제 26 회 한국정보처리학회 추계학술발표대회 논문집 제 13 권 제 2 호 (2006. 11)

식스시그마-TSP 통합프레임워크에 관한 연구

박영규, 최호진, 백종문
한국정보통신대학교 공학부
e-mail : {youngkyupark, hjchoi, jbaik}@icu.ac.kr

A Study on Six Sigma-TSP Integrated Framework

Youngkyu Park, Hojin Choi, Jongmoon Baik
Information and Communications University, School of Engineering

요 약

소프트웨어 프로세스 개선에 대한 원리와 방법은 CMM/CMMI 와 같은 프로세스 모델의 등장으로 인식하였으나 개인과 팀 차원에서 이를 구체적으로 구현하기 위한 운영적 차원의 절차와 수단의 부족으로 실제현장에 적용하여 성과를 창출하기에는 어려움을 겪어왔다. 이러한 문제를 해결하고 개발자와 개발팀의 차원에서 CMM/CMMI 의 목표와 프랙티스를 구현하기 위해 SEI(Software Engineering Institute)에 의해 PSP/TSP 가 개발되었다. 그러나 TSP 가 팀 차원에서 소프트웨어 개발에 사용할 수 있는 구체적인 기법들을 기술하고 있더라도 TSP 에서 수집되는 메트릭에 대한 분석기법은 여전히 부족하다. 따라서 TSP 수행시 발생할 수 있는 문제를 방지하고 프로세스가 변경되고 유지 관리될 수 있도록 하기 위해서는 식스시그마의 다양한 통계 기법과 의사 결정도구의 사용이 필요하다. 본 논문에서는 TSP 의 각 스크립트에 식스시그마의 통계 기법과 의사 결정도구를 포함 시킴으로써 TSP 를 확장한 식스시그마-TSP 통합 프레임워크와 활용 가이드라인을 제시함으로써 팀 차원에서의 프로세스 개선의 수행을 지원하며 팀 차원에서 발생할 수 있는 이슈를 식스시그마의 분석, 정량화 도구를 사용하여 해결하고 아울러 팀 성과를 향상할 수 있는 방법을 모색해본다.

1. 서론

소프트웨어 개발 프로세스는 설비 중심의 제조공정과는 달리 눈에 보이지 않기 때문에 이러한 무형의 소프트웨어 개발 프로세스를 구조화하기 위해서 다양한 프레임워크가 제시 되었다. 대표적인 예는 SEI 가 미국방성의 지원을 받아 IBM 과 HP 등의 베스트 프랙티스를 참조하여 조직차원에서 소프트웨어 개발의 5 단계 프로세스 성숙도를 정립한 CMM/CMMI[1,2]가 있다. 하지만 이러한 프로세스 모델을 통하여 소프트웨어 프로세스 개선에 대한 원리와 방법은 인식하였으나 현실에 적용하여 성과를 창출하기에는 어려움이 있었다. 이는 개발모형과 방법론을 통해서 수행해야 할 일은 인식하였으나 개인과 팀 차원에서 이를 구체적으로 구현하기 위한 운영적 차원의 절차와 수단이 부족한 것이 주 원인이 되었다. 이러한 문제를 해결하고 개발자와 개발팀 차원에서 CMM/CMMI 의 목표와 프랙티스를 구현하기 위해서 PSP/TSP 가 개발되었다[3,4].

"본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었음" (IITA-2006-(C1090-0603-0032))

비록 TSP 가 개인과 팀 차원에서 소프트웨어 개발에 사용될 수 있는 구체적인 기법들을 기술하고 있다 하더라도 TSP 에서 수집되는 메트릭에 대한 분석기법은 여전히 부족하기 때문에 TSP 수행시 발생할 수 있는 문제를 방지하고 프로세스가 변경되고 유지 관리될 수 있도록 하기 위해서는 식스시그마의 다양한 통계 기법과 의사 결정도구의 사용이 필요하다.

TSP 가 식스시그마를 성공적으로 적용할 수 있는 정량적인 기반을 프로젝트 차원에서 제공함에 반해 식스시그마는 TSP 에서 식별된 문제의 원인 파악과 분석에 필요한 통계적 기법의 제공과 더불어 문제를 방지하기 위해 프로세스가 변경되고 유지 관리될 수 있도록 하는 프로세스 관리 방법론을 제공한다. 따라서 식스시그마와 TSP 는 상호 보완적인 관계에 있으며 양자가 동시에 추진될 때 전략적, 관리적 목표 달성에 시너지 효과를 발휘한다고 볼 수 있다. 본 논문에서는 TSP 프로세스에 식스시그마 도구를 통합한 식스시그마-TSP 통합프레임워크를 제시함으로써 팀 차원에서의 프로세스 개선의 수행을 지원하는 방법에 대해서 알아본다.

