애자일 방법론의 동향 및 ESA 지상국 소프트웨어 개발 분야의 도입사례 분석

서석배*, 강지훈**

An Analysis of Agile Methodologies' Trends and Introduction cases of the Methodologies at the ESA Ground Segment Software Development

Seok-Bae SEO* and Jihoon KANG**

Abstract

Agile is a software development methodology which was established for four objectives; people, collaboration, responsiveness, and working software. Thus the Agile methodologies have been applied not only software development fields but also various special areas of technology. ESA (European Space Agency) adopt several Agile methodologies, including Scrum as the key technique, for the software development of ground segment. This article analyzes trends of Agile methodologies and introduction cases of the methodologies for ESA's ground segment software development.

초 록

애자일(Agile)은 논증 가능한 구체적인 목표(사람, 협조, 반응성, 작동하는 소프트웨어)를 위해서 창시된 소프트웨어 개발 방법론으로, 소프트웨어 엔지니어링뿐만 아니라 여러 전문 분야에 적용되고 있다. ESA (European Space Agency)의 경우 애자일의 대표적인 방법인 스크럼 (Scrum)을 중심으로 지상국 분야의 소프트웨어 개발에 애자일을 적용하였다. 본 논문에서는 애자일 기술의 동향을 살펴보고, 이를 도입한 ESA 지상국 소프트웨어 개발의 최근 사례를 분석한다.

키워드 : 애자일(Agile), 스크럼(Scrum), 소프트웨어(Software), 지상국 개발(Ground Segment Development), 소프트웨어 개발 방법론 (Software Development Methodology)

1. 서 론

소프트웨어 개발 실패의 원인을 분석하고 개선하는 노력의 결과로 문서화의 절차와 규격, 버그의 처리 과정, 통합 테스트의 절차, 소프트웨어 품질의 보장 방법 등과 관련한 표준과 권고사항

접수일(2012년 5월 15일), 수정일(1차 : 2012년 6월 14일, 게재 확정일 : 2012년 7월 1일)
* 위성지상시스템개발팀/sbseo@kari.re.kr ** 위성지상시스템개발팀/jhkang@kari.re.kr

이 제안되었으며[1] 이는 소프트웨어 개발 방법론 (Software Development Methodology)으로 정립되었다. 여기에서 방법론 (methodology)은 모델 (model), 방법 (method), 실천법 (practice), 기술 (technique), 프레임워크 (framework) 등의 용어로 혼용하는데, 본 논문에서는 모델로 통일하여 기술한다.

소프트웨어의 기능과 복잡도는 지속적으로 증가하고 있다. 초기의 소프트웨어 개발 프로젝트가 기능과 복잡도 관리를 위한 프로세스의 결핍으로 인해서 어려움을 겪었다면, 최근에는 프로세스의 과잉이 실패의 원인이다[1]. 애자일 모델은 프로세스 과잉을 줄이면서 기능과 복잡도 관리를 할 수 있는 해결책으로 주목받고 있는데, 이를 제대로 활용하기 위해서는 기존의 기술 (소프트웨어 개발 모델) 및 애자일 모델 등장의 이유를 이해하는 것이 선행되어야 한다.

본 논문에서는 먼저 기존 소프트웨어 개발 모델을 살펴보고 (2장), 현재 소프트웨어 개발 모델로 주목 받고 있는 애자일 모델의 동향에 대해서 설명한다 (3장). 이후 최근 애자일 모델을 지상국 소프트웨어 개발에 도입한 ESA의 사례를 분석한다 (4장).

2. 소프트웨어 개발 모델

소프트웨어 개발에서 완전한 모델은 존재하지 않으며 계속 발전하고 있다. 따라서 본 장에서는 소프트웨어 개발 모델의 근본이 되는 폭포수 모델에 대해서 설명하고, 이에 근간하여 발전한 두 가지 모델 (변형된 폭포수 모델, 애자일 모델)에 대해서 정리한다.

2.1 Waterfall Model

포수 모델 (Waterfall Model)은 '착수, 요구사항 분석, 기초설계, 상세설계, 코딩, 단위테스트, 시스템테스트, 설치와 인도, 유지보수와 성능개선'의 개발 단계 (9개)로 구성된다. 각 개발 단계에서, 다음 단계로 입력은 주지만 이전 단계로의 피드백은 없다[2].

