SQA활동 지원을 위한 방법론 및 그 활용방향

김 성 근* · 편 완 주**

A Software Quality Assurance Methodology and a Direction for Its Usage

Sung-Kun Kim* · Wan-Joo Pyun**

Abstract

As software projects become larger and more complex, we need to take a more systematic approach to quality assurance. One plausible approach is the use of SQA (software quality assurance) methodology. Since this SQA methodology was not available, our study presents a SQA methodology. This methodology has a repository in which a set of quality assurance tasks with their related techniques and deliverables is defined and from which one can select only appropriate tasks based upon characteristics of project. This study further suggests a rule-based approach for supporting task selection process.

* 중앙대학교 경영학과 교수
** 중앙대학교 대학원 경영학과

1. 서 론

오늘날 기업은 정보시스템에 크게 의존하고 있다. 기업의 모든 경영활동이 정보시스템의 활용을 통해 수행되고 있는 것이다. 하지만 정보시스템에 대한 의존성이 높아지면 높아질수록 시스템 오작동으로부터의 피해는 더욱 커지는 법이다. 그러므로 이러한 피해를 사전에 예방하기 위해 정보시스템의 품질을 높이기 위한 노력이 절실히 요구된다.

정보시스템의 품질을 향상하기 위해서는 소프트웨어 품질의 중요성을 정확히 인식하여야 한다. 소프트웨어는 유형의 제품과는 달리 업무상의 논리를 집합적으로 규정해 놓은 무형재산인 관계로 기능과 성능을 분리하기 어렵다는 특징이 있다. 또한 소프트웨어는 특정업무에 맞도록 개발되어지는 것이므로 공산품에 비해 품질관리가 더욱 어려워질 수 밖에 없다. 실제 현업에서 품질관리를 담당하고 있는 SQA(Software Quality Assurance) 조직과의 면접조사에서도 소프트웨어 품질에 대한 낮은 인식과 소프트웨어의 고유특성으로 인해 품질보증 활동의 어려움이 나타나고 있다.

소프트웨어 품질에 대한 중요성 인식도 매우 중요한 연구대상이지만, 본 연구에서는 품질 향상을 위한 구체적인 지원도구 측면에서 연구를 하고자 한다. 즉, 소프트웨어 품질을 제고하기 위해서는 이를 체계적이고 계획적으로 지원하기 위한 품질보증 방법론 개발이 시급한 것이기 때문이다.

품질 활동을 지원하기 위한 방법론에 대한 연구는 다양한 노력들이 진행되어 왔다. 특히, 외국에서는 품질보증 방법론을 크게 검사(Inspection) 중심의 방법론과 SQA 형태의 방법론으로 구분하고 있다. 검사중심의 방법론은 품질향상을 위한 노력의 일환으로 이해할 수 있으나 일반적인 품질보증이라고 보기보다는 품질통제의 차원에서 이해될 수 있다. 반면 미국방성의 SEPO (Software Engineering Process Office, 1997) 방법론은 완전한 방법론의 형태를 취하고는 있지만 품질보증 활동의 여러 측면(프로덕트, 프로세스, 지원 등)중에서 프로세스 평가측면만을 강조하고 있는 방법론이라 할 수 있다. 이러한 노력 외에도 SQA 계획서 작성을 지원하는 SQA 계획서 템플레이트(template) 방법론과 프로젝트 관리방법론에 품질보증 활동을 포함하는 형태의 방법론도 있어왔다(김성근, 1999).

국내에서는 과학기술처의 생산성 향상 및 품질관리 기술 개발에 관한 연구과제에서 품질관리 방법론이 개발되었다(양해술, 1996). 그의 연구는 소프트웨어 품질관리 방법론의 체계와 품질관리의 주요 대상을 품질특성의 평가측면에서 접근한 방법론으로 품질 측정을 위한 다양한 평가 매트릭스가 제시되고 있지만 방법론의 실행단계에서 수행할 작업이 구체적이지 못한 것으로 파악되었다.

기존 방법론에 대한 연구는 일부 품질보증방법론의 형태를 지니고 있지만 실행단계의 품질보증 활동에 있어서 구체적이지 못하며, 품질보증 활동의 대상 측면에서도 단편적이라는 인상이 짙다. 따라서 본 연구는 이러한 한계점을 해소하기 위해 구체적이면서도 모든 활동 대상을 고려한 방법론을 개발하고자 하였다. 개발될 방법론은 품질활동의 시작 단계부터 운영유지 단계를 포괄하고 있으며, 품질보증 활동을 계획적이고 체계적으로 지원할 것이다. 또한, 일반적으로 모든 프로젝트에 적합한 단일 방법론은 없다(Martin and Odell, 1996)는 전제하에 방법론을 보다 쉽고 효과적으로 사용할 수 있는 커스터마이징(Customizing)기법을 제안하고자 한다. 이는 방법론을 프로젝트에 적용함에 있어 해당 프로젝트의 특성과 조직 특성을 고려한 것으로 방

법론의 활용성을 제고하기 위한 노력인 것이다.

본 연구는 기존 문헌연구와 SI업계의 품질보증 요원 등의 면접조사를 토대로 품질보증방법론을 개발하여 제시하고, 이의 효과적 활용을 위한 커스터마이징 기법의 하나로 규칙기반 시스템 방안을 제시하고자 한다.

본 논문은 다음과 같이 구성되어있다. 2절에서는 소프트웨어 품질관련 기존 연구를 분석하고, 3절에서는 분석된 결과를 바탕으로 개발된 SQA 방법론을 제시한다. 4절에서는 프로젝트 특성에 따라 필요한 태스크를 선정하게 해주는 규칙기반 시스템을 설명한다. 5절에서는 요약과 향후 연구방향을 제시한다.

2. 소프트웨어 품질에 관한 기존 연구 분석

본 절에서는 이론적 고찰을 통해 품질보증 활동의 제약요인을 도출한다. 제약요인을 바탕으로 품질보증 활동을 제고할 수 있는 방안을 도출한다.

2.1 품질관리 관련 국·내외 연구 동향

오늘날 제품과 서비스에 대한 품질이 매우 강조되고 있으나 소프트웨어 품질에 대한 지난 10년간 축적된 품질 수준은 기대이하라 할 수 있다. 품질관리의 태동은 일본에서 시작되었고 현재는 글로벌 시장에서 경쟁우위를 획득하기 위한 수단으로써 전세계적으로 널리 확산되고 있는 개념이다. 그러나, 이에 대한 국내·외적으로 소프트웨어 품질관리를 위한 체계적 실증연구는 매우 부족한 편이다(Rai, Song and Troutt, 1994).

그러므로 향후 품질관련 연구 방향을 세우기 위해 기존의 문헌을 분류할 필요가 있다.

Henderson과 Cooprider(1992)는 품질관리의 연구영역을 기술적 측면(technical aspect)에 대한 생산(production)차원, 품질관리 활동과 인적 자원간의 관계와 의존을 초점으로 하는 조정차원, 그리고 연관된 프로세스와 기술 환경을 다루는 조직적 기술(organizational technology)차원으로 구분하고 있다.

이러한 분류차원을 바탕으로 Rai등(1998)은 연구의 중요도가 점진적으로 높아 가는 정보기술(Information Technology), 품질 주도권(quality initiative) , SQA의 경제성(economics)을 포함시킨 4번째 차원을 추가하여 기술적, 관리적, 조직적, 경제적 차원의 네 가지 유형의 차원으로 SQA(Software Quality Assurance) 연구영역에 대한 분류스키마를 제시하고 있다.

이러한 분류스키마를 바탕으로 국·내외의 연구실적을 분류함으로써 현재의 품질관리 관련 이론연구의 현주소를 진단할 수 있으며 나아가 소프트웨어 품질관리의 이론연구 방향을 가늠할 수 있다.

<표 1>은 국·내외 품질관리 연구를 분류하고 있다. 표와 같이 연구문헌의 대부분은 주로 소프트웨어 품질특성, 소프트웨어 매트릭스, 기법과 도구를 주제로 하는 기술적 차원에 집중되고 있었다. 또한 실증연구보다는 기술적(descriptive)이고 규범적(prescriptive)인 연구가 많은 부분을 차지하고 있다. 그나마 수행된 실증연구 조차도 탐험적(exploratory) 연구가 대부분이었다. 실증연구가 제한적이라는 의미는 SQA 연구영역이 여전히 진화적(evolutionary) 단계에 있다는 것을 의미한다.

