항공우주산업기술동향 4권2호 (2006) pp. 76~85
기술동향
www.kari.re.kr
/lib/index.html
에서 보실 수 있습니다.

국방사업의 소프트웨어 개발 프로세스와 소프트웨어 정부품질보증

글/서 장 원 jwsuh@kari.re.kr
한국항공우주연구원 항공우주안전・인증센터 항공인증팀

초 록

최근 정보화 사회의 진전과 함께 각계각층에서 컴퓨터의 활용이 크게 늘어나고 있으며 따라서 소프트웨어를 이용하여 각 산업 활동이 신속성, 효율성, 생산성 및 편리성 등의 측면에서 많은 효과를 보게 됨에 따라 소프트웨어의 비중은 날로 높아지고 그 중요성도 새롭게 인식되고 있다. 국내 민간분야에서의 소프트웨어 개발은 다양한 민간규격(ISO/IEC-12207, ISO/IEC-9126 등)을 따라 수행되고 있으며 국방분야에서는 “소프트웨어 개발 프로세스 (방위사업청 지침 2006-9, 2006.1.25)”이 소프트웨어 개발지침으로 최근 채용되어 개발에 적용되고 있다. 본 논문에서는 “소프트웨어 개발 프로세스 (방위사업청 지침 2006-9, 2006.1.25)”의 고찰을 통하여 소프트웨어 개발 프로세스의 요구조건을 소개 하였으며, 국방사업에서의 소프트웨어 품질관리 방법에 대하여 살펴보았다.

주제어 : 소프트웨어 개발 프로세스, 소프트웨어 품질보증

1. 서 론

최근 정보화 사회의 진전과 함께 컴퓨터 소프트웨어 산업이 급속히 발달하기 시작하고 많은 소프트웨어가 개발됨으로써 컴퓨터는 하드웨어 제품이나 시설을 개발, 설계, 생산, 설치 또는 운전(사용)하는데 있어 또한 다양한 서비스를 제공하는데 있어 여러 방면으로 활발히 이용되고 있다. 이와 같이 소프트웨어를 이용하여 각 산업 활동이 신속성, 효율성, 생산성 및 편리성 등의 측면에서 많은 효과를 보게 됨에 따라 소프트웨어의 비중은 날로 높아지고 그 중요성도 새롭게 인식되고 있다.

이러한 상황에서 일찍이 군사 및 항공산업 분야에서는 MIL-STD-498, ISO/IEC-12207, ISO/IEC-9126, RTCA DO-178 등과 같은 소프트웨어 개발 요건을 표준으로 개발, 시행해오고 있으며, 우리나라 국방사업의 경우 최근 소프트웨어 개발 프로세스 (방위사업청 지침 2006-9)를 발행하여 국방사업의 소프트웨어 개발 기준으로 설정한 바 있다.

소프트웨어 품질보증은 소프트웨어의 근본적인 결함이나 에러의 잠재력을 제거하고자 하는 것으로써 소프트웨어 품질보증 프로그램은 소프트웨어의 라이프사이클을 통하여 적절하고 체계적인 품질보증 기법을 적용함으로써 소프트웨어 에러의 가능성을 감소시키자는 데 그 목적이 있다.

그러나 근본적인 결함의 제거만이 소프트웨어 품질보증의 목적이 되는 것은 아니다. 문서들이 제대로 확보되지 않았거나 실제의 프로그래밍기법 보다 더 복

잡하게 생성된 프로그램들은 이해하기 어렵고 검증(Verification)이나 수정(Debugging)하기도 어렵다.

이러한 소프트웨어와 관련된 문제점들은 신뢰성 없는 소프트웨어, 사용하기 어려운 소프트웨어, 규격서나 요건이 갖추어지지 않은 소프트웨어, 시스템 자원을 효율적으로 사용하지 못하는 소프트웨어, 검증을 하지 못한 소프트웨어, 문서화가 되지 않은 소프트웨어 등을 들 수 있다.

다음에서 소프트웨어 개발 프로세스(방위사업청 지침 2006-9, 이하 개발지침)와 소프트웨어 품질보증에 대하여 살펴본다.

