DOI: 10.3745/KIPSTD.2009.16-D.1.55

SOA 기반 애플리케이션 개발을 위한 Agile 프레임워크

신 승 우[†] · 김 행 곤[††]

요 약

최근 다양한 비즈니스 모델 및 컴퓨팅 환경이 웹 서비스로 집결됨에 따라 웹 애플리케이션 형태의 다양한 제품들이 개발되고 있다. 이에 따라 국내외 대부분의 기업/조직들이 웹 소프트웨어 개발에 있어서 SOA(Service Oriented Architecture)를 적용한 사례들이 늘어나고 있다. SOA는 네트워크가 가용한 소프트웨어 자원에 대해 느슨한 결합과 프로토콜 독립 그리고 표준화 분산 컴퓨팅 접근방법 이다. SOA는 다양한 기업의 서비스 조합을 통한 프로세스의 통합을 요구하는 비즈니스 사용자의 신속함과 융통성을 제공하는 향후 기업의 관심 있는 기술이다. 하지만 SOA의 표준모델에서는 특정한 개발 방법론이 제시되지 않아 기존의 방법론들을 적용하여 개발하거나 SOA 솔루션 업체에서 제안하는 방법론으로 SOA 기반 애플리케이션을 구축하고 있다. 이로 인해 SOA를 초기 도입하는 기업의 경우 개별 프로젝트 단위에서 부분적 도입에 그치고 있어서 SOA의 장점 활용하는 것이 제한적이다.

본 논문에서는 소규모 웹 프로젝트의 생산성 향상과 SOA의 효과적 적용을 위해 Agile 개발 방법론을 SOA에 적용하는 프레임워크를 제안한다. SOA 아키텍처를 기반으로 하여 Agile 방법론을 도입한 아키텍처를 설계 구현하며 프레임워크 개발과정에서 필요한 다양한 Practice요소를 도입하여 프로세스 모델을 제안한다. 프레임워크 실행을 통해 향상된 개발속도와 고객의 변화하는 요구 수용성 및 유지보수성 향상을 평가하게 된다.

키워드 : 애자일 개발 방법론, 서비스 지향 아키텍처, 익스트림 프로그래밍, 스크럼

Agile Framework for SOA-based Application Development

Seung-Woo Shin[†] · Haeng-Kon Kim[††]

ABSTRACT

Various business model and computing environments are currently merged into web services and many web related application products are also develop. Most of IT enterprises in Korea use the Service-oriented architecture (SOA) whenever they develop the web applications. SOA is an approach to loosely coupled, protocol independent, standards-based distributed computing where software resources available on the network are considered as Services. SOA is believed to become the future enterprise technology solution that promises the agility and flexibility the business users have been looking for by leveraging the integration process through composition of the services spanning multiple enterprises.

But, There are no specific development methodology to apply into SOA standard model until now. The developer uses the currently existing methodology to develop the application with SOA. The users have some limitations to use it.

In this paper, we suggest a Frameworks for applying agile methodology into SOA to address the productivity and quality of small web related project. We design and implement a frameworks architecture for applying the agile method into SOA and describe the process model to implement it. We finally evaluate the frameworks with productivity, flexibility and maintainability.

Keywords : Agile Development Methodology, SOA(Service Oriented Architecture), XP(eXtreme Programming), Scrum

1. 서 론

오늘날 많은 기업에서 이미 기업업무의 환경이 웹 서비스 환경으로 변화되고 있다. 대기업의 경우는 대부분 웹 서비스 환경을 바탕으로 한 기업업무의 환경을 가지고 있다. 이에 따라 웹 서비스 환경을 새로이 구축하거나 기존의 구축된 웹 서비스 환경을 SOA를 기반으로 하여 재구축하는 추세이며 새로운 애플리케이션을 도입할 때 SOA를 통해 구축된 환경에 적합한 애플리케이션의 도입을 많은 기업에서 적용중이다 해외의 경우 금융권을 중심으로 하여 많은 선진기업들이 SOA를 적용하고 있으며 국내 기업도 파일럿 프로젝트의 형태로 많은 기업에서 SOA의 도입을 시도하고 있다[1, 2].

SOA는 네트워크가 가용한 소프트웨어 자원에 대해 느슨한 결합과 프로토콜 독립, 그리고 표준화된 분산 컴퓨팅 접근 방법이다. SOA는 다양한 기업의 서비스 조합을 통한 프로세스의 통합을 요구하는 비즈니스 사용자의 신속함과 융

† 준 회 원 : 대구가톨릭대학교 컴퓨터정보통신공학과 석사과정
†† 종신회원 : 대구가톨릭대학교 컴퓨터공학과 교수
논문접수 : 2008년 5월 19일
수 정 일 : 1차 2008년 9월 23일
심사완료 : 2008년 11월 6일

통성을 제공하는 향후 기업의 관심 있는 기술이다[3]. 대기업 뿐만 아니라 중소규모의 조직의 경우도 최근 들어 웹 환경에서 업무처리가 이루어질 수 있도록 웹 서비스 기반의 업무환경을 도입하고자 하는 기업이 늘고 있으며 역시 최근 비즈니스 기술에서 많은 관심을 받고 있는 SOA를 통한 IT 거버넌스 환경과 서비스를 도입하고자 하는 조직도 꾸준히 증가하고 있는 추세이다[1].

기존의 아키텍처 기술에서 보면 적절한 구조를 효율적으로 이용하기 위해서는 적절한 개발 방법론이 제시되어야 한다. 따라서 SOA에서도 적절하게 적용 가능한 새로운 개발 방법론이 제안되어야 한다. 그러나 SOA를 위해 제안된 표준화된 방법론이 없어 개발 조직에서 도입을 하는 방법에 대한 제약사항이 되고 있다. 특정 회사의 플랫폼과 개발 방법론을 활용하지 않는 경우 기존에 제안된 개발 방법론을 SOA에 적용하는 방법 이외에는 명확한 방법론이 없어 도입에 대한 어려움을 겪고 있으며, 개발 제품의 생산성과 품질에도 영향을 미치고 있다. 이러한 사정으로 인해 SOA를 도입하는 경우에도 하나의 파일럿 프로젝트 단위에서 도입을 하는데 그치고 있어 SOA 각 단계의 재사용 자산을 효과적으로 활용하지 못하는 단점이 있다[1, 2, 6].

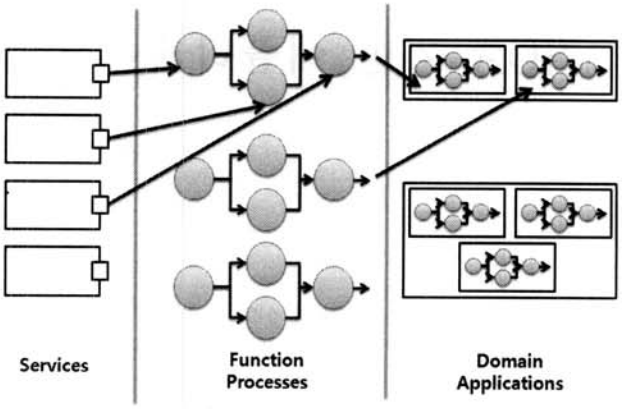
따라서 본 논문에서는 특정 회사의 플랫폼을 기반으로 하지 않아도 SOA를 도입할 수 있도록 SOA 애플리케이션 도입을 위한 프레임워크를 제안하여 중소규모의 기업에서 플랫폼 도입에 대한 부담 없이 SOA 환경을 바탕으로 한 서비스 지향 웹 애플리케이션 개발을 가능하도록 지원하고자 한다. 이에 대한 방법론으로 신속한 개발주기로 관심을 받고 있으며 도입에 따른 별도의 비용이 들지 않아 소규모 프로젝트 환경에 적합한 Agile 방법론을 기반으로 한 SOA 애플리케이션 개발에 적용하는 방법론을 제안한다. 기존에 정의된 SOA 아키텍처를 기반으로 Agile 방법론을 위한 아키텍처를 정의하고, Agile 방법론 개발 프로세스의 관리적인 요소를 중심으로 XP(eXtreme Programming) 방법론의 개발과정에서 필요한 다양한 프랙티스 요소를 도입한 새로운 프로세스 모델을 제안한다. 즉, 소규모 웹 프로젝트의 생산성 향상과 SOA의 효과적 적용을 위해 Agile 방법론을 SOA에 적용하는 프레임워크를 제안한다. 또한, SOA 아키텍처를 기반하여 Agile 방법론을 도입하는 아키텍처를 설계 구현하며 프레임워크 개발과정에서 필요한 다양한 요소를 도입하여 프로세스 모델을 제안한다. 프레임워크 실행을 통해 향상된 개발속도와 고객의 변화하는 요구수용성 및 유지보수성 향상을 평가하게 된다.

