소프트웨어 설계기술서 표준

한국정보통신기술협회

TTA Standard

정보통신단체표준 제정일 : 2000년 12월 20일

TTAS.KO-11.0023

소프트웨어 설계기술서 표준

(Standard for Software Design Descriptions)

한국정보통신기술협회
Telecommunications Technology Association

서 문

1. 표준의 목적

이 표준은 소프트웨어 설계기술서(SDD : Software Design Descriptions)를 위한 구성을 권고하고 필요한 정보 내용을 기술함을 목적으로 한다.

2. 참조 권고 및 표준

2.1 국제 표준(권고) : 해당사항 없음.

2.2 국내 권고 : 해당사항 없음.

2.3 기타 : IEEE Std 1016-1998 IEEE Recommended Practice for Software Design Descriptions

3. 국제 표준(권고)과의 비교 : 해당사항 없음.

4. 지적 재산권 관련 사항 : 해당사항 없음

5. 적합인증 관련 사항 : 해당사항 없음

6. 표준의 이력

판 수	제·개정일	개정판 내용
제 1 판	2000년 12 월 20 일	제정

Preface

1. The Purpose of Standard

This standard specifies the necessary information content and recommends an organization for software design descriptions. This document does not explicitly support, nor is it limited to, any particular software design methodology or descriptive technology.

2. Reference Recommendations and/or Standards

2.1 International standards : None.

2.2 Domestic standards : None.

2.3 Other standards : IEEE 1016-1998 IEEE Recommended Practice for Software Design Description

3. Relationship to International Standards(Recommendations) : None.

4. The statement of Intellectual Property Rights : None

5. The statement of Conformance Testing and Certification : None

6. The history of standard

Edition	Issued date	Contents
The 1st edition	2000. 12. 20.	Established

목 차
CONTENTS

소프트웨어 설계기술서 표준
Standard for Software Design Descriptions

1. 개요

1.1 범위

이 표준은 소프트웨어 설계서를 기술하기 위한 실무활동을 정의한다. 이것은 소프트웨어 설계기술서(SDD : Software Design Description)에 필요한 정보 내용과 구성에 대하여 기술한다. SDD는 소프트웨어 설계 정보를 상호교환하기 위한 전달 수단으로 사용되는 소프트웨어 시스템의 표현이다.

이 표준은 상업용, 과학용 또는 임의의 디지털 컴퓨터에서 실행되는 군사용 소프트웨어에 적용할 수 있으며, 이 표준의 적용성은 소프트웨어의 크기, 복잡도 또는 중요도에 의해 제한되지 않는다.

이 표준은 설계, 형상관리 또는 품질보증 등의 특정 방법론에 의해 제한을 받지 않으며, 품질 설계 정보와 기술서 설계 변경은 다른 프로젝트 활동에서 관리될 것이라고 가정한다. 또한 이 문서는 특정 소프트웨어 설계 방법론이나 서술 기술을 명시적으로 지원하거나 제한하지 않는다. 이 표준은 문서, 자동화된 데이터베이스, 설계 기술 언어, 다른 기술 수단에 적용 가능하다.

2. 참고문헌

본 표준은 다음과 같은 출판물과 함께 사용될 수 있다.

IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminolgy.

IEEE Std 730-1998, IEEE Standard for Software Quality Assurance Plans.

IEEE Std 828-1998, IEEE Standard for Software Configuration Management Plans.

IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements

Specifications.

Freeman, P. and A. I. Wasserman, Tutorial on Software Design Techniques. 4th Edition, IEEE Computer Society Press Annotated Bibliography, pp. 715-718, 1983

3. SDD 생성을 위한 고려사항

이 절에서는 SDD를 생성하기 전에 고려해야 할 정보를 제공한다. 소프트웨어 생명주기에 맞는 SDD는 어떤 것이고, 어디에 적합하며, 왜 사용되어야 하는가를 설명한다.