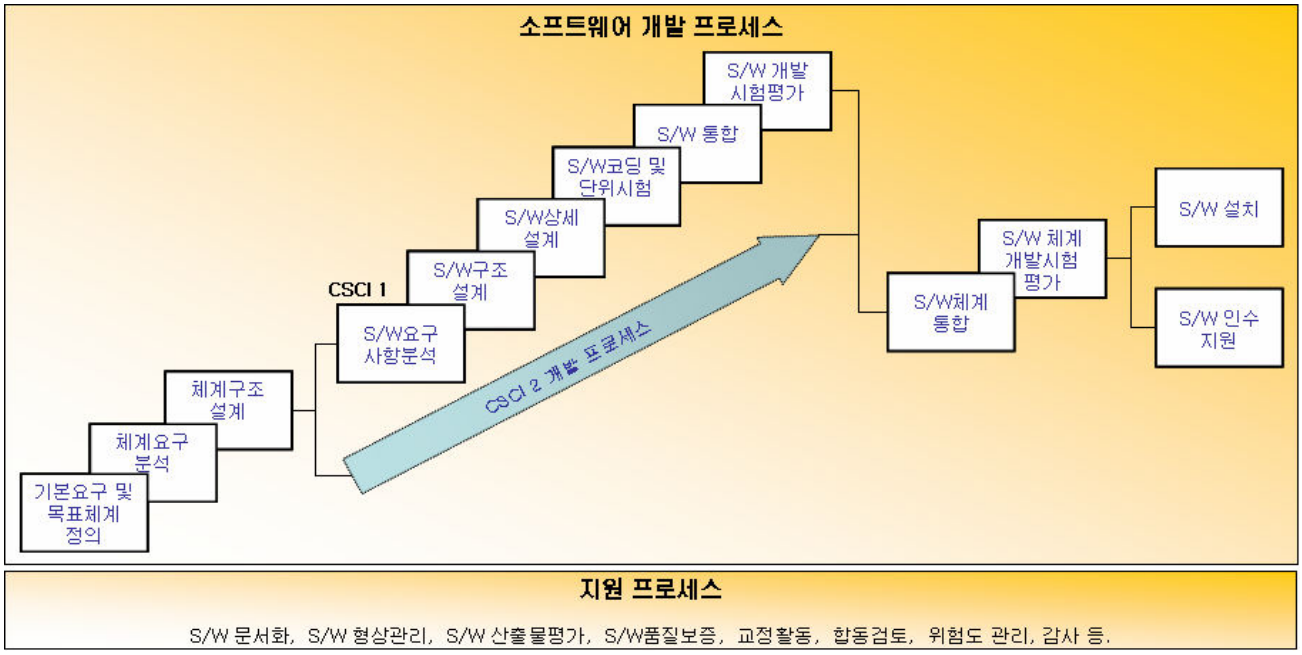

그림 1. 소프트웨어 획득 단계

2. 소프트웨어 개발 프로세스

소프트웨어 개발 프로세스(방위사업청 지침 2006-9)는 국방사업의 소프트웨어 개발 프로세스에 적용되는 지침으로서 기존의 무기/비무기체계 내장형 소프트웨어 개발관리 지침(국방부)를 대치하여 무기체계로 분류되는 소프트웨어와, 각종 무기체계에 포함되는 내장형 소프트웨어의 개발과 관련하여 공정에 관한 표준 절차를 규정한다.

이 지침은 소프트웨어 수명주기 프로세스를 정의한 국제표준(ISO/IEC-12207) 및 정보통신부에서 작성한 발주·관리 표준 프로세스별 지침을 참고하여 국방 정보체계 획득 및 개발에 필요한 프로세스를 재구성하여 획득자와 공급자의 역할을 정의한다.

2.1 일반요구사항

개발지침은 연구개발을 통하여 획득되는 소프트웨어 체계는 그림 1과 같이 탐색개발과 체계개발단계로 구분하여 추진하며, 각 단계별 추진계획을 수립하는 공정(프로세스)를 제시한다.

개발지침은 소프트웨어 체계 획득시 체계 규모 및 특성에 따라 제시된 공정을 적용하기 위한 방안을 검토·조정(Tailoring)을 허용하고 있으며, 그 결과를 체계개발계획서에 반영할 것과, 적용방

안에 따라 인수하여야 하는 문서를 사전 식별하여 계약요구자료목록(CRDL)으로서 계약서에 포함시킬 것을 요구 한다. 또한 CRDL으로 포함되는 문서는 필요시 유사 관련문서를 통합하여 작성할 수 있도록 허용하고 있다. 내장형 소프트웨어 개발에 적용하는

경우, 개발지침은 체계개발단계중 소프트웨어 요구사항분석부터 소프트웨어 개발시험평가까지의 활동에 적용됨을 원칙으로 하되 체계 규모 및 특성에 따른 적용방안을 검토·조정할 수 있도록 하고 있다.

2.2 세부 요구사항

가. 기본요구 정의 프로세스

기본요구 정의 프로세스는 체계 및 소프트웨어나 서비스 획득에 필요한 개념을 정의하는 활동을 정의하며, 개념 및 요구 상세화, 기본 체계 요구사항 정의 및 기본 소프트웨어 요구사항 정의 활동을 획득자가 수행한다.

나. 탐색개발계획 수립 프로세스

조직의 내·외부 환경, 업무 분석을 통해 새로운 목표 체계를 정의하고, 목표 체계를 획득하기 위한 추진계획을 수립하는 활동으로서, 탐색개발계획 준비 활동을 획득자가 수행하며, 현행 체계 분석, 목표 체계 정의 및 탐색개발계획 수립 활동은 탐색개발을 수행하는 공급자(수주자)가 수행한다.

다. 체계요구분석 프로세스

개발하고자 하는 체계의 요구사항을 분석하고, 이에 대한 검토 및 평가를 목적으로, 개발자는 체계 요구사항을 정의하고, 체계 요구사항의 검토 및 평가의 수행을 주관한다.

라. 체계 구조설계 프로세스

체계 구조설계에서는 체계를 구성하는 주요 요소와 기능을 식별하고, 이들 간의 상관관계를 정의하는 활동을 수행한다. 체계를 구성하는 요소는 크게 소프트웨어 항목, 하드웨어 항목, 수작업 항목으로 구분하여 정의한다.

개발자는 체계 구조설계 활동을 통하여 체계의 전체 모습을 조감할 수 있는 최상위 구조를 설계한다. 또한, 설계후 개발자는 체계 구조에 대하여 체계 요구사항의 추적성, 설계 표준 및 방법의 적절성, 운영 타당성 등의 기준으로 검토하고, 최종안을 확정하는 활동을 주관한다.

마. 개발계획 및 인수전략 수립 프로세스

획득자는 공식적인 계약절차를 통하여 획득하고자 하는 내용을 전체적으로 정의하는 체계개발추진계획서를 작성하고, 대상 사업에 대한 인수전략을 정의한다.

