□기술해설□

ISO 9001 Compliant 객체지향 소프트웨어 품질시스템

숭실대학교 김수동*

목 차

1. 서 론

1.1 ISO 소프트웨어 품질 표준

정보처리 기술의 발전과 함께 소프트웨어의 종류와 적용 범주가 크게 늘어나고, 그 결과 소프트웨어는 점차 대형화 복잡화되어 지고 있다. 빠른 속도로 발전하는 하드웨어 기술을 따라가지 못하고 상대적으로 낙후되어 있는 소프트웨어의 기술 수준을 가르켜 소프트웨어 위기(Software Crisis)라 일컫는데, 이런 문제의 주된 현상으로 소프트웨어의 생산성과 품질이 낮은 것을 들 수 있다. 소프트웨어의 품질을 보증하기 위한 대표적인 국제적 표준으로 ISO (International Organization for Standardization) 기구의 ISO 9001 [1] 과 ISO 9000-3 [2] 품질 기준을 들 수 있다. ITU 시대를 맞아 소프트웨어의 시장이 국제화되면서 ISO 품질 인증의 중요성이 더 해 가고 있으며, 우리나라에서도 대기업들을 중심으로 ISO 인증을 획득했거나 인증을 활발히 추진중에 있다.

1.2 객체지향 프로그래밍 기술

소프트웨어 분야의 혁명적 (Revolutionary) 기술이라 불리우는 객체지향 프로그래밍 기술

*종신회원

은 소프트웨어 생산성 향상과 효율적인 유지보수 및 효과적인 재사용을 지원하는 새로운 기술이다. 객체지향 프로그래밍은 구현 과정에만 적용되는 기술이 아니며, 자연스런 모델링(Natural Modeling)을 통해 도메인 분석 및 설계 과정에도 효과적으로 사용되는 소프트웨어 개발 방법론으로서, 소프트웨어 생명주기의 전 단계에 걸쳐 적용되는 기술이다 [3].

객체지향 기술의 핵심 개념은 객체 (Object), 클래스 (Class), 속성상속 (Inheritance), 다형성 (Polymorphism), 제네릭 클래스 (Generic Class)가 있으며, 이 개념은 객체지향 언어, 객체지향 분석 및 설계 방법, 객체지향 데이터베이스등 여러 객체지향 분야에 공통적으로 적용된다 [4]. 객체지향 소프트웨어 개발 과정과 직접 관련이 있는 객체지향 개발 방법론은 대부분 분석과 설계 방법들로 구성되어 있어 ISO 기준의 생명주기에 비해 좁은 범주의 방법론이라 할 수 있다.

알려져 있는 수십개의 객체지향 방법론들은 사용하는 용어들이 다소 차이가 있으나 그 절차나 방법들이 유사하다. 이들 중, 미국 General Electric 연구소에서 개발된 Object Modeling Technique (OMT) 방법론[5] 이 가장 널리 사용되고 있으며 [6], 이 방법론은 이론적 바탕도 상당히 정립되어 있어 본 논문에서 제안하는 객체지향 ISO 품질 시스템의 표준 방법론으로 채택한다.

1.3 논문의 Scope 및 구성

ISO 품질 기준과 객체지향 기술은 모두 소프트웨어 개발의 생산성을 향상시키고, 소프트웨어 운용 과정에서 효율성을 증진하며, 보다 용이하고 체계적인 유지보수를 통하여 근본적으로 소프트웨어 위기의 문제를 해결하는데 목적이 있다. 그러나, 그 목적을 달성하는 과정과 방법은 상이한 데, ISO 품질 기준은 소프트웨어 생명주기의 전 과정에 걸친 표준화와 품질 평가를 통하여, 객체지향 기술은 보다 자연스럽고 효율적인 소프트웨어 개발 방법론과 효과적인 재사용을 통하여 그 목적을 달성한다. 따라서, 이 ISO 품질 기준과 객체지향 기술의 적합한 접목을 통하여 보다 실효성 있는 소프트웨어 개발 및 운용 관리 장치를 마련할 수 있으며, 이것이 본 논문의 주된 목적이다.

본 논문에서는 최근 새로운 소프트웨어 개발 방법론으로 널리 활용되고 있는 객체지향 방법론을 기반으로 하여, 소프트웨어 개발에 효과적으로 적용할 수 있는 객체지향 ISO (이하 OO-ISO) 품질시스템을 제안한다. ISO 품질 기준은 품질 보증의 대상, 항목, 단계등 'WHAT' 부분만 명시하고 있으므로, 일련의 원칙과 철학에 맞게 ISO 품질 기준을 구체화 세분화하여야 인증도 가능하고 원래 목적인 품질 보증의 실효를 거둘 수 있다. 본 논문에서는 그림 1 처럼 객체지향 개발 방법론, User Interface 방법론, 점진적 개발 주기의 원칙에 입각하여 ISO 9001 기준을 구체화하는 과정을 설명하며, 그 결과인 OO-ISO 품질 시스템에 대하여 설명한다.

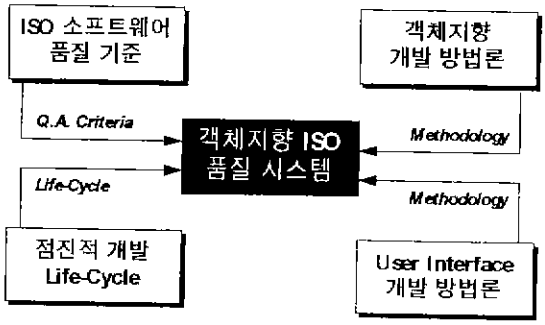

그림 1 객체지향 ISO 품질 시스템의 Design Principles

본 논문의 구성은 2장에서 ISO 9001과 9000-3 품질 기준의 범위와 한계등을 살펴보며, 3장에서는 객체지향 개발 방법론에 기반을 둔 OO-ISO 품질 시스템 개발의 원칙과 요구사항을 정의하며, 4장에서는 OO-ISO 품질시스템의 생명주기 활동등 전체적 Framework을 설명한다. 5장에서는 OO-ISO 품질 시스템 각 단계의 작업(Task)들에 관한 구체적인 정의를 하며, 6장에서는 ISO 기준에서 요구하는 품질 보증 관련 절차서와 지침서들을 정의하며, 7장에서는 ISO 기준 문서와 OO-ISO 품질 시스템과의 연관성(Cross-reference)을 관찰하여 이 품질 시스템이 ISO 기준을 어떻게 만족하는가를 알아보며, 8장에서는 결론을 다룬다.

1.4 Related Work

ISO 소프트웨어 품질 보증과 객체지향 프로그래밍 기술은 최근 들어 소프트웨어 분야에서 매우 중요한 추세 및 기술로 인정되고 있다. ISO 품질 인증은 전 세계적으로 소프트웨어 산업계에서 반드시 획득하여야 하는 중요한 과정으로 인정이 되었으며, 객체지향 기술은 소프트웨어의 분석 과 설계 및 구현의 방법론 중심으로 빠른 속도로 발전해 가고 있다. 그러나, ISO 품질 시스템 구축에 객체지향 방법론을 적용한 객체지향 ISO 품질 보증 및 품질 시스템에 관한 연구와 보고는 거의 전무한 상태이다.

이는 객체지향 기술이 최근에 들어서야 활발히 보급되기 시작하였고, 객체지향 품질 측정과 품질 보증에 관한 연구가 아직도 성숙한 단계가 아니기 때문이다. 그러나, ISO 품질 인증이 소프트웨어 사업에 필수적 요구사항으로 인정되어 가며, 객체지향 방법론이 소프트웨어 산업계의 핵심 기술로 자리를 굳혀가는 추세를 미루어 보면, 앞으로 객체지향 ISO 품질보증에 관한 연구가 보다 활발히 진행되어 질 것으로 믿어진다.

2. ISO 품질 기준의 범위 및 한계

2.1 ISO 품질 기준의 범위

ISO 9001 품질 시스템은 설계, 개발, 생산, 설치 및 서비스 분야에 적용되는 산업 분야의 공통된 품질 기준이며 [1], ISO 9000-3은 이 ISO 9001을 소프트웨어의 개발, 인도, 및 유지보수에 적용할 수 있는 세부적인 지침을 제시하고 있다 [2]. 따라서, ISO 소프트웨어 품질 인증과 적용에는 이 ISO 9000-3 품질 기준이 실질적으로 널리 사용된다.

소프트웨어 품질 기준의 중심 문서인 ISO 9000-3은 제 5장의 생명주기 관련 업무 (Life-Cycle Activities)에 관한 기준과 제 6장의 지원 관련 업무 (Supporting Activities)가 주 내용이다. 이 문서의 소프트웨어 생명주기 관련 업무를 보면 그림 2에서 처럼 과제의 계약 단계에서 부터 유지보수 단계까지 포함하는 전체 생명주기를 다루는 포괄적인 내용이다. 그러나, 소프트웨어 개발 주기의 중요한 단계인 분석 단계가 별도로 정의되어 있지 않다. 따라서, ISO 9000-3문서 5. 6절의 설계(Design) 활동에 분석활동을 포함한 것으로 간주하여 품질 시스템을 개발하여야 한다.

