The Journal of IIBC. Vol. 14, No. 1, pp.61-67, Feb. 28, 2014. pISSN 2288-0879, eISSN 2288-2618

http://dx.doi.org/10.7236/JIIBC.2014.14.1.61
JIIBC 2014-1-8

SaaS 클라우드 서비스를 위한 소프트웨어 개발 방법론

Software Development Methodology for SaaS Cloud Service

황만수*, 이관우**, 윤성혜***

Mansoo Hwang*, Kwanwoo Lee, Seonghye Yoon*****

요 약 SaaS 클라우드 서비스는 사용자가 소프트웨어를 온라인 서비스로 이용할 수 있도록 클라우드 플랫폼에 배치되어 구동되는 모델을 의미한다. 본 연구에서는 SaaS 클라우드 서비스의 효율적인 개발을 위해 적합한 개발 방법론을 제안한다. 이를 위해 우선 국내 SaaS 클라우드 서비스 개발 업체들의 현황을 분석하여 개발 핵심 요소를 도출하고, 이를 토대로 기존 소프트웨어 개발 방법론 중에서 SaaS 클래스 서비스 개발에 가장 적합한 개발 방법론을 선정하여 테일러링 하였다. 그리고 제안한 개발 방법론의 적용가능성을 검증하기 위해 현재 SaaS 클라우드 서비스를 개발하고 있는 업체에 적합한 개발 방법론을 테일러링하는 사례 연구를 수행하였다.

Abstract A SaaS cloud service represents a model deployed and running on a cloud platform to enable users to use software as an online service. This work proposes a development methodology adequate for the effective development of SaaS cloud services. For doing this, we first analyzed the current state of companies developing SaaS cloud services and identified key factors for the development of SaaS cloud services. Then, we selected and tailored the methodology that is best suited for the development of SaaS cloud services among existing software development methodologies. To validate the applicability of the proposed methodology, we performed a case study tailoring the development methodology adequate for the company developing SaaS cloud services.

Key Words : SaaS Cloud Service, Software Development Methodology, Industry Case

Ⅰ. 서 론

SaaS (Software As A Service) 클라우드 서비스는 사용자가 소프트웨어를 온라인 서비스로 이용할 수 있도록 클라우드 플랫폼에 배치되어 구동되는 모델을 의미한다[1]. SaaS 클라우드 서비스를 이용하는 고객은 초기 투자 비용이나 시스템 관리의 부담이 적은 대신 서비스 기간이나 사용량에 따라 정해진 금액을 지불하는 방식을 사용한다[2]. 이러한 SaaS 클라우드 서비스는 기존의 IT 환경을 공급자 중심에서 구매자 중심으로, 소규모 전문 업체의 솔루션이 시장성을 확보하는 환경으로, 그리고 기존 소프트웨어의 서비스화를 촉진시키는 변화를 일으키고 있다[2]. 이에 따라 현재의 소프트웨어 시장은 SaaS 클라우드 서비스의 급성장을 예측하며[3], 서비스 제공자들에게 새로운 환경에 대한 적절한 대처를 요구하고 있다[4].

그러나 소프트웨어 시장의 변화에도 불구하고, SaaS 클라우드 서비스를 개발하는 국내 업체들은 기존 개발 방법론을 사용하여 전통적인 방식으로 시장의 요구에 대응하고 있다. 이로 인해 아직 공급자 중심의 환경에서 벗어나지 못하고 있어, 기존 ASP (Application Service

*정회원, 신한대학교 IT융합공학부
**정회원, 한성대학교 정보시스템공학과
***정회원, 서강대학교 컴퓨터공학과
접수일자: 2014년1월 16일, 수정완료 2014년 2월 3일
게재확정일자: 2014년 2월 7일

Received: 16 January, 2014 / Revised: 3 February, 2014
Accepted: 7 February, 2014
*Corresponding Author: kwlee@hansung.ac.kr
Dept. of Information Systems Engineering, Hansung University, Korea

Provider) 시장이 경험했던 것과 같은 과도한 커스터마이제이션으로 인한 생산성 저하를 경험하고 있다. 이는 기존 개발 방법론들이 SaaS 클라우드 서비스의 가장 큰 특징인 다중 테넌트(multi-tenant) 특성을 지원하기가 어렵기 때문이다.