바. 개발 준비 프로세스

소프트웨어를 개발하기 위하여 사업의 범위, 규모, 복잡도 등에 적합한 소프트웨어 수명주기 모델을 정의하되, 개발에 필요한 문서화, 형상관리, 위기관리 등을 위하여 관련 프로세스의 이용방법을 포함한다. 또한 개발자는 체계개발계획서를 작성한다.

개발자는 사업의 범위, 규모, 복잡도 등에 따라 적합한 소프트웨어 수명주기 모델을 정의한다. 소프트웨어 수명주기 모델은 개발 프로세스의 활동 및 작업 수준까지 정의한다.

개발자는 해당 사업에 대한 체계개발계획서를 작성한다. 개발계획에는 해당 사업에 적용하기 위한 개발 프로세스의 활동 및 작업에 대한 표준, 접근방법, 도구, 일정, 책임 등이 정의된다.

사. 소프트웨어 요구사항 분석 프로세스

체계 구성항목 중에서 소프트웨어 항목에 대한 기능 및 성능에 대한 요구사항을 분석하고 정의한다. 또한, 정의된 소프트웨어 요구사항이 사용자 요구사항에 부합되는지 검토하고 확정한다.

개발자는 소프트웨어 요구사항 정의 활동을 통하여 소프트웨어의 기능 및 성능에 대한 요구사항을 분석하고 정의한다. 또한 소프트웨어 요구사항 검토 및 평가 활동을 주관하여 기 정의된 소프트웨어 요구사항을 체계 요구사항, 시험 가능성 등의 기준으로 검토한다. 개발자는 획득자, 공급자와 합동 검토를 통해 소프트웨어 요구사항을 평가할 수 있다.

아. 소프트웨어 구조설계 프로세스

체계 구성항목에서 소프트웨어를 구성하는 요소가 무엇인지를 식별하고, 이들 간의 상관관계를 정의한다. 소프트웨어 구조는 전체 체계에 대한 기능 및 성능에 영향을 줄 수 있는 부분이므로 체계의 특성을 고려하여 정의되어야 한다.

소프트웨어 구조설계에서 개발자는 소프트웨어

구성요소에 대한 외부 인터페이스 및 소프트웨어 컴포넌트 간의 최상위 수준 인터페이스를 설계하여야 한다. 또한 최상위 수준의 데이터베이스를 설계하고, 초기 사용자문서를 개발하여야 한다.

개발자는 소프트웨어 통합시험을 위한 요구사항과 통합시험 계획을 작성하고, 소프트웨어 항목의 구조, 인터페이스 및 데이터베이스 설계문서를 검토하여야 한다. 개발자는 획득자, 공급자와 합동 검토를 통해 소프트웨어 구조설계를 평가하고 승인할 수 있다.

자. 소프트웨어 상세 설계 프로세스

소프트웨어 상세 설계는 소프트웨어 컴포넌트로부터 단위 소프트웨어를 식별하고, 단위 소프트웨어 간의 인터페이스, 단위 소프트웨어 내부의 처리 작업을 상세하게 설계하는 활동으로 이루어진다.

소프트웨어 컴포넌트는 코딩할 수 있고, 컴파일할 수 있고, 시험할 수 있는 단위 소프트웨어를 포함하는 하위 수준으로 세분화되어야 한다. 모든 소프트웨어 요구사항은 소프트웨어 컴포넌트로부터 단위 소프트웨어로 할당되어야 한다.

개발자는 소프트웨어 항목에 대한 외부 인터페이스와 소프트웨어 컴포넌트간의 인터페이스, 단위 소프트웨어간의 인터페이스에 대한 상세설계를 개발하고 문서화하여야 한다. 인터페이스의 상세설계는 추가적인 정보없이 코딩할 수 있을 정도로 상세화 되어야 한다. 또한, 데이터베이스에 대한 상세설계를 실시하며, 필요한 경우 사용자문서를 갱신하는 활동을 주관한다.

개발자는 소프트웨어 단위시험을 위한 요구사항과 계획을 작성하고, 단위 소프트웨어를 요구사항의 한계까지 시험하여야 한다.

소프트웨어 통합시험을 위한 요구사항과 계획을 수정 및 보완한다. 개발자는 인터페이스와 데이터베이스를 포함하는 소프트웨어 상세설계를 획득자, 공급자와 합동 검토하여야 한다. 개발자는 기술 및 관리 검토를 위해 획득자와 합동 검토를 수행한다.

차. 소프트웨어 코딩 및 단위시험 프로세스

소프트웨어 상세설계 내역을 이용하여 단위 소프트웨어 및 데이터베이스에 대한 코딩을 수행하고, 소스 코드를 생성한다. 계획에서 정한 프로그래밍 언어를 이용하여 단위 소프트웨어 프로그램을 개발하고, 단위 소프트웨어 및 데이터베이스 시험을 위한 절차와 시험자료를 준비하여, 정해진 절차에 따라 단위시험을 수행한다.

카. 소프트웨어 통합 프로세스

개발자는 단위 소프트웨어와 소프트웨어 구성요소를 소프트웨어 형상항목으로 통합하기 위한 활동을 수행한다. 개발자는 소프트웨어 통합계획을 작성하고, 계획에 따라 단위 소프트웨어와 소프트웨어 컴포넌트를 통합하고, 통합결과가 개발되는 대로 시험을 하여야 한다. 각각의 통합결과는 소프트웨어의 통합 요구사항을 만족시켜야 하며, 소프트웨어 항목이 통합 활동의 결과로써 통합되었음을 보장하여야 한다. 통합 및 시험 결과를 문서화 하여야 한다.