2 장에서는 TSP 와 식스시그마에 대한 기본개념을, 3 장에서는 식스시그마와-TSP 통합프레임워크에 대해서 기술한다. 4 장에서는 결론 및 향후 연구에 대해서 기술한다.

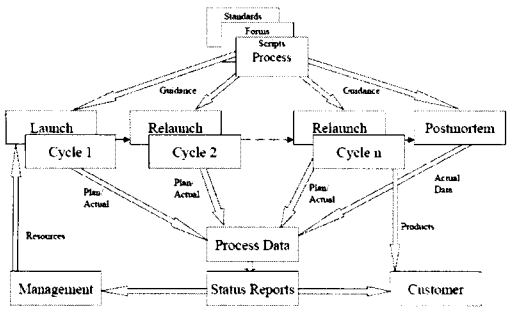
2. 기본 개념

2.1 Team Software Process

개발자 개인적인 차원의 PSP 를 기반으로 하여 CMM 의 프레임워크를 개발팀 차원에서 적용함으로써 소프트웨어 품질개선과 생산성 향상을 실현하기 위한 방법론이 TSP 이다. 개발자의 설계에서 테스트까지의 개발단계만을 다루는 PSP 와는 달리 TSP 는 프로젝트 차원에서 수행되는 모든 활동 계획과 측정작업을 정의하며 CMM 의 KPA 를 실행하기 위한 팀의 수립과 팀 활동을 전개하는 영역으로 확장된다.

TSP 는 일련의 단계(Phase)와 활동(Activity)으로 구성되어 있다. 계획-실행-사후관리로 구성된 활동은 요구사항 정의에서 테스트에 이르는 각 단계 속에서 반복되면서 실행되는데, 각 단계에서 개발팀은 프로젝트가 개발할 산출물의 크기와 개발시간, 결함을 추정하고 실행결과 데이터를 수집하여 계획 대비 진행상황을 모니터링 하면서 프로젝트를 진행한다.

TSP 는 4 일간의 TSP 착수회의(Launch)를 통해 프로젝트 목표를 확립하고 팀 역할을 정의하며, 위험평가와 전체적인 계획을 결정하는 것으로부터 시작한다. 이 결과로 팀의 목표, 팀원의 역할, 산출물 목록 및 규모산정, 전체 일정, 위험 평가 등을 얻을 수 있게 되며 이후에는 위의 결과에 따라 각자의 업무를 진행한다. 착수회의 과정은 한 번으로 끝나는 것이 아니라 각 단계 또는 주기별로 TSP 재착수(Re-Launch)를 수행하면서 전체 계획을 수정하거나 다음 단계의 상세한 계획을 수립하게 된다. 이러한 사이클은 프로젝트가 끝날 때까지 수행하게 되며 (그림 1)은 이러한 TSP 의 전체 흐름을 보여준다.



(그림 1) TSP 프로세스 흐름

2.2 식스시그마

식스시그마는 1987 년 모토로라(Motorola)의 마이클 해리(Mikel J. Harry)에 의해 제창된 통계적 기법을 활용한 품질 개선혁신활동이다. 식스시그마는 원래 통계적인 측정단위로 표준편차를 의미하는데, 프로세스를 정량적으로 평가하고 프로세스 변동을 줄이고자 품질 개선을 위한 경영전략으로 발전하여 구체화되었다[5]. 즉 식스시그마는 일반적으로 경영의 전 부문에서 식스시그마 품질목표를 달성하기 위해 수집된 프로세스의 주요 데이터들을 식스시그마 지원도구들을 사용하여 결함의 원인을 측정, 분석한 뒤 결함을 제거하는 체계적인 품질혁신활동으로 정의할 수 있다[6]. 식스시그마는 프로세스의 낭비를 제거하고 결함과 변동을 줄임으로써 품질 향상이라는 목표를 달성하고자 하며 각 분야의 선도 기업들은 설계에서부터 제조, 영업, 마케팅, 엔지니어링 등에 걸친 모든 프로세스에 식스시그마 방법론을 적용하고 있다.