본 연구에서는 SaaS 클라우드 서비스 개발에 적합한 개발 방법론을 제시하고자 한다. 먼저, 국내 SaaS 클라우드 서비스 개발 업체들의 현황을 분석하여 개발 핵심 요소를 도출한다. 이를 토대로 기존 개발 방법론 중 가장 적합한 개발 방법론을 선정하고, 이를 바탕으로 테일러링하여 SaaS 클라우드 서비스 개발을 위한 개발 방법론을 제안한다. 또한 제안된 개발 방법론의 적용가능성을 검증하기 위해 현재 SaaS 클라우드 서비스를 개발, 운영하고 있는 업체에 적합한 개발 방법론을 테일러링하는 사례를 수행하였다.

본 논문의 구성은 다음과 같다. 2장에서는 국내 업체들의 SaaS 클라우드 서비스 개발 현황을 소개한다. 3장에서는 SaaS 클라우드 서비스 개발 방법론은 제안하고, 4장에서는 사례를 통해 제안된 개발방법론에 대한 적용가능성을 검증한다. 그리고 5장에서는 본 연구의 결론을 맺으면서, 향후의 연구 과제에 대해 언급한다.

II. 국내 개발 현황 분석

국내에서 SaaS 클라우드 서비스를 개발 및 운영을 하는 8개의 기업(대기업 2, 중소기업 5, 연구기관 1)을 대상으로 설문 조사를 진행하였다. 설문 조사를 통한 현황 분석 결과를 기반으로 새로운 개발 환경 변화에 대처하기 위한 핵심 요소를 식별하였다.

1. SaaS 클라우드 서비스 요구사항 정의

SaaS 클라우드 서비스 요구사항 정의에 필수적인 활동으로는 *이해 관계자 식별 및 요구사항 도출*, *공통 및 개별 사용자 요구사항 정의 및 정제*, 그리고 *요구사항 변경 관리 및 추적* 활동들이 식별되었다. 이는 국내 소프트웨어 개발이 특정 고객사를 대상으로 한 SI 형태의 프로젝트가 주를 이루고 있기 때문에 마켓 요구사항 정의나 공통 및 개별 사용자 요구사항 정의 등에 어려움을 겪고 있는 것으로 분석된다. 또한 점점 요구사항이 복잡하고 다양해짐에 따라 요구사항의 변경 관리 및 추적 활동에 대한 요구가 커지고 있음을 나타낸다.

2. SaaS 클라우드 서비스 개발

SaaS 클라우드 서비스 개발에 필수적이 활동으로는 *참조/공통 아키텍처 설계 및 재사용*, *상세 프로세스 및 데이터 설계*, 그리고 *재사용 가능한 부분식별* 활동들이 도출되었다. 그러나 공통성과 가변성에 대한 고려가 되지 않았기 때문에 참조/공통 아키텍처에 대한 고려도 미흡하며, 재사용 가능 부분을 식별하여 아키텍처 레벨에서 재사용하기보다는 소스 코드의 일부를 복제하여 사용하는 수준에 그치고 있었다.

III. SaaS 클라우드 서비스 개발방법론

1. 기존 개발 방법론 분석

본 연구에서는 새로운 개발 방법론의 개발보다 체계화된 기존 방법론의 분석과 테일러링을 통해 SaaS 클라우드 서비스에 적합한 개발 방법론을 제안한다. 그래서 현업에서 소프트웨어 개발 방법으로 많이 사용하는 XP (eXtreme Programing)[5], CBD (Component Based Development)[6], SPLE (Software Product Line Engineering)[7]를 도출한 SaaS 클라우드 서비스 개발을 위한 핵심 요구사항과 SaaS 클라우드 서비스의 특징 관점에서 지원 여부를 비교하였고, 그 결과는 표 1과 같다.