본 논문의 구성은 제 2장에서는 관련 연구를 서술하고 제 3장에서는 관련 프레임워크를 제안한다. 제 4장에서는 프레임워크 적용 및 실행 예를 제시하며 5장에서는 결론 및 향후연구를 제시한다.

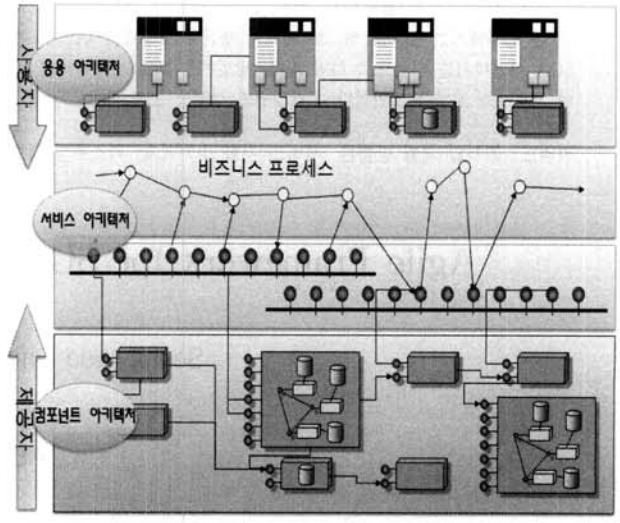
2. 관련 연구

2.1 SOA(Service Oriented Architecture)

SOA(Service-Oriented Architecture)는 잘 정의된 인터페이스와 서비스들 간 콘트랙트(contracts)를 통해 서비스라고 하는 애플리케이션의 다양한 기능 단위를 상호 연관시키는 컴포넌트 모델이다. 인터페이스는 하드웨어 플랫폼, 운영체제, 프로그래밍 언어에 독립적인 방식으로 정의된다. 따라서 다양한 시스템들에 구현된 어떤 서비스라도 일반적이고 통합된 방식으로 상호 교환이 가능하다. (그림 1)은 SOA에서 서비스들이 어떻게 애플리케이션으로 구성되는지에 대해 개념적으로 표현한 그림이다[3, 4].



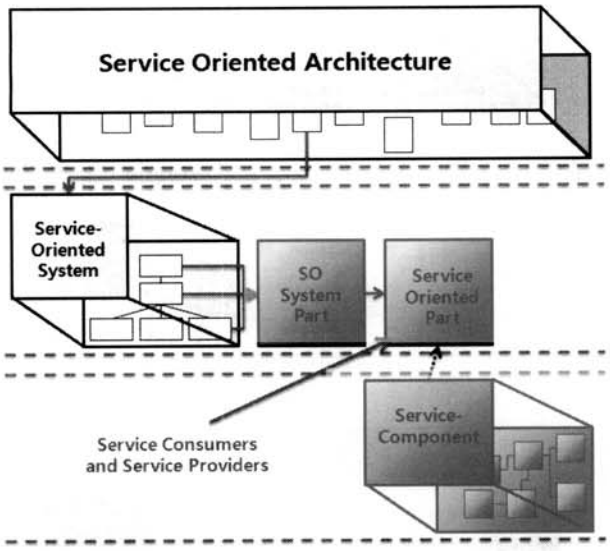
(그림 1) SOA의 기본개념



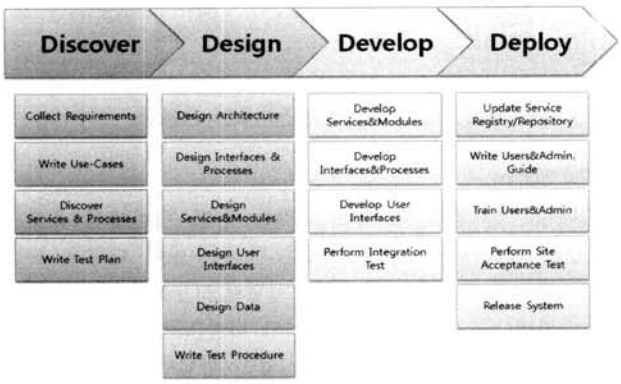
(그림 2) SOA의 3단계 아키텍처적 조망

SOA는 특정 구현에 얽매이지 않은 중립적인 인터페이스를 가졌기 때문에 서비스들 간 약결합(loose coupling)으로 알려져 있다. 약결합 시스템의 장점은 기민성과 각 서비스의 내부 구조 및 구현의 변화에 대응할 수 있는 능력을 가진다는 것이다. 약결합 시스템의 필요성은 비즈니스 애플리케이션이 변화하는 환경에 빠르게 적응해야 하는 데서 기인한다. 정책, 주력 비즈니스, 비즈니스 포커스, 파트너쉽, 산업 표준, 비즈니스의 본질에 영향을 미치는 관련 요소들은 늘 변화하기 때문에 이러한 환경에 유연하게 대처할 수 있는 비즈니스를 온디맨드 비즈니스(On demand business)로 명명하고 있다[3].

SOA는 세 가지의 아키텍처적인 요소들로 구성되어 있으며 이를 (그림 2)에서 보여주고 있다. 각각의 요소를 간단하



(그림 3) 소프트웨어 시스템의 3단계



(그림 4) 기존 SOA 개발방법론

게 살펴보면 아래와 같다[5]:

- 응용 아키텍처 : 하나 이상의 서비스 제공자나 통합하는 비즈니스 프로세스로부터 어떤 서비스를 소비하는 비즈니스 측면의 솔루션이다.
- 서비스 아키텍처 : 구현과 애플리케이션 소비사이에 다리를 제공하는 아키텍처로서 서비스 집합의 논리적인 뷰를 생성하고 공통 인터페이스와 관리 아키텍처에 의해 호출된다.
- 컴포넌트 아키텍처 : 다양한 환경을 지원하는 구현된 애플리케이션으로서 비즈니스 객체 또는 그들의 구현이다.

(그림 3)은 엔터프라이즈 소프트웨어 구조에서 SOA의 위치를 나타낸 것으로서 가장 핵심적인 정보는 서비스 컴포넌트이다. 아키텍처에서 나타나듯이 SOA의 경우 전사적인 시스템의 변경을 수반하여야 하므로 초기 도입 시에 많은 어려움을 겪고 있다. 이를 위해 국내기업인 Tmax Soft는 통합프레임워크를 통해 초기지원의 방안을 제시하고 있고(그림 4), IBM사의 경우 2007년 11월 발표한 Smart SOA라는 기술로 SOA를 도입하고자 하는 기업의 수준에 따라 도입 레벨을 달리하여 기본적인 도입에서 매우 복잡한 수준에 이르기까지 SOA의 도입을 지원하고 있다[7, 8]. 하지만 아직까지 중소 규모의 기업에 적합한 경량화된 SOA 프레임워크는 제안되지 않았으며 기존 플랫폼에서 일부만을 묶어 제공하는 형태에 머무르고 있다. 이에 본 논문에서는 중소규모의 기업에 적합한 경량화된 SOA의 도입에 관한 연구를 수행한다.

2.2 Agile 개발 방법론

Agile 소프트웨어 개발방법론은 Kent Beck이 소프트웨어 플래닝, 코딩, 디자인, 테스팅의 가치와 원리와 방법론을 통해 Extreme Programming (XP)을 시발점으로 하여 전통적인 개발방법론이 표준화된 프로세스에 가치를 두는 것과 달리 의사소통과 협동, 빠른 변화의 가치를 중시하는 90년대 후반에 태생된 개발방법론이다[6, 10].