3.1 소프트웨어 생명주기

소프트웨어 시스템의 생명주기는 일반적으로 소프트웨어 제품을 구상하고 계획하는 시점으로부터 제품이 더 이상 유용하지 않을 때까지의 기간으로 정의한다. 이 생명주기 접근방법은 효과적인 공학적 관리 도구로서 SDD의 준비와 사용에 관한 내용을 논의하는 모형을 제공한다. 특정 표준의 생명주기를 정하는 것은 이 문서 범위를 벗어나므로 전형적인 생명주기에서 SDD를 위한 내용을 정의하고자 한다.

이 생명주기는 IEEE Std 610.12-1990에 기초하며 개념 단계, 요구분석 단계, 설계 단계, 구현 단계, 시험 단계, 설치 및 점검 단계, 작동과 유지보수 단계, 폐기 단계로 구성된다.

3.2 생명주기에서의 SDD

새로운 소프트웨어 시스템과 유지보수 상태에 있는 시스템에 대하여 해당 시스템의 설계와 구현이 요구사항을 만족시키도록 보장하는 것은 매우 중요하다. 이를 위하여 요구되는 최소한의 문서가 IEEE Std 730-1998에 정의되어 있는데 그 중 하나가 SDD이다. 여기에는 설계 단계동안 수행되는 설계 공정들의 결과가 기록된다.

3.3 SDD의 목적

SDD는 소프트웨어 요구 명세서 IEEE Std 830-1998에서 정의된 요구사항을 만족시키기 위하여 소프트웨어 시스템을 어떻게 구성할 것인가를 보여준다. 요구사항은 소프트웨어 구조, 소프트웨어 부품, 인터페이스 그리고 구현 단계에 필요한 데이터들에 관한 기술로 변환된다. 근본적으로 SDD는 구현 활동의 세부적인 청사진이 된다. 완전한 SDD는 각각의 요구사항이 하나 이상의 설계 요소들과 연관되어 추적 가능해야 한다.

4. 설계기술서 정보 내용

4.1 개요

SDD는 생성할 소프트웨어 시스템의 모형 또는 표현이다. 이 모형은 소프트웨어 시스템의 계획, 분석, 구현에 필요한 정확한 설계 정보를 제공해야 한다. 이는 시스템이 분할된 설계 엔티티들로 표현되고 이들의 중요한 속성과 이들 엔티티간의 관계를 기술해야 한다.

하나의 소프트웨어 시스템을 표현하는데 사용되는 설계기술서 모형은 설계 엔티티들의 집합, 각 소유 속성들과 관계들로 표현될 수 있다. 이 모형을 단순화하기 위하여 각 설계 엔티티의 속성과 관계들은 속성들의 표준 집합으로 기술되어진다. 프로젝트에 필요한 설계 정보들은 엔티티의 식별과 그들간의 관련 속성들로 구성된다. SDD는 모든 설계 엔티티에 대한 속성들이 명시되었을 때 완성된다.

4.2 설계 엔티티

설계 엔티티는 구조적으로나 기능적으로 다른 엔티티들과 구별되며 별도의 다른 이름으로 명명되고 참조되는 설계 요소(구성요소)를 말한다.

설계 엔티티는 소프트웨어 시스템 요구 사항을 분해한 결과물이며 그 목적은 다른 엔티티들에게 최소의 영향을 주는 독립된 구성요소로 나누는데 있다.

엔티티들은 시스템, 서브시스템, 데이터 저장소, 모듈, 프로그램 그리고 프로세스로 존재할 수 있다(IEEE Std 610.12-1990을 참조). 엔티티의 수와 유형은 시스템 복잡도, 사용되는 설계 기법, 프로그래밍 환경과 같은 다수의 요인에 따라 달라진다.

비록 엔티티들은 본질적으로는 서로 다르지만 서로 다른 엔티티들 사이에도 공통된 특성들을 갖는다. 각 설계 엔티티들은 이름, 목적, 기능을 갖는다. 공유 데이터 또는 인터페이스와 같은 엔티티들간의 공통 관계들이 있다. 엔티티들의 공통 특성들은 설계 엔티티 속성으로 기술된다.

4.3 설계 엔티티 속성

설계 엔티티 속성은 설계 엔티티의 명명된 특징이나 속성을 말하며 엔티티에 대한 사실적인 설명을 제공한다.