표 1. 소프트웨어 개발 방법론의 비교
Table 1. Comparison of SW development methodologies

SaaS 클라우드 서비스 개발을 위한 핵심 요구사항		XP	CBD	SPLE
요구사항 정의에 필수적인 활동	이해관계자 식별 및 요구사항 도출	△	△	O
	공통/개별 사용자 요구사항 정의 및 정제 활동	X	X	O
	요구사항 변경 관리 및 추적활동	X	△	O
서비스 개발에 필수적인 활동	참조/공통 아키텍처 설계 및 재사용	X	△	O
	상세 프로세스 및 데이터 설계 활동	X	O	O
	재사용 가능한 부분 식별	△	O	O
SaaS 특징	확장성	X	X	X
	커스터마이제이션	X	△	O

표 2. SCoD 활동별 상세 프로세스
Table 2. The detailed process by SCoD activities

단계	목적	세부 단계	세부 목적	입력	출력
마켓 요구사항 분석	목표 마켓의 요구사항 및 특징 분석, 비즈니스 가치 기반으로 마켓 요구사항의 중요도 결정	비즈니스 동향 분석	마켓의 핵심 요구사항 및 향후 마켓의 예상되는 요구사항 도출 준비	마켓 현황 정보	비즈니스 동향 분석서
		법률 및 규범 분석	마켓 및 서비스를 제공하는 지역의 법률 및 규범을 제약사항으로 도출	관련 마켓 법률 및 규범	법률 및 규범 분석서
		테넌트 분석	주요 테넌트 식별, 이들의 성향 및 특징을 분석하여 테넌트 별 핵심 요구사항 도출	테넌트 특성 및 배경	테넌트 요구사항 분석서
		요구사항 우선순위화	도출된 요구 사항에 대해, 비즈니스 가치 관점에서 우선순위를 정함	비즈니스 동향 분석서, 법률 및 규범 분석서, 테넌트 요구사항분석서	테넌트 요구사항 목록
테넌트 요구사항 명세	테넌트 요구사항 명세는 테넌트 별로 구체적인 요구사항을 정의하는 활동	기능 요구사항 명세	테넌트들의 서비스 사용 목적을 바탕으로 요구되는 기능을 사용자 시나리오 기술	테넌트 요구사항 목록	기능 요구사항 명세서
		품질 요구사항 명세	테넌트들의 사용목적과 운영환경 등에 따라 시스템이 만족시켜야 하는 품질 요구사항기술	테넌트 요구사항 목록	품질 요구사항 명세서
		데이터 요구사항 명세	각 테넌트들의 업무를 위한 데이터를 식별, 이를 관리시 필요한 관련된 제약사항을 정의	테넌트 요구사항 목록	데이터 요구사항 명세서
		제약사항 명세	테넌트의 특성 및 환경을 고려하여 필요한 제약사항 등의 요구사항 등을 기술	테넌트 요구사항 목록	제약 사항 명세서
가변성 분석	테넌트 별로 요구사항의 공통성과 가변성을 분석하여 가변성 모델인 휘처 모델을 도출하는 활동	휘처 도출	테넌트 별 요구사항의 가변성 분석을 통해 휘처를 추출하여 정의	테넌트 요구사항 명세서	휘처 목록, 휘처 명세서
		휘처 모델작성	도출된 휘처를 휘처모델로 계층적으로 구조화하고 가변 휘처 간에 합성규칙 작성	휘처 명세서, 요구사항 간의 의존성	휘처 다이어그램, 합성규칙 명세서
		휘처 모델 정련	작성된 휘처 모델을 검토하여 수정하는 활동, 휘처 모델의 타당성 및 일관성을 검토	휘처 모델	휘처 모델 검토 의견서, 변경된 휘처모델
		요구사항 가변성 관리	휘처 모델과 테넌트 요구사항을 대응시켜 테넌트 요구사항의 가변성을 추적, 관리	휘처 모델, 테넌트 요구사항 모델	휘처-요구사항 추적표