Agile 개발 방법론들이 기존의 방법론과 비교할 때 공통적으로 가지는 특성은 다음과 같다[6, 15] :

- 태생 : 경험에서 시작된 방법들이 많다. XP의 경우 C3 프로젝트에서의 경험을 바탕으로 발전했으며, 크리스탈은 코오번이 여러 프로젝트 참여자들에게 수행한 인터뷰와 관찰을 통해 시작되었다.
- 가벼움 : '모든 것에 다 적용됨'을 강조하지 않는다. 또한 작은 단계로 시작해볼 수 있는 일들이 많다. 실천적이고 구체적이다.
- 의사소통 : 대화와 협력을 통한 상호작용을 강조한다. 또한 이를 강조하고 뒷받침하기 위한 여러 장치들을 둔다.
- 피드백에 기반한 적응적 시스템 : 짧은 반복(Iteration)을 강조하며, 그 반복 과정에서 얻은 경험들을 최대한 반영해 팀을 개선하도록 장려한다. 이를 위한 여러 장치들을 두며, '가볍다'는 장점은 짧은 반복을 실제로 가능하게 한다.

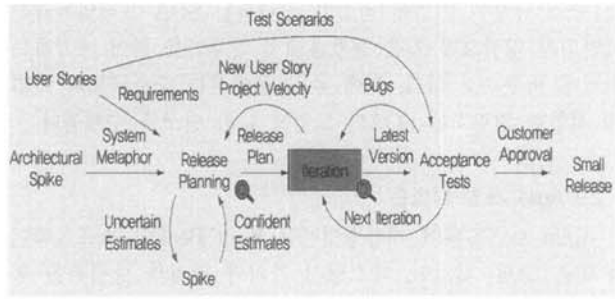
모든 Agile 소프트웨어 개발 방식에는 여러 가치(Values)들이 있다 :

- 검사와 적용(Frequent inspection and adaptation)
- 주기적인 배포(Frequent delivery)
- 협업과 긴밀한 커뮤니케이션 (Collaboration and close communication)
- 반응적 향상(Reflective improvement)
- (점증적인) 요구 사항, 기술, 팀 기능의 탄생(Emergence of requirements (incremental), technology, and team capabilities)
- 권한부여와 자가 구성(Empowerment and self-organization)

2.3 XP (eXtreme Programming)

XP는 대표적인 Agile 기법을 실천한 응용예로서 1999년 C3 프로젝트의 참여자인 켄트 벡과 론 제프리즈, 마틴 파울러 등의 경험에서 비롯된 방법이다. 그 이후 저자들의 경험과 유저 그룹 등에서의 여러 의견들이 오고가며 지속적으로 조금씩 수정되고 있다. XP의 전반적인 흐름은 (그림 5)와 같다[9].

초기에는 계획 게임(Planning Game)을 진행하게 된다. Planning Game에서는 요구사항에 대한 분석을 진행하고 고



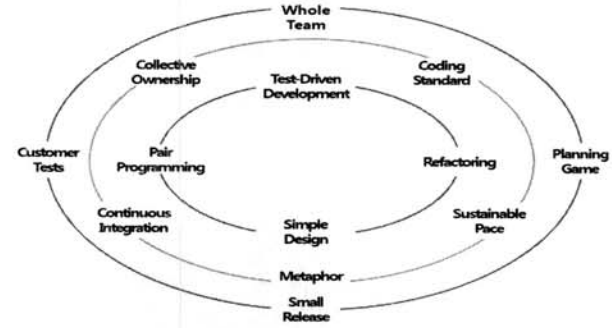
(그림 5) Agile 개발방법론 중 XP프로젝트의 진행과정

객과 함께 사용자 스토리(User Story)를 작성한다. 사용자 스토리는 추후 테스트 시나리오를 생각하여 고객 테스트(Customer Test)를 작성하는 데 이용한다. 사용자 스토리에 대해 개발자들은 스토리에 대한 구현 기간을 추정한다. 추정이 명확하지 않을 경우에는 명확하지 않은 부분에 대해 스파이크 솔루션(Spike Solution)을 시도한 뒤 추정할 수도 있다. 혹은 탐사 단계(Exploration)라 하여 해당 기술에 대한 초기 실험 기간으로 함께 두기도 한다.

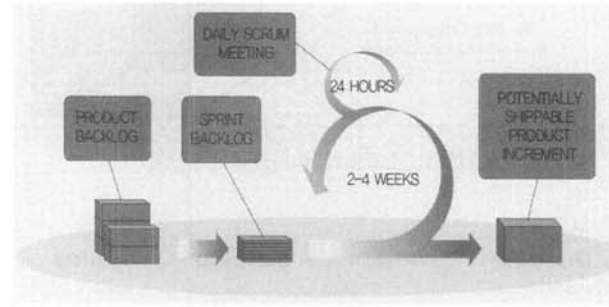
사용자 스토리는 다시 개발자들이 실제 진행할 작업들(Engineering Task)로 나눈다. 작업에 대한 추정을 거친 뒤 포인트들을 할당한다. 해당 포인트의 기준은 프로그래머의 이상적인 시간(programmer's ideal day; 아무런 방해를 받지 않는 상태에서 프로그래머가 최적의 효율을 발휘한다고 했을 경우의 기준)으로 계산한다[9, 10, 11].

최종적으로 고객에게 해당 사용자 스토리의 우선순위를 매기게 함으로써 구현할 사용자 스토리의 순서를 정하게 한다. 그리고 반복(Iteration)이 진행된다. 반복 중에는 사용자 스토리에 대한 일들을 분담해 작업을 진행한다. 반복은 일종의 타임 박스(timebox)로 기간을 정한다. 매 반복마다 스토리들에 대한 추정치들을 다시 계산하고, 이를 다음 반복의 추정에 적용한다. 반복이 끝날 때마다 실제 수행한 스토리들을 체크한 뒤, 고객 역할을 맡은 사람과 다시 계획 게임을 진행한다. 다음 반복에서는 이전 반복에서 수행한 포인트만큼의 일을 할당한다. 반복 중에는 매일 기립회의(Standup Meeting)를 통해 해당 프로그램의 전반적인 디자인과 짝 프로그래밍, 스토리의 진행 정도 등을 점검한다. 디자인은 이른바 '끊임없고 지속적인 디자인'을 지향한다. 이는 디자인 세션이나 테스트 주도 개발, 리팩토링 등을 통한 지속적인 디자인 개선과 매일 매일의 기립회의나 짝 프로그래밍 과정에서 개발자들 간의 대화를 통해 지속적으로 디자인되는 것을 의미한다.

(그림 6)은 XP방법론에서 실천해야할 요소들을 그룹별로 나타낸 것이다. 'Whole Team, Customer Tests, Small Release, Planning Game' 요소들은 프로젝트 팀의 전체 인원들이 고려를 해야 할 요소들이며 'Test-Driven Development, Pair Programming, Simple Design, Refactoring'요소들은 개발자들이 고려를 해야 할 요소 들이다.프로젝트 인원들의 실천 요소들 중에서 팀에게 필요한 프랙티스 요소들을 도입하여 프로젝트를 실시하게 된다[11].



(그림 6) XP Practice 요소



(그림 7) Scrum의 전반적인 진행도

2.4 Scrum

Scrum(스크럼)이란 단어 자체는 본래 럭비에서 유래된 것으로, 반칙이 일어났을 때 양 팀 선수들이 대형을 짜는 것을 의미한다. Scrum은 제프 서덜랜드(Jeff Sutherland)와 켄 슈와버(Ken Schwaber), 마이크 비들(Mike Beedle) 등에 의해 소개된 후 점차 확산되고 있는 방법이다. (그림 7)은 이런 Scrum의 전반적인 진행도를 그림으로 도식화 한 것이다[10, 12].