폭포수 모델에서는 상위 계발 단계인 요구사항 분석과 설계 단계에 완벽을 기하면 신뢰할 수 있는 소프트웨어를 만들 수 있다는 생각이 지배적이다. 미 국방성 2167 표준에서는 요구사항 분석과 설계에 필요한 문서의 템플릿을 제공해주며, 프로젝트 성격에 맞게 2167 표준을 어떻게 변형 (tailoring)할지는 DOD-HDBK-287에 기술하고 있다[2].

폭포수 모델의 도입으로 '코딩부터 시작하던 소프트웨어 개발'의 단점이 극복되었으나[2], 개선된 폭포수 모델의 출현으로 미루어 여전히 문제점이 존재함을 예상할 수 있다. 폭포수 모델에서, 초기에 결정되는 요구사항이 불완전하다는 것과 사용자와 개발자 사이의 의사소통이 어렵다는 것이 대표적인 문제점이다[2].

또 하나의 시각으로는 폭포수 모델은 결코 모델이 될 수 없으며 오히려 프로젝트 관리 기법이라고 주장하고 있다. 따라서 사용자가 필요할 때마다 임의의 모델을 적용할 수 있다고 이해할 수 있다[2]. 프로토타이핑 모델은 폭포수 모델의 외부에서 효과적으로 사용되는 모델로 폭포수 모델을 적용하기 전에 사용하거나 추가적인 개발이나 유지보수를 하려고 폭포수 모델이 종료된 시점에 사용한다. 그리고 폭포수 모델 수행 중에 요구사항을 추가하기 위해서 사용하기도 한다[2].

2.2 Modified Waterfall Model

폭포수 모델에서는 각 단계가 완벽하지 않을 때 다음 단계 수행에 문제가 발생한다. 특히 테스트 단계에서 (상위 단계의) 문제가 발견되는 경우가 많다. 문제 유발의 원인이 소프트웨어 제작일 수도 있지만, 불완전한 요구사항과 개발자와 사용자의 의사소통의 부재로 인하여 발생하는 경우가 많으며 이 문제에 대한 해결은 더욱 어렵다. 이러한 폭포수 모델의 단점은 다음의 두 모델 (반복 모델, 사시미 모델)에서 개선하고자 노력하였다.

2.2.1 Iteration Model

반복 모델 (Iteration Model)은 소프트웨어 개

발 프로젝트의 원활한 진행을 위해서 여러 번에 걸쳐 개발 단계를 반복하여 수행하는 것이다. 각 개발 단계 (폭포수 모델)를 반복적이고 점진적으로 진행하기 때문에 요구사항이 점점 명확해지고 개발자가 기술 및 요구사항에 익숙해져서 우수한 소프트웨어를 개발할 수 있다[3]. 폭포수 모델에 비하여 시간과 비용이 추가되는 단점이 있지만, 테스트 단계에서 발생하는 문제를 분산시키고 요구사항이 점진적으로 구체화되는 장점을 가진다.

2.2.2 Sashimi Model

사시미 모델 (Sashimi Model)은 폭포수 모델에서 개발 단계를 축소하고 각 단계를 겹치게 한 것이다. (회를 겹치게 배열한다는 데에서 이름이 유래되었음) 여기서 겹치는 부분은 폭포수 모델의 일부가 아니라 확인절차를 의미하는데, 이를 통과하지 못하면 그 전 개발 단계로 돌아간다[2]. 사시미 모델에서 개선된 점은 겹치는 부분에서 의사소통과 통합을 밀접하게 하고 인접한 개발 단계 사이에서 협력을 높이는 것이다[2].

2.3 Agile Model

폭포수 모델은 소프트웨어 개발과정을 모델로 정립하여 체계화하는 업적을 이루었다. 하지만 기능과 복잡도가 증가하는 소프트웨어 개발에서 의사소통이 원활하지 않아서 부차적인 문제를 유발하였다. 애자일 모델 (Agile Model)은 기존의 모델을 거부하는 것이 아니라 이를 적극적으로 수용하되[1], 과정을 매우 잘게 분해해서 가벼운 형태로 변경하고 의사소통을 향상시키기 위한 노력이다. 애자일 모델에 대해서 3장에서 자세히 기술한다.