3. 식스시그마-TSP 통합프레임워크

식스시그마 기법을 성공적으로 적용하기 위한 선결조건은 체계적으로 정의된 프로세스와 일관성 있는 표준측정지수이다. 또한 인프라적인 측면에서는 개선활동에 대한 동기부여와 개발자에 대한 교육과 데이터관리 체계가 마련되어야 한다. TSP 는 식스시그마가 성공적으로 적용될 수 있는 이러한 정량적인 기반을 프로젝트 차원에서 제공하며 이에 대해 식스시그마는 TSP 에서 파악된 문제의 원인을 분석하는 작업인 사후 데이터 분석(Postmortem Data Analysis)에 필요한 분석도구와 통계적 기법을 제공하고, 문제를 방지하기 위해 프로세스가 변경되고 유지 관리될 수 있는 프로세스 관리 방법론을 제공한다.

<표 1> TSP 스크립트 이름과 약어

이름	약어
Cycle 1 Team Launch	LAU1
Cycle n Team Launch	LAUn
Cycle 1 Development Strategy	STRAT1
Cycle n Development Strategy	STRATn
Cycle 1 Development Planning	PLAN1
Cycle n Development Planning	PLANn
Cycle 1 Requirements Development	REQ1
Cycle n Requirements Development	REQn
Cycle 1 Design	DES1
Cycle n Design	DESn
Cycle 1 Implementation	IMP1
Cycle n Implementation	IMPn
Cycle 1 Integration and System Test	TEST1
Cycle n Integration and System Test	TESTn
Cycle 1 Postmortem	PM1
Cycle n Postmortem	PMn

<표 1>은 TSP 에서 사용하는 스크립트와 각 스크립트의 약어를 보여주고 있다. 식스시그마-TSP 통합프레임워크는 TSP 의 각 스크립트에 사용 가능한 식스시그마 통계 기법과 의사 결정도구를 포함시킴으로써 TSP 수행시 발생할 수 있는 문제를 방지하고 프로세스가 변경되고 유지 관리될 수 있도록 한다. 이러한 식스시그마-TSP 통합 프레임워크는 기존의 TSP 에 식스시그마의 통계적 데이터분석 기법을 접목한 TSP 의 확장버전으로 볼수 있다. 본 프레임워크를 사용하여 팀 차원에서 발생할 수 있는 이슈를 식스시그마의 분

석, 정량화 도구를 사용하여 팀의 성과를 향상 시킴으로써 프로젝트 및 조직 차원에서 식스시그마 적용을 위한 기반을 마련할 수 있다.

다음 장에서는 식스시그마-TSP 통합프레임워크에서 스크립트의 예를 통해서 식스시그마 도구가 TSP 프로세스와 어떻게 사용 되는지 알아보고 다음으로는 실제 적용 프로세스를 자세히 살펴본다.

3.1 식스시그마-TSP 통합프레임워크 스크립트의 실제 예

<표 2>는 식스시그마-TSP 통합프레임워크에서 Cycle 1 Implementation 스크립트의 일부를 보여준다. Implementation 스크립트는 TSP 팀이 프로젝트 사이클 1 에서 소프트웨어를 구현하고 검사하도록 가이드하며 Cycle 1 Implementation 의 4 번째 단계에서는 개발중인 소프트웨어의 상세 설계와 설계 검토를 수행한다. 설계 검토 수행시 식스시그마의 잠재적 고장형태 및 영향분석(FMEA: Failure Modes and Effects Analysis) 기법을 활용하여 개발하려는 프로그램의 설계 결함을 식별하는 것뿐만 아니라 결함의 근본원인을 파악할 수 있다.