프로젝트는 2~4주간의 Sprint로 나뉜다. 각 팀들은 팀 별 Sprint Backlog를 가지고 있으며, 이 또한 Product Backlog처럼 늘 변할 수 있다. Sprint의 시작 날에는 Sprint 계획 미팅을 가지는데, 이때 모든 스테이크 홀더들(stake holder)은 다음 번 Sprint에 무엇을 할 것인지를 결정을 한다. Sprint 시 전달되는 결과물은 반드시 가시적이며 사용 가능한 형태로 완료되어 있어야 한다. Product Backlog 항목들의 추가와 우선순위의 설정은 고객에 의해 이뤄진다. 이 때 Sprint가 일어나는 동안에는 외부적인 변경이 불가능하다는 원칙을 가진다.

매일 혹은 정해진 날마다 팀 리더나 Scrum 마스터가 스크럼미팅(Scrum Meeting)을 마련한다. Scrum 미팅의 목표는 개발자들이 각 백로그의 일들에 집중하게 하고 진척 현황을 확인하며 팀 멤버들과 백로그 작업들의 우선순위에 대해 대화하고, 장애물과 리스크들을 함께 이야기하며 이를 해결하거나 완화하는 것이다. 이를 위해 Scrum 미팅에서 Scrum 마스터는 각 팀원들에게 다음과 같이 간단한 세 가지 질문을 던지고, 팀원들은 이에 대해 대답한다[10].

Scrum 미팅은 15~30분 정도 내의 짧은 시간 동안 진행한다. 문제점들에 대해 충분히 이야기할 시간을 주되, 문제

점에 대한 해결책을 만드는 시간은 별도로 둔다. Sprint의 마지막 날에는 전체 팀원들과 고객이 Sprint 리뷰 미팅을 가진다. 여기서는 전달된 증분들, 발견된 문제점들, 변화되어야 할 모든 것에 대한 이야기(프로젝트 취소도 포함) 그리고 다음 번 Sprint를 위한 새로운 추정과 이에 대한 팀 할당 등에 대해 점검한다.

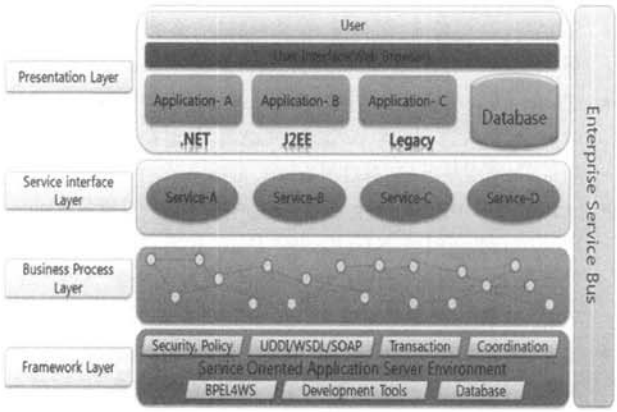
Scrum의 경우 개발 방법과 관련해 그 실천 사항을 따로 명시하진 않았고, 상대적으로 팀의 개선과 관리에 초점을 맞추고 있다. Scrum은 다른 방법론의 랩퍼(wrapper)로써 적용될 수 있고, 구체적인 개발과 관련해서는 다른 방법론의 실천 사항들을 이용할 수 있다. 즉, 프로젝트를 진행할 때 일종의 기본 가이드라인으로 활용될 수 있는 것이다[10, 12].

3. Agile 애플리케이션 프레임워크 설계 및 구현

Agile 방법론인 XP와 Scrum의 요소들은 독립적으로 존재 할 수도 있지만 Scrum을 가이드라인으로 하고 XP의 프랙티스 요소를 결합한 형태를 생각해보면 중소규모 기업에 적합한 SOA의 도입의 이상적인 방법론으로 적용이 가능하다. 따라서 본 논문에서는 Scrum과 XP를 결합한 방법론을 제안한다. 본 장에서는 SOA 애플리케이션 개발을 위한 프레임워크를 설계 구현한다. 이 프레임워크는 SOA기반의 애플리케이션이 구동되는 아키텍처와 이 아키텍처에서 구동될 서비스 모듈을 구현할 수 있는 절차를 제시하고 있는 프로세스 모델로 구성한다.

3.1 Agile 프레임워크 아키텍처

SOA를 도입하는 기업의 목표는 주로 개발에 대한 비용을 줄이고 빠른 시간 안에 구현을 가능하게 하며 기존 모델을 표준화된 형태로 랩핑하여 각 애플리케이션간의 효율적인 의사소통을 가능하게 하는 것이다. 이를 위해 하부 계층의 프로세스에 대한 지식 없이 상위 계층에 대한 정보만으로 애플리케이션을 구현할 수 있도록 추상화되어야 한다. 따라서 Agile 개발방법론을 적용하기 위한 프레임워크의 (그림 8)과 같이 아키텍처를 제안한다. 아키텍처는 Service Oriented Architecture reference model을 바탕으로 한다[19].



(그림 8) SOA 기반 애플리케이션 개발지원 Agile 프레임워크 아키텍처

제안하는 아키텍처는 총 4개의 레이어로 나누어지며 각 레이어는 다음과 같은 기능을 수행한다 :

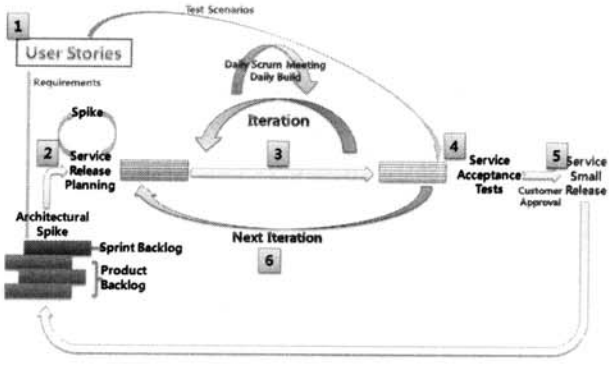
- Framework Layer : 프레임워크 레이어 에서는 서비스를 중심으로 처리하기 위한 기반이 되는 표준을 정의하고 제공한다. 또한 이 레이어 에서는 보안정책, 트랜잭션, 개발 툴 및 데이터베이스를 포함한다.
- Business Process Layer : 실질적인 프로세스의 영역으로 비즈니스 요구사항의 구현단위를 제약사항이나 관계를 바탕으로 하여 비즈니스 로직으로 표현하는 레이어이다.
- Service Interface Layer : 비즈니스 로직에서 나오는 일련의 과정들을 모듈화 하여 애플리케이션에서 이용이 가능하도록 하고 애플리케이션(Presentation Layer에 존재하는)이 이질적인 플랫폼이나 시스템에서 구축된 것을 표준화된 방식으로 처리하는 레이어 이다.
- Presentation Layer : 다양한 플랫폼 위에서 구동되는 다양한 애플리케이션을 위한 애플리케이션으로 기존 애플리케이션은 Wrapping등의 방법을 통하여 지원하고, 새로이 개발되는 서비스 지향 애플리케이션은 프레임워크에서 지원하는 다양한 표준기술(HTML, XML, SQL, JSP, ASP)과 보안처리규격 등을 통해 애플리케이션을 개발하고 운용 할 수 있는 레이어 이다.
- Enterprise Service Bus: 위의 4계층 레이어의 통합이 가능하도록 연결해주는 가교역할을 하는 미들웨어로 약결합(Loosely Coupling) 연결을 지원하여 유연성과 민첩성을 보장 해주는 역할을 수행한다.

3.2 Agile 프레임워크 적용 프로세스 모델

Agile 방법론을 SOA 애플리케이션에 적합하고 개발자가 쉽게 도입 적용할 수 있도록 하기 위해 향상된 프로세스 모델을 제안한다. Scrum의 경우 프로젝트 관리적인 측면에 있어서는 매우 우수한 모델이지만 그것을 개발하는데 필요한 구체적인 프랙티스에 대해서는 별다른 가이드라인이 존재하지 않는다. 반면에 XP의 경우 개발자를 위한 다양한 프랙티스들이 존재하며 이를 통해 합리적으로 개발할 수 있는 모델이다. 하지만 XP의 경우 개발자 중심의 측면이 강조되어 전체적인 일정을 관리하고 보완하는 부분에 있어서는 부족한 부분이 있다. 따라서 본 논문에서 제안하는 모델은 Scrum 프로세스를 중심으로 XP의 프랙티스 요소들을 결합한 형태의 모델이다.