3. 애자일 모델동향

2001년 2월, 소프트웨어 개발 모델의 돌파구를 모색하던 17명은 서로 다른 모델 사이에 존재하는 공통의 철학을 논의하기 위해서 미국 유타주 스노우버드 스키리조트에 모였다. 이들의 공통점은 사람이 중심이 되는 새로운 협력모형을 찾는 것이었으며[1], 스크럼 (Scrum), 익스트림 프로그래밍 (eXtream Programming), 기능중심개발 (Feature-Driven Development), 실용주의 프로그래밍 (Pragmatic Programming), 크리스탈 (Crystal) 등 다양한 분야의 전문가들이었다. 이 모임을 통하여 애자일 소프트웨어 개발을 위한 성명서 (Manifesto for Agile Software Development)가 합의되었으며, 12가지 가치 원칙이 제정되었다[4]. 애자일 모델의 주요 가치는 다음과 같다[4].

- 소프트웨어 개발자들 간의 상호작용
- 작동하는 소프트웨어
- 고객과의 협력
- 변화에 대응

애자일 컨설팅업체 VersionOne의 2009년 7월에서 12월까지의 조사에 따르면, 애자일 모델의 적용 중 스크럼이 50%, 스크럼과 익스트림 프로그래밍의 조합이 24%를 차지하고 있었다. 본 장에서는 대표적인 애자일 모델인 스크럼과 익스트림 프로그래밍에 대해서 설명하고, 스크럼과 익스트림 프로그래밍을 결합한 애자일 엔터프라이즈를 간단히 소개한다.

3.1 Scrum

럭비의 대형에서 명칭이 유래한 스크럼 (Scrum)은 성공한 소프트웨어 개발에 대한 분석이 그 출발점이었으며, 현재 애자일의 대표적인 모델이 되었다.

스크럼은 단순하면서도 기존 모델의 장점을 포함하고 있어 애자일 모델 적용에서 많이 선택된다. 스크럼을 도입하게 되면 즉시 많은 문제점이 발견되는데, 이는 스크럼 도입에 따른 것이 아니라 소프트웨어 개발 후반부에 생기는 문제들이 미리 드러나는 것으로 이해할 수 있다. 발견된 문제는 더 이상 문제가 아니다. 왜냐하면 스크럼에서는 소프트웨어 개발이 예상하기 힘들다는 것 (사용자 요구사항 및 소프트웨어 요구사항을 변경하는 것)을 기본적인 전제로 하기 때문이다.

스크럼은 스크럼 마스터를 주축으로 스프린트를 만들고, 일일 스크럼회의, 스프린트 검토회의, 배포 스프린트 회의 등을 통하여 소프트웨어를 개발한다. 스크럼에서는 소프트웨어의 기능, 특성, 기술을 나열하여 제품 백로그로 관리한다. 스크럼에서 주요 항목에 대한 설명은 다음과 같다.

- 제품 백로그 (Product Backlog) : 우선순위가 매겨진 요구사항으로 ID, 이름, 중요도 (각 항목의 상대적인 중요도 설정), 최소추정치 (업무 처리에 필요한 예상 시간), 데모방법, 참고사항이 핵심 항목이다[4]. 제품 백로그는 결코 확정되지 않고 소프트웨어 개발과 함께 진화하며, 제품소유자 (Product Owner, PO)만이 중요도의 순서를 결정 및 변경할 수 있다.

- 스프린트 (Sprint): 전체 업무를 분할하는 단위로 보통 30일이다. 하나의 스프린트가 완성되고 나면 제품 백로그를 업데이트 한다.

- 스크럼 마스터 (Scrum Master) : 스크럼을 실천하도록 관리하며, 결정을 내리거나 방해요소를 제거하도록 지원하는 역할을 수행한다.