따라서 향후 소프트웨어 품질에 대한 연구 방향으로 정보시스템 개발조직의 품질관리 실태에 관한 실증적 연구가 필요할 것이다. 일과성에 그치지 않고 주기적인 실증적 연구를 함으로써 국내 개발조직의 소프트웨어 품질관리 수준을 진단하고, 이를 바탕으로 보다 낳은 품질시

〈표 1〉 국·내외 품질관리 연구 분류

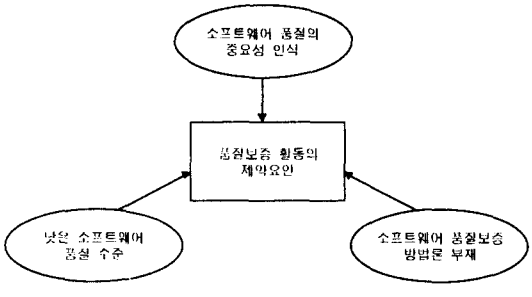
분류 차원	내 용	주요연구	비 고
기술적 차원	- 소프트웨어 품질특성(6개 특성) - 소프트웨어 매트릭스 - 기법과 도구	- 국외문헌 : Boehm et al.(1978), Feinawer(1991), Halasz(1988), Wilden et al.(1991), Napier et al.(1989), Carrol and McKendrea(1987), Rush et al.(1990), Apte et al.(1990), Pfleeger and Bollinger(1994),Ince(1990), Grumen(1991), Siegel(1992), Ahituv and Zelek(1987), Miller(1989), Van Treeck and Thackeray(1991), Kim and Stohr(1998)외 - 국내문헌 : 한국통신(1992), 양해술 · 이창석(1989), 양해술 · 임춘봉 · 정호원(1992), 이용근 · 양해술(1993), 양해술(1994), 권기태외 1(1990), 오상헌외 2인(1992), 이경환(1990)	- 연구의 대부분이 기술적 주제를 다루고 있음 - 기법과 도구, 품질 특성, 매트릭스 순으로 연구가 많음 - 지금도 연구가 가장 활발히 진행되고 있음
관리적 차원	- 자원관리(프로젝트 관리, 선발과 교육, 자원할당) - 시스템 개발프로세스 관리(개발 프로세스, 프로세스 운영유지, 사용자 참여) - 기타 관리적 이슈	- 국외문헌 : Boehm and Ross(1989), Swanson et al.(1991), Carpenter and Hallman(1985), Kann(1992), Rahman(1987), Apte et al.(1990), Cerveny et al.(1986), Swanson et al.(1991),Dagwell and Weber(1983) , Gould and Lewis(1985), Engler(1997) - 국내문헌 : 이주헌(1983), 양해술 · 정재학(1994)	- 자원관리, 개발프로세스, 기타 관리적 이슈 순으로 연구가 많음 - 최근 들어 개발 프로세스에 대한 연구는 줄어들고 기타 관리적 이슈에 대한 연구가 증가함
조직적 차원	- SQA기능의 조직구조 - 개발팀(구조, 개발자간 의사소통, 개발자 자격) - 기타 조직적 이슈	- 국외문헌 : Brelsford(1988), Bucklty and Poston(1984), Grumen(1991), Nenz(1985), Apte et al.(1990), Karimi(1990), Bendifallah and Scacchi(1987), Newman and Robey(1992), Cartwright, Andrews and Webley(1999) - 국내문헌 : 민병욱 · 김형배(1992), 양해술 · 임춘봉 · 정호훤(1992)	- 개발팀, 기타 조직적 이슈, 조직구조 순으로 연구가 많음 - 최근들어 조직구조의 연구가 증가함
경제적 차원	- SQA를 위한 경제적 모델	- 국외문헌 : Abdel-Hamid(1988), Barnes and Bollinger(1991), Levendel(1990), Murine(1988), Hollocken(1986), Paughtrey(1988), Ramin, Ali and Seiichi(1999) - 국내문헌 : 한국소프트웨어산업협회(1996), 이용근 · 양해술(1994)	- SQA의 비용대 효익에 관한 연구 - 대부분은 SQA의 비용 측면이며 일부는 효익 측면

스템을 구축할 수 있도록 지표로 삼아야 한다.

또한, Chou, Yen and Chen(1998)은 소프트웨어 품질보증 수준을 소프트웨어 개발업체의 주요경쟁우위 요소로 지적하면서 체계적이고 계획적인 SQA 활동의 중요성을 강조하고 있다. SQA 활동에 대해 기업이 안고 있는 문제에 대한 연구에서는 품질보증에 대한 능력을 갖춘 전문 인력의 부족(Boehm, 1981 ; Arthur, 1985), 개발조직과 소프트웨어 유지·보수 조직과의 분리문제, 정보시스템 업계의 낮은 품질보증 수준(Ramin, Ali and Seiichi, 1999), 인간적 기업문화(Cartwright, Andrews and Webley, 1999),

적합한 SQA 방법론의 부재(Ince, 1994 ; Ginac, 1998, 황준규, 1997 ; 김성근외, 1999) 등의 요인이 제시된 바 있다.

앞에서 지적된 문제는 효과적인 SQA 활동을 위해 반드시 해결되어야 할 문제들이다. 제 학자들이 제시한 문제와 더불어 국내 SI(System Integration)업계의 실무적 문제를 파악하기 위해 면접조사를 실시하였다. SQA 요원과의 면접조사에서도 소프트웨어 품질에 대한 인식 문제, SQA 방법론의 부재, 전문가 부족 등이 파악되었다. 이처럼 문헌연구와 면접조사 결과는 거의 일치된다 하겠다. 다음은 문헌 연구와 면접조사를 통해 파악된 품질보증 활동의 제약요인을 묘사한 것이다.



(그림 1) 품질보증 활동의 제약요인

소프트웨어 품질의 인식에 대한 문제는 품질 중요성에 대한 꾸준한 교육을 통해 해결되어야 할 요인다. 이에 대한 조사에서 최고경영자의 인식이 무엇보다도 중요한 요인으로 나타나고 있다. 소프트웨어 품질 수준은 단 기간에 해결되는 요인은 아니다. 자사의 수준을 정확히 파악하고 이를 바탕으로 점진적인 수준 향상계획을 수립하고 실천하여야만 해결될 수 있는 요인이라 하겠다.

본 연구는 품질보증 활동의 지원도구 측면에서 접근하고 있다. 즉, 품질보증 활동을 향상하기 위해 해결해야 할 여러 요인 중에서 SQA 방법론의 개발이 궁극적으로 소프트웨어 품질 수준과 품질에 대한 중요성 인식문제에도 많은 영향을 미칠 것으로 판단하였다. 그러므로 본 연구는 소프트웨어 프로젝트가 대규모화되고 복잡해지는 추세를 반영하여 SQA 조직의 품질 보증 활동을 체계적으로 지원하기 위한 방법론을 개발하고자 한다.

2.2 방법공학에 관한 기존 연구

방법론에 관련되어 파악된 또 다른 요구사항은 프로젝트 특성에 적합한 방법론의 커스터마이징 기법의 요구이다. Martin and Odell(1996)은 그의 저서에서 "일반적으로 모든 프로젝트에 적합한 단일 방법론은 없다,"고 지적하고 있다. 미국의 연구사례에서도 기업의 자체 방법론을 커스터마이징하지 않고 그대로 적용한 기업은 10%에도 미치지 못한 것으로 나타났다(Russo, 1995). 즉, 방법론은 해당 프로젝트 특성에 따라 적절히 재정의되어야 한다는 것이다. 기존 연구와 더불어 국내 SI업계의 관련 요원과의 인터뷰에서도 SQA 방법론의 개발요구와 함께 방법론을 효과적으로 사용할 수 있는 커스터마이징 기법의 필요성도 제기되었다(김성근, 1999).

아직, 방법론의 커스터마이징 연구는 초기단계라 할 수 있다. 국내·외의 문헌 연구를 통해 파악된 기존 연구는 손으로 꼽을 수 있는 정도이다. 국내 연구로 정기원(1998)은 프로젝트 및 조직의 유형에 따른 개발경로를 제시하고 있다. 즉, 프로젝트의 조직유형과 개발형태에 따라 개발경로가 달라야 한다는 것으로 정보시스템 개발경로는 소규모 시스템, 재개발 시스템, 클라이언트/서버 시스템, 호스트 시스템, 인트라넷 기반 시스템, RAD(Rapid Application Development) 도구를 이용한 시스템, 실시간 시스템, 데이터베이스처리/OLTP 시스템 등으로 제시하

고 있다. 이러한 유형의 실제 구현사례로 삼성데이타시스템(SDS)의 INNOVATOR를 들 수 있다. INNOVATOR는 현장 경험을 기반으로 구축된 SDS의 품질시스템으로써 정보화계획수립에서부터 다양한 개발유형(클라이언트/서버, 호스트, 인트라넷, 패키지 개발 및 적용, 객체지향 개발, 공정제어 개발, 데이터워어하우징, 소프트웨어 유지보수)에 따라 적합한 방법론을 작성하도록 지원하는 시스템이다.