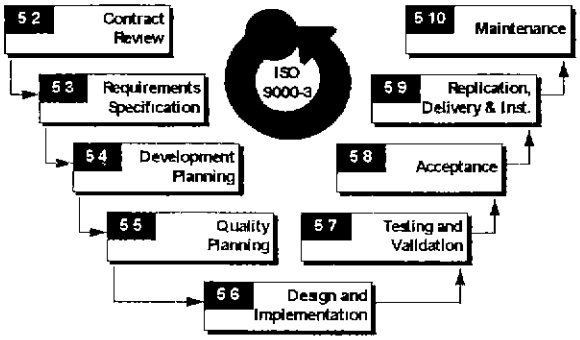

그림 2 ISO 9000-3의 생명주기 관련 업무

ISO 9000-3의 지원 관련 업무 범위도 그림 3에서 처럼 형상 관리, 문서 관리등 소프트웨어 개발 지원 업무에 관한 포괄적인 내용이다.

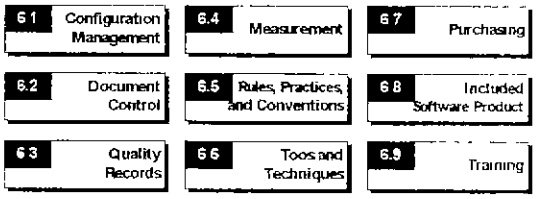

그림 3 ISO 900-3의 지원 관련 업무

따라서 이 ISO 9000-3 기준에 근거하여 개발 및 지원 업무의 표준이 마련되면, 중요한 요소들을 모두 포함하는 포괄적 품질 시스템을 마련할 수 있다.

2.2 ISO 품질 기준의 한계

ISO 9001 및 9000-3 기준은 품질 표준을 제정시 고려해야 할 항목들 (WHAT 부분) 을 명시하고 있으나, 각 항목들에 대한 절차, 방법 및 구체적인 내용 (HOW 부분) 은 포함하고 있지 않다. 따라서, ISO 기준을 그대로 실무에 적용하기는 어렵고, 상당 부분 실제적이며 세부적인

품질 보증 표준이 마련되어야 하는데 이 과정을 Tailoring Process라 한다. 예로서 ISO/IEC DIS 12207-1은 소프트웨어 생명주기 관련업무를 Tailoring한 표준이다 [7]. ISO 9001, 9000-3의 기준에서 Tailoring해야 할 몇 가지 중요한 항목들을 다음과 같다.

- 소프트웨어 생명주기(Lif-Cycle)
- 소프트웨어 개발 방법론(Methodology)
- 검증 및 확인(Verification and Validation)
- 품질 보증 계획(Quality Assurance Planning)
- 형상 관리등 생명주기 지원 관련 업무

ISO 기준에서 명시하고 있는 항목들은 상당히 포괄적이며 큰 범위의 업무들이므로, 이 항목들도 보다 구체적으로 세분화하거나 각 회사의 환경에 알맞게 수정될 수 있다. 따라서, ISO 기준은 품질 보증 대상 (WHAT 부분)과 각 항목별 수행 방법(HOW 부분)에서 Tailoring이 요구된다. 이러한 ISO 기준의 성격을 잘 표현할 말로 "Say what you do, Do what you say, and Prove it."이 있다.

ISO 품질 기준은 소프트웨어 개발의 생산성 및 품질과 관련이 높은 생명주기 모델이 제시되어 있지 않으므로 (ISO 9000-3, 5.1절), 각 회사나 조직의 목적과 환경에 적합한 생명주기를 정의해야 한다. 몇가지 대표적인 모델을 정의하여 각 과제에 최적의 생명 주기를 선택할 수 있도록 하여도 된다.

소프트웨어 개발 방법론은 도메인의 분석, 소프트웨어 설계 및 구현에 이르는 체계적인 절차와 도식 및 방법들로 이루어져 있는 매우 중요한 기술이다. 방법론도 각 회사나 조직 및 과제의 성격을 반영하여 정의해야 하는데, 개발 방법론이 소프트웨어 개발 생산성과 품질에 미치는 영향은 매우 크므로 신중히 정의되어야 한다.

개발 지원 업무에 있어서도 ISO 기준은 고려해야 할 품질 보증의 항목은 나열하고 있지만, 각 항목에 대한 절차, 방법등 세부 내용은 포함하고 있지 않다. 예를 들면, ISO 9000-3, 6. 1절에서 형상 관리등 주요 항목들에 대한 개괄적인 요구사항만 명시되어 있을 뿐, 형상관리의 방법, 절차, 표준등은 명시되어 있지 않아 Tailoring 과정을 거쳐야 한다.

2.3 ISO 품질 시스템의 Risk

개괄적인 ISO 기준을 효율적이며 적절하게 Tailoring 하지 못하면 소프트웨어 품질보증과 표준화의 원래 목표가 유명무실하게 되고 오히려 개발자들을 혼란시키거나 품질을 저해하는 경우가 흔히 발견된다. 실제로 ISO 품질 인증을 받은 기업에서 제정된 소프트웨어 품질 시스템이 개발에 실제적인 도움이 되지 않고 오히려 개발 절차와 방법론 적용에 혼란을 초래하고 불필요한 표준 문서화 작업의 요구로 인하여 생산성이 저하되는 사례가 국내에서도 여러번 관찰 되었다. 그 결과, ISO 품질 인증은 받았으나 품질 시스템이 실제로 품질의 보증에 도움이 되지 않고, 품질의 표준이 마련되어 있으나 표준으로 자리잡지 못하게 된다.

효율적인 품질 시스템이 되기 위해서는 각 회사나 조직의 환경에 따른 요구사항이 반영되어야 하며, 개발 과제에 가장 적합한 생명주기, 효과적인 개발 방법론, 순차적이며 구체적인 품질 보증 기준, 개발자 뿐만이 아니라 관리자도 활용할 수 있는 소프트웨어 개발과 관리의 종합적인 지침이 마련되어야 한다.

3. OO-ISO 품질 시스템의 Design Principles

효과적인 품질 시스템을 구축하기 위해서는 개발 Process, 개발 Paradigm등 여러 요구사항 및 원칙들이 정해져야 한다. 본 장에서는 객체지향 ISO 품질 시스템을 설계, 개발하기 위한 요구사항과 원칙을 다음과 같이 정의한다.

- 점진적 개발 생명주기 채택
- 객체지향 프로그래밍 패러다임 적용
- OOA/D 방법으로 Object Modeling Technique 지원
- User Interface 개발을 지원하는 통합 방법론

3.1 점진적 개발 생명 주기

소프트웨어 개발 과제의 대형화와 복잡화로 시스템의 요구사항 도출과 도메인 분석이 점차 어려워지고 있다. 이에 고객의 검토 의견(Feedback)을 개발 과정에 효과적으로 반영하여 보다 안정적인 개발을 지원하는 점진적 개발 (Incremental Development) 방식이 최근 들어 널리 활용되고 있다. 이 방식은 객체지향 개발 방법론에 가장 적합한 생명주기이기도 하므로 OO-ISO 품질 시스템이 이를 체계적으로 지원하여야 한다.

3.2 객체지향 프로그래밍 패러다임

OO-ISO 품질 시스템의 기본 개발 패러다임으로 객체지향 프로그래밍 방식을 선택한다. 즉, 객체지향 프로그래밍의 기본 개념인 객체, 클래스, 속성 상속, 추상클래스, 제네릭 클래스 등의 단위, 장치 및 개념에 근거하여, 문제 영역을 분석, 설계, 구현, 시험, 유지보수등을 수행하도록 OO-ISO 품질 시스템이 개발되어야 한다. 이는 분석과 설계의 과정도 객체지향 방법론에 의해 수행되어야 하며, 구현도 객체지향 언어나 도구들을 사용함을 의미한다. 객체지향 프로그래밍이나 방법론의 소개는 문헌 [4]를 참고로 하도록 한다.

3.3 Object Modeling Technique 방법론

OO-ISO 품질 시스템은 객체지향 분석 및 설계 방법론중에서 가장 많이 사용이 되며 연구가 활발히 진행되어 온 Object Modeling Technique (OMT) [5] 를 선택한다. 지난 수년간 중. 대형 소프트웨어 개발에 널리 활용되어 온 이 방법론은 적용 대상 도메인이 다른 방법론에 비해 상대적으로 넓고, 방법론 구성이 매우 체계적이어서 보편성과 적용성이 우수한 것으로 알려져 있다 [6]. OO-ISO 품질 시스템은 OMT 방법론의 모든 단계를 ISO 기준이 제시하는 단계와 작업에 매핑을 시켜야 하며 OMT 방법론의 방법과 절차들을 모두 지원하여야 한다. OMT 방법론에 관한 소개는 문헌 [5]를 참고로 하도록 한다.

3.4 User Interface 개발 방법론

그래픽 단말장치와 그래픽 처리기술의 발달로 그래픽 사용자 인터페이스가 일반화되면서, 보다 효율적이며 사용이 편리한 사용자 인터페이스의 개발이 전체 소프트웨어에 차지하는 비중이 점차 증가하고 있다. 따라서, 사용자 인터페이스를 소프트웨어의 프로세싱 부분과 병행하여 효과적으로 개발할 수 있는 방법론이 요구되고 있다.

따라서, 제안될 OO-ISO 품질 시스템에서 사용자 인터페이스의 개발 단계와 지침이 제공되어야 하며, 사용자 인터페이스 개발의 각 단계가 프로세싱 부분의 개발 단계와 논리적으로 잘 매핑이 되어야 한다.

