**An American National Standard**

**IEEE Standard for Software Unit Testing**

**1. Scope and References**

**1.1 Inside the Scope**

Software unit testing is a process that includes the performance of test planning, the acquisition of a test set, and the measurement of a test unit against its requirements.

소프트웨어 단위 테스팅은 테스트 계획의 수행과 test set의 획득, 그리고 그들이 요구하는 것과 반대하는 단위 테스트의 측정을 포함하는 process이다.

Measuring entails the use of sample data to exercise the unit and the comparison of the unit's actual behavior with its required behavior as specified in the unit's requirements documentation.

측정은 unit을 활용시키기 위한 샘플데이터의 사용법과 unit’s 요구사항 문서에 명세 된 것처럼

unit’s의 실제 행동과 그것이 요구되는 행동의 비교가 필요하다.

This standard defines an integrated approach to systematic and documented unit testing.

이 표준은 조직적이고 문서화되었던 단위 시험의 통합된 접근법(연구법-길잡이)을 정의한다．

The approach uses unit design and unit implementation information, in addition to unit requirements, to determine the completeness of the testing.

시험의 완전함을 결정하기 위해，unit의 요구 사항들에 더하여，이 접근법은 unit 디자인과 unit 구현 정보를 사용한다.

This standard describes a testing process composed of a hierarchy of phases, activities, and tasks and defines a minimum set of tasks for each activity.

이 표준은 단계의 계층, 활동들, 그리고 task 그리고 각 activity의 최소화된 tasks set의 정의로 구성된 testing process를 묘사한다.

Additional tasks may be added to any activity.

추가의 작업들은 어떤 활동이라도 추가하게 될지도 모른다．

This standard requires the performance of each activity.

이 표준은 각 활동의 실행을 필요로 한다.

For each task within an activity, this standard requires either that the task be performed, or that previous results be available and be reverified.

활동 내의 각 task를 위해 이 표준은 어느 쪽이든지 요구된다. task는 수행되어야 한다 또는 이전의 결과들은 이용 가능하고 검사된다.

This standard also requires the preparation of two documents specified in ANSI/IEEE Std 829-1983 [2]

이 표준도 ANSI/IEEE Std 829-1983[2]에 명세 된 2개의 문서들의 준비를 필요로 한다

These documents are the Test Design Specification and the Test Summary Report.

이 문서들은 Test 설계 명세서 와 Test 요약 보고서이다．

General unit test planning should occur during overall test planning.

전반적인 단위 시험 계획은 전반적인 test planning동안 일어나야 한다．

This general unit test planning activity is covered by this standard, although the balance of the overall test planning process is outside the scope of this standard.

비록 전반적인 test planning process의 균형은 이 표준의 범위 밖에 있지만, general unit test planning activity는 이 standard에 있다.

This standard may be applied to the unit testing of any digital computer software or firmware.

이 표준은 어떤 디지털 컴퓨터 소프트웨어 또는 firmware의 unit testing에 적용하게 될지도 모른다．

However, this standard does not specify any class of software or firmware to which it must be applied, nor does it specify any class of software or firmware that must be unit tested.

그러나, software 또는 firmware의 어떤 클래스도 반드시 지정 되어야 한다거나 반드시 지정되지 않아야 한다는 것이 이 표준에서는 소프트웨어의 어떤 class 또는 firmware도 지정되지 않았다.

This standard applies to the testing of newly developed and modified units.

이 표준은 새로이 발전되고 수정되었던 units를 testing하는 것을 적용한다．

This standard is applicable whether or not the unit tester is also the developer.

Unit 시험자가 또한 개발자 일지라도 하여간 이 표준은 적용할 수 있다．

**1.2 Outside the Scope**

The results of some overall test planning tasks apply to all testing levels (for example, identify security and privacy constraints).

약간의 전반적인 test planning tasks의 결과들은 모든 testing 수준들(예를 들어 identify security(기밀 보호성 확인)와 privacy constraints(비밀제약))에 적용한다.

Such tasks are not considered a part of the unit testing process, although they directly affect it.

비록 그들이(tasks) 직접 그것에(unit testing process)영향을 끼치지만, 그러한 tasks는 unit testing process의 일부로서 간주되지 않는다.

