 유형을 분류할 수 있다[7].

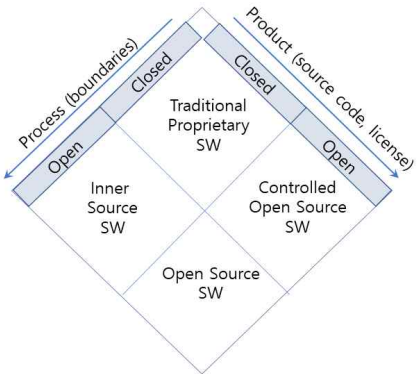

그림 2. 개발과정과 제품에 따른 SW 유형 분류[7]
Fig. 2. Classification of SWs by their process and product[7]

이너소스는 다양하게 설명되고 있으며, 조직내의 개발자가 이미 오픈소스를 사용하고 있더라도 이너소스라는 용어는 전혀 사용하지 않고, 대신 방화벽 뒤에서 오픈소스 개발 방법론을 적용한 소프트웨어 개발을 진행하고 있다고 설명한다. 일반적으로 많은 회사는 자신들의 소프트웨어 자산을 공개하거나 외부의 개인들이 자신들의 소스코드를 검토하거나 이너소스 프로젝트에 접근하는 것을 원치 않는다. 즉, 회사는 공개되지 않은 소스코드가 회사 내의 개발 환경에서 안전하게 유지되기를 원하고 각 개발자가 적절한 접근 권한을 가지고 기여하기를 원한다. 그렇지만 회사 내부의 프로젝트를 진행하면서 오픈소스 개발 방법론의 장점을 적용하기를 희망한다. 오픈소스 모델을 적용하면 투명한 협력 체제를 구축할 수 있으며, 더 낳은 소프트웨어를 구축할 수 있는 지식이나 기술들을 축적할 수 있기 때문이다.

만약 특정 조직 내에서 소프트웨어를 구축한다고 하면, 아마도 오픈소스 프로젝트를 이용하여 자신들의 환경에 맞게 수정하거나 그 위에서 새로운 기능을 추가할 것이다. 삼성 오픈소스 컨퍼런스 2018에서 리눅스 재단 대표인 짐 젠린이 오픈소스의 활용도는 전체 SW 중 약 90%이상을 차지하고 있으며, 10%의 사용자 코드가 앱이나 SW의 가치를 차별화한다고 한 것처럼, 오픈소스 없는 소프트웨어 개발은 생각할 수가 없다.

그렇다면 왜 회사들은 이너소스를 도입하려고 할까? 그것은 이너소스를 도입하게 되면, 소프트웨어를 더 빠르게 구축할 수 있고 협업을 효율적으로 진행할 수 있기 때문이다. 특히 개발 자체의 품질을 향상하고 문서화 작업(기록)을 더 잘 할 수 있다. 특히 다음과 같은 이유로 회사는 SW 개발에서 효율성을 향상할 수 있다.

- 자원의 중복과 낭비를 피하고, 코드를 쉽게 파악할 수 있으며 재사용이 가능하다.
- 회사 규모와 상관없이 개발 속도를 높일 수

있다.

- 사일로 문화를 줄일 수 있으며, 팀 내외부를 막론하고 전체 팀과 사업 조직을 총망라한 전체 조직을 통한 협업을 단순화시킬 수 있다.
- 엔지니어와 경영자 또는 관심이 있는 누군가든 투명성/명료성을 증가시킬 수 있다.
- 오픈소스 프로젝트에 대한 참여를 독려하고 개방 문화를 조성할 수 있다.
- 필요한 적재적소에 팀원들이 도움을 줄 수 있고, 그것으로부터 자부심, 성장, 직무만족도 등을 향상시킬 수 있다.

PayPal이나 Bloomberg, 그리고 Walmart 등이 자신들의 고객과 팀을 위해서 이너소스를 채용하고 있으며, 자신들만의 경쟁 이점을 제공하고 모범 사례를 도입함으로써 회사가 관련성을 유지할 수 있도록 도움을 주고 있다.