- 일일 스크럼 회의 (Daily Scrum Meeting) : 매일 짧은 시간 진행하며 (의자에 앉지 않고 서서 수행하는 회의도 많이 이용됨), 개발에 직접 참여하는 사람들이 돌아가며 세 가지 주제에 대해서 이야기한다[5]: 1) 지난 일일 스크럼 회의 이후 무엇을 했고, 2) 다음 일일 스크럼 회의 전까지 무엇을 할 계획이며, 3) 무엇이 작업을 방해하고 있는가.

- 스프린트 점검회의 (Sprint Review Meeting) : 스프린트 점검회의는 개발자뿐만 아니라 관리자도 함께 참석하여 진행사항을 검토하고 업무를 진행함에 있어서 문제점을 확인하고 제거하는 방법을 모색한다.

- 배포 스프린트 (Release Sprint) : 제품이 잠정적으로 배포 가능하다고 판단되는 시점에 배포를 대비하여 수행되는 스프린트이다.

3.2 eXtream Programming, XP

익스트림 프로그래밍(eXtream Programming, XP)은 고객에게 최고의 가치를 가장 빨리 전달하려는 목적으로[3], 단순성 (simplicity), 상호소통 (communication), 의견 (feedback), 용기(courage)에 원칙을 두고 있다. 이를 위해서 페어 프로그래밍, 테스트주도 개발, 지속적 통합을 핵심 실천사항으로 설정하였다.

- 페어 프로그래밍 (Pair Programming) : 익스트림 프로그래밍의 개념을 창시한 켄트 백 (Kent Beck)이 익스트림 프로그래밍의 실천을 위한 최고의 아이디어로 제시한 것으로, 두 팀에서 한 명씩의 인원이 소프트웨어를 동시에 개발하는 것이다. 한 명은 소프트웨어 제작을 주도권을 가지고 나머지 한 명은 소프트웨어 제작이 원활하게 이루어지도록 돕는 역할을 한다. 이 때 각각의 역할을 수시로 바꿔어야 한다.

- 테스트 주도 개발 (Test-Driven Development, TDD) : 실제 소프트웨어를 작성하기 전에 테스트 소프트웨어를 먼저 작성하는 방법으로, 론 제프리 (Ron Jeffries)가 정의한 '잘 동작하는 깔끔한 코드 (clean code that works)'가 목표이다. 테스트 주도개발은 테스트 소프트웨어 작성, 동작하는 소프트웨어 작성, 소프트웨어 보완 (잘 동작하는 소프트웨어)하는 과정을 반복하며 수행된다.

- 지속적인 통합 (Continuous Integration) : 대규모 소프트웨어는 여러 명이 동시에 다른 부분을 개발하게 되는데, 이 과정에서 각각의 개발자가 작성 또는 수정한 코드를 자주 그리고 지속적으로 통합 (Build)해야 한다. 자주 통합하는 것은 보통 일일 통합 (Daily Build)을 통해서 이루어지며, 이 모든 과정이 자동화되어야 소프트웨어 개발 효율이 높아진다.

3.3 Agile Enterprise

최근 스크럼과 익스트림 프로그래밍을 통합한 엑스브리드 (XBreed)라는 개념이 등장했는데, 여기에서 스크럼은 일일 프로젝트 관리를 강화시키고 익스트림 프로그래밍은 소프트웨어의 품질을 향상시켰다. 현재 엑스브리드는 기업을 관리, 설계, 감독하는 기민한 방법으로 확장되어 애자일 엔터프라이즈 (Agile Enterprise)로 명명되었다[5].

4. ESA의 애자일 모델 도입사례 분석

본 장에서는 ESA (European Space Agency)의 위성운영센터인 ESOC (ESA's Operation Center)의 소프트웨어 개발 표준에 대한 설명과 애자일 모델 도입 배경 및 결과, 그리고 애자일 모델 도입을 통해서 습득한 교훈에 대해서 정리한다[6].

4.1 European Cooperation for Space Standard, ECSS

ESOC의 지상국 소프트웨어 개발은 ECSS (European Cooperation for Space Standard)을 준수한다. ECSS는 폭포수 모델에 근간하며, 개발을 네 개의 단계 (phase)로 구분하고 각 단계마다 리뷰회의를 수행한다.