국외 사례로 Ginsberg and Quinn(1995)은 소프트웨어 CMM(SW-Capability Maturity Model)에서 제반 환경을 고려한 핵심 활동을 도출하는 방법을 제안하고 있다. 즉, CMM의 모든 핵심 프로세스를 그대로 적용하기 보다는 환경에 따라 적절히 조정(tailoring)되어야 한다는 것이다. Budlong, Szulewski and Ganska(1996)는 조직의 표준 프로세스에서 프로젝트 특성에 적합한 표준 프로세스만을 선정하고 선정된 표준 프로세스를 바탕으로 프로젝트에 적합한 SDP(Software Development Plan)를 조정하는 방법을 제시하고 있다. 즉, SDP에 영향을 미치는 조직의 모든 표준 프로세스들은 프로젝트 특성에 맞게 선정되고 수정되어야 한다는 것이다. 그러나 이러한 연구들은 조직의 표준 프로세스를 개발하고 개발된 표준 프로세스에 대한 조정이 필요하다는 것이지 실제 선정된 표준 프로세스를 어떻게 커스터마이징하는 가에 대해서는 구체적이지 못하다.

ISO 12207은 소프트웨어 산업계에서 참고할 수 있는 잘 정의된 용어를 사용하여 소프트웨어 생명주기 프로세스의 공통 프레임워크를 제공하고 있다. 이것은 소프트웨어를 포함하는 시스템과 소프트웨어 제품에 대하여 공급기간 중의 소프트웨어 서비스, 개발, 운영과 유지보수 기간동안 적용할 수 있는 프로세스, 활동 및 작업으로 구성된 표준이다(정기원 · 윤창섭 · 김태현, 1997).

ISO 12207에서도 조정 프로세스를 제공하고 있다. 이는 모든 프로젝트에 ISO 12207의 프로세스, 활동, 작업을 그대로 적용할 수 없다는 것이다. 따라서 관련 작업을 조정시 조직의 정책과 절차, 획득방법과 전략, 프로젝트 규모 및 복잡도, 시스템 요구사항 및 개발 방법 등의 요인은 물론 환경적인 요인을 충분히 고려해야 한다고 제안하고 있다. 그러나 이러한 조정지침을 그대로 SQA 방법론에 적용하는 것은 생각해야 할 문제이다.

왜냐하면 ISO 12207은 기본 생명주기 프로세스, 지원 생명주기 프로세스, 조직 생명주기 프로세스로 구성된 소프트웨어의 생명주기 동안 수행될 수 있는 모든 활동을 포함하고 있다. 이에 비해 SQA 활동은 지원프로세스를 구성하는 하나의 프로세스 중 하나의 활동에 불과함으로 ISO 12207의 조정요인을 그대로 사용하기에는 어렵다는 것이다. 또한, SQA 활동은 개발 모형이나 프로젝트의 비용, 기간, 적용 방법론 및 개발 유형 등에 따라 매우 큰 영향을 받는 활동이다. 따라서 SQA 방법론을 커스터마이징하기 위해 모든 영향요인을 고려하는 것은 다소 어려움이 있다. 즉, SQA 방법론의 커스터마이징 지침은 좀더 세부적이면서도 모든 영향요인을 고려하기 보다는 보다 영향정도가 높은 요인이 추출되어야 한다는 것이다.

따라서 본 연구는 방법론을 실제 현장에서 보다 쉽고 효율적으로 활용할 수 있도록 하는 방법을 제시하고자 한다. 첫 번째로 조직의 표준 프로세스라 할 수 있는 SQA 방법론을 프로젝트 특성에 따라 커스터마이징할 수 있는 연구모형을 제시하고, 다음으로 커스터마이징 된 품질보증 활동 작업을 작업간의 연관관계를 이용하여 실제 수행할 작업을 확정할 수 있는 방안을 제시하고자 한다.

2.3 전문가시스템 접근방법

일반적으로 방법론을 커스터마이징하는 방법에는 여러 가지의 시도가 가능할 것이다. 하지만 기존 연구방법을 고려해보면 다음과 같이 두 가지 접근방법이 주류를 이루고 있음을 알 수 있다.

먼저, 시스템 특성에 따라 개별 템플레이트를 완성하는 접근방법이 그것이다. 이러한 방법의 예로 위에서 언급하였던 정기원의 연구를 들수 있다. 그이 연구는 시스템 특성별로 개별 템플레이트를 완성해 놓은 형태이다. 개별 템플레이트는 개발 시스템의 유형에 맞게 개발경로별로 프로세스를 카스터마이징하는 형태를 지닌다. 이는 정의된 8개의 개발경로에 따라 방법론을 선택하는 것으로 사전에 정의하기 어려운 실시간 시스템 개발이나 지식-기반 프로젝트에는 여전히 기존의 고착형(rigid)방법론을 따라야하는 것 문제가 있다(Harmsen, 1994).

다음으로 전문가 시스템의 규칙-기반 접근방법을 들 수 있다. 규칙-기반 전문가시스템 접근방법은 프로젝트나 SQA 조직에 따라 실제 수행하는 품질보증 활동이 영향을 받는다는 것에 주안점을 두고 있다. 즉, 정해진 개발경로를 따를 수 없는 프로젝트 혹은 실시간으로 품질보증 활동을 수정할 수 있는 환경에 적합한 접근방법인 것이다. 규칙-기반 전문가시스템은 프로젝트의 형태나 SQA활동 특성을 정의하여 이에 따라 생성된 규칙-기반 기법을 적용, 필요한 결과물을 활용하는 형태이다. 이러한 접근방법은 앞에서 언급한 Budlong et al.의 연구와 Ginsberg의 연구가 이에 속한다.

3. 품질보증방법론의 성격

본 절에서는 SQA 방법론의 주요 구성요인을 프레임워크, 프로세스, 리포지토리의 관점에서 정의하고자 한다.

3.1 SQA 방법론 개발 및 활용

품질보증방법론 개발을 위해 기존 문헌연구와 SI업체의 관련 요원과의 면접조사 결과 품질보증 활동을 지원하기 위한 품질보증방법론 개발은 크게 두 가지 유형으로 구분해 볼 수 있었다. 한 유형은 기존의 정보시스템 개발방법론이나 프로젝트 관리방법론, 정보시스템 계획수립 방법론과 같은 고정형(rigid) 방법론으로 일반적인 SQA 활동을 절차적으로 묘사하는 형태이다. 이러한 유형의 접근방법은 SQA 활동이 프로젝트의 초기 단계에서 생산되는 SDP에 따라 종속되는 특징이 있어 모든 프로세스의 활동과 작업을 순차적으로 묘사하기 어렵다는 한계가 있다. 즉, SDP의 개발 일정이나 개발 유형에 따라 품질보증 활동이 수립 및 실행되고, SDP의 작업 산출물(work product)에 따라 품질보증 작업이 정해진다는 것이다. 그러므로 이러한 유형의 방법론을 개발할 경우 모든 프로젝트에 적용가능한 방법론을 개발하기는 어렵다.

또 다른 유형의 방법론은 방법론을 프로세스와 리파지토리(repository)로 분리하여 개발하는 형태로 방법론의 실행단계에서 수행하는 작업을 커스터마이징(customizing)할 수 있도록 하는 유형이다. 이는 미국방성의 SEPO(Software Engineering Process Office)와 유사한 방법론이라할 수 있다. 이를 수정형(Customizable) 방법론이라 부르기로 한다. 수정형 방법론은 실행단계에서 수행되는 모든 품질보증 작업을 리파지토리내에 묘사하고 프로젝트의 특성이나 조직 특성에 따라 필요한 작업만을 커스터마이징하고, SQA 방법론 프로세스에는 필수적인 활동을 프로젝트의 개발 단계에 따라 수행

하도록 순차적으로 묘사해주는 형태를 취한다. 이러한 유형의 접근 방법은 SDP에 따라 적합한 품질보증 활동을 커스터마이징하는 것은 가능하지만, 품질보증 활동에 적합한 작업을 선정할 때는 적지 않은 품질보증 활동의 수행경험이 요구되는 특징이 있다.