개발자는 사용자문서를 갱신하고 소프트웨어 개발시험평가를 위한 요구사항 및 일련의 시험종류, 시험케이스(입력, 출력 및 시험기준), 시험절차 등을 정의한다.

개발자는 소프트웨어 통합계획, 설계, 코드, 시험, 시험결과, 사용자문서를 검토하고, 기술 및 관리 검토를 위해 획득자, 공급자와 합동검토를 수행한다.

타. 소프트웨어 개발시험평가 프로세스

개발자는 소프트웨어 항목에 대한 자격 요구사항에 따라 소프트웨어 개발시험평가를 준비하고, 계획에 따라 개발시험평가를 수행하고 시험결과를 검토하여야 한다. 또한, 사용자문서를 갱신하고, 소프트웨어 개발시험평가를 위한 요구사항 및 절차를 정의한다. 개발자는 소프트웨어의 설계, 코드, 시험, 시험결과, 사용자문서를 검토하고, 감리를 통해 획득자를 지원한다. 감리를 수행할 경우 감리의 결과를 통해 체계 통합, 체계 개발시험평가, 소프트웨어 설치, 소프트웨어 인수지원을 위하여 소프트웨어를 준비 및 필요한 갱신을 처리하며 소프트웨어 항목의 설계 및 코드에 대한 기준선을 설정한다.

파. 체계 통합 프로세스

개발자는 소프트웨어 형상항목을 하드웨어 항목,

작업 및 기타의 체계 등과 함께 시스템 차원에서 통합하기 위한 활동을 수행한다. 개발자는 체계를 통합하고 결과를 시험하여야 한다. 시험을 위하여 개발자는 각 체계의 자격 요구사항에 대하여 체계 개발시험평가를 수행하기 위한 일련의 시험, 시험케이스(입력, 출력 및 시험기준), 시험절차 등을 개발하고, 문서화 한다.

하. 체계 개발시험평가 프로세스

개발자는 체계 자격 요구사항에 따라 개발시험평가를 준비하고 시험을 수행한다. 개발자는 시험계획에 따라 개발시험평가를 수행하고 시험결과를 검토하여야 한다.

개발자는 체계 개발시험평가를 위한 요구사항 및 절차를 정의하고, 체계 요구사항에 대한 시험적용범위, 예상결과와의 일치성 등을 검토하고 감리를 통해 획득자를 지원한다. 감리를 수행할 경우 감리의 결과를 통해 소프트웨어 설치, 소프트웨어 인수지원을 위한 소프트웨어 준비 및 필요한 갱신을 처리하며, 소프트웨어 형상항목의 설계 및 코드에 대한 기준선을 설정한다.

거. 소프트웨어 설치 프로세스

개발자는 시험이 완료된 소프트웨어에 대하여 계약서에 정해진 실제 사용자(획득자)의 운영환경에 설치하기 위한 활동을 수행한다. 개발자는 소프트웨어를 설치하기 위한 계획을 작성하고, 계획에 따라 소프트웨어 제품을 설치하여야 한다.

너. 소프트웨어 인수지원

개발자는 설치된 소프트웨어를 기초로 획득자가 최종적으로 수행하여야 하는 인수 검토 및 시험을 지원한다. 개발자는 계약에 명시되어 있는 대로 완료된 소프트웨어를 획득자에게 인도하여야 한다. 또한 개발자는 계약에 명시되어 있는 획득자에 대한 초기 교육 및 후속적인 교육을 지원하여야 한다.

2.3 소프트웨어 획득 단계별 산출물

그림1의 소프트웨어 획득 단계에 따라 개발지침이 규정한 산출물은 다음 표와 같다.

표1. 소프트웨어 개발지침에 따른 개발단계별 산출물

구 분	산출물
기본요구 및 목표체계 정의	· 운용개념기술서 · 기본 체계요구사항 기술서 · 기본 소프트웨어 요구사항기술서 · 탐색개발계획서
체계요구분석	· 체계요구사항명세서 초안 · 체계요구사항명세서
체계 구조설계	· 체계설계기술서 초안 · 체계설계기술서 · 소프트웨어 수명 주기 모델 기술서 · 체계개발계획서
소프트웨어 요구사항 분석	· 소프트웨어요구사항 규격서 초안 · 소프트웨어요구사항 규격서
소프트웨어 구조설계	· 초기 수준) 소프트웨어 통합 시험 계획서 · 소프트웨어설계 기술서 (소프트웨어 구조부문) · 최상위 수준) 인터페이스설계 기술서 · 최상위 수준) 데이터베이스설계 기술서
소프트웨어 상세 설계	· 소프트웨어단위 시험계획서 · 소프트웨어통합 시험계획서 (수정 및 보완) · 소프트웨어설계 기술서 · 인터페이스설계 기술서 · 데이터베이스설계 기술서
소프트웨어 코딩 및 단위시험	· 소프트웨어 단위 및 데이터베이스 시험 절차서 · 소프트웨어 통합 시험 계획서 (수정 및 보완) · 소스코드 기록
소프트웨어 통합	· 소프트웨어통합 결과보고서 · 소프트웨어개발 시험평가절차서 · 소프트웨어통합 계획서
소프트웨어 개발시험평가	· 소프트웨어 개발시험평가 결과보고서 · 사용자문서
체계 통합	· 체계통합시험결과 보고서 · 체계개발시험평가 절차서
체계 개발시험평가	· 체계개발시험평가 결과보고서
소프트웨어 설치	· 소프트웨어설치 계획서 · 소프트웨어전이 계획서 · 펌웨어설치계획서
소프트웨어 인수지원	· 체계 인수시험 결과 보고서 · 계약서에 명시된 납품 산출물 · 교육 및 지원기록

3. 소프트웨어 품질보증

소프트웨어의 품질보증업무는 일반 하드웨어의 품질보증업무와 마찬가지로 단지 최종결과물의 검사와 시험에 의해서만 수행되는 것이 아니고 소프트웨어의 전 개발단계에 적용함으로서만이 가능하다. 물론 소프트웨어의 결함을 제거하고 해석하는 일이 소프트웨어의 품질보증을 위해서 중요한 기능이기는 하지만 보다 중요한 것은 사전에 이들 결함이 발생되지 않도록 예방하는 일이라 할 수 있다.