2.4 DevOps 모델

DevOps는 개발 (Development)과 운영(Operation)의 합성어로, 고객에게 응용과 서비스를 빠른 속도로 제공할 수 있도록 조직의 역량을 향상하는 문화 철학, 방식 및 도구의 조합이다. 전통적인 개발과정에는 개발자와 운영 전문가들이 분리되어 각각의 부서 이익에 맞는 활동을 진행하였다. 운영팀은 개발이 완료된 제품에 대한 사후 관리를 담당하였으며 제품의 개발 역할은 기능을 요구하는 등의 소극적 참여에 한정되었다. 그러나 DevOps에서는 개발팀과 운영팀은 서로 밀접한 협업을 진행하고 전 단계에 공동으로 참여하게 된다. 즉, 개발과정에서 DevOps는 개발, IT 운영, 품질 엔지니어링, 보안 등 서로 단절된 역할들을 조율하고 협업하여 안정적이고 뛰어난 제품을 생산할 수 있도록 지원한다. 따라서 기존의 소프트웨어 개발 및 인프라 관리 프로세스를 사용하는 조직보다 제품을 더 빠르게 혁신하고 개선할 수 있다. 빠른 개발 속도와 고객에 대한 효과적인 대응을 통해 회사는 고객을 더 잘 지원하고 시장에서도 효과적으로 경쟁할 수 있다 [8][9].

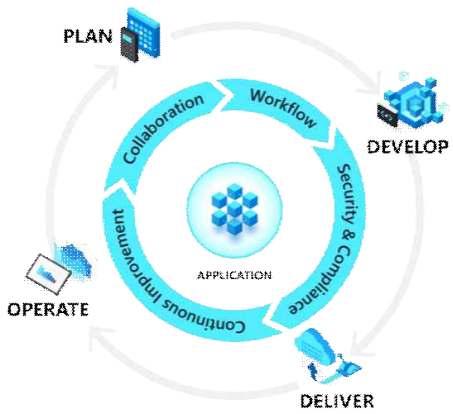

그림 3. DevOps와 응용 생명 주기[9]
Fig. 3. DevOps and application lifecycle [9]

DevOps는 계획(plan), 개발(develop), 제공(deliver), 운영(operator) 단계 전반에 걸쳐 응용 프로그램의 개발에 영향을 준다. 각 단계는 서로 영향을 주며 개발이나 운영의 특정 역할에 한정되지 않는다. DevOps 모델의 중요한 특징은 개발팀과 운영팀이 부서 이기주의에서 벗어나, 단일팀처럼 활동하여 엔지니어가 개발에서 테스트, 배포, 운영에 이르기까지 전체 응용 개발과정에 걸쳐 작업할 수 있으며, 단일 기능에 한정되지 않는 광범위한 기술을 개발할 수 있다는 점이다.

DevOps와 응용 개발의 생명주기를 간단하게 다음과 같이 요약할 수 있다. 프로그램 개발(develop)단계에서는 개발자들이 코드를 작성하고, 검토 및 통합하는 과정뿐만 아니라 다양한 환경으로 배포할 수 있는 작업이 포함된다. DevOps 팀들은 품질, 안정성 및 생산성의 저하 없이 신속하게 혁신할 방안을 모색하며, 생산성

이 뛰어난 도구를 사용하고 자동화된 시험 및 지속적 통합을 통하여 작은 개발 주기로 각 단계를 반복한다. 제공(deliver) 단계는 일관성 있고 안정적인 방식으로 제품을 배포한다. 개발팀은 제공 단계를 자동화하여 확장성과 반복성, 관리 용이성을 향상시킬 수 있다. 운영(operate) 단계는 응용 프로그램을 관리, 모니터링하고 문제를 해결하는 작업이 이뤄진다. DevOps를 도입한 팀들은 시스템 안정성과 고가용성을 보장하기 위해 협력하고, 가동 중단 시간을 최소화할 수 있다. DevOps를 사용하면 속도가 느리고 수동으로 수행되던 공정을 자동화할 수 있다. 이외에도 DevOps의 몇 가지 장점을 표 3에 정리하였다.

표 3. DevOps의 장점 및 이유
Table 3. Advantages of DevOps

장점	내용
속도	작업속도가 빨라져서 고객의 요구사항을 쉽게 반영할 수 있으며, 시장변화에 더 잘 적응하고 효율적으로 사업성과를 도출할 수 있음
신속한 제공	소프트웨어의 배포 빈도와 속도를 개선하여 제품을 더 빠르게 수정하고 개선가능 새로운 기능의 배포와 버그 수정 속도가 빨라질수록 고객의 요구에 더 빠르게 대응하여 경쟁 우위 강화 가능.
안정성	최종 고객에게 지속적으로 긍정적인 경험을 제공할 수 있고, 더욱 빠르게 안정적으로 제공할 수 있도록 응용의 업데이트와 인프라 변경의 품질 보장 가능
확장	회사 규모에 따라 인프라와 개발 프로세스를 운영 및 관리 가능
협업강화	개발자와 운영팀은 긴밀하게 협력하고, 많은 책임을 공유하며 작업 흐름을 결합하기 때문에 비효율성을 줄이고 시간을 절약할 수 있음
보안	자동화된 규정 준수 정책, 세분화된 제어 및 구성관리 기술을 사용함으로써 보안을 유지하면서 DevOps 모델 도입가능