3.5 기타 요구사항

위의 네 가지 요구사항이외에 제안될 품질 시스템은 매우 체계적이고 순차적인 작업 단계를 명시하여야 하며, 각 단계별 입. 출력, 주요 업무, 품질 기준등이 명확히 명시되어야 한다. 이렇게 개발된 품질 시스템은 개발자나 관리자가 실무에 매우 효율적으로 적용될 수 있기 때문이다. 또한, 소프트웨어 전 생명주기에 걸쳐 산출될 문서의 종류와 분량을 최소화하되, 각 문서들은 매우 핵심적인 중요한 정보를 최대한 표현하는 Minimal Documentation, Maximal Information의 원칙을 적용되도록 한다.

4. OO-ISO 품질 시스템의 Framework

ISO 기준을 Tailoring하여 OO-ISO 품질 시스템을 개발하기 위해서 우선 ISO 품질 기준 항목들에 대해 개발 방법론과의 관련도(Dependency)를 살펴보아야 한다. 표 1은 ISO 9000-3 제5장의 생명주기 관련 업무 항목에 대한 방법론과의 상관도를 나타내고 있다.

생명주기 관련 업무중에는 5.4-5.7절의 개발 계획, 품질 계획, 설계 및 구현, 시험 및 확인 업무들이 개발 방법론과 관련성이 높다. 예로서 구조적 개발 방법론에서의 설계 과정과 객체지향 방법론에서의 설계 과정은 매우 상이하므로, 5.6절의 설계 및 구현은 방법론에 따라 그 지침이 매우 달라진다.

표 2는 ISO 9000-3 문서 제6장의 지원 관련

업무와 방법론과의 관련도 나타내고 있다. 6.1, 6.4-6.6절의 업무들이 방법론과 높은 관련성이 있는 것으로 알려져 있다. 예로서 6.4절의 측정 업무는 개발 패러다임에 따라 측정 대상 및 방법이 매우 달라진다.

표 1 생명주기 관련 업무와 방법론의 관련도

ISO 9000-3 생명주기 관련 업무	방법론과의 관련도
5.1 General	Low
5.2 Contract Review	Low
5.3 Requirement Specification	Medium
5.4 Development Planning	High
5.5 Quality Planning	High
5.6 Design and Implementation	High
5.7 Testing and Validation	High
5.8 Acceptance	Medium
5.9 Replication, Delivery & Installation	Low
5.10 Maintenance	Medium

표 2 지원 관련 업무와 방법론과의 관련도

ISO 9000-3 지원 관련 업무	방법론과의 관련도
6.1 Configuration Management	High
6.2 Document Control	Medium
6.3 Quality Records	Low
6.4 Measurement	High
6.5 Rules, Practices and Conventions	High
6.6 Tools and Techniques	High
6.7 Purchasing	Low
6.8 Included Software Product	Low
6.9 Training	Medium

개발 방법론과 관련성이 높은 ISO 품질 기준 항목들을 중점적으로 Tailoring하여 개발된 OO-ISO 품질 시스템의 각 요소에 대하여 알아본다.

4.1 생명 주기 모델

점진적 개발을 지원하는 생명주기 모델로서 Spiral Model을 선택한다. 그림 4는 도메인 분석, 소프트웨어 설계, 프로그램 구현 및 시험 과정이 여러번 반복하면서 점진적으로 목표 시스템의 수준에 접근하는 Spiral Model의 주요 프로세스를 나타내고 있다 [8].

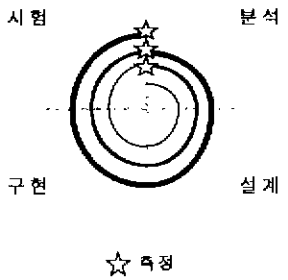

그림 4 점진적 개발을 지원하는 Spiral Model

이 모델에서는 분석-설계-구현-시험의 한 사이클이 진행된 다음의 중간 산출물에 대해 관련된 측정을 하여 완성도, 사용성, 효율성등의 평가를 실시한다. 이 모델을 적용하는데는 몇 번의 사이클이 반복되어야 하는지에 대한 체계적 논리적 결정과 각 사이클에 대한 시간 및 인력에 대한 비용 산출등 개발 계획의 기술을 필요로 한다.

4.2 객체지향 Spiral Model

사용자 인터페이스를 방법론 차원에서 지원하기 위하여 그림 4의 Spiral Model의 설계 단계를 User Interface 설계 과정과 프로세싱 설계 과정으로 나누어야 한다. 그림 5는 User Interface 개발 과정을 포함하고 객체지향 개발 패러다임을 기본으로 하는 확장된 Spiral Model의 개발 주기를 나타내고 있다. 이 모델은 개발 초기에 객체지향 분석을 개괄적으로 수행하고, User Interface 설계와 객체지향 설계 (OOD)를 병행 실시한 후, 객체지향 구현에서 단일 시스템으로 통합 개발되는 과정을 보여 주고 있다. 별 표로 표시된 부분이 Spiral Model의 한 사이클에서 시스템 완성도에 대한 측정 시점이다.

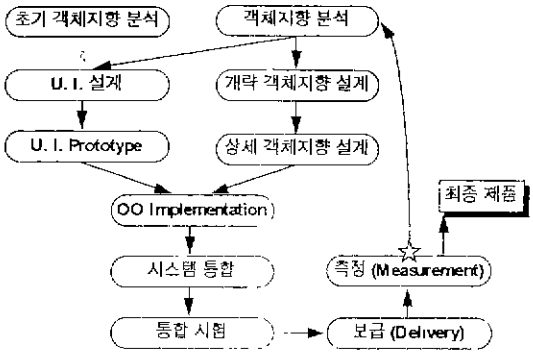

그림 5 객체지향 Spiral Model

4.3 OO-ISO 품질 시스템의 Life-cycle 업무

ISO 9000-3에서는 소프트웨어의 전체적인 생명주기를 다루고 있으므로, 그림 5의 개발 주기를 ISO의 기준에 매핑하여 전체적인 소프트웨어 생명주기를 완성하여야 한다. ISO의 5장에서 제시되어 있는 생명주기 관련 업무들과 절차에 근거하여 개발된 객체지향 ISO 표준 시스템은 생명주기 관련 업무들을 계획, 분석, 설계, 구현, 시험/확인, 및 후속업무 등의 6단계(Phases)로 분류한다. 그림 6은 이러한 단계와 각 단계별 세부 업무들을 나타내고 있다.

제안된 OO-ISO 품질 시스템의 생명주기는 그림 2의 ISO 9000-3의 생명주기 업무를 모두 포함하고 있으며, ISO 9000-3에는 별도의 나타나 있지 않는 분석 단계를 포함하고 있다. 따라서 OO-ISO 품질 시스템은 전부 6개의 단계(Phase)들로 구성되어 있으며 각 단계는 여러 개의 세부작업(Task)으로 세분화 되어 있다. 그림 6에서 점선으로 표시되어 있는 화살표는 구현 단계와 시험 및 확인 단계후 전 단계의 작업으로 돌아가 Spiral Model의 사이클을 반복할 수 있는 작업 흐름을 나타낸다.

OO-ISO 품질 시스템의 요구사항인 User Interface 개발 과정이 그림 6의 생명 주기에 세부적인 작업들로 포함되어 있는데 이는 A4-D1-D3-D4-I3-I4의 작업 과정이다. 이 User Interface 개발 과정은 프로세싱 부분인 객체지향 설계 과정과 병행되어 지며 적절한 단계에서 통합, 상호 참조하게 된다.

4.4 OO-ISO 품질 시스템의 Process Map

그림 6의 확장된 OO-ISO 생명주기에 근거하여, 각 작업에 대한 산출물, 작업의 절차와 수행 방법등을 정의하는 절차서와 지침서를 포함한 완성된 OO-ISO 품질 시스템의 개요를 그림 7이 보여 주고 있다. 각 작업을 나타내는 사각형에는 작업 번호 (ID)와 작업 명이 있으며, 그