과거의 소프트웨어 품질보증 프로그램은 시험계획, 시험절차, 시험범주, 시험형식, 시험방법에 대한 규격서와 같은 시험프로그램이 전부였다. 이와 같이 시험에 국한된 소프트웨어 품질보증 프로그램의 취약점은 시험을 통해서는 단지 소프트웨어 제품의 품질을 제한적으로 확인할 수는 있어도 결코 과정 중에 품질을 만들어 넣지는 못한다는 것이다.

기본적으로 소프트웨어 품질보증 프로세스와 그 관리 정도는 소프트웨어 제품의 품질에 대한 판단기준에 따라 결정된다. 구체적인 목표가 없는 프로세스는 결코 목적한 곳에 도달할 수 없다. 물론 특정된 소프트웨어 품질보증 기법은 프로세스상의 방법과 기준에 따라 적용할 수 있다.

예를 들면 전체 시스템에 대한 소프트웨어 결함의 영향도를 평가하는데 위험도 분석(Risk Analysis)을 이용할 수 있으며 위험도와 그 영향도가 클수록 소프트웨어 품질보증 효과가 나타나게 된다.

소프트웨어 품질보증이란 요구되어지는 소프트웨어 품질을 성취하기 위해서 계획되어 지고 체계화된 모든 활동을 의미하는 것이며 따라서 품질보증 활동은 materials, 데이터, supplies 또는 서비스 등이 기확립된 기술적인 요구사항과 일치하는 지를 확인케 하여 주며 또한 소프트웨어가 만족스럽게 수행되는 지를 확인케 한다.

소프트웨어 품질보증프로그램은 조직의 목표, 특성 및 업무에 따라 수립되어야 한다.

소프트웨어를 개발하는 조직은 소프트웨어 제품을 이용하는 조직보다는 설계와 시험과정에 대해 더 관심을 기울여야 한다. 또한 후자 조직은 형상관리(Configuration Management) 및 코드관리, 합부시험 및 구매 행위에 중점을 두어야 한다.

따라서 소프트웨어 품질보증의 본질은 프로그램의 결점(Defects)을 사전에 방지하고 그들이 발견되면 제거함으로써 소프트웨어의 사용도(Usability)와 보전도(Maintainability)를 높이는데 있다.

소프트웨어 품질보증 계획은 최종결과물 검사(Inspection)와 검증(Testing)은 물론 전 소프트웨어 개발과정에 대하여 적용되어지는 것이다. 소프트웨어 품질보증의 목적이 결점의 제거와 그 분석이지만 더 중요한 것은 결점의 방지에 있는 것이다.

과거에는 소프트웨어 품질보증계획이라 하면 검증계획, 검증절차서, 검증의 범위, 검증 종류 및 검증방법 등에 대한 규격서(Specification)와 같은 검증계획(Test Program)을 일컬었다. 이와 같이 검증에 중점을 둔 소프트웨어 품질보증계획의 주요 단점은 품질은 소프트웨어 결과물로서 검증되어 지는 것이 아니라는 점이다. 즉, 품질은 소프트웨어 결과물로 만들어지면서 결정되는 것이다.

소프트웨어의 품질을 판단하기 위해서는 사용하는 품질기준에 따라 소프트웨어 품질 보증계획의 방향이 결정된다. 따라서 각 조직은 그들의 업무(Activities)에 적용시킬 소프트웨어 품질보증 계획을 작성하여야 한다. 소프트웨어를 개발하는 조직은 소프트웨어를 사용해야 하는 조직보다 설계와 검증과정(Testing Process)에 관심을 가져야 하며, 소프트웨어 사용 조직은 Configuration Management, Acceptance Testing 및 구매업무에 중점을 두어야 한다.

3.1 소프트웨어 품질과 하드웨어 품질

소프트웨어와 하드웨어를 비교하면 다음 표와 같다.

표2. 소프트웨어와 하드웨어의 비교

구 분	소프트웨어	하드웨어
수리	새로운 소프트웨어의 개발	원상태로의 복귀
고장	고장 징후 (경고) 없음	고장 징후 (경고) 감지 가능
고장원인	설계상의 잘못	설계, 생산, 운영 등
표준화	표준화 불가능	표준화 가능
시험	무한한 시험	소모적 시험 가능
	물리적 측정 불가능	물리적 측정 가능(초음파시험, 재료시험 등)
신뢰도	Redundancy에 의한 향상 불가능	Redundancy에 의한 향상 가능

표2의 내용으로 볼 때, 소프트웨어의 품질은 개발 과정을 통하여 소프트웨어에 반영되어야 한다는 것을 알 수 있다.

가. 소프트웨어 품질의 계량화 방법

소프트웨어 품질보증 프로그램을 확립하기 위해서는 소프트웨어의 품질에 대한 정의가 있어야 한다. 소프트웨어의 품질은 측정 가능할 때만 성취할 수 있고 품질이 정의되었을 경우에 만 측정 가능하다.