설계 엔티티 속성은 설계 엔티티에 대한 질문으로 생각할 수 있는데 이들 질문에 대한 답변이 속성의 값이 된다. 모든 질문은 답변을 얻게 되며 답변의 내용은 엔티티의 상태에 따라 달라진다. 이 답변들의 모음은 엔티티에 대한 완전한 설명을 제공한다.

이 절에서 제시하는 설계 엔티티 속성 리스트는 모든 SDD에서 요구하는 최소한의 것이다. 이러한 속성들의 선정은 다음 세 가지 기준에 근거한다.

a. 속성은 모든 소프트웨어 프로젝트에서 필수적이다.
b. 속성 값의 부정확한 명세는 개발되어질 소프트웨어 시스템의 결함을 초래할 수 있다.
c. 속성은 본질적인 설계 정보를 기술하는 것이지 설계 공정과 관계된 정보를 기술하는 것이 아니다.

두 번째와 세 번째 기준에 맞지 않는 속성의 실례로는 설계자의 이름, 설계 상태와 교정 기록들이 있다. 이러한 중요한 공정 정보는 IEEE Std 730-1998과 IEEE Std 828-1998에서 설명한 것과 같이 다른 소프트웨어 프로젝트 활동에 의해 유지 보수된다.

각 엔티티에 대한 모든 속성들은 명시되어야 한다. 속성 기술서에는 참고문헌과 설계 고려사항들이 포함되어야 한다. 속성 기술서가 값을 갖지 않는 경우도 있다. 부가적인 속성들이 특정한 소프트웨어 프로젝트에서 식별되었을 때 이는 설계기술서에 포함되어야 한다.

속성들과 그와 관련된 정보 항목들을 4.3.1에서 4.3.10까지 정의하였다.

4.3.1 식별

엔티티 이름 : 각 엔티티들은 동일한 이름을 가질 수 없다. 엔티티 이름은 그들의 본질을 특성화할 수 있는 것으로 선택하고 이는 엔티티들을 식별할 수 있을 뿐만 아니라 참조와 추적을 하는데 사용할 수 있다.

4.3.2 유형

엔티티 종류에 대한 설명 : 유형 속성은 엔티티의 본질을 나타낸다. 이것은 서브프로그램, 모듈, 절차, 프로세스 또는 데이터 저장과 같이 단순히 엔티티의 종류를 명명할 수 있다. 다른 한편으로 설계 엔티티는 정보의 한 특정 유형을 다루는 엔티티를 자리잡게 하는데 도움을 주는 주요 클래스들로 그룹화 될 수 있다. 주어진 설계기술서에 의해 선택된 엔티티 유형은 일관성있게 적용되어야 한다.

4.3.3 목적

엔티티가 존재해야 되는 이유에 대한 설명 : 목적 속성은 엔티티가 생성되는 근본적인 이유를 설명한다. 그러므로 해당 엔티티의 생성을 위한 기능적, 성능적 요구사항을 명확히 나타내야 한다(IEEE Std 830-1998 참조). 또한 목적은 소프트웨어 요구명세서에 포함되지 않는 엔티티들에 의해 만나게 될 수도 있는 특별한 요구사항도 서술해야 한다.

4.3.4 기능

엔티티가 하는 일에 대한 설명 : 기능 속성은 입력이 바람직한 출력을 생성할 수 있도록 엔티티에 의해 적용되는 변환을 기술한다. 데이터 엔티티의 경우라면 엔티티에 의해 저장되거나 전달되는 정보의 유형을 나타낸다.

4.3.5 종속

해당 엔티티를 구성하는 모든 엔티티에 대한 식별 : 종속 속성은 특정 엔티티에 대한 구성 관계(*composed of*)를 나타낸다. 이 정보는 엔티티들을 설계하기 위한 요구 사항을 추적하고, 소프트웨어 시스템의 분해를 통하여 부모/자식간의 구조적 관계를 식별하기 위하여 사용된다.