-SRR : Software Requirement Review
-PDR : Preliminary Design Review
-CDR : Critical Design Review
-FAR : Final Acceptance Review

(과제의 성격에 따라서 FAR 다음에 유지보수 단계가 포함되기도 한다.)

ESOC에서는 모든 중요 소프트웨어를 외주를 통하여 개발하고 있으며, 다음 두 가지 형태의 계약이 있다.

- Firm Fixed Price, FFP : 범위 (scope)를 고정하고, 일정 (schedule)에 대한 위험을 비용 (cost) 및 품질 (quality)로 조정하는 계약
- Firm Unit Price, FUP : 일정과 품질을 고정하고, 비용 및 범위를 조정하는 계약

실제로 ESOC에서는 FFP를 기본으로 하고 소프트웨어 개발 과정 중에 발생하는 사소한 변경은 FWP (Flexibility Work Package)로 관리하는 방식 (Hybrid Approach)으로 대부분의 소프트웨어 개발 계약을 체결하고 있으며, FWP의 규모는 전체 개발의 10~15%수준이다.

4.2 Applying Agile to Ground Segment Software Development

ESOC에서는 다음의 이유로 스크럼을 포함한 애자일 모델을 도입하였다:

- 생산성 개선
- 프로젝트의 가시성 및 관리 향상
- 개발 기간 단축
- 소프트웨어 품질 개선
- 사용자 만족도 증대

애자일 모델을 사용한 소프트웨어의 개발 경험을 위해서, ESOC은 위험도가 낮은 웹기반 시스템 (Web-Based System) 개발 (Project A)에 처음으로 애자일 모델을 도입하였다. 매 주 회의를 수행하고, 개발환경에 웹으로 접속하는 방법을 도입하였으며, (요구사항 수립 후 구현/검증을 수행하는 방식이 아닌) 점진적인 개발방법을 채택하였다. 계약의 형태는 FFP를 근간으로 하였으나 소프트웨어 개발을 총 2개의 단계로 구분하였다. 첫 번째 단계에서는 사용자 요구사항을 추가로 수집하고 이에 따라 소프트웨어 구조 및 인터페이스 설계, 기능 검증에 집중하였다. 두 번째 단계에서는 본격적인 구현 및 검증을 수행하였다. 본 개발은 엄밀하게 애자일 모델을 적용한 것은 아니지만 애자일 모델의 주요 가치를 달성하기 위한 노력으로 개발 기간 내에 90% 요구사항을 달성하였으며, 높은 사용자 만족도를 얻을 수 있었다.

애자일 모델 경험을 위한 다음 소프트웨어 개발 (Project B)에서는, 몇 가지 형태의 사용자와의 회의를 진행하며 DOORS, Mercury Quality Center 등의 도구를 이용해서 관리하였다. 그리고 스크럼의 스프린트 회의와 유사하게 한 주 단위로 소프트웨어 변경 요청서 (Software Change Request, SCR) 및 변경 결과에 대한 보고서 (Software Problem Report, SPR)를 검토하는 회의를 수행하였다. 결과적으로 의사소통과 가시성이 증대되었으며, 일정 내에 요구사항을 모두 만족시키는 소프트웨어 개발을 완료하였다.

본격적인 스크럼의 도입은, 운영되고 있는 위성의 데이터 배포 시스템과 대규모 소프트웨어의

일부인 GUI 소프트웨어 개발에 스크럼을 동시에 적용한 것이다.

위성 데이터 배포 시스템 개발 (Project C)은 스크럼을 도입하기 2년 전 (2009년 2분기)에 시작되었으며, 이미 상세한 소프트웨어 요구사항 및 상세설계 결과가 도출된 상황이었다. 스크럼은 작업기술서 (Statement of Work, SoW)의 상세화를 위한 도구로 사용하기 시작하였다. 스크럼 도입은 소프트웨어 요구사항 및 상세설계 결과에서 제품 백로그를 생성하고, 4주 간격 (팀 개편으로 인해서 2주로 간격이 변경된 적도 있었음)의 8개 스프린트를 구성하여 스크럼 마스터에 의해 수행되었다. 상세한 요구사항으로 부터 제품 백로그 생성 및 업데이트를 수행하였으며, 회의 수행 (스프린트 회의, 일일 스크럼 회의), 스프린트 별 업무할당, 제품 백로그 관리 등에 어려움이 있었으나 이를 극복하고 성공적으로 소프트웨어 개발을 완료하였다. 프로젝트 성공의 주요인은 매 스프린트마다 가시적인 결과를 생성함으로써 개발자와 사용자 간의 신뢰가 쌓였기 때문으로 분석하였다.