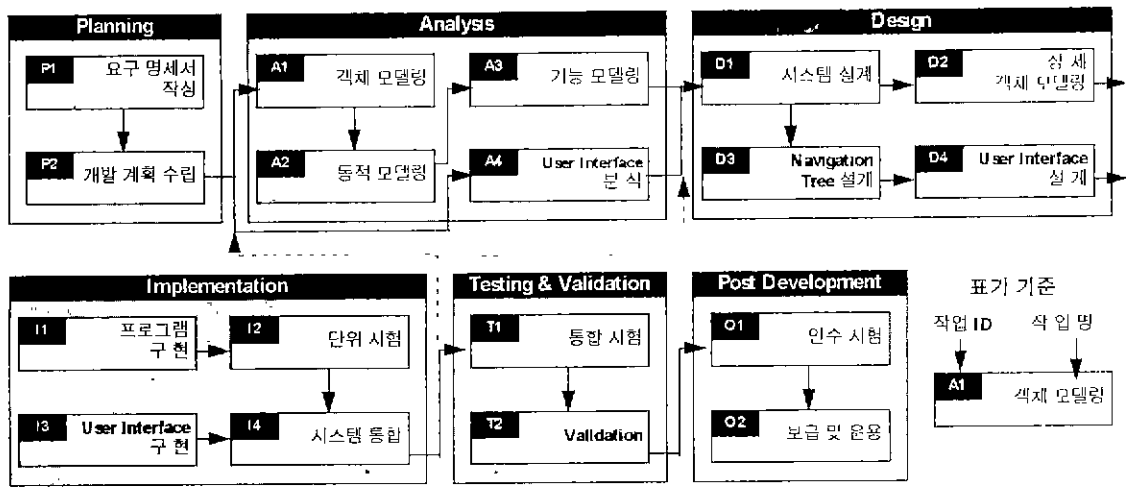

그림 6 OO-ISO 품질 시스템의 소프트웨어 생명주기

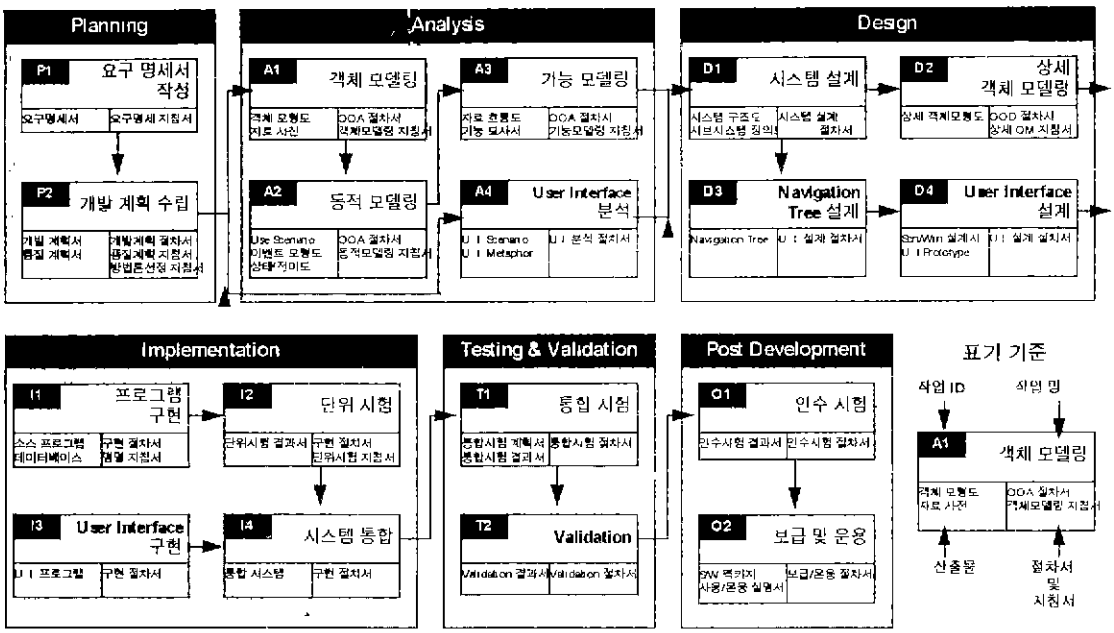

그림 7 완성된 OO-ISO 품질 시스템의 개요

아래에 산출물 리스트와 관련된 절차서와 지침서 리스트가 정의되어 있다. 산출물과 관련 절차서 및 지침서는 본문 3.5절의 원칙에 의하여 문서 작업과 기준서를 최소화하였다.

그림 7에 의하여 과제를 수행시, 과제별로 개발 주기와 작업 내용을 변형 적용할 수 있다. 예를 들어, 사용자 인터페이스가 중요시 되지 않는 시스템의 경우 관련된 작업들을 제외할 수 있으며, 시스템 도메인에서 데이터 흐름이나 계산등이 거의 없는 경우 A3의 기능 모델링을 생략해도 된다. 이렇게 표준 생명주기 모델을 수정 적용할 경우, 개발계획서와 품질계획서에 그 내용을 적절히 반영하여야 한다.

OO-ISO 품질 시스템의 작업별 구체적인 정의는 5장에서 다루고, 개발 및 품질 보증 절차서 및 지침서들을 6장에서 설명하도록 한다.

5. OO-ISO 품질 시스템의 Task 정의

본 장에서는 OO-ISO 품질 시스템 생명주기의 각 작업들을 상세히 정의한다. 각 작업은 그 목적, 작업내역, 입력 자료, 산출물, 개발 및 품질 기준 문서, ISO 9000-3의 관련된 절 번호등으로 정의된다. 각 작업은 Task Matrix로 다음의 각 절에서 설명되어 지며, 각 작업을 수행하는데 필요한 절차나 세부지침은 6장에서 다룬다. 품질 기준 문서는 ISO의 표준에 따라 절차서와 지침서로 구분되는데 절차서는 작업 순서나 작업 범위를 정의하며 지침서는 구체적인 작업 방법등을 제시한다.

5.1 Planning 단계

소프트웨어 개발 계획 단계는 요구 명세서 작성 (P1)과 개발 계획 수립 (P2)의 두 개의 작업으로 이루어 진다.

5.1.1 요구 명세서 작성 (P1)

이 작업은 개발에 앞서 소프트웨어의 기능적 요구 사항을 구체적으로 명시하는 작업인데, 주로 고객이 작성하지만 ISO 9000-3의 5.3.1에서도 허용된 것 처럼 공급자 (개발자)가 작성할 수도 있다. 고객이 작성하는 요구사항 명세서는 이 후 모든 개발 단계의 중요한 입력자료로 사용이 되지만 이 명세서를 완전한 것으로 가정

하고 개발하면 개발이 진행중에 많은 에러를 발생시킨다. 일반적으로 이 요구명세서는 그 내용이 불분명하고 (Ambitious), 불완전하며 (Incomplete), 불일치 (Inconsistent)한 부분이 많으므로 고객과의 인터뷰나 업무 조사등을 통하여 도메인을 보다 정확한 이해해야 한다.

작업 명	P1/요구 명세서 작성
목 적	• 목표 시스템에 대한 고객의 요구 사항을 명시
작업 내역	• 문제 도메인 대한 명시
	• 기능 및 성능에 관한 요구사항 명시 • 시스템 운영 및 과제의 제약사항 명시
입력 자료	• 목표 도메인의 업무 현황 • 기존 시스템 및 체계에 대한 현황 • 시스템 개발의 동기 및 목표
산 출 물	• 요구사항 명세서
품질 기준	• 요구 명세 지침서
9000-3 관련	• Section 5.3

작업 명	P2/개발 계획 수립
목 적	• 목표 시스템 개발에 필요한 일체의 세부 계획 수립 및 명시
작업 내역	• 과제의 정의, 목표 및 관련 사항 명시 • 수행 조직, 인력 및 자원 계획 • 개발 단계 정의 및 방법론 선정 • 과제 수행 일정 및 Task Matrix 작성 • 각 단계/작업별 품질 보증 계획 • 도구선정 등 기타 사항
입력 자료	• 요구사항 명세서 • 고객 인터뷰, 현업조사 등 도메인에 대한 추가 정보 • 사내 표준 시스템 및 개발 지침서 • 유사 시스템 개발 경험
산 출 물	• 개발 계획서
	• 품질 계획서
품질 기준	• 개발 계획 절차서 • 품질 계획 지침서 • 방법론 선정 지침서
9000-3 관련	• Section 5.4, 5.5

5.1.2 개발 계획 수립 (P2)

이 작업은 P1 작업에서 작성된 요구명세서를 중심으로 ISO 기준에서 요구하는 사항들에 대한 구체적인 개발 계획을 세우는 일이다. ISO 기준에서 요구하는 개발 계획서의 내용은 다음 표의 작업내역에 나열된데로 포괄적이다.

이 작업의 산출물로는 개발 계획서와 품질 계획서 두 가지가 있는데, 품질 계획서는 OO-ISO 생명주기의 각 단계에 관련된 품질 보증 활동에 대한 구체적 계획을 포함하는 것으로 품질 시스템을 수행하는데 매우 중요한 문서이다. 이 품질 계획서는 품질 계획 지침서에 따라 작성되어야 하나, 과제별 특성을 반영할 수 있다.

5.2 Analysis 단계

ISO 9001이나 9000-3의 생명주기 관련 업무에는 분석 단계가 별도로 정의되어 있지 않다. 9000-3 기준 5.6절의 설계 단계가 분석 작업을 포함하고 있는 것으로 간주할 수 있으며 따라서 이 5.6절에 명시된 내용을 참고하여 OO-ISO 품질 시스템을 개발한다. 또한, ISO 9000-3 관련 근거로는 ISO 기준에 분석 단계가 정의되어 있지 않으므로 5.6.2의 설계 작업을 부분적으로 관련이 있는 것으로 정의한다.

객체지향 분석 및 설계 기법인 OMT 방법론은 시스템 도메인을 세가지 측면에서 관찰하는데, 이는 시스템의 정적인 측면, 동적인 측면 및 기능적 측면이다. 따라서, 분석 단계는 객체 모델링, 동적모델링, 기능 모델링의 세가지에 사용자 인터페이스 분석을 포함한 네가지 작업으로 구성된다.

5.2.1 객체 모델링 (A1)

시스템의 정적인 측면을 객체지향 방식으로 관찰하는 과정이 객체 모델링이다. 객체지향 개발에서 이 객체 모델링 작업이 가장 중요한데, 이는 이 작업의 산출물인 객체 모형이 설계 및 구현 과정에 일관성있게 사용되기 때문이다.