개념 아키텍처 설계	시스템의 논리적인 기능 단위들을 찾고, 이들 간의 논리적인 상호작용 정의 활동	개념 아키텍처 드라이버 식별	테넌트 요구사항 명세서로부터 개념 아키텍처 설계에 중요한 요인으로 작용하는 주요 기능 요구사항을 식별	테넌트 요구사항 명세서	아키텍처 기능 요구사항 목록, 아키텍처 품질 요구사항 목록
		개념 아키텍처 설계 뷰 작성	아키텍처 요구사항을 만족시키는 개념 아키텍처 모델을 도출	아키텍처 요구사항	요구사항별 아키텍처 설계 패턴리스트, 개념 아키텍처 모델 명세서
		개념 아키텍처 설계 검증	아키텍처가 목표로 하는 품질을 만족도, 각 품질 속성 간의 연관성 등을 분석, 아키텍처의 상충된 문제를 식별하고 해결 방안 논의	개념 아키텍처 명세서, 아키텍처 요구사항 목록	테넌트 고유 서비스 배치 단계 아키텍처 평가 보고서
데이터 아키텍처 설계	시스템의 논리적인 데이터 단위들을 찾고, 이들 간의 관계를 정의 활동	데이터 아키텍처 요구사항 식별	테넌트 요구사항 명세서로부터 데이터 아키텍처 설계에 중요한 요인으로 작용하는 주요 데이터 요구사항을 식별	테넌트 요구사항 명세서	아키텍처 데이터 요구사항 목록
		데이터 아키텍처 설계 뷰 작성	주어진 아키텍처 데이터 요구사항을 만족시킬 수 있도록 데이터 아키텍처 모델을 구체화	아키텍처 데이터 요구사항 목록	데이터 구조 및 배치 아키텍처 모델 명세서
		데이터 아키텍처 설계 검증	이해관계자들을 대상으로 미처 파악하지 못한 아키텍처 요구사항을 찾아내고, 아키텍처가 요구사항을 모두 만족하였는지를 평가	데이터구조 아키텍처모델 명세서, 아키텍처 요구사항 목록	아키텍처 평가 보고서
아키텍처 변경 관리	테넌트 요구사항에 따라 변경되는 아키텍처 요소를 식별, 관리하는 활동	개념 아키텍처 추적 테이블 작성	휘처 모델의 휘처와 대응되는 개념 아키텍처 구성요소를 식별, 이들 간의 대응관계를 추적 테이블로 작성	요구사항 명세서, 개념 아키텍처 모델 명세서	휘처-개념 아키텍처 추적 테이블
		데이터 아키텍처 추적 테이블 작성	휘처 모델의 휘처와 대응되는 데이터 아키텍처 구성요소를 식별, 이들 간의 대응관계를 추적 테이블로 작성	요구사항 명세서, 데이터 아키텍처 모델 명세서	휘처-데이터 아키텍처 추적 테이블
서비스 컴포넌트 개발	적응형 컴포넌트를 구현하고 추적관리하는 활동	적응형 컴포넌트 구현	다양한 테넌트의 요구를 SaaS 클라우드 서비스가 동적으로 수용할 수 있도록 동적 적응 가능한 컴포넌트를 구현	개념 아키텍처 모델 명세서, 데이터 아키텍처 모델 명세서	클래스 설계서
		컴포넌트 변경 관리	컴포넌트 변경 관리는 구현된 컴포넌트를 테넌트 요구사항의 변경으로 인해 영향 받는 컴포넌트를 추적 관리하는 활동	클래스 설계서, 휘처 모델 명세서	휘처-클래스 추적 테이블

현업에서 가장 많이 사용되는 있는 XP는 코딩 위주의 활동에 초점을 두고 있어 가변성 분석이나 재사용에 중점을 두고 있는 개발 핵심 요구사항을 대부분 만족하지 못하고 있다.

또한 개발 자산의 재사용을 목적으로 하는 CBD는 개발 측면에서 재사용을 고려하고 있지만, 요구사항 정의 단계에서 공통성과 가변성을 고려하지 않고 있다. 그리고 개발 시 묵시적으로만 참조 및 공통 아키텍처의 설계 및 재사용을 고려하고 있다.