DevOps 방식을 도입하면 조직 내부의 문화뿐만 아니라 인력 관리까지도 포함하여 개발단계의 공정을 자동화 및 최적화할 수 있다. 그러나 DevOps 문화를 조성하기 위해서는 사람들의 작업 및 협업하는 방식이 근본적으로 변화해야 한다. DevOps 문화의 구성 요소와 그 영향을 표 4와 같이 정리할 수 있다.

표 4. DevOps 문화 구성요소와 영향
Table 4. Components and Effects of DevOps Cultures

요소	영향
협업, 투명성 및 조율	· 효과적인 협업은 투명성에서 출발한다. 개발 팀, IT 운영 팀 등 여러 팀은 팀의 DevOps 공정, 우선순위, 우려 사항을 다른 팀과 공유해야 하며, 효과적인 협업을 계획하는 한편 사업 관련 목표와 성과 측정 기준을 조율해야 함
팀 역할 범위와 책임의 변화	· 팀들은 조율 과정에서 팀 역할에 핵심적인 생명 주기 단계뿐만 아니라 부차적인 단계에도 주도성을 가지고 관여하게 됨. · 개발자는 개발 단계에서 설정한 혁신 및 품질뿐만 아니라 변경 내용이 운영 단계에 미치는 성능과 안정성도 고려해야 함. · IT 운영자들은 계획 및 개발 단계에서 관리, 보안, 규정 준수들을 적용가능
짧은 제공 주기	· DevOps 팀이 제공 주기를 짧게 가져가게 되면 개선 사항이 작은 변화 단위로 누적되기 때문에 계획과 위험 관리가 쉬어지게 되고 응용 프로그램의 이해도가 높아짐 · 전체적인 시스템의 안정성에 미치는 영향이 줄어들게 되므로 그 효과를 정확히 확인할 수 있어 안정성 또한 향상됨 · 제공 주기가 짧아지게 되면서 변화하는 고객의 요구사항에 경쟁 업체의 압박 등에 유연하게 대응가능
지속적인 학습	· 높은 성과를 내는 DevOps 팀은 긍정적 사고방식을 갖게 됨. · 이들은 실패로부터 빠르게 학습하고 공정에 적용하여 끊임없이 개선하고 고객 만족도를 높일 수 있어 혁신을 가속화하고 시장 적응력을 향상시킬 수 있음

3. 개발 방법론 비교

폭포수 모델과 애자일 모델과 같은 전통적인 소프트웨어 개발 방법론들은 관련 연구가 많이 소개되었기 때문에, 두 개발 방법에 대한 비교보다는 최근의 개발 방법론인 오픈소스, 이너소스, DevOps에 대한 비교 및 분석에 초점을 맞추고자 한다.

오픈소스는 소스코드를 공개하여 누구나 자유롭게 접근 및 확인하고 자신의 목적에 맞게 수정 및 배포할 수 있도록 하는 작업방식을 말한다. 오픈소스의 가장 큰 특징은 투명성과 커뮤니티를 통한 협업으로 정의할 수 있으며 이를 통하여 최종 결과물의 안정성과 양질의 성과를 얻을 수 있다. 특히 오픈소스가 제안되던 시기의 전통적인 소프트웨어 개발 방법과 다른 특징을 보이고 있으며, 최근 고객의 요구와 시장 변화에 능동적으로 대응하는 애자일 방식과 비슷한 개발 방식을 보인다. 오픈소스 개발 방식은 작은 프로토타입으로부터 출발하여, 수정 및 기능 개선이 작은 단위로 진행되며 고객의 만족도를 신속하게 개선하는 방식으로 각광을 받고 있다.