작업 명	A1/객체 모델링
목 적	• 시스템의 정적인 정보를 추출, 명시
작업 내역	• 클래스를 추출, 자료 사전을 작성 • 클래스간의 관계 추출 • 각 클래스의 속성 추출 • 클래스간 상속 구조 정의 • 제약 사항 분석
입력 자료	• 요구 명세서 • 도메인 상식 및 경험
산 출 물	• 객체 모형도 (Object Diagram) • 자료 사전 (Data Dictionary)
품질 기준	• 객체지향 분석 절차서 • 객체 모델링 지침서
9000-3 관련	• 직접 관련 없음, 5.6.2와 부분 관련

5.2.2 동적 모델링 (A2)

OMT 방법론의 동적 모델링은 시간에 따라 변화하는 시스템 내부 객체들의 상태 변화와 객체들간의 이벤트를 관찰하는 과정이다. 사용 시나리오를 작성한 후, 각 시나리오별 이벤트 추적 모형을 작성한 후, 이 이벤트 추적 모형을 이용하여 주요 클래스에 대한 상태전이도를 작성한다. 따라서, 이 상태전이도를 보면 해당 클래스의 객체들에 대한 동적인 행위(Behavior)를 상세히 알 수 있다. OMT 방법론은 사용 시나리오를 분석 단계에 속하는 동적 모델링에 반영함으로서 고객이 원하는 시스템을 개발하도록 하는 장점이 있다. 즉 Validation의 활동이 개발 초기에 병행되어 지는 것이다.

작업 명	A2/동적 모델링
목 적	• 시스템의 동적인 정보를 추출, 명시
작업 내역	• 사용 시나리오 작성 • 이벤트 모형도 작성 • 상태 전이도 작성
입력 자료	• 요구 명세서 • 객체 모형도
산 출 물	• 사용 시나리오 • 이벤트 모형도 • 상태 전이도
품질 기준	• 객체지향 분석 절차서 • 동적 모델링 지침서
9000-3 관련	• 직접 관련 없음, 5.6.2와 부분 관련

5.2.3 기능 모델링 (A3)

OMT 방법론의 기능 모델링은 시스템의 각 함수가 수행될 때 입력과 출력이 되는 데이터들을 규명하고, 각 함수의 수행 알고리듬을 개략적으로 정의한다. 세부적인 알고리듬 정의는 D2 의 상세 객체 모델링 단계에서 수행한다. 이 모델은 데이터 입출력이나 계산이 많이 일어나는 시스템에 매우 중요한 모델이다.

작업 명	A3/기능 모델링
목 적	• 시스템의 기능 요구사항을 명시
작업 내역	• 시스템 외부 I/O 분석 • 여러 계층의 자료 흐름도 작성 • Process 별 Function Description 작성
입력 자료	• 요구 명세서 • 객체 모형도
산 출 물	• 자료 흐름도 • Function Description
품질 기준	• 객체지향 분석 절차서 • 기능 모델링 지침서
9000-3 관련	• 직접 관련 없음, 5.6.2와 부분 관련

5.2.4 User Interface 분석 (A4)

User Interface 개발 방법론에 관한 연구는 최근 들어 활발해 지기 시작했으나, 프로세싱 부분의 개발 과정과 User Interface 개발 과정을 논리적인 매끄러운 통합 방법론의 연구는 부족한 상태이다. 최근들어 객체지향 User Interface 개발이[9] 보편화되고 있으며, 프로세싱 부분의 방법론과 통합된 방법론의 요구가 높아지고 있다. OO-ISO 품질 시스템에서는 User Interface 개발의 주기를 전체 개발 주기에 논리적으로 포함하고 있으나, 특정 User Interface 개발 패러다임을 지정하고 있지는 않다. 따라서, OO-ISO 품질시스템의 User Interface 개발 절차와 작업 내역을 지원하는 어떤 방법론도 사용될 수 있다.

User Interface 분석 작업에서는 사용자의 시스템 사용 행위를 U. I. 시나리오로 표현하며, 또한 목표 시스템에 최적의 U. I. Metaphore를 정의한다. 대표적인 U. I. Metaphore로는 책, 서류철(Folder) 등이 있다.

작업 명	A4/User Interface 분석
목 적	• 시스템의 효율적인 U. I. 모델을 분석
작업 내역	• U. I. 에 관한 추가 요구사항 분석 • U. I. Scenario 작성 • 최적의 U. I. Metaphore 정의
입력 자료	• 요구 명세서 • U. I. 에 관한 추가 요구사항
산 출 물	• U. I. Scenario • U. I. Metaphore
품질 기준	• U. I. 분석 절차서
9000-3 관련	• 직접 관련 없음, 5.6.2와 부분 관련

5.3 Design 단계

5.3.1 시스템 설계 (D1)

소프트웨어의 대형화 복잡화 추세로 잘 정의된 소프트웨어 구조의 유용성이 점차 인식되어 지고 있다. 시스템 설계 작업은 주요 계층을 정의하고, 각 계층내 구성 컴포넌트들과 계층간 표준 인터페이스를 정의하는 일이다. 또한, 전체 시스템을 서브 시스템의 단위로 나누는 작업도 수행되어져, 서브 시스템들이 복수 개발팀에 의해 동시에 개발되게 한다.

이 작업을 수행하는데 일관성있고 체계적인 표준 시스템 구조가 마련되어 있으면, 시스템들 간의 이식성, 호환성 등이 우수한 개방형 시스템 개발이 가능해 진다. 대표적인 객체지향 분산 시스템 구조로 Object Management Group (OMG)의 CORBA [10] 구조가 있는데, 이미 이 구조를 기반으로 개발된 상용 소프트웨어들이 다수 개발 운용 중이다.

작업 명	D1/시스템 설계
목 적	• 시스템의 계층과 각 계층의 주요 컴포넌트들을 논리적으로 규명하여 시스템의 구조를 정의
작업 내역	• 시스템 구조상 제약 조건 분석 • 최적의 계층 및 컴포넌트 구조 설계
입력 자료	• 요구 명세서 • 객체 모형도
산 출 물	• 시스템 구조도 • 서브 시스템 정의서
품질 기준	• 시스템 설계 절차서
9000-3 관련	• Section 5.6.1, Section 5.6.2

5.3.2 상세 객체 모델링 (D2)

이 과정은 전형적인 소프트웨어 개발 주기의 상세 설계 부분에 해당이 되는데, 우선 객체지향 분석의 산출물인 상태 전이도의 트랜지션 (Transition) 과 자료 흐름도의 프로세스를 객체 모형도의 해당 클래스의 함수로 고려한다. 이 작업의 지침이 되는 상세 객체 모델링 지침서는 클래스의 함수 추출과정에 대해 다루며, 추출된 함수들에 대한 품질 측정 및 평가 방법을 제시한다. 이 작업에는 클래스 함수 추출이 외에도 데이터 구조 확정, 알고리듬 설계등 전형적인 상세 설계 활동이 포함되어 있다.

작업 명	D2/상세 객체 모델링
목 적	• 상세설계를 통하여 함수, 알고리듬, 자료구조 정의등으로 객체모형도를 완성
작업 내역	• 상태전이도와 자료흐름도에서 객체 모형도의 함수를 추출 • 알고리듬, 자료구조등의 최적화
입력 자료	• 객체 모형도 • 상태 전이도 • 자료 흐름도
산 출 물	• 상세 객체 모형도
품질 기준	• 객체지향 설계 절차서 • 상세 객체 모델링 지침서
9000-3 관련	• Section 5.6.1, Section 5.6.2

5.3.3 Navigation Tree 설계 (D3)

이 작업은 A4의 User Interface 분석 작업의

작업 명	D3/Navigation Tree 설계
목 적	• U. I. 시나리오 분석을 통하여 최적의 Navigation 흐름을 명개
작업 내역	• U. I. 시나리오에서 Navigation 흐름를 정형화 • 정의된 U. I. Metaphore에 최적인 Navigation Tree로 설계
입력 자료	• U. I. Scenario • U. I. Metaphore
산 출 물	• Navigation Tree
품질 기준	• U. I. 설계 절차서
9000-3 관련	• Section 5.6.1, Section 5.6.2

결과물인 U. I. 시나리오를 기반으로 화면, 윈도우, 아이콘등 U. I. 컴포넌트들을 정의하고 이들간의 동적인 Navigation 흐름을 모델링하는 과정이다. Navigation 스타일과 방법은 A4 작업에서 정의된 U. I. Metaphore에 의해 상당 부분 정해진다.

5.3.4 User Interface 설계 (D4)

이 작업은 설계된 Navigation Tree를 바탕으로 U. I. Prototype을 개발하는 과정이다. User Interface는 사용자의 사용성(Usability)이 중요하므로 Prototype을 개발하여 조기에 사용자의 의견을 듣고 수정 보완하는 것이 보편적이다. 최근에는 Windows나 Motif등 윈도우 환경에서 User Interface Prototype을 쉽고 빠르게 개발할 수 있는 도구가 상당수 있어 이들을 활용하면 더욱 효과적으로 Prototype을 개발할 수 있다. 이 단계에서 개발된 U. I. Prototype은 I3의 User Interface 구현단계에 그대로 사용될 수 있으므로 구현 작업의 일환으로 볼 수 있다.

작업 명	D4/User Interface 설계
목 적	• Navigation Tree와 U. I. Metaphore를 기반으로 U. I. 프로그램 설계
작업 내역	• Navigation Tree에 기반한 화면 및 윈도우 설계 • U. I. Metaphore에 근거한 U. I. Prototype 개발
입력 자료	• Navigation Tree • U. I. Metaphore
산 출 물	• 화면/윈도우 설계서 • U. I. Prototype
품질 기준	• U. I. 설계 절차서
9000-3 관련	• Section 5. 6. 1, Section 5.6.2

5.4 Implementation 단계

5.4.1 프로그램 구현 (I1)

설계의 모든 과정이 완료되면, 설계 결과물을 컴파일러나 DBMS등을 이용하여 프로그램으로 변환한다. 영구적으로 자료를 저장해야 하는 시스템의 경우 데이터베이스가 이 과정에서 만들어 진다. 객체지향 구현 과정은 절차적 방법에 비해 상당히 손쉽고 체계적인 특징을 가지고 있는데, 이는 객체지향 분석에서 구현에 이르기 까지 같은 종류의 모델들을 일관성 있게 사용하기 때문이다.