반면, SPLE는 요구사항의 다양성에 따라서 소프트웨어를 적응시키기 위해 가변성 관리 및 설계 기법을 지원하고 있다. 이는 SaaS 클라우드 서비스의 커스터마이제이션에 직접 활용하여 테넌트의 요구사항에 따라 정적 혹은 동적으로 서비스가 적응하게 할 수 있다. 그러나 다른 방법론들과 같이 SaaS 클라우드 서비스의 특징 중 하나인 확장성 측면에서는 고려되지 않고 있다.

2. SaaS 클라우드 서비스 개발방법론 개념

SaaS 클라우드 서비스 개발은 다양한 테넌트들의 요구사항에 따라 쉽게 커스터마이징이 가능한 서비스 개발을 지원할 수 있는 체계가 필요하므로, 테넌트의 고유 요구사항에 따라 다양한 형태의 SaaS 클라우드 서비스를 배치할 수 있는 것이 중요하다. 본 연구에서는 이를 지원할 수 있도록 그림 1과 같은 개념을 가진 SCoD (SaaS Cloud oriented Development)방법론을 제안한다.

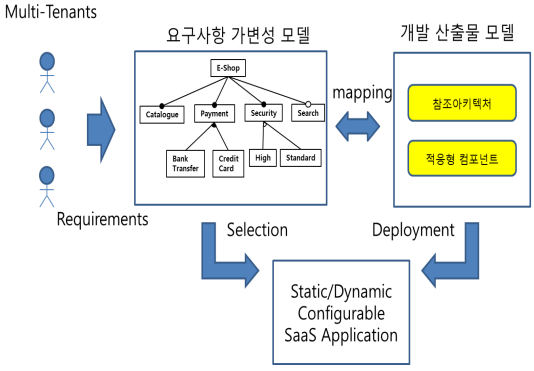

그림 1. 다중 테넌시 지원 SCoD 개념
Fig. 1. The concept of SCoD to support multi-tenancy

다중 테넌트의 고유 요구사항 지원을 위해 휘처 모델링(Feature Modeling)[8]을 사용하여 다중 테넌트들 간의 요구사항 가변성을 분석한다. 또한, SaaS 클라우드 서비스의 논리적인 추상화를 제공하는 모델인 참조 아키텍처를 설계함으로써, 다중 테넌트들 간의 공통 요구사항 및 테넌트별 고유 요구사항도 지원할 수 있다. 각 공통 및 가변 요구사항과 개발 산출물 모델 (참조아키텍처 및 적응형 컴포넌트) 사이에 명시적인 대응관계가 설정되어 있으므로, 요구사항 가변성의 선택에 따라 SaaS 클라우드 서비스를 클라우드 플랫폼에 배치 할 수 있다.

3. SCoD 방법론 프로세스

본 연구에서 제안한 SCod 방법론의 프로세스는 그림 2와 같이 두 단계로 나뉘며. 표 2에서는 그림 2에 나타난 활동별 상세 프로세스 및 산출물을 보여주고 있다.

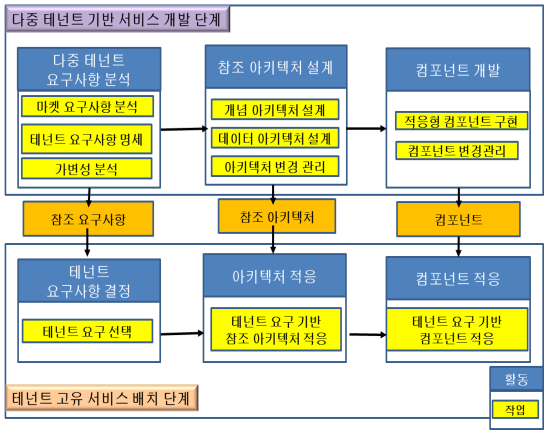

그림 2. SCoD 프로세스 개요
Fig. 2. The overall process of SCoD

가. 다중 테넌트 기반 서비스 개발 단계

이 단계는 다중 테넌트들의 요구사항을 모두 만족시킬 수 있는 참조 아키텍처와 컴포넌트를 개발하는 과정으로, 다중 테넌트 요구사항 분석, 참조아키텍처 설계, 컴포넌트 개발로 세분화 된다.