4.3.6 의존

해당 엔티티와 다른 엔티티간의 관계 설명 : 의존 속성은 특정 엔티티에 대한 필요 관계(*requires the presence of*)나 사용 관계(*uses*)를 나타낸다. 이 관계들은 구조도, 자료 흐름도와 트랜잭션 다이어그램에 의해 도형적으로 표기되기도 한다.

이러한 속성들은 상호작용의 타이밍, 조건과 같은 특성을 포함하는 각 상호작용의 특징을 기술한다. 이러한 상호작용은 초기화, 실행순서, 데이터 공유, 생성, 중복, 용도, 저장 또는 엔티티의 파기 등을 포함한다.

4.3.7 인터페이스

해당 엔티티와 다른 엔티티간의 상호 작용 방법에 대한 설명 : 인터페이스 속성은 상호작용의 방법과 이러한 상호작용들을 관리하는 규칙으로 나타낸다. 상호작용의 방법은 호출이나 간섭 엔티티를 위한 매카니즘, 매개변수, 공통 데이터 영역 또는 메시지를 통한 통신 메카니즘, 그리고 내부 데이터를 직접 검색하기 위한 메카니즘을 포함한다. 상호 작용을 관리하는 규칙들은 통신 프로토콜, 데이터 포맷, 액세스 가능 값과 각 값의 의미를 포함한다. 이 속성은 입력 값의 범위, 입·출력 값의 의미, 입·출력 양식과 형태, 그리고 출력 에러 코드에 대한 설명이 있어야 한다. 정보 시스템에서 이것은

입력, 스크린 포맷과 상호작용 언어의 완전한 설명이 포함되어야 한다.

4.3.8 자원

엔티티에 의하여 사용되는 설계 외부 요소에 대한 표현 : 자원 속성은 설계를 위한 외부 자원 모두를 식별하고 기술해야만 한다. 자원을 사용하기 위한 상호작용 규칙과 방법들은 이 속성에 의하여 명시되어야 한다.

이 속성은 물리적 장치(프린터, 디스크-분할, 메모리), 소프트웨어 서비스(라이브러리, 운영체제 서비스), 프로세스 서비스(CPU 사이클, 메모리 할당, 버퍼)와 같은 항목에 대한 정보를 제공한다.

이 자원 속성은 자원이 획득되는 시간, 사용되는 버퍼의 크기와 양을 포함하는 규모 등에 대한 사용 특성을 설명한다. 또한 자원 관리 기능 뿐 아니라 교착상태 조건과 잠정적인 레이스(race)의 식별을 포함해야 한다.

4.3.9 처리

기능을 달성하기 위하여 엔티티에 의해 사용되는 규칙에 대한 표현 : 처리 속성은 특정 작업을 수행하기 위하여 엔티티에 의해 사용되는 알고리즘을 설명해야 하며, 돌발 사고 처리를 포함해야 한다. 이 기술서는 기능 속성을 정리한 것으로 이 엔티티에 대한 가장 상세한 수준이다.

이 기술서는 타이밍, 사건 또는 프로세스의 순서, 프로세스 초기화를 위한 선행 요구, 사건의 우선순위, 프로세스 수준, 실제 프로세스 단계, 경로 조건, 그리고 반복의 종료 기준을 포함해야 한다. 돌발 사고에 대한 조치로는 확인 점검이 실패했을 경우나 오버플로우 조건인 경우에 취해지는 활동을 설명한다.

4.3.10 데이터

내부적 데이터 요소에 대한 기술 : 데이터 속성은 표현 방법, 초기값, 용도, 의미, 포맷과 내부 데이터 액세스 가능 값에 대하여 기술한다.

데이터에 대한 기술은 자료 사전의 형태일 수 있는데 이는 내용, 구조, 모든 데이터 엔티티들의 용도에 대하여 기술한다. 데이터 정보로는 이 엔티티에 의한 내부 데이터 구조 또는 데이터 사용에 관한 모든 것을 기술해야 한다. 이것은 포맷, 요소의 수, 초기치와 같은 자료 명세를 포함해야 하고, 파일 구조, 배열, 스택, 큐와 메모리 분할과 같은 데이터를 표현하는데 사용되는 구조들을 포함해야 한다.