3.1.1 SQA 방법론의 특징 및 구성

본 연구의 SQA 방법론은 수정형 방법론 유형을 기반으로 하고 있다. 이를 채택한 배경은 다음과 같다.

첫째, 개발 프로젝트에 따라 품질보증 활동이 상이하므로 품질보증 활동을 순차적으로 묘사하는 접근방법에는 다소 문제가 있으며,

둘째, 국내 품질보증 담당자와의 인터뷰 결과 현재 사용되고 있는 대부분의 품질보증 방법이 너무 엄격한 절차와 계량적인 관리 기법을 요구하고 있어 개발조직의 성숙도가 상대적으로 낮은 국내실정에서는 이러한 방법을 그대로 적용하기가 어렵고,

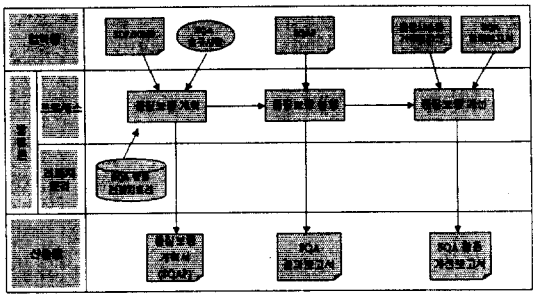
셋째, 현실을 반영하여 프로젝트의 특성에 따라 재구성이 가능하고, 사용하기 용이한 품질보증 방법론의 필요하다는 것이다.

또한, 본 연구의 품질보증방법론은 다음과 같은 특징을 염두에 두고 있다. 먼저, 종합적 시각의 품질보증방법론으로 개발자나 발주자의 일방적 시각에서의 품질보증보다는 개발자와 발주자의 공통적인 시각이 반영될 수 있도록 흔히 프로젝트 조직과 별도로 구성되는 품질보증조직을 위한 방법론이라고 볼 수 있다. 또한, SI 업체에의 적용을 염두에 둔 방법론으로 사내의 정보시스템 개발에 관련된 품질보증 노력보다는 별도의 발주조직으로부터 수주받은 개발프로젝트에의 적용을 염두에 둔 방법론이다. 즉 발주자와 개발자의 객관적인 시각을 반영하고 품질에 관련된 문제의 발생시 이를 근원적으로 해결하기 위해 수행될 노력을 체계적으로 묘사하고있다. 끝으로, 정보시스템 감리에 대한 연계를 고려한 방법론이라 할 수 있다. 감리란 발주조직의 요청에 의해 개발조직과 발주조직이 아닌 제삼자의 입장에서 정해진 품질요구대로 시스템 개발이 이루어지고 있는가를 감사하는 기능이다. 이러한 감리의 노력도 소프트웨어의 품질을 높이기 위한 노력의 일환이므로 감리활동 및 품질보증활동의 노력과 결과물이 상호 참조될 수 있다면 매우 효과적일 것이다. 이에 본 품질보증방법론은 정보시스템 감리와의 상호 참조를 염두에 두고 있다.

품질보증방법론은 프로세스와 리파지토리로 구분된다. 프로세스는 실제 방법론의 단계 및 활동을 묘사하는 접근방법을 띠고 있으며, 프로세스는 계획단계, 실행단계 그리고 개선단계로 진행된다. 리파지토리는 품질보증 활동에 필요한 실행 작업을 담고 있는 저장소라 할 수 있다. 리파지토리에 저장된 작업은 프로젝트의 특성에 따라 재구성이 가능하도록 품질보증 활동에 필요한 대부분의 작업을 포함한다. 그러므로 품질보증방법론은 실제 방법론의 단계 및 활동에 대해서는 품질보증방법론의 프로세스를 참조하고, 리파지토리에서는 품질보증 실행활동의 작업만을 담고 있는 이중적 구조를 띄고 있다.

(그림 2)는 품질보증 방법론의 프로세스와 리파지토리 그리고 주요 입력물과 산출물을 묘사한 프레임워크이다.

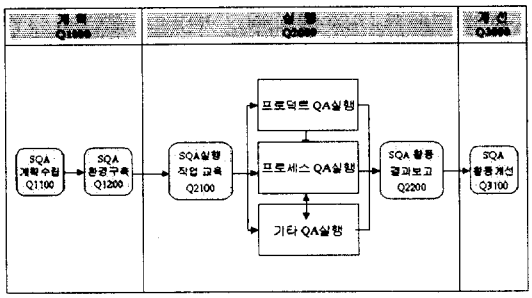
SQA 방법론 프레임워크는 방법론을 적용하는 과정을 개괄적으로 묘사해주고 있다. 즉, SQA 방법론의 주요 구성요소와, 프로세스의 각 단계에 요구되는 입력물, 해당 단계의 수행 후 생산되는 주요 산출물을 개발 프로세스에 따라 묘사한 것이다.



(그림 2) SQA 방법론 프레임워크

3.1.2 SQA 방법론의 프로세스

다음 (그림 3)은 SQA 방법론 프로세스의 단계 및 활동을 묘사하고 있다. SQA 방법론의 프로세스는 크게 계획, 실행, 그리고 개선 단계로 구성되어있다. 각 단계는 몇 개의 활동으로 구분되어 수행되며, 각 활동 또한 이를 수행하기 위해 필요한 작업으로 구성된다.



(그림 3) SQA 방법론 프로세스 단계별 주요활동 묘사

본 연구에서는 지면관계로 활동을 구성하는 작업에 대한 세부묘사를 생략하고 있다. 이는 본 연구의 커스터마이징 기법이 방법론 리파지토리의 작업을 대상으로 하고 있기 때문이다.

각 활동에 대한 작업들은 대략 다음과 같은 기술체계를 따른다. 즉, 전체적인 활동별 단계의 구성은 단계별 주요활동과 유사한 형태를 지니며, 각 작업의 기술체계는 작업의 목적, 절차 설명, 입력물, 산출물, 필요 양식의 체계로 기술되어져 있다.

3.1.3 SQA 방법론의 리파지토리 작업

SQA 방법론에는 품질보증 활동을 지원해주고, 성공적인 품질보증을 반복적으로 수행할 수 있도록 하기 위해서 품질보증을 위한 리파지토리를 제공한다. 여기서 리파지토리란 품질보증 활동의 주요 실행 작업에 대한 표준내용을 담고 있는 저장소라 할 수 있다. 이와 같은 리파지토리의 활용은 주요 실행작업에 대한 품질보증 계획서 작성을 용이하게 해주고, 또한 프로젝트 특성과 조직 특성에 따라 방법론 커스터마이징을 가능하게 한다.

본 SQA 방법론의 리파지토리는 크게 세 가지 유형으로 구분하였다. 다음 <표 1>은 리파지토리를 구성하고 있는 요소를 보여준다.

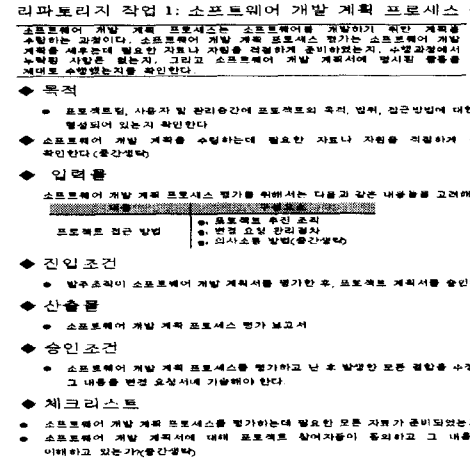
〈표 1〉 SQA 방법론 리파지토리의 구성요소

구성요소	설 명
프로덕트 QA 작업	소프트웨어 개발계획서 검토 작업외 38개 작업
프로세스 QA 작업	소프트웨어 개발 계획 프로세스 평가 작업외 38개 작업
지원 QA	매체인증 작업외 5개 작업

리파지토리의 작업은 품질보증 활동의 선택/필수 작업을 망라하여 결정되었다. 리파지토리 작업을 선정하기 위해 미국방성 SEPO 방법론의 품질보증 태스크, ISO 12207의 기본 생명주기의 모든 개발 프로세스의 태스크, 지원 및 조직 생명주기 프로세스의 일부 태스크, MIL-STD-498의 대부분 개발활동과 태스크 및 모든 작업 산출물을 검토하였다.