작업 명	I1/프로그램 구현
목 적	• 시스템 설계와 상세 객체 모형도를 기반으로 프로그램 및 DB 구현
작업 내역	• 각 서브 시스템에 해당하는 객체 모형을 프로그래밍 언어로 구현 • 상세 설계 내용을 구현
입력 자료	• 시스템 구조도 • 서브 시스템 정의도 • 상세 객체 모형도
	• 알고리듬, 자료구조등 상세설계 내용
산 출 물	• 소스 프로그램 • 데이터베이스
품질 기준	• 구현 절차서 • 명명 지침서
9000-3 관련	• Section 5.6.3

5.4.2 단위 시험 (I2)

이 작업은 객체지향의 단위별로 단위 시험을 실시하는 일로서 단위 시험 절차서의 절차와 방법데로 단위 시험을 실시한 후, 결과서를 작성하고 필요시 적절한 수정 작업을 시행한다.

작업 명	I2/단위 시험
목 적	• 작성된 프로그램에 대해 객체, 클래스, 상속구조등 객체지향 단위별 시험
작업 내역	• 단위 시험의 계획 수립 • 단위별 시험 실시 및 결과 분석
입력 자료	• 상세 객체 모형도 • 시스템 구조도 • 소스 프로그램
산 출 물	• 단위 시험 결과서
품질 기준	• 구현 절차서 • 단위 시험 지침서
9000-3 관련	• Section 5.6.3

5.4.3 User Interface 구현 (I3)

D4 작업의 산출물인 User Interface Prototype에 대한 사용자의 의견을 반영 수정한 후,

U. I. 프로그램을 개발하고, 이를 I4작업에서 프로세싱 부분의 프로그램과 매핑을 시켜 단일 프로그램으로 통합한다. U. I. 프로그램의 형태은 윈도우 플랫폼, U. I. 개발 도구 및 방법에 따라 상당한 차이가 있을 수 있으나, U. I. Prototype을 확장하여 쉽게 개발할 수 있다.

작업 명	I3/User Interface 구현
목 적	• U. I. Prototype을 보완 전체 U. I. 프로그램 작성
작업 내역	• U. I. Prototype을 U. I. Scenario로 검증 및 보완 • 화면/윈도우, 프로세싱 부분과 인터페이스등 전체 U. I. 프로그램으로 개발
입력 자료	• Navigation Tree • U. I. Metaphore • U. I. Prototype
산 출 물	• U. I. 프로그램
품질 기준	• 구현 절차서 • U. I. 구현 지침서
9000-3 관련	• Section 5.6.3

5.4.4 시스템 통합 (I4)

이 작업은 정의된 시스템 구조에 따른 서브 시스템/모듈들을 단일 프로그램으로 통합하고, I3 작업의 산출물인 U. I. 프로그램을 통합하는 과정이다. 사용하는 운용체계나 시스템 플랫폼에 따라서는 시스템 컴포넌트들과 통합해야 할 경우도 있다. 예를 들면, Windows 에서 개발된 프로그램일 경우 MFC 클래스 라이브러리와 통합하는 경우가 많다.

작업 명	I4/시스템 통합
목 적	• 서브 모듈, U. I. 프로그램들을 통합, 단일 S/W 시스템 구성
작업 내역	• 여러 서브 모듈, U. I. 프로그램, 실행 라이브러리등을 통합 단일 시스템으로 구성
입력 자료	• 서브 모듈 프로그램 • U. I. 프로그램 • 실행 라이브러리
산 출 물	• 통합 시스템
품질 기준	• 구현 절차서
9000-3 관련	• Section 5.6.3

5.5 Testing & Validation 단계

5.5.1 통합 시험 (T1)

통합 시험은 통합 시스템에 대한 품질 측정 및 시험을 통하여 종합적인 평가를 하는 과정이다. 통합 시스템은 시스템의 특성을 반영한 통합 시험 계획을 수립한 후, 시험을 수행하고 결과를 처리한다. 통합 시험은 시스템의 컴포넌트들간의 조합성 시험이 주 목적이지만, 시스템의 종합적인 시험을 통하여 개발된 시스템이 주어진 규격(Specification)에 의하여 정확히 개발 되었는가를 시험하는 검증(Verificaton) 작업의 성격도 가진다.

작업 명	T1/통합 시험
목 적	• 시스템의 기능 및 성능의 통합 시험
작업 내역	• 시스템 특성을 반영한 통합 시험 계획 작성 • 통합 시험 수행 및 결과 처리
입력 자료	• 통합 시스템 • 시스템 구조
산 출 물	• 통합 시험 계획서 • 통합 시험 결과서
품질 기준	• 통합 시험 절차서
9000-3 관련	• Section 5.7.2, Section 5.7.3

5.5.2 Validation (T2)

ISO 9000-3의 5.7.1절은 Validation, Field Testing, Acceptance Testing이 하나의 작업으로 합쳐지거나 혹은 이들을 같은 단일 시험 과정으로 볼 수 있다고 한다. 본 OO-ISO 품질 시스템에서는 Field Testing을 Acceptance

작업 명	T2/Validation
목 적	• 시스템이 고객의 요구사항을 명시
작업 내역	• 요구 명세서와 사용 시나리오에 근거하여 시스템의 완성도를 측정
입력 자료	• 요구 명세서 • 사용 시나리오 • 통합 시스템
산 출 물	• Validation 결과서
품질 기준	• Validation 절차서
9000-3 관련	• Section 5.7.4

Testing에 포함하여 정의하고 있다.

확인 (Validation) 작업이란 개발된 시스템이 고객이 기대하고 있던 시스템인가를 시험하는 과정이다. 따라서, 시스템의 정확성, 효율성 등 성능적 측면보다는 시스템 기능과 영역이 고객이 염두에 둔 시스템의 기능과 영역인가의 일치성을 측정한다.

5.6 Post Development 단계

5.6.1 인수 시험 (O1)

확인 작업이 완료된 시스템을 고객이 직접 계약서와 요구 명세서의 내용데로 수행되는가를 시험해 보고 인수 유무를 결정하는 과정이다. 이 작업은 개발 방법론이나 생명주기와 무관하며, 고객의 입장에서 고객이 직접 검증 및 확인을 해 보는데 의미가 있는 과정이다.

작업 명	O1/인수 시험
목 적	• 완성된 시스템을 고객이 계약내용을 근거로 최종 시험
작업 내역	• 계약서와 요구 명세서에 근거하여 완성된 시스템을 고객(구매자)가 시험, 인수 여부를 결정
입력 자료	• 요구 명세서 • 통합 시스템
산 출 물	• 인수 시험 결과서
품질 기준	• 인수 시험 절차서
9000-3 관련	• Section 5.7.5, Section 5.8

5.6.2 보급 및 운영 (O2)

작업 명	O2/보급 및 운영
목 적	• 시스템을 복사, 패키지화 하여 고객의 Site에 설치하고 고객이 운영할 수 있게 함.
작업 내역	• 시스템을 보급할 형태로 패키지화 • 고객의 Site에 설치 • 필요시 운용 교육 실시
입력 자료	• 통합 시스템
산 출 물	• S/W 패키지 • 사용자 메뉴얼 • 운용자 메뉴얼
품질 기준	• 보급/운용 절차서
9000-3 관련	• Section 5.9

이 작업은 인수 시험을 통과한 시스템을 고객의 Site에 설치하고 필요한 지원을 하는 업무이다. 전형적인 작업으로 시스템 설치, 사용자 및 운용자 교육 실시, 메뉴얼 전달등이 있으며, O1의 작업처럼 개발 방법론과 생명주기와는 무관하다.

6. OO-ISO 품질 시스템의 절차서 및 지침서

6.1 산출물, 절차서 및 지침서의 종류

ISO 기준에 근거한 품질 시스템은 여러 개의 절차서 (Procedure)와 지침서(Instruction)들로 구성된다. 생명주기 관련 및 지원 업무의 각 단계별 혹은 작업별로 관련된 절차서와 지침서들이 마련된다.

절차서는 각 단계나 작업의 세부 절차를 정의하고 그 작업 내역을 명시하는 반면, 지침서는 관련된 절차서에서 제시된 작업 순서나 내역을 세부적이며 구체적으로 정의하고 있다. 단계나 작업의 내용에 따라서 절차서만으로 충분한 경우도 있으며, 어떤 작업들은 지침서의 구체적인 지침을 필요로 한다.

표 3은 OO-ISO 품질 시스템의 절차서와 지침서를 각 작업의 산출물과 연관하여 나타내고 있다. 모든 품질 시스템은 여러 절차서와 지침서를 통하여 품질 보증 활동을 구체화하게 되는데, 이런 표준 문서의 종류나 분량은 최소이면서 필수적인 절차와 지침을 모두 반영하도록 해야 한다. 과제 관리자의 입장에서는 필수적인 지침을 충분히 제공하면서, 개발자에게 문서화 작업의 부담은 최소화하는 것이 바람직하기 때문이다.