데이터 요소의 용도와 의미는 명시되어야 한다. 이 기술서는 정적, 동적인 것에 대한 것, 트랜잭션에 의해 공유되는지 여부, 제어 매개 변수로 사용되는지, 혹은 값으로 사용되는지 여부, 루프 반복 카운터, 포인터 링크 필드로 사용되는 것 등을 포함해야 한다. 추가로 데이터 정보는 프로세스에 대하여 필요한 데이터 확인에 대한 기술을 포함해야 한다.

5. 설계기술서 구성

5.1 개요

각 설계기술서 사용자는 소프트웨어 설계의 본질적인 측면이 무엇인지에 대해 서로 다른 관점들을 가질 수 있다. 다른 모든 정보들이 그 사용자에게 모두 필요한 것은 아니다. 특정한 사용자에 대해 유용한 정보 부분은 소프트웨어 프로젝트의 크기 또는 복잡도로 인하여 줄어들 수 있다. 따라서 필요한 정보들을 기술서로부터 발췌하는 것이 어렵거나 비현실적일 수 있으며, 그것으로의 동화가 불가능 할 수도 있다. 그러므로 필요한 설계 정보의 실제적인 구성은 사용성에 초점을 두어야 한다.

이 장은 4장에서 정의된 설계 속성 정보를 구성하는데 도움이 되는 설계 뷰(View)의 개념을 소개한다. 이것은 추가적인 설계 정보를 제공하여 4장을 보충하려는 것은 아니며, 설계 뷰에 대한 포맷 또는 문서화 활동을 규정하지도 않는다.

이 절에서는 다양한 기술적인 관점으로부터 설계 정보에 대한 접근을 용이하게 하기 위하여 설계 엔티티의 권고된 구성과 연관된 속성들을 제시한다. 여기서 권고하는 구성은 유연성을 갖고 조정될 수 있으며, 서류 문서, 설계 언어, 자동 보고서 생성기를 갖는 데이터베이스 관리 시스템, 질의 언어 처리와 같은 다른 매체를 통해 구현될 수 있다. 설계기술서를 위한 접근 구조가 어떻게 준비될 수 있는가를 보여주는 내용 실례가 부록 Ⅰ에 있다.

5.2 설계 뷰

엔티티 속성 정보는 설계의 중요한 모든 측면을 표현하기 위해 여러 방법으로 구성될 수 있다. 이런 과정에서 사용자는 다른 시각 또는 관점으로부터 상세한 설계에 초점을 둘 수 있다. 설계 뷰는 소프트웨어 프로젝트 활동의 요구에 명백하게 부합하는 적합한 설계 엔티티 속성 정보의 부분 집합이다.

각각의 설계 뷰는 소프트웨어 시스템에 대한 별도의 관심사항을 표현하는 것이다. 이러한 설계 뷰는 정보 접근과 동화를 간편하게 하는 간결하고 유용한 형식을 가진 이

해가 쉬운 설계기술서를 제공한다.

정보 접근과 동화를 쉽게 하기 위한 SDD의 권고 구성은 <표 5 - 1>과 같이 설계 뷰를 갖는다.

<표 5 - 1> 권고된 설계 뷰들

설계 뷰	범위	엔티티 속성	실례 표현
분해 기술	설계 엔티티로의 시스템 분할	식별, 유형, 목적, 기능, 종속자	계층적 분해도, 자연 언어
종속성 기술	엔티티들과 시스템 자원들 간의 관계 기술	식별, 유형, 목적, 종속성, 자원	구조도, 자료흐름도, 트랜잭션 다이어그램
인터페이스 기술	시스템을 구성하는 설계 엔티티를 사용하기 위해서 알 필요가 있는 모든 목록	식별, 기능, 인터페이스	인터페이스 파일, 매개 변수 테이블
상세 기술	엔티티의 상세한 내부 설계의 기술	식별, 처리, 데이터	순서도, N-S 차트, PDL

5.2.1 분해 기술

5.2.1.1 범위

분해 기술은 소프트웨어 시스템을 설계 엔티티로 나누어 기록한다. 이는 시스템이 구조화된 방식과 각 엔티티의 목적과 기능을 기술한다. 이것은 식별 속성을 통하여 상세한 기술에 대한 참조를 제공한다.