부서 간의 이기주의를 타파하고, 투명성과 협업, 문서화 작업 등의 오픈소스 개발 방식의 장점을 도입하여 회사 소유의 소프트웨어 개발에 적용하는 이너소스는 개발과정을 회사 내부에만 공개하고, 개발 결과의 소유권 및 소스 코드의 접근 권한을 회사 내부로 규정하고 있다. 오픈소스 개발 방식의 경우 방향성 설정이나 기능에 대한 요구사항은 초기 프로젝트를 주도한 개발자나 소스코드에 많이 기여한 사람이 주도적으로 진행하나, 이너소스 소프트웨어의 경우 내부 위원회나 부서의 목적에 따라 방향성이 설정되기 때문에 오픈소스 개발 방법론과 유사하면서도 독특한 특성이 존재한다. 이너소스 소프트웨어 개발이 오픈소스 개발 방법론을 회사에 적용하였다 할지라도 커뮤니티 기반의 협업 방식이 아닌 각 개발 부서나 사업 부서의 역할을 넘는 참여가 확보되고, 자발적 참여를 회사에서 어떻게 지원하느냐에 따라 이너소스 소프트웨어의 적용 성패 여부가 결정된다. 즉, 이너소스 소프트웨어 개발 방법은 오픈소스 소프트웨어의 개발 방법론뿐만 아니라 조직의 지원, 예산, 인력 분배 등에 여러 면에서 고려할 사항이 많다.

마지막으로 개발팀과 운영팀의 밀접한 상호 협력 하에 고객의 요구사항과 시장의 변화에 능동적으로 대처하는 DevOps 방식을 살펴보았다. 기존의 오픈소스 개발 전략이 명시적이지 않은 개발자와 사용자의 협업이고, 이너소스 개발 방법론이 회사 내부에서 오픈소스 개발 방법론을 적용한 협업이라면 DevOps는 개발과 운영의 효과적인 협업에 대한 모델이라 할 수 있다. DevOps는 오픈소스나 이너소스 등의 소프트웨어 개발 전략을 사용할 수 있으나 운영, 품질 관리, 보안 등을 결합한 개발 방식으로 생각할 수 있다. 결국 오픈소스 개발 방법론이나 이너소스 개발 방법론, DevOps 개발 방법론은 별개의 독립적 개발 방식이 아니라 효과적인 소프트웨어 개발과 고객의 요구와 시장의 변화에 대응하는 유연하고 통합된 개발 방식으로 상호 보완적인 관계라 할 수 있다.

위에서 요약한 3가지 방법론을 자세히 관찰하면 몇 가지 공통점을 찾을 수 있다. 바로 순차적 개발모델이 아닌 점진적 개발모델이며, 최종 사용자 또는 고객, 커뮤니티의 의견을 반영하여 지속적인 개선과 통합과정을 거친다는 것이다. 물론, 개발참여자와 공개범위, 프로그램 개발 방식 등이 일부 다를 수 있지만, 안정성, 확장성, 사용자 참여와 협업, 투명성, 유연성 등은 공통으로 3가지 방법론을 설명하는데 중요한 용어가 될 수 있다.

전통적인 소프트웨어 개발모델들과 최근 활발

표 5. SW 개발 모델 비교
Table. 5. Comparison of SW development methodology

	폭포수모델	애자일모델	오픈소스모델	이너소스모델	DevOps모델
제안시기	1950년대	1970년대	1980년대	2000년[10]	2008~9년[11]
개발정책	하향식	하향식+상향식	상향식	하향식	하향식
개발방법	순차적	점진적	점진적	점진적	점진적
개발구조	구조적	유연성	유연성	유연성	유연성
개발초점	프로젝트 완료	속도	확산	참여	속도+자동화
요구사항	변경 어려움	변경 가능	변경 가능	변경 가능	변경 가능
요구주체	고객	고객	커뮤니티	내부 구성원	자체
팀구성	매우 제한적 개발팀	전담 개발팀	공개	전담 개발팀 + 내부 구성원	개발+운영팀
공개범위	개발팀	개발팀	전체	내부 구성원	개발+운영팀
시험단계	명시적	묵시적	명시적+묵시적	명시적+묵시적	명시적+묵시적
사업분석	시작 전	개발 전과정	-	개발 전과정	개발 전과정
사업적합성	중	중	하	중	상

히 연구되고 있는 오픈소스, 이너소스, DevOps 모델들을 특징별로 표 5와 같이 정리할 수 있다.