리파지토리 작업의 세부묘사는 대략 방법론 프로세스의 작업 묘사와 유사하다. 단, 리파지토리 작업은 품질보증 실행단계에서의 구체적인 진입조건, 참여자 및 역할, 승인조건, 체크리스트 등을 추가로 포함하고 있다. 다음 (그림 4)는 하나의 리파지토리 작업을 기술체계에 따라

묘사한 예이다.


리파토리지 작업 1: 소프트웨어 개발 계획 프로세스 평가

소프트웨어 개발 계획 프로세스는 소프트웨어를 개발하기 위한 계획을 수립하는 과정이다. 소프트웨어 개발 계획 프로세스 평가는 소프트웨어 개발 계획을 세우는데 필요한 자료나 자원을 적절하게 준비하였는지, 수행과정에서 누락된 사항은 없는지, 그리고 소프트웨어 개발 계획서에 명시된 활동을 제대로 수행했는지를 확인한다.

◆ 목적
- 프로젝트팀, 사용자 및 관리층간에 프로젝트의 목적, 범위, 접근방법에 대한 공감대가 형성되어 있는지 확인한다.

◆ 소프트웨어 개발 계획을 수립하는데 필요한 자료나 자원을 적절하게 준비했는지 확인한다.(중간생략)

◆ 입력물

소프트웨어 개발 계획 프로세스 평가를 위해서는 다음과 같은 내용들을 고려해야 한다.

제품	구성요소
프로젝트 접근 방법	●: 프로젝트 추진 조직 ●: 변경 요청 관리 절차 ●: 의사소통 방법(중간생략)

◆ 진입조건
- 발주조직이 소프트웨어 개발 계획서를 평가한 후, 프로젝트 계획서를 승인해야 한다.

◆ 산출물
- 소프트웨어 개발 계획 프로세스 평가 보고서

◆ 승인조건
- 소프트웨어 개발 계획 프로세스를 평가하고 난 후 발생한 모든 결함을 수정해야 하며, 그 내용을 변경 요청서에 기술해야 한다.

◆ 체크리스트
- 소프트웨어 개발 계획 프로세스를 평가하는데 필요한 모든 자료가 준비되었는가?
- 소프트웨어 개발 계획서에 대해 프로젝트 참여자들이 동의하고 그 내용을 제대로 이해하고 있는가?(중간생략)

(그림 4) 리파지토리 작업의 묘사 사례

4. 방법론 커스터마이징을 위한 규칙기반 시스템

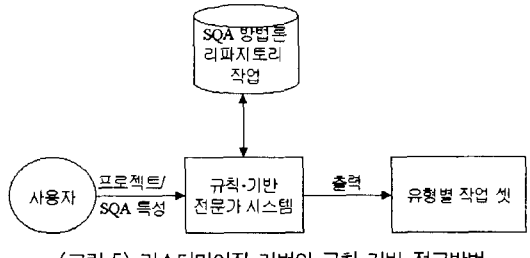
본 절에서는 방법론 커스터마이징을 효과적으로 수행하게 해주는 방법의 하나로서 규칙기반 접근방법을 제시한다.

4.1 리파지토리 작업의 커스터마이징 기법

SQA 방법론 리파지토리의 작업들은 프로젝트 특성과 SQA 조직 특성에 따라 적절히 커스터마이징되어 사용하여야 효과적이다. 프로젝트에 관계없이 모든 리파지토리 작업을 수행한다는 것은 품질보증 활동에 적지 않은 인적자원과 비용/시간이 소모될 것이다.

일반적으로 방법론을 커스터마이징하는 접근방법에는 여러 가지의 시도가 가능하다. 먼저, 대부분의 시스템 처럼 일정한 절차에 따라 모든 상황을 고려하여 프로그래밍하는 방법으로 정형화된 시스템을 개발하는 접근방법이다. 이는 순차적인 문제해결 접근방식으로 전문가의 지식을 모두 표현하기에는 부적절하다. 다음으로 상황에 따라 모든 절차를 거치지 않고 특정한 조건에 맞춰 작업을 진행하는 방법이다. 즉, 비정형화된 전문가의 지식이나 경험에 따라 일정한 규칙과 사실을 통해 결과를 추론하는 접근방법인 것이다. 즉, 프로젝트의 특성과 SQA 조직 특성을 정의하여 이에 따라 생성된 규칙-기반 기법을 적용, 필요한 결과물을 제공받는 형태인 것이다. 여전히 SQA 활동이 정형화되지 않은 분야이며 전문가의 경험이나 지식에 따라 작업셋의 선정이 가변적이라 후자의 접근방법을 택하였다.

규칙-기반 전문가 시스템으로 리파지토리 작업의 커스터마이징 기법을 개발하기 위해서는 SQA 활동 유경험자의 휴리스틱(heuristics)이 요구된다. 이는 이러한 전문가의 휴리스틱에 따라 적절한 품질보증 활동의 작업이 선정되기 때문이다. 본 연구에서는 전문가의 경험을 규칙으로 도출하고 이를 시스템으로 구현할 계획이다. 즉, 프로젝트의 특성과 SQA 조직 특성 요인이 정의되고, 유경험자로부터 규칙이 도출되면 정해진 규칙과 특성요인에 따라 커스터마이징 프로세스가 진행되고 프로세스의 결과로 유형별 작업 셋(set)들이 결정되는 것이다.



(그림 5) 커스터마이징 기법의 규칙-기반 접근방법

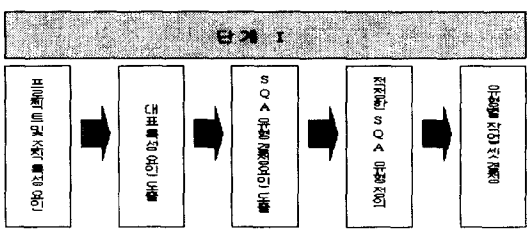
(그림 5)는 일반적인 전문가 시스템의 규칙-기반 접근방법을 묘사하고 있다. 규칙-기반 전문가 시스템은 전문가의 지식을 기반으로 추출

된 규칙을 바탕으로 개발된다.

4.2 커스터마이징 수행단계

본 연구에서 접근하는 리파지토리 작업의 선정 방법은 크게 두 단계로 구분된다. 단계 I은 프로젝트 특성요인과 SQA 조직 특성요인으로부터 적정한 SQA 유형을 결정하는 프로세스이다. 단계 II는 단계 I의 최종 산출물인 유형별 작업 셋(set)을 작업간의 관계를 이용하여 수행할 작업을 재조정하는 프로세스이다.

다음 (그림 6)은 커스터마이징 수행 단계 I을 묘사하고 있다.

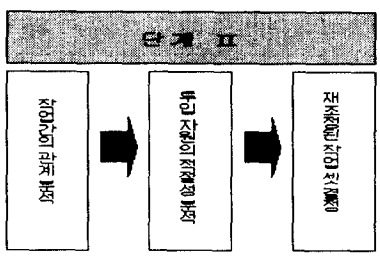


(그림 6) 단계 I 프로세스

단계 I의 최종 목표는 정의된 SQA 유형에 일정한 규칙을 도출하고 이러한 규칙에 따라 유형별 작업 셋을 선정하는 것이다. 작업 셋을 결정하는 규칙은 SI현업의 전문가에 의해서 추출되어진다. 규칙을 도출하는 절차는 먼저, 작업 셋에 선정될 작업의 개별 특성 요인을 추출하고 다음으로 추출된 개별 특성 요인의 속성에 따라 값을 부여한다. 계량화된 값은 리파지토 모든 작업을 중요도, 난이도, 자원투입 요인으로 구분하여 속성 값을 부여한다. 부여된 속성값은 작업 셋을 결정하는 규칙에 의거하여 작업의 선정여부를 결정하는 기준으로 사용된다.

새로운 프로젝트 착수시 SQA 요원은 프로젝트의 특성과 SQA 조직 특성을 바탕으로 SQA 방법론의 리파지토리 작업을 선정하게 된다. 리파지토리의 선정작업은 적지 안은 경험과 실무지식이 필요하며, 많은 시간 비용을 요구하는 작업이다. 작업의 선정은 전문요원의 판단과 경험에 의존됨으로써 신규요원이나 무경험자일수록 더욱 어려운 작업이 될 것이다. 즉, SQA 방법론이 제공된다 하더라도 이를 효과적으로 사용할 수 없는 경우가 빈번할 것으로 예상됨으로 전문가의 추론 방법을 추출하여 이를 시스템으로 구현하는 것은 방법론을 보다 효과적으로 활용하려는 시도라 할 수 있다.

단계 II의 프로세스는 다음 (그림 7)과 같이 묘사할 수 있다.