식별, 유형, 목적, 기능, 종속자에 대한 속성 기술들은 이 설계 뷰에 포함되어야 한다. 이 속성 정보는 모든 설계 엔티티에 대해 제공되어야 한다.

5.2.1.2 용도

분해 기술은 시스템의 주요 설계 엔티티를 식별하기 위해서 엔티티가 특정한 기능들을 이행할 책임이 있는지를 결정하는 것과 설계 엔티티에 대한 요구사항을 추적하는 것을 목적으로 설계자와 유지보수자에 의해 사용될 수 있다. 설계 엔티티들은 정보의 특정한 유형을 담거나 완전성을 위하여 분해서를 재검토하는데 도움을 주기 위해서 주

요한 클래스들로 그룹화될 수 있다.

분해 기술의 정보는 소프트웨어 프로젝트 계획, 감시, 제어를 위하여 프로젝트 관리에서 사용될 수 있다. 이들은 각각의 소프트웨어 구성요소, 목적, 기본 기능을 식별한다. 다른 프로젝트 정보와 함께 이 설계 정보는 비용과 인원에 대한 산정, 개발 노력에 대한 일정을 예측하는데 사용될 수 있다.

형상관리에서는 구성, 추적, 최근에 만들어진 제품들의 변경 관리를 설정하기 위하여 사용될 수 있다. IEEE Std 828-1998을 참조하라. 메트릭 개발자들도 또한 초기 복잡도, 크기와 인원 산정, 개발 시간 매개변수들을 위해 이 정보를 사용할 수 있다. 소프트웨어 품질 보증 수행자는 요구사항 추적 메트릭을 구성하기 위해 이 설계 뷰를 사용할 수 있다.

5.2.1.3 표현

소프트웨어공학에서는 엔티티 분해를 위한 일관된 표준을 제공하는 몇가지 방법론을 설명한다. 이들 방법론들은 쉬운 설계, 독립적인 엔티티를 고려하고 구조적 설계와 정보 은닉과 같은 원칙에 기초를 둔다. 시스템 분해를 기술하기 위해 사용되는 주요 그래픽 기법은 계층적 분해 다이어그램이다. 이 다이어그램은 각 엔티티에 대한 목적과 기능을 기술하는 자연어와 함께 사용될 수 있다.

5.2.2 종속성 기술

5.2.2.1 범위

종속성 기술은 엔티티간의 관계를 기술한다. 이것은 종속 엔티티들을 인식하고 이들의 결합도를 기술하고 요구되는 자원들을 정의한다.

이 설계 뷰는 설계 엔티티간의 상호작용을 위한 전략을 정의하고 시스템 실행이 어떻게, 왜, 어디서, 어떤 수준으로 발생하는가를 쉽게 이해하는데 필요한 정보를 제공한다. 이는 공유된 정보, 실행의 규정된 순서 또는 잘 정의된 매개변수 인터페이스들과 같은 엔티티들 간에 존재하는 관계의 유형을 기술한다.

이 설계 뷰에는 식별, 유형, 목적, 종속성과 자원을 위한 속성 기술이 포함되어야 한다. 이 속성 정보는 모든 설계 엔티티들을 위해 제공되어야 한다.

5.2.2.2 용도

종속성 기술은 요구사항과 설계사항이 변경될 때, 그 영향을 평가하기 위하여 시스템이 어떻게 작업하는지에 대한 전반적인 그림을 제공한다. 이것은 유지보수자들이 시스템 고장 또는 자원 병목현상을 일으키는 엔티티들을 분리하도록 도와줄 수 있다. 이것은 다른 엔티티에 필요하고 처음으로 개발되어야 하는 엔티티들을 식별함으로써 시스템 통합 계획을 생성하는데 도움이 될 수 있다. 이 기술서는 통합 시험 케이스를 생성하는데 도움을 주기 위해서 통합 시험에서 사용될 수 있다.