<표 2> 식스시그마-TSP 통합프레임워크 cycle 1 Implementation 스크립트

목적	TSP 팀이 프로젝트 사이클 1 에서 소프트웨어의 구현과 검사를 수행하도록 가이드	

투입물	• 팀이 개발 전략과 계획을 수립하였다.
	  • SRS, SDS 명세와 용어해설
	  • 문서화된 코딩표준과 다른 표준들	

개요	구현 프로세스를 통해서 산출물을 검토, 검사, 단위 테스트하며 산출물은 다음 사항을 만족시켜야 한다. • SDS, SRS 에서 명시하는 기능과 유스케이스를 완벽히 포함한다. • 수립된 코딩, 설계 표준에 부합한다. • PSP2.1 또는 PSP3프로세스를 따른다.	
단계	활동	설명
1	구현 프로세스 개요	강사는 구현 프로세스를 설명한다. • 품질 높은 구현의 중요성 • 코딩표준의 내용과 필요성 • 품질 낮은 컴포넌트의 처리 방안
2	구현 계획	개발 관리자는 팀이 구현 태스크(SUMP, SUMQ)를 정의하고 계획하도록 이끈다.
3	작업 할당	팀 리더는 팀원들 사이에 작업을 할당하는 것을 돕고 다음과 같은 활동을 수행한다. • 이러한 작업의 완료날짜에 대한 공약을 획득한다.
4	상세 설계	팀원들은 상세 설계를 수행한다. • 철저한 설계검토 방법을 사용하여 설계검토를 수행한다. • *[식스시그마 도구] 설계 검토시 개발하려는 프로그램의 설계 결함을 식별뿐만 아니라 결함의 근본원인을 알아내기 위해서 잠재적 고장형태 및 영향분석을 활용한다.* • LOGD 와 LOGT 양식을 완성한다.
5	단위 테스트 계획	팀원은 단위테스트 계획을 산출한다.
…		

잠재적 고장형태 및 영향분석은 프로세스가 복잡하고 창의력과 정신적 활동의 비중이 크며 타 산업에 비해 위험 큰 소프트웨어 개발에서 프로젝트 관리의 정확성을 높이기 위해 사용한다[7]. 잠재적 고장형태 및 영향분석은 기술, 개발비용, 일정, 시장 등의 주요 관리영역에서 위험 요소를 도출하고 위험의 발생 가능성과 발생 영향을 정량적으로 표현하여 우선순위가 높은 위험요소를 줄이기 위한 대응방안을 수립하고 실행한다. 위험요소별 위험지수는 주기적으로 평가하고 대응방안의 실행 효과를 추적하여 소프트웨어 개발 라이프사이클 전반에 걸친 총체적인 위험지수의 추이를 정량적으로 관리함으로써 효과적인 위험 관리를 실현한다.

3.2 식스시그마-TSP 통합프레임워크 가이드라인의 실제 예

다음은 <표 2>의 4 번째 단계에서 설계 검토시 수행하는 잠재적 고장형태 및 영향분석에 대한 설명이다.

[예제] 설계 검토시 프로그램의 설계 결함을 식별하는 것뿐만 아니라 결함의 근본원인을 알아내기 위해서 잠재적 고장형태 및 영향분석(Failure Modes and Effects Analysis) 기법을 활용한다. 동일한 결함이 앞으로 개발할 프로그램에서 반복되지 않도록 하기 위해 적절한 조치가 행해져야 하며 이러한 잠재적 고장형태 및 영향분석 문서는 새로운 결함 발생시마다 개정되어야 한다.

[단계 1] 프로세스 단계란에 평가하고 있는 프로세스의 단계를 입력한다.
[단계 2] 잠재적 고장 형태란에 발생할 수 있는 실패 종류를 나열한다. 하나의 잠재적 고장은 각각 하나의 열에 기술되어야 한다.
[단계 3] 고장의 잠재적 영향란에 실패가 고객의 요구사항에 미칠 수 있는 영향을 기술한다.

[단계 4] 심각도(SEV)에 실패가 고객의 요구사항에 미칠 수 있는 영향을 정량화하여 기입한다. 잠재적 고장들간의 명확한 구분을 위해서 크기 척도로 1(영향 없음), 4,7,10(아주 위험한 영향)을 사용한다.
[단계 5] 고장의 잠재적 원인란에 실패를 유발하는 원인을 나열한다. 하나의 실패에 여러 가지 원인이 있을 수 있다면 각각을 다른 열에 기입한다.
[단계 6] 발생도(OOC)란에 해당 원인이 발생 했을 때 실패로 이어질 수 있는 확률을 기입한다. 크기 척도로 1(실패 희박), 4,7,10(실패 매우 높음)을 사용한다.
[단계 7] 현재 관리 방법란에 고장 형태의 방지 및 발견에 사용하는 현재 절차를 기술한다.
[단계 8] 실패확률(FAIL PROB)란에 현재 관리방법이 실패할 확률을 크기척도 1(실패 희박), 4,7,10(실패 매우 높음)을 사용하여 기입한다.
[단계 9] 위험 우선순위(RPN)란에 심각도와 발생도, 실패확률을 곱한 값을 기입한다. 높은 위험 우선순위를 가진 고장 형태의 경우 위험을 줄이기 위한 적절한 계획을 수립하여야 한다.
[단계 10] 권고 조치 사항란에 실패 확률을 줄일 수 있는 조치를 기술한다.
[단계 11] 책임자란에 조치를 취하는데 책임이 있는 사람을 기술한다.
[단계 12] 목표 완료 예정일란에 조치가 완료될 날짜를 기입한다.
[단계 13] 완료 날짜란에 조치가 완료된 날짜나 설명을 기입한다.
[단계 14] 결과 심각도(SEV)란에 실패가 고객의 요구사항에 미칠 수 있는 영향을 정량화하여 기입한다. 잠재적 고장들간의 명확한 구분을 위해서 크기척도로 1,4,7,10 을 사용한다. 보통은 고장 형태에 대한 조치에 의해서 이 값이 변하지는 않는다.
[단계 15] 결과 발생도(OOC)란에 수정조치가 행해진 후에 해당원인이 발생하여 실패를 유발할 수 있는 확률을 크기척도 1,4,7,10 을 사용하여 기입한다.
[단계 16] 결과 실패확률(FAIL PROB)란에 수정조치가 행해진 후에 관리 방법이 실패할 확률을 크기척도 1,4,7,10 을 사용하여 기입한다.
[단계 17] 결과 RPN 란에 심각도와 발생도, 실패확률을 곱한 값을 기입한다. 수정조치가 효과가 있다면 이 단계에서의 값은 [단계 9]의 RPN 값보다 작아야 한다.