OO-ISO 품질 시스템은 11가지 절차서와 10가지 지침서들로 구성되어 있다. 이 품질 문서들의 내용과 그 깊이는 각 회사의 특성과 목적에 따라 다소 차이가 날 수 있는데, 너무 구체적으로 작업의 절차나 지침을 지정할 경우 일관성 있고 체계적인 개발을 할 수 있으나 과제별 특성을 반영하여 개발 할 수 있는 유연성 (Flexibility)은 낮아진다.

표 3의 여러 문서들중에서 대표적으로 객체지향 분석 절차서, 객체지향 모델링 지침서의 내용을 좀 더 알아보고, P2의 개발 계획 수립

표 3 OO-ISO 품질 시스템의 절차서 및 지침서

	Phase/Task	산 출 물	절 차 서	지 침 서
P1	요구 명세서 작성	요구 명세서	–	요구명세 지침서
P2	개발 계획 수립	개발 계획서 품질 계획서	개발계획 절차서	품질계획 지침서 방법론 선정 지침서
A1	객체 모델링 (Object Modeling)	객체 모형도 자료 사전	객체지향 분석 절차서	객체 모델링 지침서
A2	동적 모델링 (Dynamic Modeling)	Use Scenario 이벤트 모형도 상태 전이도		동적 모델링 지침서
A3	기능 모델링 (Functional Modeling)	자료 흐름도 Function Description		기능 모델링 지침서
A4	User Interface 분석	U. I. Scenario U. I. Metaphore	U. I. 분석 절차서	–
D1	시스템 설계	시스템 구조도 서브 시스템 정의서	시스템 설계 절차서	–
D2	상세 객체 모델링	상세 객체 모형도	객체지향 설계 절차서	상세 객체 모델링 지침서
D3	Navigation Tree 설계	Navigation Tree	–	
D4	User Interface 설계	화면/윈도우 설계서 U. I. Prototype	U. I. 설계 절차서	–
I1	프로그램 구현	소스 프로그램 데이터베이스	구현 절차서	명명 지침서
I2	단위 시험	단위 시험 결과서		단위 시험 지침서
I3	User Interface 구현	U. I. 프로그램		U. I. 구현 지침서
I4	시스템 통합	통합 시스템		–
T1	통합 시험	통합 시험 계획서 통합 시험 결과서	통합 시험 절차서	–
T2	Validation	Validation 결과서	Validation 절차서	–
O1	인수 시험	인수 시험 결과서	인수 시험 절차서	–
O2	보급 및 운용	S/W 패키지 사용/운용 설명서	보급/운용 절차서	–

작업의 한 지침서인 품질 계획 지침서에 대하여 살펴본다.

6.2 객체지향 분석 절차서

객체지향 분석 (이하 OOA) 절차서는 주어진 도메인을 이해하고 정형적으로 표현하는 과정을 순차적이며 체계적으로 정의하여야 한다. OMT 방법론의 경우, OOA 작업은 객체 모델링, 동적 모델링, 기능 모델링의 순서데로 진행된다. 따라서, OOA 절차서는 전체적인 작업과정을 종합적으로 설명한 후, 각 작업을 소단위 작업들로 세분화하여 정의한다. 그림 8은 OMT 방법론의 세가지 모델링 작업 과정을 요약한 것이며, 관련된 표준 문서인 객체모델링 지침서, 동적모델링 지침서, 기능모델링 지침서는 각 모델을 작성하는 과정에 세부적이며 구체적인 방법과 지침을 제공한다.

OOA 절차서에서는 각 작업의 목적, 작업 방법, 입. 출력 자료등을 정의하는데, 객체 모델링에 관한 순서는 다음과 같이 요약된다.

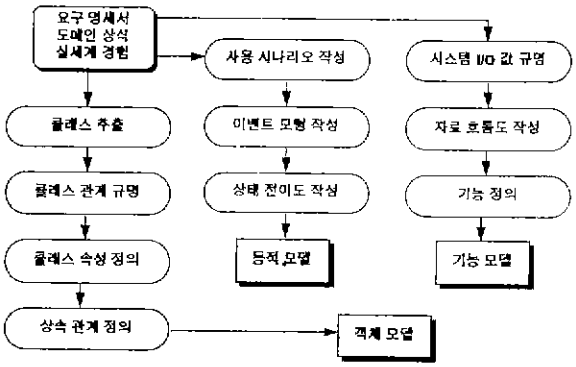

그림 8 객체지향 분석의 모델링 과정

1. 클래스를 추출한다.
2. 클래스 단위로 자료 사전을 생성한다.
3. 클래스/객체간의 관계를 규명한다.
4. 클래스의 속성(Attribute)를 정의한다.
5. 클래스간의 상속 관계 (Inheritance)를 정의한다.
6. 생성된 객체 모델을 검증 보완한다.

또한, OOA 절차서는 객체 모델링의 각 순서에 대해 보다 구체적인 절차를 정의하게 된다. 그림 9는 객체모델링의 첫 순서인 클래스 추출 과정을 그림으로 요약한 것이다. 우선 요구 명세서등에서 명사구를 찾아 후보 클래스로 등록하고, 후보 클래스에서 부적절한 클래스를 찾아 삭제하고 좋은 클래스들을 찾아내는 과정을 정의하고 있다. 일반적으로 절차서는 구체적인 작업의 방법들은 명시하고 있지 않으므로 OOA 절차서의 경우 부적절한 클래스를 찾고 삭제하는 방법은 객체모델링 지침서에서 다루도록 한다.

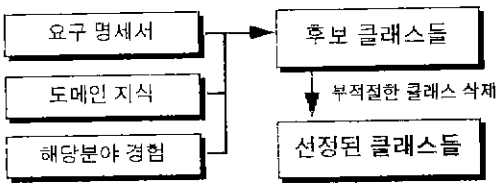

그림 9 객체모델링에서 클래스 추출 과정

6.3 객체 모델링 지침서

객체 모델링 지침서는 OOA 절차서에서 정의된 각 작업에 대하여 세부 지침을 제공하는데, 계속해서 그림 9의 클래스 추출 과정을 예로 들어 보자. 객체 모델링 지침서는 다음과 같이 부적절한 클래스를 찾아 내는 방법과 기준을 제공한다.

- 중복 이름의 (Redundant) 클래스
- 관련성이 없는 (Irrelevant) 클래스
- 불분명하거나 범위가 너무 큰 (Vague) 클래스
- 속성 (Attribute)으로 사용되어야 할 클래스
- 기능 (Operation)으로 사용되어야 할 클래스
- 역할 (Role)로 사용되어야 할 클래스
- 구현에 밀접한 성격의 클래스

부적절한 클래스의 각 항목에 대한 보충 설명과 필요시 예제들도 지침서에 포함되어야 이 지침서가 개발자에게 유용하게 사용될 수 있다.

방법론 적용 측면에서는 도형, 도표등 필요한 양식이나 템플레이트 (Template)등도 매우 유용하게 사용될 수 있다. 예를 들면 객체 모델링의 클래스 추출 과정에서 그림 10와 같은 템플레이트가 제공될 수 있다. 이 템플레이트를 보면, 삭제할 클래스를 분류하는 7개의 항목들이 나열되어 있어 삭제될 후보 클래스 이름을 써넣게 되어 있고 아래의 네모 칸에는 남은 클래스들 즉 좋은 클래스들의 이름들을 기록하게 한다. 이렇게 지침서들은 개발자나 관리자가 쉽고 효과적으로 방법론과 표준을 적용할 수 있도록 체계적이며 구체적으로 작성되어져야 한다.

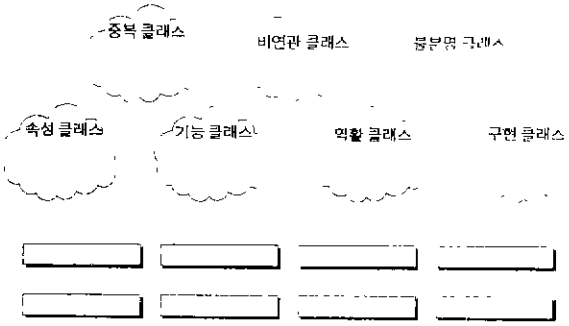

그림 10 클래스 추출을 위한 템플레이트(Template)

6.4 품질 계획 지침서

ISO 9000-3의 5. 5절에는 개발 계획 수립 단계의 한 산출물로 품질 계획서를 요구하고 있다. 각 생명주기의 각 단계별 관련된 품질 보증 계획을 수립, 적용하도록 해야 한다. 같은 문서 5.5.2절에는 품질 계획이 포함해야 할 내용이 열거되어 있어 이를 기준으로 채택된 방법론에 적합한 품질 보증 기준이 마련된다.

OO-ISO 품질 시스템은 OMT 방법론을 채택하였으므로 객체지향 품질 보증 항목의 정의 및 품질 측정 방법등이 제시되어야 한다. 그러나, 객체지향 품질 보증 및 측정에 관한 연구가 문헌 [11]등 활발히 진행되고 있으나 아직 성숙된 단계가 아니므로 품질 계획 수립에 세심한 주의를 필요로 한다. OO-ISO 품질 시스템의 품질 계획 지침서에서 A1의 객체모델링과 D2의 상세 객체 모델링 작업을 예로 들어 품질 보증 대상 항목들을 살펴본다.