(그림 7) 단계 II의 프로세스

단계 I의 최종 산출물은 리파지토리 작업 셋이다. 리파지토리 작업 셋은 일정한 규칙에 의해 선정된 결과이지만 이에 대한 검증작업이 필요한 것이다. 단계 II는 단계 I의 결과물을 다시 한번 검토하는 프로세스라 할 수 있다. 작업 셋의 검토 작업은 선정된 작업을 작업간의 관계 측면과 투입자원 측면에서 고려하는 것으로 선정된 작업의 타당성을 높여주는 작업이라 할 수 있다.

단계 II는 먼저, 선정된 작업간의 관계를 묘사하여 작업의 선·후 관계 및 대체관계를 파악하는 것이 다. 리파지토리의 작업들은 작업간에 매우 밀접한 관계를 지닌 작업들과 어떤 작업들은 다른 작업으 로 대체될 수 있는 속성을 지니고 있다. 이러한 속성을 바탕으로 작업의 관계

를 도식화하여 대체 관 계가 성립되는 작업들은 자원 측면을 고려하여 작 업 셋에서 삭제할 수 있다. 또한 선행관계에 따라 삭제가 불가능한 작업은 투입자원과 관계없이 선정되어야 하는 경우도 있다. 이러한 작업간의 관계에 따라 작업 셋은 재조정된다.

작업의 관계에 의해 재조정된 작업들은 전체 자원투입 측면에서 작업의 적정성을 검증한다. 작업의 적정성은 준비 프로세스와 개발 프로세스의 단계로 구분하여 각 단계별 자원 투입 가중치를 바탕으로 검토한다. 각 단계별 가중치는 균등하게 수행하는 것을 원칙으로 정할 수 있다. 단계별 적정성을 검토한 후, 프로젝트 착수시 정한 SQA 활동의 전체 자원 투입규모에 대한 자원 적정성을 평가함으로써 선정된 작업 셋의 타당성을 제고하게 된다.

4.3 활동수준 제고를 위한 추론 모형

여기서는 커스터마이징 기법을 적용하기 위한 모형을 제시하고자 한다. 커스터마이징 기법의 추론 모형은 프로젝트 특성과 SQA 조직 특성 외에도 개발에 적용하는 방법론, 개발 시스템 유형, 개발 경로 등에 따라서도 SQA 활동이 달라질 수 있다. 그러므로 본 연구에서는 SQA 활동에 영향을 미치는 모든 요인을 고려하기 보다는 활동에 보다 직접적으로 영향을 미치는 프로젝트 특성과 SQA 조직의 특성요인을 바탕으로 이를 추론 모형에 적용하고자 하였다.

추론 모형을 적용하여 작업 셋을 결정하는 방법은 앞 절에서 언급한 2단계의 프로세스를 거친다. 단계 I에서는 정의된 SQA 유형에 따라 작업 셋을 결정하고, 단계 II에서는 선정된 작업 셋의 적정성을 평가하는 것이다.

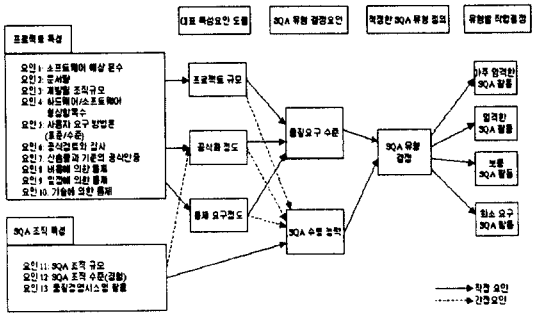
단계 I의 첫 번째 프로세스는 면접조사를 통해 프로젝트와 SQA 조직 특성요인을 추출하는 것이다. 다음은 면접조사를 통해 파악된 영향요인들이다. 프로젝트 특성으로 선정된 요인은 소프트웨어 예상본수, 문서량, 개발팀 조직규모, 하드웨어/소프트웨어 형상항목 수, 사용자의 요구방법론(표준), 공식검토와 감사, 산출물과 기준의 공식인증, 비용관리, 일정관리, 기술관리 등이다. 또 다른 특성인 SQA 조직 특성으로는 SQA 조직 규모, 조직의 수준(경험), 품질경영시스템 등을 선정하였다.

SQA 유형별 작업을 결정하기 위한 절차는 다음과 같다. 첫 번째 단계는 프로젝트와 SQA 조직의 특성요인을 분석하여 대표 특성요인을 도출하는 것이다. 대표 특성요인은 특성요인의 속성을 군집화한 것으로 반드시 모든 특성요인이 대표 특성요인으로 도출되는 것은 아니다. 본 연구에서 특성요인으로부터 도출한 대표 특성요인은 프로젝트 규모, 공식화 정도, 통제 요구정도 등이 추출되었다. 상황에 따라 어떤 특성요인은 대표 특성요인에 영향을 주기보다는 다음 단계에서 도출할 SQA 유형결정요인에 직접적으로 영향을 줄 수 있다. 대표 특성요인이 도출되면 다음 단계는 대표 특성요인을 바탕으로 SQA 유형결정요인을 도출하는 것이다. 도출방법은 전 단계와 유사하며, SQA 유형결정에 가장 영향을 끼치는 요인을 선정하여야 한다. 본 사례에서는 SQA 유형결정요인으로 발주자의 품질에 대한 요구수준과 개발자의 SQA 수행능력을 도출하였다. 최종 단계는 도출된 SQA 유형결정요인을 기반으로 적정한 SQA 유형을 정의하는 것이다. SQA 유형이 정의되면 이에 따라 유형별 작업이 결정되어진다. SQA 유형결정요인을 분석한 결과 SQA 유형으로는 아주 엄격한 SQA 활동, 엄격한 SQA 활동, 보통 SQA 활동, 최소요구 SQA 활동 유형이 결정되었다.

다음 (그림 8)은 SQA 유형의 정의 및 유형별

작업결정 절차를 상세 묘사한 것이다.

결정된 작업유형에 따라 적정한 작업 셋이 선정된다. 작업 셋을 선정하기 위해 리파지토리 작업에 대한 개별 특성 요인을 추출하였고 이러한 개별 특성요인을 바탕으로 리파지토리의 모든 작업에 정해진 속성치를 부여하였다.



(그림 8) SQA 활동 유형정의 프로세스

작업 셋은 개별 특성요인의 속성치를 기준으로 선정된 작업들의 집합이다.

다음 <표 2>는 리파지토리 작업의 개별 특성요인에 속성치를 부여하는 양식의 예이다. 리파지토리의 개별 특성요인은 SQA 조직의 전문가에 의해 도출된 것으로 해당 작업 선정에 가장 큰 영향을 미치는 요인이다.

<표 2>에서 수행단계는 해당 작업이 속한 개발 프로세스 상의 단계를 의미한다. 예를 들어 작업1은 프로젝트의 준비단계에서 수행되는 작업을 의미한다. 선행 작업은 해당 작업 전에 반드시 수행해야 할 작업을 의미한다. 병행 작업은 해당 작업과 시간적으로 같은 시기에 수행가능한 작업을 말한다. 대체 작업은 해당작업을 다른 작업으로 대체할 수 있는 작업을 의미한다. 예를 들어 기술검토와 경영층 검토 작업은 상호 대체적인 관계를 지닌 것으로 파악되었다. 중요도는 작업의 필수/선택을 의미하며 이를 조금 더 세분화하여 3개 유형-매우 중요, 중요, 보통, 으로 구분한다. 인력투입은 man/date 단위로 작업당 투입자원을 묘사한 것이다. 예를 들어 1M/D는 한 사람이 하루에 수행할 수 있는 작업량을 의미한다. 끝으로 작업의 난이도는 해당 작업 수행시 어려움을 계량화한 것으로 중요도와 같이 3개 유형-매우 어려움, 어려움, 보통, 으로 구분한다.

〈표 2〉 리파지토리 작업의 개별 특성요인에 대한 속성치 양식

작업 번호	작업명	내용	수행 단계	선행 작업	병행 작업	대체 작업	중요도	인력투입 (M/D)	작업 난이도
리파지토리 작업1	소프트웨어 개발 계획서 검토	소프트웨어 품질보증 계획을 수립하기 위해 개발 유형, 개발 경로 및 기간등을 파악한다							
리파지토리 작업2	형상 관리 계획서 검토	하드웨어/소프트웨어에 대한 형상관리 및 변경관리 계획을 검토							
리파지토리 작업3	시스템 테스트 계획서 검토	개발될 소프트웨어의 테스트 계획이 적절한지, 사용자의 요구는 만족하는지를 검토							
⋮	⋮	⋮							

결론적으로 이러한 작업의 개별 특성요인들은 그들의 속성치를 바탕으로 일정한 규칙에 따라 조합되어 작업 유형 셋을 구성하게 된다.