GUI 소프트웨어 개발 (Project D)은 그 결과가 대규모 시스템의 일부로 포함되어야 하므로 개발일정을 반드시 준수해야하는 제약이 있었다. 제품 백로그를 초기 사용자 소프트웨어 요구사항으로부터 도출하고, 스프린트 회의와 스프린트 데모 회의를 수행하였으며, 사용자 요구사항을 지속적으로 업데이트하며 명료한 제품 백로그를 생성하였다. 스크럼에 익숙하지 않은 개발자 및 사용자, 이들로 인하여 첫 번째 스프린트 미팅은 개발자만 참석하여 진행하는 등의 어려움이 있었지만, 스크럼을 이용하여 사용자 의견 (feedback)을 수용하고 의미 있는 스프린트 데모회의를 수행하며 난관을 극복하였다. GUI 소프트웨어 개발에 스크럼을 도입함으로써, 불충분한 사용자 요구사항을 명확하게 업데이트하고 모든 요구사항을 만족시킨 것을 큰 성과로 분석하였으며, 프로젝트의 기술과 관리에 대한 가시성이 증가되었음을 확인하였다.

폭포수 모델에서는 소프트웨어 요구사항을 개발 초기에 확정하며, 이로 인하여 사용자의 의견을 충분히 만족하는 소프트웨어 개발에 어려움이 있다. 반면 스크럼 모델을 도입하면 지속적으로 사용자 의견을 반영하여 소프트웨어 요구사항의 업데이트를 수용하는 것이 가능하다. 즉, 소프트웨어 요구사항은 개발과 함께 진화한다.

ESOC은 지상국 소프트웨어 개발에 스크럼을 도입함으로써 '사용자 의견 반영으로 인한 소프트웨어 품질 향상' 및 '자주 수행되는 테스트로 인한 지속적인 소프트웨어 요구사항 업데이트'의 장점이 있다고 결론 내렸다.

4.3 Lessons Learned

본 절에서는 ESOC에서 기존의 폭포수 모델에 애자일 모델을 적용함으로써 습득하였던 교훈에 대해서 정리한다.

-User Requirements : 기능적 요구사항 (functional requirements)을 명확하게 정의할 수 있었다. 잘 정의된 사용자 요구사항은 스프린트 데모에서 우수한 결과 산출이 가능하게 하였다.

-SoW template update : 작업기술서 (SoW)를 보다 상세하게 기술할 수 있었다. (작업기술서의 '기대, 비용, 결과물, 책임과 역할' 항목에 대한 명확성 증대)

-Scrum at ESOC : SixSq사 (EOSC에 애자일 자문을 수행한 업체)는 제품소유자 (PO)의 역할을 수행할 수 있는 핵심인원을 현재 ESOC 조직에 추가해야 함을 권고하였다. 그리고 스크럼을 이용한 개발은 기존의 폭포수 모델에 비하여 더 많은 시간이 필요한 대신에 스프린트 진행을 통하여 예측가능한 장점이 있다는 것을 이해해야 한다.

-User Involvement : 사용자는 스크럼 적용에 따라서 스프린트 데모 회의에 참석하는 것에 대해서 많은 관심을 보였다. 사용자가 참여하여 제공하는 의견은 프로젝트를 성공으로 이끄는 중요한 요소이다.