4. 검토 및 결론

최근 4차 산업혁명 관련 용어가 소개되고, 인공지능, 클라우드, 로봇, 빅데이터 등과 같이 최신의 기술이 날로 발전하면서 개인이나 팀 독립단위의 연구 및 개발 작업방식의 한계를 가져오고 있다. 이에 따라 최근 인공지능과 빅데이터, 자연어처리 기술이 발전함에 따라 많은 소스 코드들이 공개되고 오픈소스 개발 모델에 따라 관련 연구가 진행되고 있다. 또한 회사 내부에서도 오픈소스 소프트웨어 개발 방법론의 장점을 적용하여 회사 구성원들의 참여를 독려하고 품질을 개선하는 시도가 진행되고 있으며, 공개적으로 오픈소스와 이너소스에 대한 지원 정책들을 제시하고 있다.

본 논문에서는 오픈소스 모델과 소프트웨어 개발모델의 진화에 따른 각 모델을 소개하고, 널리 알려진 전통적인 폭포수 모델과 애자일 모델과의 비교를 진행하였다. 분석 및 비교 결과는 매우 주관적일 수 있으나 최근 소프트웨어 개발 정책의 흐름인 공개와 투명성, 참여와 협업, 유연과 확장성 등의 개념은 이제 소프트웨어 개발과정에서 중요한 고려사항이 되고 있다. 따라서 본 논문이 소프트웨어 개발 흐름을 파악하는 데 조그만 도움이 되길 바라면서, 개발자들의 자발적인 참여와 협업 속에서 더 좋은 기술개발들이 공개되어 많은 사용자가 혜택을 받을 수 있는 소프트웨어 생태계가 구축되기를 기원한다.

참 고 문 헌

[1] RedHat, What is Open Source?, 2020.03.10. https://www.redhat.com/ko/topics/open-source/what-is-open-source
[2] Brahim Haddad, The Open Source Development Model: Overview, Benefits and Recommendations, 2020.03.10. http://aaaea.org/Al-muhandes/2008/February/open_src_dev_model.htm

[3] Ibrahim Haddad and Brian Warner, Understanding the Open Source Development Model, The Linux Foundation, 2011. http://www2.thelinuxfoundation.org/understanding-open-source-development-model
[4] RedHat, What is agile methodology?, 2020.03.10. https://www.redhat.com/ko/devops/what-is-agile-methodology
[5] Github whitepaper, An introduction to innersource, 2018.01.22. https://resources.github.com/whitepapers/introduction-to-innersource/
[6] D. Cooper and K.-J. Stol, Adopting Inner Source: Principles and CAse Studies, O'Reilly, June, 2018. ISBN: 9781492041856. https://www.oreilly.com/library/view/adopting-innersource/9781492041863/
[7] M. Host, K.-J. Stol and A. O.-A., Inner Source Project Management, Software Project Management in a Changing World, 2014. https://doi.org/10.1007/978-3-642-55035-5_14
[8] Amazon, What is DevOps?, 2020.03.10. https://aws.amazon.com/ko/devops/what-is-devops/
[9] Microsoft, What is DevOps?, 2020.03.10. https://azure.microsoft.com/ko-kr/overview/what-is-devops/
[10] Chapter 1. The InnerSource Approach to Innovation and Software Devleopment, 2020.11.08. https://www.oreilly.com/library/view/adopting-innersource/9781492041863/ch01.html
[11] Steve Mezak, The Origins of DevOps: What's in a Name?, Jan. 25, 2018. https://devops.com/the-origins-of-devops-whats-in-a-name/

저 자 소 개

손경아(Kyung A Son)

2003.2 한양대학교 교육공학박사
2003.11-2014.1 한국방송통신대학교 책임연구원
2005.9-2008.8 한양대학교 컴퓨터교육과 겸임교수
2014.2-2015.12 국가평생교육진흥원 국정과제추진단 부단장
2015.1-현재 UNIST U교육혁신센터 연구교수
<주관심분야> 에듀테크, 컴퓨터교육, 멀티미디어, 학습관리시스템(LMS), 이러닝 개발 및 운영 등

윤영선 (Young-Sun Yun)

1990.2 KAIST 전산학과 졸업
1992.2 KAIST 전산학과 석사
2001.2 KAIST 전산학과 박사
2006.4-2007.2 한국전자통신연구원 초빙연구원
2012.8-2013.7 University of Washington 방문학자
2001.3-현재 : 한남대학교 교수
<주관심분야> 음성인식, 음성변환, 화자인식, 인공지능, 내장형시스템, 저작권침해, 유사도, 완성도 감정, 오픈소스 등