다음은 단계 II의 프로세스로 선정된 작업 셋의 적정성을 평가하는 절차이다. 단계 I의 산출물인 작업 셋들은 일정한 규칙을 통해 선정된 리파지토리 작업들이다. 선정된 작업들은 실제

SQA 활동을 수행하기 위한 여러 조건을 갖추고 있다. 즉, 선정된 작업들은 그 동안 SI현업에서 수행되어왔던 품질보증 실행작업을 전문가의 지식을 바탕으로 추출된 것들이기 때문이다.

하지만 이렇게 선정된 작업들도 전체적인 적정성 평가차원에서 한번 더 검토할 필요가 있다. 이는 모든 작업들이 그러하듯이 투입자원에 따라 삭제되거나 추가되는 경우가 매우 빈번하기 때문이다. 또한 작업간의 관계에 따라 삭제되거나 대체되는 경우도 있다. 즉 작업간에는 선행관계, 병행관계, 그리고 대체관계가 존재하기 때문이다.

대체관계가 성립되는 작업을 모두 수행하는 것은 자원의 소모로 이어져 전체 SQA 활동에 적지 않은 부담으로 작용될 수 있다. 그러므로 선정된 작업 셋은 재정의 되어 사용할 때 효과성이 제공될 것이다.

선정된 작업 셋의 적정성은 앞에서도 언급하고 있지만 크게 두 가지 측면에서 평가할 수 있다. 먼저, 작업간의 관계를 이용한 평가이다. 작업간의 관계를 도식화하여 보면 작업간의 선·후 관계와 대체관계가 묘사되고 SQA 요원은 이를 기반으로 필수작업과 선택작업을 선별할 수 있다. 다음으로 SQA 활동에 대한 전체 자원 측면의 평가이다. 전체 자원은 SQA 활동 계획에 명시될 것이다. 전체 투입자원은 개발 프로세스의 중요도에 따라 적정한 비율로 분산되고 이러한 분산 비율에 따라 선정된 작업을 평가하여 단계별 투입자원의 비율이 적정한가를 볼 수 있다. 또한 선정된 작업의 총 투입자원을 계산하여 SQA 활동 계획의 전체 투입자원과 비교하여도 작업의 적정성을 평가할 수 있다.

5. 요약 및 향후 연구방향

본 연구의 주요 목표는 품질보증 활동을 체계적으로 지원해주기 위한 품질보증방법론을 제시하고 이의 효과적 활용을 위한 규칙기반 접근방법을 제시하는 것이다. 이 품질보증방법론은 프로세스와 리파지토리로 구분된다. 프로세스는 실제 방법론의 단계 및 활동을 묘사하는 접근방법을 띠고 있다. 이 프로세는 크게 계획단계, 실행단계 그리고 개선단계로 진행된다. 리파지토리는 품질보증 활동에 필요한 실행 작업을 담고 있는 저장소라 할 수 있다. 리파지토리에 저장된 작업은 프로젝트의 특성에 따라 재구성이 가능하도록 품질보증 활동에 필요한 대부분의 작업을 포함한다. 그러므로 품질보증방법론은 실제 방법론의 단계 및 활동에 대해서는 품질보증방법론의 프로세스를 참조하고, 리파지토리에서는 품질보증 실행활동의 작업만을 담고 있는 이중적 구조를 띄고 있다.

여기서 제시된 방법론은 몇 가지 특성을 갖고 있다. 우선, 프로젝트 특성에 따라 커스터마이징이 용이하도록 개발된 수정형 방법론이다. 또한 이 방법론은 개발조직과 별도로 구성되는 품질보증조직을 위한 방법론이다.

아울러 본 연구에서는 이 방법론의 커스터마이징을 체계적으로 지원해주는 규칙기반 접근방법을 제시하고 있다. 이는 방법론 연구의 최근 연구방향인 방법공학(method engineering)의 일종이다. 즉, 일반적으로 모든 프로젝트에 적합한 단일 방법론을 제시하기 보다는 필요가능한 다양한 내용을 리포지토리에 넣어두고 프로젝트 특성에 적합한 내용만을 추출하여 적용하는 접근을 말한다. 향후 대부분의 방법론이 이런 방향으로 개발, 활용되어야 한다는 Martin & Odell[1996]의 지적을 반영하고 있다는 점이 큰 의의라고 하겠다.

본 연구에서 제시한 방법론과 커스터마이징 방법을 통해 조직의 품질보증 활동수준을 어느 정도 향상할 수 있을 것으로 기대된다. 그러나

여전히 몇 가지의 한계점을 안고 있다. 우선 개발된 방법론에 대한 실증적 사례연구가 요구된다. 즉, 개발된 방법론이 국내 기업에서 얼마나 효과적으로 적용될 수 있는 지에 대한 실증적 자료가 제시되어야 할 것이다. 또한 국내 SQA 전문가 부재로 프로젝트 특성으로부터 태스크를 선정하는 규칙의 신뢰성 역시 해결해야 할 문제라 할 수 있다.

참 고 문 헌

[1] 과학기술처, *소프트웨어 품질관리체계정립 및 지원도구 개발에 관한 연구*, 1989.

[2] 과학기술처, *생산성향상 및 품질관리 기술개발에 관한 연구(I)*, 1996.

[3] 김성근 · 이진실 · 배이철 · 김지혜 · 편완주, "국내 정보시스템 개발의 품질보증 활동의 이해", *한국 DB 학회 '99춘계학술대회 발표논문*, 1999.

[4] 김성근 · 이진실 · 배이철 · 김지혜 · 편완주, *정보시스템 품질보증 및 감리 방법론 개발에 관한 연구*, 쌍용정보통신, 1999.

[5] 양해술, "소프트웨어 공학과 품질관리 방법론의 동향", 한국정보처리학회학회지, 1994.

[6] 양해술 · 이창석, "프로그램의 복잡성과 계산적 복잡도에 관하여", 한국정보과학회 학술발표논문집, 1989.

[7] 양해술 · 임춘봉, 정호원, "소프트웨어의 품질 보증과 평가 방법", 전산망기술 및 표준화 심포지움, 1992.

[8] 이경환, "소프트웨어 품질관리 기술의 현황과 전망", *정보과학회회지*, 1990.

[9] 이순용, *현대품질관리론*, 법문사, 1988.

[10] 이용근 · 양해술, "소프트웨어 품질 보증 기술의 동향", 한국정보과학회 소프트웨어 공학회지, 1993.

[11] 이용근 · 양해술, "소프트웨어의 품질을 고려한 비용평가 모델의 제안과 평가", 한국정 보처리학회 논문지, 1994.

[12] 이주헌, "Application of OR/MIS Concepts to Management of Lange Software Projects," 일리노이공대 박사학위논문, 1983.

[13] 정기원 · 윤창섭 · 김태현, *소프트웨어 프로세스와 품질, 홍릉과학출판사*, 1997.

[14] 정재학 · 양해술, "소프트웨어 개발 공정에서의 정량적인 관리 방법", 한국정보과학회지 학술발표논문집, 1994.

[15] 한국소프트웨어산업협회, "소프트웨어개발비 산정기준 개정안", 1996.

[16] 한국통신, "CASE Tool 개발 기술연구", 소프트웨어 연구소, 1992.

[17] 정보통신부, *소프트웨어 품질보증기준*, 1998.

[18] 한국정보산업연합회, *소프트웨어 품질/평가 관리기법 세미나*, 1997.

[19] 한국전기통신공사, *소프트웨어 품질보증을 위한 경영정보처리 소프트웨어 관리 및 개발에 관한 연구*, 1987, p.11.

[20] 한국전자통신연구원, *프로젝트 및 조직 유형 분류 및 개발경로 개발*, 1998.

[21] Abdel-Hamid, T., "The Economics of Software Quality Assurance : A Simulation-Based Case Study," *MIS Quarterly*, 1988.

[22] Ahituv, N. and M. Zelek, "Instant Quality Control of Large Batch Processing Jobs," *MIS Quarterly*, 1987.

[23] Arthur, L. J., *Measuring Programmer Productivity and Software Quality*, New York, John Wiley & Sons, 1985.

[24] Apte, U., Sanker, C., Thakur, M. and J. Turner, "Reusability-Based Strategy for Development of Information systems :

Implementation Experience of a Bank," *MIS Quarterly*, 1990.