표 4 객체 모델링 (A1)작업의 품질 보증 대상 항목

평 가 항 목	평 가 방 법
객체분석모형의 도메인 반영도	객체분석모형의 클래스, 상속성, 관계등이 주어진 도메인의 정적인 (Static) 정보의 반영 정도
클래스 추출의 논리적 합리성	객체분석모형에 나타난 각 클래스에 대해 1. State정보, 2. 고유 기능성, 3. 객체의 구별성 유무
클래스간의 관계 간결성	복수 클래스간의 Ternary 관계를 피하고 Binary 관계로 정의된 비율
클래스간의 중복성 배제 정도성	클래스간의 속성및 고유기능면에서의 중복성을 Superclass로 규명한 정도
클래스 속성의 완전도	각 클래스에서 요구되는 속성자들의 정의에 대한 완전도
클래스 속성의 일반성(Generality)	표현된 속성이 특정 언어나 DBMS 데이터 타입과의 독립성 정도
객체분석모형의 추적 적용성	사용자 요구사항에 근거한 개략적 기능수행 Path가 객체모형의 관계 Path로 추적의 적용성
관계의 복수성 명확도	클래스간의 복수성 (Multiplicity)과 순서등에 관한 표현 정도

표 4는 객체 모델링 작업에 대한 품질 보증 항목과 평가 방법을 나타낸 것으로, 품질 계획 지침서에는 각 항목에 대한 측정 방법과 중요도(Weight) 등 품질 보증 활동이 구체적으로 정의되어야 한다. 각 과제의 개발 계획 수립단계에서 이 지침서를 바탕으로 과제에 특성을 반영한 품질 계획서를 작성하며, 이 계획서를 바탕으로 생명주기의 각 단계에 해당되는 품질 보증 활동이 수행된다.

표 5 상세 객체 모델링(D2) 작업의 품질 보증 대상 항목

평 가 항 목	평 가 방 법
분석모델과의 일관성	세 가지 모델과의 전반적인 일관성
상태 전이도에서 추출된 함수의 반영도	상태 전이도에서 나타난 Transition들의 객체모델의 해당 클래스의 함수로 반영된 정도
자료 흐름도에서 추출된 함수의 반영도	자료 흐름도에서 나타난 Process들의 객체모델의 해당 클래스의 함수로 반영된 정도
자료 흐름도에서 추출된 함수의 매개체들의 반영도	자료 흐름도에서 나타난 Process별 입력과 출력 매개체들의 객체모델의 함수의 시그네쳐에 적용된 정도
상속구조에 있어 서브 클래스들간의 밸런싱 정도	한 베이스 클래스에서 파생된 서브 클래스간의 속성및 함수들의 개념적인 밸런싱 정도
상속 구조의 확장성	새로운 서브 클래스를 추가하기가 용이한 확장적 구조로 설계된 정도 특히, 추상 클래스를 사용하여 추상화 작업을 최대화한 정도
상속 구조의 효율성	불 필요한 서브 클래싱을 배제하여 실행시 최소한의 서치 (Search)를 통하여 바인딩할 수 있는 정도.
클래스별 속성을 구현하기 위한 상세도	각 클래스별로 정의된 속성을 구현하기 위한 설계 단계의 충분한 내역의 정도
멤버 함수들의 설계성의 효율성	각 클래스별로 정의된 멤버 함수들의 알고리듬에 구현시 효율성을 보장하는 설계의 정도
객체설계모형의 추적 적용성	분석단계에서 작성된 시나리오에 근거한 메시지 교환을 통한 원시적인 검증시, 객체모형의 추적 적용성

개발 계획 수립단계에서 품질 계획 지침서에 정의된 모든 항목들을 다 고려해야 하는 것은 아니며, 각 과제의 특성과 요구사항에 따라 일부를 선택하거나 부분 수정하여 각 과제에 따른 개발계획서와 품질 계획서를 작성하면 된다.

표 5는 OOA의 세가지 모델을 참고하여 객체

모델을 상세히 정의하는 설계 단계에서 고려해야 할 품질 보증 항목들을 나타내고 있다.

7. ISO 기준과의 Cross-Reference

지금까지 ISO 9001의 적용 기준을 제시하는 ISO 9000-3 기준에 따라 개발된 OO-ISO 품질 시스템을 설명하였는데, 본 장에서는 ISO 9000-3 문서의 항목들과 OO-ISO 생명주기의 작업들과의 상호 관련을 정리해 본다. 표 6는 ISO 9000-3 문서의 생명주기 관련 업무의 절과 OO-ISO 품질 시스템의 작업들과의 상호 관련을 요약하고 있다.

표 6 ISO 기준과 OO-ISO 품질시스템의 상관도

ISO 9000-3 기준		OO-ISO 품질 시스템	
절 #	절 제목	작업 명	ID
5.3	Requirements Spec.	요구명세서 작성	P1
5.4	Devel. Planning	개발 계획 수립	P2
5.6.2 ?	Design ?	객체 모델링	A1
		동적 모델링	A2
		기능 모델링	A3
		User Interface 분석	A4
5.6.2	Design	시스템 설계	D1
		상세 객체 모델링	D2
		Navigation Tree 설계	D3
		User Interface 설계	D4
5.6.3	Implementation	프로그램 구현	I1
		단위 시험	I2
		User Interface 구현	I3
		시스템 통합	I4
5.7	Testing & Validation	통합 시험	T1
		Validation	T2
5.8	Acceptance	인수 시험	O1
5.9	Replication, Deliv.,.	보급 및 운용	O2

8. 결 론

ISO 9001 품질 기준과 객체지향 방법론은 소프트웨어 품질을 향상시키고 개발의 생산성을 높이는 핵심 기술로 인정을 받고 있다. ISO 품질 기준은 소프트웨어 생명주기의 전 과정에 걸친 표준화와 품질 보증활동을 통하여, 객체지향 기술은 보다 자연스럽고 효율적인 소프트웨어 개발 방법론과 재사용을 통하여 그 목적을 달성한다. 본 논문에서는 ISO 품질 기준에 근거하여 객체지향 소프트웨어 개발에 효과적으로 활용할 수 있는 객체지향 ISO 품질 시스템을 제안하였다.

OO-ISO 품질 시스템은 객체지향 프로그래밍 패러다임, 점진적 개발 방법, User Interface 개발 지원, OMT 분석 및 설계 방법등을 주된 원칙으로 하며, 문서의 최소화와 실무 적용성을 우선으로 개발되었다. ISO 기준을 모두 충족시키며, ISO 기준에 포함되어 있지 않는 품질 보증 및 개발 활동등을 보완한 OO-ISO 품질 시스템의 생명주기와 단계별 작업들을 정의하였다.

품질 시스템의 표준 문서로서 절차서와 지침서를 두어 방법론의 효과적인 적용을 꾀하도록 설계되었으며, OMT 방법론과 User Interface 개발 트랙을 포괄적인 ISO 생명주기에 매핑하여 일관성있는 방법론으로 통합하였다. 절차서와 지침서의 내용을 예를 통하여 제안하였고, 객체지향 시스템에 대한 품질 보증 기준을 정의하는 품질 계획 지침서를 제안하였다.

제안된 OO-ISO 품질시스템이 ISO 품질 인증을 통한 품질 향상과 객체 기술의 도입과 정착을 추구하는 회사나 조직에 그 기본 모델로 사용될 수 있기를 기대한다.

참고문헌

[1] International Organization for Standardization, *ISO 9001 - Quality Systems-Model for Quality Assurance in Design, Development, Production, Installation and servicing*, Second Edition, July 1, 1994.

[2] International Organization for Standardization, *ISO 9000-3 : Guidelines for the Application of ISO 9001 to the Development, Supply and Maintenance of Software*, June 1, 1991.

[3] Ivar Jacobson, *Object-Oriented Software Engineering : A Use Case Driven Approach*, Addison-Wesley, p. 524, 1992.

[4] 김수동, "객체지향 소프트웨어공학," 정보과학회지, 객체지향 패러다임 특집, 제 11권, 제2호, pp. 5-21, 1993년 4월.
[5] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-Oriented Modeling and Design*, Prentice-Hall, 1991.
[6] ACM, Conference on Object-Oriented Programming Systems, Languagues, and Applications, Tutorials and Panel Discussions, Portland, Oregon, Oct. 23-27, 1994.
[7] International Organization for Standardization, ISO/IEC DIS 12207-1 : Information Technology-Part1-Software Life-Cycle Processes, 1994.
[8] Stephen H. Kan, Metrics and Models in Software Quality Engineering, Addison-Wesley, 344 pp., 1995.
[9] Dave Collins, *Designing Object-Oriented User Interfaces*, Benjamin/Cummings Publishing Co., p. 590, 1995.
[10] Object Management Group, *The Common Object Request Broker: Architecture and Specification*, Revision 1. 2, Dec. 1993.
[11] Shyam Chidamber and Chris Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, June 1994.

김 수 동

1984 12 Northeast Missouri State Univ. 전산학 학사
1998 5 The University of Iowa, 전산학 석사
1991 5 The University of Iowa, 전산학 박사
1991~1993 한국통신 연구개발단 선임연구원/연구실장
1993~1994 The University of Iowa, 전산학과 교환교수
1994~1995 현대전자 S/W연구소 책임연구소
1995~현재 숭실대학교 컴퓨터학부, 조교수
관심분야 : S/W 공학, 객체지향 개발 방법론, S/W 품질 시스템, 분산 처리 시스템