품질의 측정은 두 가지의 측면에서 고려할 수 있는데 그 하나는 품질코스트(Quality Cost)의 측면이고 다른 하나는 요구사항(Requirements)에의 적합성(Conform-trance)의 측면이다.

(1) 품질비용 측면

품질비용(Quality Cost)은 품질관리에 드는 비용의 총칭을 의미한다.

즉 소프트웨어 제품이 사용자가 쓰는데 문제가 없다는(Defect-free) 것을 보증하는데 드는 비용을 의미하며 예방비용, 평가비용, 실패비용 등으로 구분한다.

예방비용은 소프트웨어를 개발하기 전에 에러를 방지하기 위하여 드는 비용이며, 평가비용은 개발중인 소프트웨어가 사용자의 요구사항과 일치하는 지를 확인하는데 드는 비용이며, 테스트, 검토 등이 이에 해당한다. 여기서 테스트와 검토를 정의한다면 테스트(test)는 Sample data 또는 실제 data를 가지고 소프트웨어 제품을 검사하는 것이며 검토(Review)는 테스트를 할 수 없는 경우 규격서를 서면으로 검사하는 것을 말한다.

실패비용은 개발 완료된 소프트웨어를 사용하고 유지 보수하는 단계에서 재개발(Rework)을 한다거나 또는 결점을 보완하기 위하여 수정 및 보수하는데 드는 비용을 의미한다.

표 3은 예방비용, 평가비용 그리고 실패비용의 요소를 나타내었으며 그림2에서 비용 간의 관계는 예방비용을 증가시키면 평가비용과 실패 비용이 감소하며 전체 품질비용이 감소하는 것으로 나타나 있다. 이 예는 소프트웨어 개발의 초기부터 품질보증 적용의 필요성을 강조하는 내용이다.

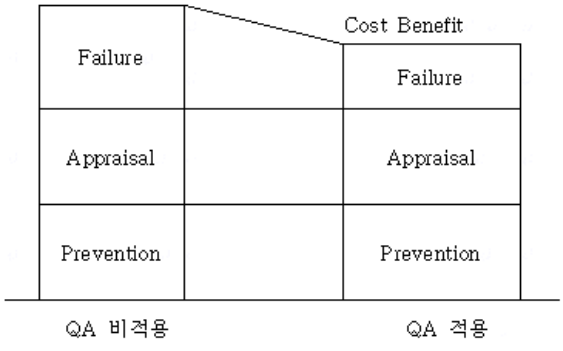

그림 2. 품질코스트에 대한 QA의 역할

표3. 소프트웨어 품질비용 요소

Prevention Cost	Consulting, Quality planning, 프로젝트 개발 방법 설정, 데이터베이스 계획, Standards와 요구사항의 정의, SQAP의 작성, 프로젝트 품질계획, 개발환경의 확립
Appraisal Cost	검토계획 작성, V/V, Audit, Inspection, Review, 점검 및 기술검토 Acceptance Test.
Failure Cost	Project rework, Malfunction, Maintenance Repair, Redundancy, Damage Claim, Retest Error Analysis

(2) 품질적합성(Conformance) 측면

품질에 대한 관점 차이로서 개발자의 관점은 프로그래밍 규약에 따라 프로그램을 올바로 제작하는 것이고 사용자의 관점은 사용하기 쉬우며 응답시간이 빠른 것으로 말할 수 있으며 운영자의 경우는 확실한 문서화(Documentation)와 이해 용이한 소스 프로그램으로 소프트웨어의 품질을 말하기도 한다.

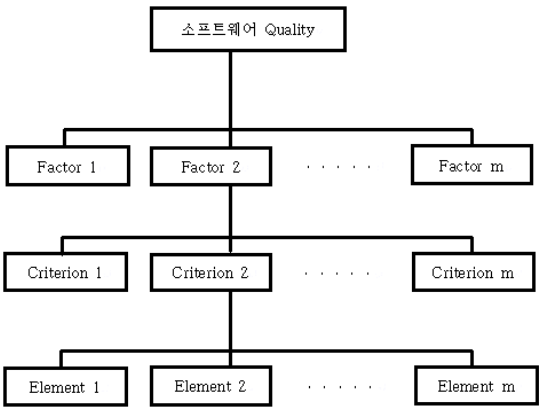

그림 3. 소프트웨어 품질측정의 체계

여기서는 이러한 관점들을 통합하여 그림 3과 같은 소프트웨어 품질계량 모델(SQM 소프트웨어 Quality Metric Model)을, (표 4 및 5)에 나타난 바와 같이 13개의 품질요건(Factors)과 26개의 품질기준(Criteria)을 고려할 수 있다. 이 품질기준은 SW를 개발하는 과정(Process)으로부터 유도된 것이 아니라 소프트웨어의 제품의 속성을 의미한다. 각각의 기준(Criteria)의 측정은 요소(Element)에 의하여 측정 가능하며 이 요소의 측정값을 metric이라고 정의한다.