<표 3> 설계검토 이후의 잠재적 고장형태 및 영향분석의 예

프로세스이름: 통합 사원 급여 시스템								초기버전날짜: 8/20/01				개정번호: 0			
작성자: 홍길동								개정자:				개정날짜:			
	잠재적 고장			이유와 빈도		예방법			조치 계획				조치 결과			
프로세스 단계	잠재적 고장 형태	고장의 잠재적 영향	심각도	고장의 잠재적 원인	발생도	현재 관리 방법	실패 확률	위험 우선 순위	권고 조치 사항	책임자	목표 완료 예정일	완료 날짜	심각도	발생도	실패 확률	위험 우선 순위
설계 검토	입력화면들을 유일하게 식별하는 식별자가 부족	헬프 데스크에서 화면 식별이 쉽지 않기 때문에 지원이 어려워 짐	10	표준부족	10	없음	10	1000	설계 표준에 유일한 식별자를 추가한다.	김철수	8/24/04					0

4. 결론 및 향후 연구

식스시그마 통계도구와 의사 결정기법을 성공적으로 적용하기 위한 선제조건은 체계적으로 정의된 프로세스와 일관성 있는 표준측정지수이다. TSP 는 식스시그마가 성공적으로 적용될 수 있는 정량적인 기반을 프로젝트 차원에서 제공하며 이에 대해 식스시그마는 TSP 에서 파악된 문제의 원인을 분석하는 작업인 사후 데이터 분석(Postmortem Data Analysis)에 필요한 분석도구와 통계적 기법을 제공하고, 문제를 방지하기 위해 프로세스가 변경되고 유지 관리될 수 있는 프로세스 관리 방법론을 제공한다. 이러한 측면에서 본 연구에서는 기존의 TSP 에 식스시그마의 통계적 데이터분석 기법을 접목한 식스시그마-TSP 통합 프레임 워크를 제시하고 예를 들어 설명하였다. 팀 차원에서 발생할 수 있는 이슈를 식스시그마의 분석, 정량화 도구를 사용하여 팀의 성과를 향상 시킴으로써 프로젝트 및 조직 차원에서 6 시그마 적용을 위한 기반을 마련할 수 있다. 따라서 조직 관점에서 발생할 수 있는 이슈를 식스시그마의 분석, 정량화 도구를 사용하여 소프트웨어 개발 프로세스의 반복성과 예측성을 강화하는 방법에 대한 연구가 필요하다.

참고문헌

[1] Mark C. Paulk, B. Curtis, M. B. Chrissis and et al., "Capability Maturity model for Software", Software Engineering institute, CMU/SEI-91-TR-24, DTIC number ADA240603, August 1991.
[2] CMMI, "Capability Maturity Model Integration", Software Engineering Institute, Carnegie Mellon University, 2002.
[3] The Personal Software Process (PSP), Humphrey, Watts S. CMU/SEI-2000-TR-022, ESC-TR-2000-022, December 2000.
[4] The Team Software Process (TSP), Humphrey, Watts S., CMU/SEI-2000-TR-023, ESC-TR-2000-023, December 2000.
[5] Pete Pande, Larry Holpp, *What is Six Sigma?*, McGraw-Hill, 2002.
[6] 이혜영, 최호진, 백종문, 소프트웨어 프로세스의 6 시그마 적용 및 향후 전망, 정보과학회지 특집호 12 월, 2005.
[7] Christine B. Tayntor, *Six Sigma Software Development*, Auerbach Publications, 2002.