[25] Barnes, B., and T. Bollinger, "Making Reuse Cost-Effective," *IEEE Software*, 1991.

[26] Bendifallah, S. and W. Scacchi, "Understanding Software Maintenance Work," *IEEE Trans. on Software Engineering*, 1987.

[27] Boehm, B.W., *Software Engineering Economic*, Prentice Hall, 1981.

[28] Boehm, B. and R. Ross, "Theory-W Software Project Management : Principles and Examples," *IEEE Trans. on Software Engineering*, 1989.

[29] Brelsford, J., "Establishing a Software Quality Program," *Quality Progress*, 1988.

[30] Buckley, F. and R. Poston, "Software Quality Assurance," *IEEE Trans. on Software Engineering*, 1984.

[31] Budling, F. C., P. A. Szulewski, and R. J. Ganska, "Process Tailoring for Software Project Plans," *The Softeware Technology Support Center*, 1996.

[32] Carpenter, M. and H. Hallmen, "Quality Emphasis at IBM's Software Engineering Institute," *IBM Systems J.*, 1985.

[33] Carrol. J. and J. McKendrea, "Interface Design Issues for Advice-Giving Expert Systems," *Communication of the ACM*, 1987.

[34] Cartwright, J., T. Andrews, and P. Webley, "A Methodology for cultural measurement and change : A case study," *Total Quality Management*, 1999.

[35] Cerveny, R., Garrity, E. and G. Sandes, "Prototyping in Systems Development," *J. of Management Information Systems*, 1986.

[36] Chou, D. C., D. C. Yen and J. Q. Chen, "Analysis of the total quality management based software auditing," *Total Quality Management*, Oct 1998.

[37] Crosby, P. B.,*Quality without Tears*, McGraw-Hill, 1984.

[38] Dagwell, R. and R. Weber, "Systems Designers' User Models : A Comparative Study and Methodological Critique," *Communication of the ACM*, 1983.

[39] Deming, E. W., Quality, *Productivity and Competitive Position*, Cambridge MA, MIT, 1982.

[40] Deutsch, M.S. and R.R. Willis, *Software Quality Engineering*, Prentice Hall, 1988.

[41] Engler, N., "Software quality : It's up to developer, too," *Computerworld*, 1997.

[42] Evans, M. W. and J. J. Marciniak, *Software Quality Assurance and Management*, New York, John Wiley & Sons, 1987.

[43] Feigenbaum, A. V., *Total Quality Control*, McGraw-Hill, 1986.

[44] Feinawer, L., "Compiler Issues Associated with Safety-Related Software," *Nuclear Technology*, 1991.

[45] Flood, R. L., *Beyond TQM*, John Wiley, 1993.

[46] Ginac, F., *Customer-Oriented Software Quality Assurance*, Prentice Hall, 1998.

[47] Ginsberg, M. P. and L. H. Quinn, "Process Tailoring and the Software Capability Maturity Model," *Software Engineering Institute*, Nov. 1995.

[48] Gould, J. and C. Lewis, "Designing for Usability : Key Principles and What De-

signers Think," *Communication of the ACM*, 1985.

[49] Grady, R. B. and D. L. Caswell, *Software Metrics : Establishing a Company-wide Program*, Prentice-Hall, 1987.

[50] Grumen, G., "How to Assure Quality : Debate Shows Divisions," *IEEE Software*, 1991.

[51] Halasz, F., "Reflections on Note Cards : Seven Issues for the Next Generation of Hypermedia Systems," *Communication of the ACM*, 1988

[52] Henderson, J. and J. Cooprider, "Dimensions of I/S Design Teams : A Control Theories Perspective," *Management Science*, 1992.

[53] Hollocken, C., "Finding the Cost of Software Quality," *IEEE Trans. on Software Engineering*, 1986.

[54] Ince, D,. *An Introduction to Software Quality Assurance and its Implementation*, McGraw-Hill, 1994, pp.3-6.

[55] _____, "Software Metrics : Introduction," *Information and Software Technology*, 1990.

[56] International Standard, *Information Technology-Software Evaluation*, ISO/IEC 9126, 1991.

[57] Jackson, P. and D. Ashton, *Implementing Quality Through BS5750(ISO 9000)*, Kogan Page, 1993.

[58] Juran, J. M., *Quality Planning and Analysis*, McGraw-Hill, 1980.

[59] Kane, E., "Implementing TQM at Dun & Bradstreet Software" *National Productivity Review*, 1992.

[60] Karimi, J., "An Asset-Based Systems Development Approach to Software Reusability," *MIS Quarterly*, 1990.

[61] Kim, Y. and E. A. Stohr, "Software Reuse : Survey and Research Directions," *Journal of Management Information Systems*, 1998.

[62] Levendel, Y., "Reliability Analysis of Large Software Systems : Defect Data Modeling," *IEEE Trans. on Software Engineering*, 1990.

[63] Martin, J. and J. J. Odell, *Object-oriented Methods*, Prentice Hall, 1996.

[64] Miller, H., "Quality Software : The Future of Information Technology," *J. of Systems Management*, 1989.

[65] Murine, G., "Integrating Software Quality Metrics with Software Quality Assurance," *Quality Progress*, 1988.

[66] Napier, H., Lane, D., Batsell, R. and N. Guadango, "The Impact of a Restricted Natural Language Interface on Ease of Leaning and Productivity," *Communication of the ACM*, 1989.

[67] Nenz, J., "Software Quality Assurance : Systems 12," *Electrical Communication*, 1985.

[68] Newman, M. and D. Robey, "A Social Process Model of User-Analyst Relationships," *MIS Quarterly*, 1992.

[69] Paughtrey, T., "The Search for Software Quality," *Quality Progress*, 1988.

[70] Pfleeger, S. and T. Bollinger, "The Economics of Reuse : New Approaches to Modeling and Assessing Cost," *Information and Software Technology*, 1994.

[71] Pressman, R. S., *Software Engineering : A practitioner's Approach(Fifth Edition)*, Mc-

Graw-Hill, 1992.

[72] Rahman, W., "Software Quality by Management : Learning from the Manufacturing Industries," *Information and Software Technology*, 1987.

[73] Rai, A., H. Song and M. Troutt, "Software Quality Assurance : An Analytical Survey and Research Prioritization," *Journal of Systems Software*, 1994.

[74] Ramin, M. K., G. P. Ali, and K. Seiichi, "ISO Standards : Perceptions and experiences in the UK construction industry," *Construction Management and Economics*, 1999.

[75] Rush, K., Draving, S. and J. Kerley, "Technical Challenges to a Decentralized Phone System," *IEEE Spectrum*, 1990.

[76] Siegel, S., "Why We Need Checks and Balances to Assure Quality," *IEEE Software*, 1992.

[77] Software Engineering Process Office(SEPO), *Software Quality Assurance Plan (SQAP) Template*, Version 1.2, 1997.

[78] Software Engineering Process Office(SEPO), *Software Quality Assurance(SQA) Process*, Version 1.4, 1997.

[79] Swanson, K., McComb, D., Smith, J. and D. McCubbrey, "The Application software Factory : Applying Total Quality Techniques to Systems Development," *MIS Quarterly*, 1991.

[80] Van Treeck, G. and R. Thackeray, "Quality Function Deployment at Digital Equipment Corporation," *Concurrent Engineering*, 1991.

[81] Walters, G. F. and J. A. McCall, "The Development of Metrics for Software R&M," *The Proceedings of ARM Symposium*, 1978.

[82] Wilden, J., Wolft, A., Rosenblatt, W. and P. Tarr, "Specification-Level Interoperability," *Communication of the ACM*, 1991.

■ 저자소개

김 성 근

미국 New York University에서 Information Systems 박사 학위를 취득하였다. 동 대학에서 전임강사를 거친 후, 한국산업투자자문(주)의 전문위원과 고문을 역임하였다. 미 국방성 산하 전자상거래센터에서 연구원을 역임하였으며, 현재 중앙대학교 경영학과 교수로 근무하고 있다. 주요 관심분야는 정보계획수립, 데이터웨어하우스, 지능형 정보기술 활용 등이다.

편 완 주

공군사관학교 전자공학과를 졸업하고, 중앙대학교 대학원 경영학과에서 경영학석사, 그리고 동대학원에서 경영학 박사 과정을 수료하였다. 현재 공군 소령으로 공군본부 항공사업단에서 공중조기경보통제기 사업을 담당하고 있으며, 주요관심분야는 소프트웨어품질보증 방법론분야이다.