표4. 소프트웨어 품질 요소(factor)

정 확 성 (Correctness)	소프트웨어의 목적을 달성하는가?
신 뢰 성 (Reliability)	프로그램이 의도된 기능을 항상 정확히 수행하는가?
효 율 성 (Efficiency)	시간이나 자료를 낭비하지 않는가?
순 수 성 (Integrity)	시스템 및 데이터의 보호기능이 있는가?
재 사 용 성 (Reusability)	일반적인 독립모듈로 구성되어 있는가?
사용의 편의성 (Usability)	사용하기 쉬운가?
유지보수성 (Maintainability)	새로운 환경에서 용이하게 수정되는가?
테스트 용이성 (Testability)	테스트가 가능한가?
범 용 성 (Potability)	타 기종에서도 수행되는가?
상호작동성 (Interoperability)	타 시스템과 연계성이 있는가?
모듈 통신성 (Intraoperability)	구성 프로그램 모듈간에 통신이 잘되는가?
유 연 성 (Flexibility)	수정이 가능한가?
생 존 성 (survivability)	시스템의 어떤 부분이 동작 불가능할 때 중요기능이 수행되는가?

예를 들어서 완전성(Completeness)은 프로그램의 전 부분이 완성되었는가를 나타내는 척도로서 그 요소(Elements)는

- 각 의사결정 분기점(Decision Point)의 모든 조건과 처리형태가 정의되었는가의 여부
- 각 내부데이터 구조가 정의되었는가의 여부
- 각 입력 조건 값이 정의되었는가의 여부

등으로 나타내어 평가할 수 있다는 것이다. 수백개의 이러한 요소가 26개의 기준과 연관되어 있고 또한 한 개의 기준에 대하여 여러 개의 요소들이 소프트웨어적용분야, 개발단계, 요소(Factors) 등에 의해 변화한다. 이러한 요건과 기준의 관계가 표6.에 나타나 있으며 기준에 대한. 요소의 내용들을 점검표로 작성하여 소프트웨어의 품질을 측정할 수 있을 것이다.

표5. 소프트웨어 품질 기준(criteria)

접근 감시성 (Access Audit)	일반성 (Generality)
접근관리성 (Access control)	장치독립성 (Hardware Independence)
정확성 (Accuracy)	기계사용성 (Modularity)
자율성 (Autonomy)	모듈성 (Modularity)
통신보편성 (Communication commonality)	작동성 (Operability)
통신용이성 (Communicativeness)	재구성 용이성 (Reconfigurability)
완벽성 (Completeness)	견고성 (Robustness)
간결성 (Conciseness)	자기 기술성 (Self-Descriptiveness)
일관성 (Consistency)	단순성 (Simplicity)
데이터 보편성 (Data commonality)	독립성 (SW System Independence)
에러 허용성 (Error tolerance)	저장 효율성 (Storage Efficiency)
실행효율성 (Execution efficiency)	추적 용이성 (Traceability)
확장성 (Expandibility)	훈련성 (Training)

표6. 품질요건(Factor)과 품질기준(Criteria)의 관계

품질요건	품질기준
정확성	추적용이성, 완전성, 일관성
신뢰성	일관성, 정확성, 단순성, 에러허용성
효율성	실행효율성, 접근관리성
순수성	접근감시성, 접근관리성
재사용성	일반성, 모듈성, 소프트웨어 시스템 독립성
사용 편의성	훈련용이성, 통신용이성, 작동성
유지보수성	일관성, 단순성, 간결성, 모듈성, 자기기술성
시험 용이성	모듈성, 단순성, 자기기술성, 기계사용성
범용성	모듈성, 자기기술성, 장치독립성
상호작동성	모듈성, 통신보편성, 데이터보편성
모듈통신성	모듈성, 통신보편성, 데이터보편성
생존성	에러허용성, 모듈성, 견고성, 재구성용이성, 자율성

3.2 정부 소프트웨어 품질보증 업무

국방사업에서의 소프트웨어 품질보증은 국발품질관리소의 소프트웨어 품질보증업무 지침(잠정),(세부지침(Q-지시-77)}을 통하여 살펴본다.

가. 정부 소프트웨어 품질보증 업무의 목적

소프트웨어 품질보증 업무 지침(이하 국방품보지침)에서 국방품질보증활동의 목적을 체계를 만족할 수 있도록 개발되고 있는지 기술적, 업무적으로 검토(Review)하여 개발과정에서 발생하는 산출물(소프트웨어 기술자료, 소스코드 포함)이 품질요구조건에 일치성을 보장하는 것으로 정의하고 있다. 또한 구체적인 품질보증 활동 방법으로

(1) 개발단계별로 해당 산출물에 대해 체계가 올바르게 구현되고 있는지를 검토 (Review), 검증(Verification), 확인(Validation), 시험 등 활동
(2) 소프트웨어 개발계획서에 의거 검토 작성된 소프트웨어 품질보증활동 수행계획서에 따라 수행
(3) 소프트웨어 정보체계를 통한 기술문서 검증 및 소스코드 등의 시험실시

을 제시하고 있다.

나. 성공적 품보활동을 위한 준비사항

(1) 소프트웨어 개발계획서에 의한 개발활동 확인(Validation)
(2) 소프트웨어 품질보증 계획서 작성

을 요구하고 있다

다. 소프트웨어 개발단계에 따른 업무

국방품보 지침은 소프트웨어 개발 단계에 따른 품질보증 조직의 업무를 다음과 같이 정의하고 있다.