(그림 9)에서 보는바와 같이 기본적인 흐름은 Scrum의 프로세스의 흐름을 따른다. 이는 프로젝트의 관리적인 측면을 우선시하기 위함이다. 개발일정을 관리하는 측면의 개념은 XP에서도 Scrum의 모델을 참조한 것으로 알려지고 있으며 그만큼 강력하다는 의미라고 할 수 있다. 중소기업에서 개발방법론을 도입하는 경우 실제 프랙티스나 개발 사례 등이 초기 도입에 많은 도움이 된다. 이에 Scrum의 프로세스에 XP의 프랙티스 요소들을 더하여 제안하는 것이다.

두 방법론이 공통적으로 가지고 있거나 유사한 개념을 가지고 있는 요소는 좀 더 명확하게 명시된 요소를 우선시 하



(그림 9) 제안 Agile 프레임워크 프로세스 모델

여 고려하였다. 제안한 프레임워크 프로세스 모델의 수행 절차는 다음과 같다. (그림 9)에 1~6까지의 번호는 아래의 절차에서 수행되는 순번과 같다.

1) Service Planning Game : 프로젝트 팀은 사용자와 함께 사용자스토리(User Story)를 작성하게 되는데 이를 통해 테스트 시나리오가 나오게 된다. 사용자 스토리를 바탕으로 요구사항들을 Product Backlog로 쌓는다. 작성된 사용자 스토리는 비즈니스 프로세스 서비스 후보를 식별하고 서비스 오퍼레이션 후보를 식별한다. 기존 SOA 개발방법론에서는 Discover 단계에 해당하며 서비스와 프로세스를 정의하는 단계라고 할 수 있다.
2) Service Release Planning : 사용자 스토리를 바탕으로 만든 요구사항들을 한번의 Iteration단위로 나누고 필요할 경우 Architectural Spike나 Spike를 통해 구동환경이나 스펙 등을 정의하고 릴리즈에 대한 계획을 세운다. 이때 앞서 식별된 비즈니스 프로세스 서비스 후보와 서비스 오퍼레이션 후보 중 구현 불가능 한 요소나 우선 구현 필요성이 낮은 서비스 단위는 계획에서 제거하거나 우선순위를 낮추어 차후 구현으로 Product Backlog의 차후 구현 순위로 변경한다. 필요에 따라 서비스 단위의 분할을 하거나(비즈니스 프로세스 서비스가 서비스 오퍼레이션 하나씩 가지도록 분할) 재사용 할 수 있는 애플리케이션 서비스 후보를 식별하여 랩핑(Wrapping)하여 사용할 수 있도록 서비스로 정의한다. 기존 SOA 개발방법론 에서는 Discover와 Design의 단계의 일부를 포함하고 있으며 애플리케이션의 아키텍처와 인터페이스 등을 설계한다.
3) Service Sprint Backlog : Sprint Backlog는 한번의 Iteration에 수행되는 단위로 실제적으로 구현 하는 단계라고 할 수 있다. 이때 XP의 다양한 Practice(Pair Programming, Refactoring 등)를 통하여 Iteration을 수행한다. 한번의 Iteration이 끝나면 해당하는 Sprint에서 수행된 결과를 바탕으로 다음 Sprint Backlog를 만들어내게 된다. 이때 한번의 Iteration은 10~30일정도로 정할 수 있으며 프로젝트의 성격과 팀의 규모에 따라 유동적으로 조절한다. 물론 Iteration동안 Test Scenario를 바탕으로 테스트를 수행하며 개발절차를 진행하며 TDD(Test Driven Development)를 사용하는 것이 효과적이다. 기본적으로는 서비스 오퍼레이션 단위를 하나의 구현단위로 하고 Iteration동안 이러한 서비스 오퍼레이션을 그룹화 하여 하나의 서비스로 도출하도록 한다. 기존 SOA 개발방법론에서는 Design과 Develop 단계의 일부를 포함하고 있으며 각 서비스 모듈을 구현하는 단계라고 할 수 있다.
4) Service Acceptance test : Sprint Backlog를 통해 몇 번의 Iteration이 수행된 후 사용자와 함께 Acceptance test를 통해 현재 개발 중인 애플리케이션의 테스트를 수행한다. WSDL등의 언어를 정제하는 등의 과정을 통해 테스트되는 서비스의 Refactoring을 수행할 수 있다. 기존 SOA 개발방법론에서는 Develop 단계의 일부를 포함하고 있으며 사용자 스토리를 통해 얻어진 테스트 시나리오와 추가적인 기술적인 사항을 고려하여 수행하는 단계이다.
5) Service Small Release : 사용자가 Acceptance Test를 통해 모든 것을 만족한다면 Small Release를 수행한다. 개발자와 사용자가 서로 다른 의견을 가지고 있다면 Small Release가 일어나지 않으며 의견을 조율하여 다시 Iteration을 수행하게 된다. 기존 SOA 개발방법론에서는 Deploy단계에 해당하며 서비스를 등록하고 레파지토리를 정리하는 단계이다.
6) Next Product Backlog Sprint : 하나의 Small Release를 수행하였으면 다음 Product Backlog의 아이템을 Sprint Backlog로 보내고 다시 Release Planning을 수행하거나 바로 Sprint에 들어가게 된다. 애플리케이션의 크기에 따라 Iteration을 반복하여 애플리케이션 서비스를 정의하여 비즈니스 프로세스 서비스 단위들을 프레젠테이션 계층에서 인터페이스를 통해 연결한다. 기존 SOA 개발방법론은 반복적인 개발 단계를 가지지 않으므로 이 단계와 매칭되지 않으며 하나하나의 서비스 모듈단위를 하나의 Iteration에서 구현하는 형태로 구성된다.

제안 프레임워크가 기존 SOA 모델에 비하여 가지게 되는 장점은 아래와 같다.

- 특정 회사나 기업의 솔루션이나 플랫폼과 관계없이 표준화된 SOA 환경을 구축할 수 있다.
- 좀 더 구체화된 프랙티스를 제공하여 원활한 도입이 가능한 가이드라인 역할을 제공한다.
- SOA를 도입함에 있어서 중요시되는 유연성(Flexibility)이 XP의 가치를 통해 극대화 된다.
- 중 소규모 기업에서 가장 힘든 요소인 전사적인 도입이 아닌 점증적인 XP의 도입과 SOA의 도입을 가능케 하여 기업의 위험부담을 경감시킬 수 있다.
- Scrum의 Time-Boxing등의 개발자에 대한 압박을 Pair-Programming이나 Sustainable Pace 등의 개발자 중심의 가치들로 인하여 균형 있게 하여 프로젝트를 수행하는 인원의 장기 프로젝트에 대한 부담을 줄일 수 있다.
- 짧은 개발주기의 반복으로 인해 서비스를 개선하거나 새로운 서비스 프로세스의 추가, 재정의에 대한 유연성을 보장한다.
- SOA의 거버넌스들을 Scrum을 통하여 효율적으로 통제

하면서 XP를 통해 개발자들에 대한 다양한 지원들을 제공하므로 기업 및 실무자 양쪽에게 보다 나은 가치를 제공한다.

4. Agile 프레임워크의 적용 및 실행 예

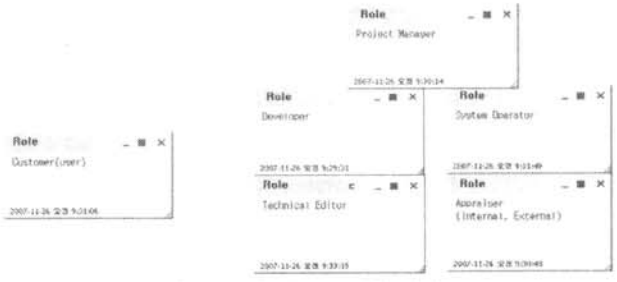
본 논문에서 제안한 Agile 프레임워크가 SOA을 기반으로 한 웹 애플리케이션 개발에 얼마나 적절한지 확인하기 위하여 제안 프레임워크 적용 예제를 제시하고 수행한다.