다중 테넌트 요구사항 분석은 먼저 마켓 요구사항을 분석하고, 이를 바탕으로 테넌트별 요구사항을 구체화한다. 그리고 테넌트별 요구사항의 공통성과 가변성을 분석한다.

참조 아키텍처 설계는 기능적 관점과 데이터 관점 설계로 구분된다. 기능적 관점 설계에서는 논리적인 기능 단위로 컴포넌트를 찾고 이들 간의 상호작용을 모델링한다. 데이터 관점 설계에서는 시스템에 필요한 논리적 데

이터와 이들 간의 관계를 모델링 한다. 참조 아키텍처는 다중 테넌트들 간의 공통 요구사항을 지원하는 부분과 테넌트 별로 고유하게 요구되는 부분이 있다. 따라서 아키텍처 변경관리에서는 테넌트 요구사항의 가변성에 따라 변경되는 부분을 식별 및 관리할 수 있도록 요구사항과 아키텍처의 가변성 간의 대응관계를 설정한다.

컴포넌트 개발은 참조 아키텍처 설계 요소를 구현 단위 요소(클래스, 웹페이지, 자료 저장소 등)로 대응시키고 테넌트별 요구사항의 가변성과 대응되는 구현 단위 요소의 가변성을 관리하는 컴포넌트 변경 관리 활동과 테넌트별 요구사항의 가변성 선택에 따라 구현 단위 요소가 정적이나 동적으로 적응될 있도록 적응형 컴포넌트를 구현하는 활동으로 구분된다.

나. 테넌트 고유 서비스 배치 단계

이 단계는 테넌트 고유 요구사항에 따라 서비스를 배치하는 단계로 테넌트 요구사항 결정, 테넌트 요구기반 아키텍처 적응, 테넌트 요구기반 컴포넌트 적응과정으로 세분화된다.

테넌트 요구사항 결정은 다중 테넌트 요구사항 분석 결과로 도출된 테넌트 요구사항 가변성 모델을 기반으로, 새로이 배치시키고자 하는 서비스의 테넌트 요구사항을 선택하는 과정이다. 이 과정의 결과는 테넌트 요구사항 집합이 산출된다.

테넌트 요구 기반 참조 아키텍처 적응은 입력된 테넌트 요구사항 집합을 바탕으로 대응된 참조 아키텍처 요소를 선택하는 과정이다. 참조 아키텍처의 가변요소 선택은 서비스가 클라우드 플랫폼에 배치하기 전에 정적으로 이루어지므로, 이 단계는 서비스를 클라우드 플랫폼에 배치한 후 동적으로 적응이 일어나는 서비스의 경우에는 해당되지 않는다.

테넌트 요구 기반 컴포넌트 적응은 정적 혹은 동적으로 일어날 수 있다. 정적 컴포넌트 적응은 서비스가 클라우드 플랫폼에 배치되기 전에 서비스 컴포넌트 코드의 일부가 테넌트 요구사항에 따라 선택되거나 비 선택되는 경우이고, 동적 컴포넌트 적응은 서비스가 클라우드 플랫폼에 배치된 후에, 테넌트의 요구사항에 따라서 컴포넌트의 적응이 동적으로 일어나는 것을 의미한다.

Ⅳ. 기업적용을 위한 테일러링 사례

제안하는 SCoD 개발방법론의 적용가능성 검증을 위해 서비스 중인 SaaS 클라우드 서비스를 대상으로 개발방법론을 테일러링하고, 실무자 인터뷰로 방법론에 대한 타당성 여부를 검토하였다.

대상으로 선정된 기업은 12년 동안 클라우드 서비스를 개발해왔으며, 개발 인력의 70%가 SaaS 클라우드 서비스에 참여하고 있는 중소기업이다. 이 업체에서 개발하고 있는 대표적인 SaaS 클라우드 서비스 중 하나인 멀티 단말을 지원하는 작업 환경 단일화 서비스는 크게 두 가지 특성을 가진다. 첫 번째는 특정 고객사를 대상으로 개발하는 SI 프로젝트와는 달리 고객의 범위가 정해지지 않고, 여러 도메인(제조, 금융, 공공 등)에서 사용된다. 두 번째는 프라이빗 클라우드 환경이기 때문에 가변성이 고객사별로, 사용자 권한에 따른 테넌트별로, 발생한다는 점이다.