5.2.2.3 표현

같은 엔티티안에서의 구성요소들간의 관계를 극대화함으로써 엔티티들간의 관계를 최소화 하는데 도움을 주는 수많은 방법론들이 있다. 이러한 방법론들은 낮은 모듈 결합도와 높은 모듈 응집도를 강조한다.

정형적 명세 언어는 잘 정의된 언어를 사용하여 시스템 기능과 데이터, 이들의 내부적 연관관계, 입력과 출력, 다른 시스템 측면에 대한 명세를 제공한다. 설계 엔티티들간의 관계는 자료 흐름도, 구조도 또는 트랜잭션 다이어그램으로 표현된다.

5.2.3 인터페이스 기술

5.2.3.1 범위

엔티티 인터페이스 기술은 엔티티에 의해 제공되는 기능을 올바르게 사용되는지 알려고 하는 설계자, 프로그래머, 시험자에게 필요한 것들을 제공한다. 이 설계 뷰는 소프트웨어 요구사항 명세서에서 제공하지 않은 외적 내적 인터페이스의 자세한 내용을 포함한다.

이 설계 뷰는 각 엔티티에 대한 인터페이스 명세의 집합으로 이루어진다. 식별, 기능, 인터페이스를 위한 속성 기술이 이 설계 뷰에 포함되어야 하며, 이 속성 정보는 모든 설계 엔티티를 위해 제공되어야 한다.

5.2.3.2 용도

인터페이스 기술은 설계자, 프로그래머, 고객 그리고 시험자간의 구속력있는 계약사항으로 작용한다. 이것은 엔티티에 대한 상세 설계 전에 필요한 합의사항를 제공한다. 추가적으로 인터페이스 기술은 고객 문서를 생성하기 위하여 전문 작성자에 의해 사용되거나 고객에 의해 직접적으로 사용될 수도 있다.

설계자, 프로그래머, 시험자는 개발되지 않은 설계 엔티티들을 사용할 필요가 있을

수 있다. 이들 엔티티들은 초기 프로젝트로부터 재사용될 수도 있고, 외부 소스로부터 계약될 수 있고, 다른 개발자들에 의해 생산될 수 있다. 인터페이스 기술은 협력하는 엔티티들이 어떻게 상호작용을 하는지에 대하여 설계자, 프로그래머, 시험자간의 동의 사항을 결정한다. 각 엔티티 인터페이스 기술은 다른 설계자나 프로그래머가 그 엔티티와 상호작용 하는 소프트웨어를 개발하기 위해 알아야하는 필요한 모든 것을 포함해야 한다. 엔티티 인터페이스에 대한 명확한 기술의 매끄러운 통합과 쉬운 유지보수를 위해서 여러 사람이 참여하는 개발에 필수적이다.

5.2.3.3 표현

인터페이스 기술은 화면 포맷, 유효한 입력, 결과 출력을 가지는 각각의 엔티티와 통신할 수 있는 언어를 제공해야 한다. 데이터 위주의 이러한 엔티티들을 위해서, 데이터 속성을 기술하는데 데이터 사전이 활용된다. 사용자에게 높은 시각적 이미지를 주고, 고객이 어떻게 시스템을 인식하고 있는지에 대한 상세 사항을 포함하는 이러한 엔티티들은 기능적 모형, 사용을 위한 시나리오, 상세한 특징들 그리고 상호작용 언어들을 포함해야 한다.

5.2.4 상세 설계 기술

5.2.4.1 범위

상세 설계 기술은 각각의 설계 엔티티에 대한 내부 상세 사항을 포함한다. 이들 상세 사항은 정의, 처리, 데이터에 대한 속성 기술을 포함한다. 이 속성 정보는 모든 설계 엔티티들을 위해 제공되어야 한다.

5.2.4.2 용도

이 설계 뷰는 구현 전에 프로그래머에게 필요한 상세 사항을 포함한다. 상세 설계 기술은 단위 시험 계획을 생성하는데 도움을 주기 위해 사용될 수도 있다.