본 논문에서 수행할 예제 프로젝트는 프로젝트를 수행함에 있어서 프로젝트의 관리를 수행하는데 도움을 주는 프로젝트 지원 도구를 개발하는 것에 대해 수행하며 예제는 실제 포스트잇을 이용하여 Task Board에 작성을 하였으나 가독성을 고려하여 본 논문에서는 컴퓨터의 메모기능을 수행해 주는 별도의 도구로 내용을 기록한 것을 캡처하여 나타낸다.

수행 순서는 제안된 프로세스 모델에 나오는 절차에 따라 수행하였으며 이에 따라 가장 먼저 사용자 스토리를 작성하기 위해 Planning Game을 수행한다.

먼저 고객과 함께 사용자 초기역할을 식별하였다. 초기역할 식별 후 식별된 사용자들을 통합하고 세분화하여 총 여섯 개의 역할(고객(의뢰인, 사용자), 개발자, 관리자, 내 외부 평가자, 프로젝트 관리자, 기술 문서 작성자)로 분류하고 연관성에 따라 (그림 10)과 같이 나타낸다.

사용자 역할 식별이 끝난 후 역할 모델링을 통해 해당 역할에 대한 좀 더 구체적인 내용을 언급하기 위해 사용자 역할 모델링을 (그림 11)과 같이 수행한다.



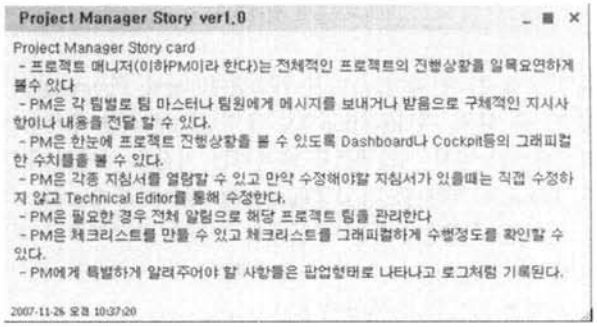
(그림 10) 사용자 역할 식별

(그림 11) 사용자 역할 모델링

사용자 역할 모델링에서는 사용자가 필요로 하는 요구사항을 정의하는 단계라기보다는 사용자가 실제 소프트웨어를 사용하는 빈도나 소프트웨어의 특성, 사용자가 소프트웨어를 사용하는 일반적인 목적 등에 대해서 정의하는 단계이므로 이에 맞춰 사용자의 기본적인 역할을 정의하고 부가적으로 사용자의 특성 등을 포함하여 역할 모델링을 수행한다.

사용자 역할 모델링을 수행한 다음에는 각 사용자에 대한 사용자 스토리 카드를 작성한다. 사용자 스토리 카드는 앞서 정의한 역할 모델링을 바탕으로 하여 구체적으로 사용자별로 수행할 수 있는 세부적인 기능 요소를 논의하여 스토리를 작성해나가는 단계이다. 이에 따라 각 사용자 별로 스토리카드를 작성하였다. (그림 12)는 이러한 스토리카드 중 프로젝트 매니저에 대한 스토리카드에 대한 예를 나타낸다. 이 스토리카드를 바탕으로 차후에 CRC카드(Class, Responsibilities and Collaboration)나 UML등을 통하여 모델링을 실시한다. (그림 12)의 스토리를 바탕으로 <표 1>과 같이 스토리 추정을 실시할 수 있다.

스토리 추정치는 예상되는 기간(일)으로 계산하고 우선순위는 사용자가 우선적으로 구현되기를 원하는 기능부터 낮은 번호를 부여한다. 이를 통해 Release Planning을 정할 수



(그림 12) 사용자 스토리카드의 예(Project Management)

〈표 1〉 스토리 추정 결과

Story	Estimate	Priority
프로젝트 매니저(PM)는 전체적인 프로젝트의 진행상황을 일목요연하게 볼 수 있다	7	1
PM은 한눈에 프로젝트 진행상황을 볼 수 있도록 Dashboard나 Cockpit등의 그래피컬한 수치들을 볼 수 있다	14	4
PM은 각종 지침서를 열람할 수 있고 만약 수정해야 할 지침서가 있을 때에는 직접 수정하지 않고 Technical Editor를 통해 수정한다.	4	2
PM은 필요한 경우 전체알림으로 해당 프로젝트 팀을 관리한다.	6	3
PM은 체크리스트를 만들 수 있고 체크리스트를 그래피컬하게 나타내 확인 할 수 있다	10	5
PM에게 특별하게 알려주어야 할 사항들은 팝업형태로 나타나고 로그처럼 기록된다.	4	6

〈표 2〉 Release planning

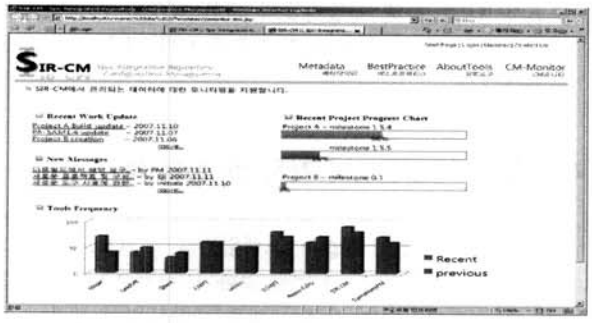
Iteration 1(10 days)	Iteration 2(14 days)
프로젝트 매니저(PM)는 전체적인 프로젝트의 진행상황을 일목요연하게 볼 수 있다(7)	PM은 한눈에 프로젝트 진행상황을 볼 수 있도록 Dashboard나 Cockpit등의 그래피컬한 수치들을 볼 수 있다(14)
PM은 각종 지침서를 열람할 수 있고 만약 수정해야 할 지침서가 있을 때에는 직접 수정하지 않고 Technical Editor를 통해 수정한다(4)	PM은 체크리스트를 만들 수 있고 체크리스트를 그래피컬하게 나타내 확인 할 수 있다(10)
PM은 필요한 경우 전체알림으로 해당 프로젝트 팀을 관리한다(6)	PM에게 특별하게 알려주어야 할 사항들은 팝업형태로 나타나고 로그처럼 기록된다(4)

〈표 3〉 사용자스토리에 따른 테스트 시나리오의 예

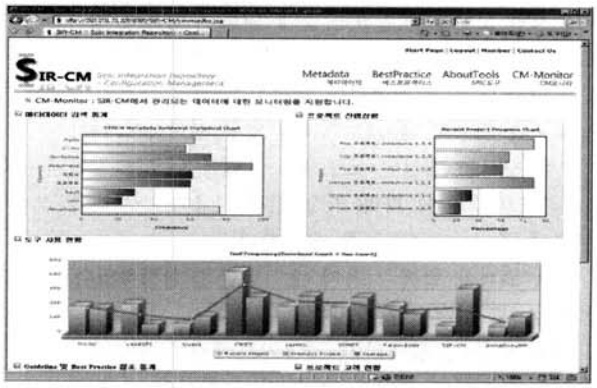
사용자 스토리	테스트 시나리오
PM은 각종 지침서를 열람할 수 있고 만약 수정해야 할 지침서가 있을 때에는 직접 수정하지 않고 Technical Editor를 통해 수정 한다	• PM외의 사람이 지침서 열람을 시도한다. • Technical Editor외의 사용자가 수정을 시도한다. • 존재하지 않는 지침서 열람을 시도한다. • PM이 지침서를 열람중일 때 Technical Editor가 지침서 수정을 실시한다. • Technical Editor가 지침서 수정을 실시할 때 PM이 해당 지침서를 열람한다. • PM보다 상위권한 사용자가 열람한다. • PM보다 상위권한 사용자가 Technical Editor에게 지침서 수정을 지시한다. • 수정이 완료되었을 때 PM에게 지침서 수정이 완료되었음을 알려준다.