이와 같은 특성은 고객 범위의 모호성에 의한 요구사항 분석의 어려움, 고객의 다양성에 따른 관련 법률 미비에 따른 요구사항 분석의 어려움, 가변성 분석의 어려움, 아키텍처 변경 관리 및 컴포넌트 변경 관리의 미흡 등의 문제를 발생시킨다.

프로젝트의 현재 개발 방법론과 기존 산출물, 프로젝트 규모, 개발 조직의 특성들을 고려하여 테일러링 된 개발 방법론은 표 3과 같다. 고객사 별 가변성을 고려하지 않는 것을 정책으로 가지고 있고 테넌트 별 가변성 역시, 공통 휘처화 하고 있기 때문에 가변성을 고려한 활동 및 작업들은 방법론에서 제외되었다. 대신, 고객사별 가변성을 차기 버전에 고려하기 위하여 마켓의 사용자 그룹 분석 시, 이를 고려할 수 있도록 입력 값에 명시적으로 고객의 커스터마이징 리스트가 추가되었다.

이 개발 조직의 경우, 다년간의 실무 경험으로 명시적으로 개발 방법론은 없지만, 암묵적 동의하에 정의된 프로세스 하에서 서비스를 개발하고 있는 조직이다. 테일러링 된 개발 방법론은 현재 사용하고 있는 개발 방법론의 활동 및 작업들을 모두 수용하고, 현 개발 조직이 앞으로 목표로 삼고 있는 요구사항-아키텍처-컴포넌트 간의 추적과 같은 형상관리 부분까지 포함하고 있어서, 실무자들로부터 제안된 개발 방법론 적용 의사를 타진한 결과, 충분한 도입 의사를 확인할 수 있었다.

표 3. 테일러링된 SCoD 방법론
Table 3. Tailored SCoD methodology

개발방법론		입력	출력
마켓 요구사항 분석	마켓의 비즈니스 동향 분석	마켓 현황	비즈니스 동향 분석서
	마켓의 법률 및 규범 분석	법률 및 규범	법률 및 규범 분석서
	마켓의 사용자 그룹(테넌트) 분석	테넌트 특성 및 배경, 고객사 커스터마이징 리스트	테넌트 요구사항 분석서
	마켓 요구사항의 우선순위화	비즈니스 동향 분석서, 법률 및 규범 분석서, 테넌트 요구사항 분석서	테넌트 요구사항 목록
테넌트 요구사항 명세	기능 요구사항 분석	테넌트 요구사항 목록	기능 요구사항 명세서
	데이터 요구사항 분석	테넌트 요구사항 목록	데이터 요구사항 명세서
	품질 요구사항 분석	테넌트 요구사항 목록	품질 요구사항 명세서
	제약사항 분석	테넌트 요구사항 목록	제약 사항 명세서
개념 아키텍처 설계	개념 아키텍처 드라이버 식별	테넌트 요구사항 명세서	아키텍처 기능 요구사항 목록
	개념 아키텍처 설계 뷰 작성	테넌트 요구사항 명세서	요구사항별 아키텍처 설계 패턴리스트
	개념 아키텍처 설계 검증	개념 아키텍처 명세서, 아키텍처 요구사항 목록	아키텍처 평가 보고서
데이터 아키텍처 설계	데이터 아키텍처 요구사항 식별	테넌트 요구사항 명세서	아키텍처 데이터 요구사항 목록
	데이터 아키텍처 설계 뷰 작성	아키텍처 데이터 요구사항 목록	데이터 구조 아키텍처 모델 명세서, 데이터 배치 아키텍처 모델 명세서
	데이터 아키텍처 설계 검증	데이터구조 아키텍처모델명세서, 아키텍처 요구사항 목록	아키텍처 평가 보고서
아키텍처 변경 관리	개념 아키텍처 추적 테이블 작성	요구사항 명세서, 개념 아키텍처 모델 명세서	휘처-개념 아키텍처 추적 테이블
	데이터 아키텍처 추적 테이블 작성	요구사항 명세서, 데이터 아키텍처 모델 명세서	휘처-데이터 아키텍처 추적 테이블
서비스 컴포넌트 개발	적응형 컴포넌트 구현	개념 아키텍처 모델 명세서, 데이터 아키텍처 모델 명세서	클래스 설계서
	컴포넌트 변경 관리	클래스 설계서, 요구사항 명세서	요구사항-클래스 추적 테이블