(1) 국방용 소프트웨어의 개발계획서 검토시 다음 분야에 대한 확인을 요구하고 있다.
  (가) 소프트웨어 개발에 대한 일반적인 수행계획: 소프트웨어 개발공정, 소프트웨어 개발방법, 소프트웨어 산출물에 대한 절차, 재사용이 가능한 소프트웨어, 중요 요구조건 취급 및 하드웨어 자원의 이용 등
  (나) 소프트웨어 개발 세부계획: 개발계획 수립 및 감독, 소프트웨어 개발환경의 설정, 시스템 요구사항 분석, 시스템 설계, 소프트웨어 요구사항 분석, 소프트웨어 설계, 소프트웨어 구현 및 단위시험, 소프트웨어 단위 통합 및 시험, 소프트웨어 형상품목 수락시험, 체계통합 및 시험, 시스템 수락시험, 소프트웨어의 사용준비, 소프트웨어의 인계준비, 소프트웨어 형상관리, 소프트웨어 산출물 평가, 소프트웨어 품질보증 등
  (다) 일정계획 및 업무체계
  (라) 개발조직 및 자원
(2) 소프트웨어 개발계획서에 의한 소프트웨어 요구사항 분석단계
  (가) 식별된 소프트웨어 형상품목별 사용자 요구사항을 소프트웨어 기술적인 측면에서 구체화하고, 각 요구사항별로 정의, 식별 및 비교 분석하여 대상체계가 달성하여야 할 기능 및 성능 요구사항을 명확히 하며, 소프트웨어 형상품목의 요구사항과 시스템 요구사항 간의 추적성 등을 정의하는 소프트웨어 요구사항 분석업무
  (나) 소프트웨어 요구사항명세서, 인터페이스 요구사항명세서 등에 대하여 각 요구사항 식별, 비교분석, 검증 활동 수행

(3) 소프트웨어 설계단계 단계
(가) 설계단계에서 요구사항을 맞족시키기 위해 다음과 같이 수행
1) 소프트웨어 형상품목 수준의 설계결정
2) 소프트웨어 형상품목 구조설계
3) 소프트웨어 형상품목 상세설계
등 소프트웨어 형상품목을 구성하는 소프트웨어 단위의 선정과 설계에 영향을 주는 결정사항 등을 정의하고, 소프트웨어 단위, 소프트웨어 단위간의 인터페이스 및 소프트웨어 단위간의 실행개념 식별과 각각의 소프트웨어단위에 대한 세부 명세 등의 타당성, 추적성 등이 부여되어 있는지 확인
(나) 소프트웨어 설계명세서, 인터페이스설계명세서, 데이터베이스 설계명세서 등에 대하여 요구사항에 대한 설계결정사항, 소프트웨어 단위간의 실행개념, 세부명세 등 비교분석, 검증 및 확인 활동을 수행
(4) 소프트웨어 구현 및 시험단계
(가) 소프트웨어 구현 및 시험단계는 소프트웨어 설계단계에서 만들어진 설계 표현들이 컴퓨터에 의해 실행될 수 있도록 프로그램 개발언어로 변환하는 구현업무와 컴퓨터 명령어와 데이터 정의의 코딩, 데이터베이스의 구축, 데이터 값의 입력, 설계 구현에 필요한 기타 활동 등을 말한다.
(나) 소프트웨어 단위 또는 컴포넌트, 형상품목 등에 따른 각각의 단위 소프트웨어를 시험하기 위하여 테스트 케이스(입력자료, 예상되는 결과 및 평가기준 등), 시험절차 및 시험데이터를 설정하여야 하며, 절차에 따라 시험을 수행하여야 한다.
(다) 필요시 소프트웨어 단위, 컴포넌트 시험을 수행할 수 있으며, 또한 각 소프트웨어 기술자료와의 추적성 등을 확인할 수 도 있다
(라) 소프트웨어 시험명세서에 따라 소프트웨어 형상품목 시험을 수행하여야 하며, 형상품목 시험시 요구사항을 충족하지 못하거나, 불만족 사항이 발견된 경우 수정보완후 재시험을 시행
(5) 체계통합 및 시험단계
: 체계통합시험에 참여하여 다음활동 실시
(가) 시험결과 확인
(나) 요구사항을 충족하지 못하거나, 불만족 사항 발견시 수정/보완 후 재시험 시행
(6) 기술시험 및 운영시험 단계 단계에서 품질조직은 다음사항 확인업무를 수행한다.
(가) 기술 및 운영 시험평가시 소프트웨어 변경사항
(나) 시험결과에 따른 산출물의 변경사항

4. 결 론

일반적으로 소프트웨어 제품도 하드웨어 제품과 마찬가지로 그 생성과정을 보면 개발, 설계, 생산 및 설치라는 일련의 과정을 거치며, 궁극적으로는 일반적으로 널리 알려진 하드웨어의 개발 프로세스와 각종 품질보증 요건들이 개념적 차원에서는 소프트웨어에 대해서도 적용될 수 있다.

그러나 항공용 소프트웨어들은 항공기 및 관련 장비의 내부에 장착되고, 비행과 관련하여 반응시간이 종속적인 실시간 처리가 요구되며, 고온, 저온 또는 다습한 환경 등 극한 상황에서도 계속 동작하도록 요구된다. 따라서, 소프트웨어의 오동작은 시스템의 성능뿐만 아니라 항공기의 신뢰성과 인간의 안전에도 큰 영향을 주게 된다. 따라서, 항공용 소프트웨어의 개발 프로세스의 수립은 매우 중요하고 복잡한 문제라 할 수 있다. 앞에서 살펴본 국방 소프트웨어 개발 프로세스 기준과 소프트웨어 정부품질보증 활동 기준은 우리 연구원에서 국방사업관련 소프트웨어 개발시 적용가능한 개발 프로세스와 품질보증 프로세스의 구성 및 수립에 활용될 수 있을 것으로 판단된다.

참고문헌

1. 소프트웨어 개발 프로세스 (방위사업청 지침 2006-9, 2006.1.25)
2. 소프트웨어 품질보증업무 지침(잠정), 세부지침(Q-지시-77), 국방품질관리소