있다. <표 2>와 같이 하나의 완료된 Release Planning을 작성 할 수 있고, 각 Iteration별로 추정된 스토리를 수행할 수 있도록 나누어서 프로젝트를 수행하는 기간을 나타낼 수 있다. Iteration 별로 Release Planning을 맞추면서 Iteration 별로 기간이 추정결과에 비해 줄어드는 것은 프로젝트에 참여하는 프로젝트인원이 동시에 수행할 수 있는 경우를 고려하여 추정된 기간이기 때문이다.

Release Planning이 완료가 되면 비로소 Iteration 1에서 Sprint를 실시한다. 팀원 간의 Pair Programming등의 XP Practice요소들을 기반으로 하여 하나하나의 사용자 스토리 카드의 내용을 토대로 프로젝트를 수행한다. CRC(Class, Responsibilities and Collaboration)카드나 기타 방법을 통해 설계를 하고 daily Scrum Meeting을 통해 매일의 진행상황을 관리하고 문제점을 파악하게 된다. 한번의 Iteration이 종료하면 처음 사용자스토리를 작성 한 후 작성된 테스트 시나리오로 인수테스트를 실시하게 되어 해당하는 스토리카드의 내용이 잘 수행되었는지를 사용자와 함께 테스트 하게 된다. <표 3>은 사용자 스토리 카드 중 하나의 테스트시나리오를 작성한 예이다. 스토리카드가 작성하던 중에 주기적으로 사용자와의 커뮤니케이션을 위해 스토리 카드에 기재된 사항이 반영된 기본적인 UI를 통하여 스토리카드의 개선을 수행한다. (그림 13)은 Project Manager 사용자 스토리를 바탕으로 하는 커뮤니케이션 UI의 예이다. 위의 과정을 통해 Iteration 1이 종료되면 Small Release를 수행하게 되고 해당 Iteration에서 부족하고 개선해야할 요소들을 Product Backlog에 넣어서 보완하여 다음 Iteration으로의 진행을 하



(그림 13) 사용자 커뮤니케이션을 위한 UI의 예



(그림 14) 하나의 서비스 모듈이 Small Release 된 결과물

〈표 4〉 SOA애플리케이션 개발 시 제안 프레임워크 모델 도입의 장점

제안 Agile Practices	SOA Principles	장점
Small Release	Loose Coupling, Service Granularity	Small Release 프랙티스는 독립적인 하나의 컴포넌트 형태로 작성하기 용이하다. 이는 SOA애플리케이션을 개발하는데 바탕이 되는 컴포넌트 요소를 구현하는데 있어서 Loose Coupling과 서비스의 입자 성을 보장 할 수 있다. 또한 기업의 민첩성을 향상시키고 고객의 요구사항을 수용하는데 있어서도 작은 단위로 변경하고 디플로이하는 것이 유리하다.
TDD Customer Tests	Autonomous	TDD(Test-Driven Development)프랙티스의 핵심요소는 자동화 테스팅에 있다. SOA원칙의 자동화도 비즈니스 프로세스 외에 프로젝트의 생성물들의 자동화된 테스트를 고려해야 한다. 고객과의 협업된 테스트는 개발업체와의 커뮤니케이션을 원활하게 하여 개발시간과 비용을 줄일 수 있도록 하는 요소가 된다.
Coding Standard, Simple Design	SOA Simplifies Development	표준화된 코딩(Pair Programming, Simple design)을 실시하게 되면 단순하고 표준과 가까운 코드의 개발이 가능해진다.
Refactoring Continuos Integration	Designed for reuse, well defined interface	코드를 리팩토링하여 코드의 품질을 향상시키는 것은 결국 잘 정의된 컴포넌트나 인터페이스를 제공할 수 있게 된다. 이에 따라 재사용에 더 적합한 컴포넌트나 모듈을 사용할 수 있게 된다. 또한 연속적인 통합 가운데에 표준화된 인터페이스를 통해 연결을 하게 되므로 통합에 발생하는 단점요소를 제거하게 되는 장점을 가진다.

게 된다. Iteration이 끝나고 나서의 Small Release된 하나의 서비스 모듈은 아래의 (그림 14)와 같다.

<표 4>는 제안 Agile 프레임워크 모델을 기반하여 SOA 애플리케이션 개발에 실제 적용, 평가하여 그 장점을 요약

〈표 5〉 기존 SOA 개발방법론과의 비교

	T 시스템	E 시스템	제안 Agile 프레임워크
프로세스 모델	정형화됨 (4단계)	세가지방법을 제시(상향식,하향식, 혼합)	반복수행으로 정형화됨
도입비용	X	O	O
생산성	X	△	O
개발환경	O	△	△
확장성	O	△	O

한 것이다.

기존 SOA방법론 중 널리 알려진 방법론인 Tmax SOA 개발방법론[8]과 Thomas Earl의 저서의 개발 방법론[17]과의 비교를 수행하면 다음 <표 5>와 같이 나타난다. 각 항목별로 간단히 비교한 내용은 아래와 같다.

- 프로세스모델 : T 시스템에서 제안한 방법론의 경우 크게 4단계로 나뉘며(Discover,Design,Develop,Deploy)4단계 별로 세부 단계로 구성되어있다. 정형적이므로 기업의 구조화가 잘 이루어진 경우에는 체계적인 적용이 가능하나 파일럿 형태로 적용하기에는 복잡하며 방법론 자체가 자사의 플랫폼위주로 맞추어져 있다. E시스템이 제안한 방법론은 기본적인 라이프사이클을 토대로 상향식 설계와 하향식 설계, 그리고 이 두 방식을 조합한 혼합 방식의 방법론으로 존재한다. E시스템은 그중 두 방법론을 수용하여 상반되는 요구사항을 수렴한 모델로 상, 하향식에 비해 복잡하나 가장 생산적인 모델로 평가하고 있다. 그러므로 소규모의 기업환경에 적합한 방법론이라기보다는 중견기업 이상의 규모의 기업에서 상, 하향식 방법론을 부서별로 적용하는 형태의 방법론으로 이루어질 가능성이 크다. 제안한 Agile 프레임워크의 경우 T시스템이나 E시스템의 모델과 같이 SOA의 틀에 완벽하게 들어맞지는 않는 모델이다. 하지만 기본적인 개념들을 가지면서도 유연하게 반응할 수 있고 프로젝트를 수행하는 구성원이 반복적인 수행을 통해 SOA 애플리케이션 구현에 적합한 형태로 정형화 해나갈 수 있다는 장점을 가진다. 또한 기존 Waterfall 방법론에 비해서는 수행과정에서의 리스크를 줄일 수 있다.
- 도입비용 : T시스템의 개발방법론을 도입할 경우 자연스레 자사의 플랫폼을 사용하게 된다. 이에 따른 비용은 특히 도입초기에 많은 컨설팅 비용 및 소프트웨어 도입비용이 발생하게 된다. 하지만 E시스템이나 제안한 프레임워크의 경우는 구체적인 솔루션이나 툴을 제안하고 있지는 않지만 오픈소스위주의 도구들을 위주로 채용할 수 있고, 특히 제안한 Agile 프레임워크 도입의 경우는 애자일 철학이 담겨있는 툴을 적용하기 용이하므로 비용뿐만 아니라 프로세스모델에 수용적인 툴 도입이 가능하다.
- 생산성 : SOA를 도입하는 가장 큰 이유는 고객의 요구사항을 빠르게 수용하여 우수한 품질의 애플리케이션을 단시간 내에 배포할 수 있도록 해주는 것이다. 물론 T시스템의 경우 체계적인 프로세스와 효율적인 도구를 제공하고 있으므로 우수한 품질의 애플리케이션을 개발할 수 있으나 문서화 위주의 프랙티스들이 존재하여 개발시간을 줄이는 데에는 효과적이지 않으며 기본적으로 하향식 개발방법론을 따르고 있으므로 시간과 비용적인 요소들은 많이 소모된다. 반면 E시스템의 방법론의 경우 애플리케이션에 맞게 상향식(통합위주), 하향식, 혼합 방식을 적용할 수 있으므로 생산성을 높이는 데에 도움을 줄 수 있으나 SOA를 처음 도입하는 경우 혼란만 가중시킬 수 있다는 사실도 고려를 해야 한다. 그러므로 제안한 프레임워크와 같이 지속적인 반복을 통해 개선해나가는 형태가 생산성 향상에 가장 큰 도움이 된다.
- 개발환경 : 솔루션을 갖춘 제품을 잘 사용하면 효율적인 것은 자명한 사실이다. 초기도입비용을 극복하고 프로세스를 프로젝트 팀이 내재화에 성공한다면 우수한 개발환경에서 빠른 속도의 애플리케이션 개발이 가능하다. 이에 비해 E시스템의 방법론이나 제안한 Agile 프레임워크의 경우 올바른 툴의 선택을 하지 못하면 중복작업의 양이 증가 하는 비효율적인 요소를 가질 수 있다.
- 확장성 : 여기서 확장성이란 서비스 모듈이 점점 증가하여 일정 수준이상의 서비스 모듈이 되었을 때의 커버리지에 관한 것을 일컫는다. T시스템이나 E시스템의 개발방법론의 하향식 전략으로 접근할 경우 초기 서비스정의 등에서 소모되는 시간이 많지만 충분히 향후 확장에 대해서 고려하여 서비스들을 정의할 수 있다. 기본적으로 SOA는 뛰어난 확장성과 유지보수성을 기본특징으로 한다. 그러므로 확장성에 대하여는 더 이상 의심할 여지가 없다고 할 수 있다. 제안한 Agile 프레임워크의 경우는 문서화 등이 미약하고 초기 서비스정의를 완벽하게 하고 구현을 수행하지 않으므로 상대적으로 확장성이나 유지보수성은 낮다고 할 수 있다. 하지만 기본적인 애자일 프랙티스를 적용하여 개발하면 가독성이 뛰어난 코드와 재사용 가능한 구현을 하게 되므로 반복적인 수행을 통해 우수한 확장성을 획득할 수 있다.