V. 결 론

본 연구에서는 기존 국내 SaaS 클라우드 서비스의 개발 현황을 분석하여 개발 핵심 요소를 도출하고, 이를 기반으로 기존 소프트웨어 개발 방법론인 SPLE를 테일러링하여 SaaS 클라우드 서비스에 특화된 SCoD 방법론을 제시하였다. 본 연구에서 제안한 SCoD 방법론에 대한 유효성을 검증하기 위해서, SaaS 클라우드 서비스를 개발 및 운영하고 있는 기업에 적합한 테일러링된 개발 방법론을 제시하였다. 그리고 현업 실무자의 피드백을 통해 실제로 SaaS 클라우드 서비스 개발에 충분히 적용 가능함을 알 수 있다.

제안된 개발 방법론을 기업이 스스로 적용할 수 있으려면 테일러링 사례가 보여주는 것과 같은 적용하고자 하는 SaaS 클라우드 서비스 및 개발 조직의 특성이 고려되어야 한다. 향후, 더 많은 사례를 확보를 통해 개발 방법론을 적용하고자 하는 조직들이 스스로 이를 테일러링해서 사용할 수 있는 가이드로 활용할 수 있도록 할 것이다.

References

[1] Gillett, F.E., "Future View: New Tech Ecosystems of Cloud, Cloud Services, and Cloud Computing," Forrester Research Paper, 2008.

[2] Turner, M., Budgen, D., and Brereton, P., "Turning Software into a Service," IEEE Computer, vol.36, no.10, pp.38-44, 2003

[3] Gartner, "Hype Cycle for Cloud Computing, 2012", 2012. 8

[4] IDC, "Worldwide Software as Service 2010-2014 Forecast: Software Will Never Be the Same,"2010

[5] Beck, K., Andres, C., "Extreme Programming Explained: Embrace Change (2nd Edition)" Addison_Wesley, 2004

[6] Heineman, G. T., Councill, W. T. "Component based software engineering : putting the pieces together", ACM Press, 2001

[7] Northrop, L., Clements P. "Software Product Lines",

Addison_Wesley, 2002

[8] Kang, Kyo C., et al. Feature-oriented domain analysis(FODA) feasibility study. No. CMU/SEI-90-TR-21. CARNEGIE-MELLON UNIV. PITTSBURGH PA SEI, 1990.

저자 소개

황 만 수(정회원)

- 1986년 : 중앙대학교 이학석사
- 2001년 : 숭실대학교 공학박사
- 1987년 ~ 1993년 : LG소프트웨어 연구원
- 1993년 ~ 현재 : 신한대학교 IT융합공학부 교수

<주관심분야: SW Requirement Engineering, Usability Engineering, SW Quality Management>

이 관 우(정회원)

- 2003년 POSTECH, 박사
- 2008년 : University of Limerick, 방문교수
- 2003년 ~ 현재 : 한성대학교 정보시스템공학과, 부교수

<주관심분야: Software Product Line Engineering, M2M, Aspect-Oriented Programming, Software Architecture>

윤 성 혜(정회원)

- 2012년 : 서강대학교 공학석사
- 2002년~2006년 : 스펙트라 연구원
- 2007년~2011년 : 웨어밸리 선임연구원
- 2012년~현재 : 서강대학교 박사 과정

<주관심분야: Software Reengineering, Software Evolution, Software Product Line Engineering>

※ 본 연구는 한성대학교 교내연구비 지원 과제임.