-Contract Update : (현재 ESOC의 FFP 기반 계약에서는 일정조정을 위하여 비용을 도구로 이

용하고 있다.) 기존의 FFP 계약에서 비용 및 소프트웨어의 변경을 수용해야 한다. (Money for noting and changes for free, by Jeff SUTHERLAND) 이것이 가능하면, 스크럼에 따라서 우선순위를 관리하다가 제품의 백로그가 완료되지 않았더라도 (중요한 것은 이미 완료되고 상대적으로 중요하지 않은 항목이 남아있는 상태) 비즈니스 관점의 가치가 만족되면 프로젝트를 종료할 수 있다.

-Coaching Model and Scrum Fluency : SixEq사의 애자일 자문은, ESOC 직원이 '팀, 제품소유자, 스크럼 마스터'의 역할 원활하게 수행할 수 있도록 지원하였다. 팀에 스크럼 경험이나 교육을 이수한 인원이 존재한다면 보다 더 좋은 결과를 산출할 수 있는 애자일 모델 프로젝트를 수행할 수 있다.

-Tools : ESOC에는 현재 소프트웨어 개발을 지원하는 몇몇 도구 (예, 요구사항 관리, 테스트 시나리오 시스템)를 사용하고 있는데 이를 스크럼에 적절히 사용하는 것이 중요하며, 스크럼을 위해서 새로운 도구를 도입하는 것에 대해서도 고민해야한다.

5. 결 론

본 논문에서는 기존 소프트웨어 개발 모델 및 현재 널리 사용되는 애자일 모델의 동향에 대해서 설명한 다음, ESA의 지상국 개발에 스크럼을 도입한 사례에 대한 분석 결과를 정리하였다.

기능과 복잡도가 증가하는 소프트웨어 개발에서 소프트웨어 개발 모델의 도입은 '코딩부터 시작하던 시기'에 비해서 많은 발전이 있었으나, 프로세스가 많아짐으로써 부차적인 문제를 초래하였다. 애자일 모델은 짧은 간격으로 의사소통을 반복함으로써 기존 소프트웨어 개발 모델의 단점을 극복하고자 노력한다.

ESA의 지상국 개발에 애자일 모델을 도입한 최근 사례를 분석한 결과, 애자일 모델의 부분적인 적용으로도 소기의 성과를 달성할 수 있음을 확인하였다. 엄밀하게는 ESA에서 적용한 방법은 애자일 모델 보다는 다단계의 폭포수 모델에 가까우나[6], 애자일 모델을 적용하고자 하는 노력이 보다 나은 소프트웨어 개발에 많은 도움이 되었다고 분석하였다.

의사소통을 핵심으로 삼고 있는 애자일 모델의 네 가지 가치 (소프트웨어 개발자들 간의 상호작용, 작동하는 소프트웨어, 고객과의 협력, 변화에 대응)만으로도 국내 지상국 소프트웨어 개발에 적용해야 하는 이유가 충분하다고 생각한다.

참 고 문 헌

1. 임백준, 임백준의 소프트웨어 산책, 한빛미디어, ISBN 89-7914-329-X, pp.120-122, 2005.
2. 정태중/신승환 역, 고약한 문제 합당한 해결 (Wicked Problems, Righteous Solutions, by Peter DEGRAS), 인사이트, ISNB 978-89-91268-86-9, p.59 p.92 p.100 p.106 p.144 p.161 p.213 p.217 p.218, 2010.
3. 김익환/전규현, 소프트웨어 개발의 모든 것, 페가수스, ISBN 978-89-96091-73-8, p.171 p.173, 2008.
4. 심우곤외 역, 스크럼과 XP : 애자일 최전선에서 일군 성공 무용담 (Scrum and XP from the Trenches, by Henrik KNIBERG). 인사이트, ISBN 978-89-91268-60-9, p.xiv p.3 p.5, 2009.
5. 박일/김기욱 역, 스크럼: 팀의 생산성을 극대화시키는 애자일 방법론 (Agile Software Development with Scrum, by Ken SCHWABER and Mike BEEDLE), 인사이트, ISBN 978-89- 91268-47-0, p.11 p.30, 2008.
6. Rui SANTOS, Felix FLENTGE, Mac-Elian BEGIN, and Vicente NAVARRO, Agile Technical Management of Industrial Contracts: Scrum Development of Ground Segment Software at the European Space Agency, Springer, XP2011 LNBIP77, pp.290-305, 2011.