5. 결론 및 향후 연구

본 논문에서는 소규모 웹 프로젝트 생산성 향상과 SOA의 효과적 적용을 위해 Agile 방법론을 SOA에 적용하는 프레임워크를 제안하였다. SOA 아키텍처를 기반하여 Agile 방법론을 도입하는 아키텍처를 설계 구현하며 프레임워크 개발과정에서 필요한 다양한 요소를 도입하여 프로세스 모델을 제안하였다. 프레임워크 실행을 통해 기존의 다른 모델에 비해 좀 더 신속하고 향상된 개발속도와 고객의 변화하는 요구 수용성 및 유지보수성 향상을 평가 할 수 있다. 이를 통해 우리는 제안한 Agile 프레임워크 모델이 SOA의 프로젝트를 수행하는데 있어서 성공적인 진행을 이끌어갈 수 있는 방법론임을 확인하였다.

본 논문은 프로젝트 전반에 관한 부분과 비즈니스 프로세스에 대한 연구보다는 개발과정에 초점이 맞춰져 있다. 또한 개발과정 이전에 프로젝트를 수주하여 웹 서비스를 따라서, 향후 프레임워크를 개선해 나가기 위해 현재 모델이 가

지는 장단점을 확인하였다. 향후 기업에서 적용한 결과(Practice Case)의 분석에 대한 지속적인 연구가 요구된다.

참 고 문 헌

[1] 고원규, "SOA 시장, 어디쯤 와 있나", 경영과 컴퓨터 2007년 6월호, 2007.

[2] 이상일, "SOA 방법론과 시장진단", 경영과 컴퓨터 2006년 7월호, 2006.

[3] IBM, "SOA와 웹서비스 입문," http://www.ibm.com/developerworks/kr/webservices/newto/, IBM Developer Works, 2008.

[4] Ing-Yi Chen, Chao-Chi Huang, "An SOA-based software deployment management system," Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, IEEE, pp.617-620, 2006.

[5] David Sprott and Lawrence Wilkes, "Understanding Service Oriented Architecture," Microsoft Architecture Journal 1, Microsoft, pp.10-17, 2004.

[6] Pal Krogdahl, Gottrried Luef, and Christoph Steindl, "서비스 지향과 기민성: 성공적인 SOA 개발, Part 1: SOA와 애자일 방식의 기초", IBM Developer Works, http://www.ibm.com/developerworks/kr/library/ws-agile1/index.html, 2006.

[7] IBM, "Smart SOA: Best Practices for agile innovation and optimization," Service oriented architecture White paper, 2007.

[8] Tmax Soft, "4Frameworks Solution," http://www.tmax.co.kr/

[9] Ron Jeffries, "What is Extreme Programming?. XProgramming.com : an agile software development resource," http://www.xprogramming.com/xpmag/whatisxp.htm, 2001.

[10] 강석천, 강규영, 김창준, "변화를 꿈꾸는 개발방법론 애자일(Agile)", 월간 마이크로소프트웨어 2007년 3월호, 2007.

[11] Kent Beck, 'Extreme Programming Explained : Embrace change,' 2nd ED., Pearson Education, 2005.

[12] Ken Schwaber, "What is Scrum?, Scrum: Its about common sense," http://www.controlchaos.com/about/, 2007.

[13] Ken Schwaber, Mike Beedle, Robert C. Martin, 'Agile Development with Scrum', Prentice Hall, 2001.

[14] Ronaled E. Jeffries, Ann Anderson and Chet Hendrickson, 'Extreme Programming Installed', Pearson Education, 2003.

[15] Korea eXtreme Programming Users' Group, "Korea eXtreme Programming Users' Group," http://www.xper.org/, 2007.

[16] Thomas Earl, 'Service-Oriented Architecture: A field guide to integrating XML and Web Service,' Pearson Education, 2004.

[17] Thomas Earl, 'Service-Oriented Architecture:Concepts, Technology, and Design,' Prentice-Hall, 2005.

[18] Mike Cohn, 'User Stories Applied: For agile software development,' Pearson Education, 2004.

[19] OASIS Open, "Reference Model for Service Oriented Architecture 1.0," http://docs.oasis-open.org/soa-rm/v1.0/, 2006.

[20] William Pietri, "An XP Team Room," http://www.scissor.com/resources/teamroom/, 2004.

[21] 박동식, 신호준, 김행곤, "SOA 기반의 웹 서비스 컴포넌트 개발에 관한 연구", Journal of Korea Multimedia Society Vol.7, No.10, pp.1496-1504, 2004.

[22] 한상우, 박선희, 노재호, "Service Oriented Architecture 적용을 위한 서비스 식별 기법," 정보과학회지, Vol.24, No.11, 2006.

[23] Stefan Tilkov, "10 Principles of SOA," Stefan Tilkovs Weblog, http://www.innoq.com/blog/st/2006/12/13/10_principles_of_soa.html, 2006.

신 승 우

e-mail : selab@cu.ac.kr

2007년 대구가톨릭대학교 컴퓨터공학과 (공학사)

2007년~현 재 대구가톨릭대학교 컴퓨터정보통신공학과 석사과정

관심분야 : SOA, Agile methodologies, Mobile Web Development, Web Frameworks, MDA

김 행 곤

e-mail : hangkon@cu.ac.kr

1985년 중앙대학교 전자계산학과 (공학사)

1987년 중앙대학교 대학원 전자계산학과 (공학석사)

1991년 중앙대학교 대학원 전자계산학과 (공학박사)

1978년~1979년 미 항공우주국 객원연구원

1987년~1989년 한국전기통신공사 전임연구원

1988년~1989년 AT&T 객원연구원

1990년~현 재 대구가톨릭대학교 컴퓨터공학과 교수

2000년~2002년 Central Michigan University 교환교수

2007년~2008년 미 SEITI 연구소 객원연구원

관심분야 : 모바일 임베디드 시스템 개발기법, 모바일 시스템 모델링, 컴포넌트 기반 개발 기법, 임베디드 소프트웨어 테스팅, 프로세스 개선 공학, 요구공학 및 도메인 공학