5.2.4.3 표현

설계 엔티티의 상세 사항을 기술하기 위하여 사용되는 많은 도구들이 있다. 프로그램 설계 언어는 엔티티를 위한 입력, 출력, 지역 데이터, 알고리즘을 기술하기 위해 사용될 수 있다. 설계 엔티티 논리를 기술하기 위한 상용 기법은 메타코드, 구조적 영어, 혹은 나시-쉬나이더만 차트 또는 순서도 같은 그래픽 방법들이 있다.

부록 I

SDD를 위한 목차 실례

다음 목차는 본 표준의 5절에서 나타난 관련 정보와 설계 뷰을 조직화하고 형식화한 여러 방법들 중의 하나이다.

(그림 I -1) SDD 목차

부록 Ⅱ

용어 정의

정의는 본 표준 내용에서 나타나는 용어를 서술한다. 이 문서에서 사용하는 다른 용어들에 대한 정의는 IEEE Std 610.12-1990에서 찾아 볼 수 있다.

Ⅱ.1 엔티티 속성(entity attributes)

설계 엔티티의 명명된 특성이나 속성. 이것은 설계 엔티티에 대한 사실의 진술을 제공한다.

Ⅱ.2 설계 엔티티(design entity)

다른 원소들과 구조적, 기능적으로 구별되고 독립적으로 명명되고 참조되는 설계의 요소(컴포넌트)

Ⅱ.3 설계 뷰(design view)

소프트웨어 프로젝트 활동의 요구에 명확하게 적합한 설계 엔티티 속성 정보의 부분 집합

Ⅱ.4 소프트웨어 설계기술서(SDD)

소프트웨어 설계기술서는 소프트웨어 시스템의 분석, 계획, 구현 및 의사결정을 용이하게 하는 소프트웨어 시스템에 관한 표현이다. 즉 소프트웨어 시스템의 청사진 또는 모형. 소프트웨어 설계기술서는 소프트웨어 설계 정보를 상호교환하기 위한 기초 수단으로 사용한다.

표준작성 공헌자

표준 번호 : TTAS.KO-11.0023

이 표준의 제·개정 및 발간을 위해 아래와 같이 여러분들이 공헌하였습니다.

구분	성명	위원회 및 직위	연락처	소속사
과제 제안		정보통신기술기준연구위원회(SC09)		한국전자통신연구원
표준 초안 제출	김 길 조	S/W개발기술연구반	042-860-6509	한국전자통신연구원
표준 초안 검토 및 작성	김 길 조	S/W개발기술연구반 의장	042-860-6509	한국전자통신연구원
	이 영 곤	S/W개발기술연구반 부의장	02-3219-3810	한국통신
	전 인 걸	S/W개발기술연구반 간사	042-860-6554	한국전자통신연구원
	정 호 원	S/W개발기술연구반 특별위원	02-3290-1938	고려대학교
	김 진 삼	S/W개발기술연구반 위원	042-860-5995	한국전자통신연구원
	신 수 정	S/W개발기술연구반 위원	031-260-2459	한국전산원
표준안 편집 및 감수	김 길 조	S/W개발기술연구반 의장	042-860-6509	한국전자통신연구원
	이 영 곤	S/W개발기술연구반 부의장	02-3219-3810	한국통신
	전 인 걸	S/W개발기술연구반 간사	042-860-6554	한국전자통신연구원
	김 진 삼	S/W개발기술연구반 위원	042-860-5995	한국전자통신연구원
표준안 심의	김 채 규	정보통신S/W기술위원회 의장	042-860-6330	한국전자통신연구원
	김 정 희	정보통신S/W기술위원회 부의장	032-779-8930	삼성전자(주)
	김 길 조	정보통신S/W기술위원회 간사	042-860-6509	한국전자통신연구원
	전 진 옥	정보통신S/W기술위원회 특별위원	02-3486-1234	(주)비트컴퓨터
	백 두 권	정보통신S/W기술위원회 특별위원	02-3290-3192	고려대학교
	이 경 환	정보통신S/W기술위원회 특별위원	02-820-5302	중앙대학교
사무국 담당	권 미 희	정보기술표준부 부장	02-723-7073	한국정보통신기술협회
	이 문 철	정보기술표준부	02-723-7073	한국정보통신기술협회