While the standard identifies a need for failure analysis information and software fault correction, it does not specify a software debugging process.

이 표준이 실패 분석 정보와 소프트웨어 결함 정정의 필요성을 확인할지라도, 그것을 software debugging process라고 지정하지 않는다.

This standard does not address other components of a comprehensive unit verification and validation process, such as reviews (for example, walkthroughs, inspections), static analysis (for example, consistency checks, data flow analysis), or formal analysis (for example, proof of correctness, symbolic execution).

이 표준은 비평들(e.g. 검토회, 검사들) 정적 분석 (e.g. 일관성 검사, data flow 분석), 또는 공식적인(formal) 분석(e.g. 정확함의 증명, 기호 실행)과 같은 광범위한 unit 확인과 유효한 process들에 관해 언급하지 않는다.

This standard does not require the use of specific test facilities or tools.

이 표준은 특정의 test 설비들 또는 도구들의 사용이 필요하지 않는다．

This standard does not imply any particular methodology for documentation control, configuration management, quality assurance, or management of the testing process.

이 표준은 자료관리, 형상관리, 품질보증 또는 testing process의 관리를 위한 어떤 특정한 방법론을 의미하지 않는다.

**1.3 References**

This standard shall be used in conjunction with the following publications.

이 표준은 동시 발행될 아래의 출판물들에 사용될 것이다.

[1] ANSI/IEEE Std 729-1983, IEEE Standard Glossary of Software Engineering Terminology.

[2] ANSI/IEEE Std 829-1983, IEEE Standard for Software Test Documentation.

**2. Definitions**

This section defines key terms used in this standard but not included in ANSI/IEEE Std 729-1983 [1] or ANSI/IEEE Std 829-1983 [2].

이 섹션은 이 표준에서 사용되는 key 단어들을 정의한다. 그러나 ANSI/IEEE Std 729-1983[1] or ANSI/IEEE Std 829-1983[2]에는 포함되지 않는다.

characteristic: See: data characteristic or software characteristic.

특징: See: 데이터 특징 또는 소프트웨어 특징

data characteristic: An inherent, possibly accidental, trait, quality, or property of data (for example, arrival rates, formats, value ranges, or relationships between field values).

data characteristic: 하나의 고유의, 아마도 비본질성，특징，품질 또는 data의 속성（예를 들면 arrival rates，formats，value ranges 또는 field values 사이의 관계들）

feature: See: software feature. (feature: 특징)

incident: See: software test incident. (incident: 우발 사건, 부수적인 것)

nonprocedural programming language: A computer programming language used to express the parameters of a problem rather than the steps in a solution (for example, report writer or sort specification languages). Contrast with procedural programming language.

비 절차형 프로그래밍 언어: 컴퓨터 프로그래밍 언어로 solution에서 명령을 하나 실행 시키는 것(step)보다도 문제의 매개변수를 나타내기 위해 사용된다. procedural programming language와 대조된다.

procedural programming language: A computer programming language used to express the sequence of operations to be performed by a computer (for example, COBOL). Contrast with nonprocedural programming language.

절차형 프로그래밍 언어: 컴퓨터(예를 들면 COBOL)로 실행되는 operations의 순서를 나타내기 위해 사용되는 프로그래밍 언어. nonprocedural programming language와 대조된다.

software characteristic: An inherent, possibly accidental, trait, quality, or property of software (for example, functionality, performance, attributes, design constraints, number of states, lines of branches).

software characteristic: 하나의 고유의, 아마도 비본질성，특징，품질 또는 software의 속성(예를 들어 기능성, 운용, 속성들(attributes), 설계 제약들, number of states, 분기의 수

software feature: A software characteristic specified or implied by requirements documentation (for example, functionality, performance, attributes, or design constraints).

software feature: 요구 사항 문서에 의해 명기 또는 내포되는 소프트웨어 특징. (예를 들어, 기능성, 운용, 속성들, 설계 제약들)

software test incident: Any event occurring during the execution of a software test that requires investigation.

software test incident: software test를 실행하는 동안 일어나는 어떤 이벤트로 조사가 요구된다.

state data: Data that defines an internal state of the test unit and is used to establish that state or compare with existing states.

test unit의 내부의 상태를 정의하고 상태를 확립하거나 또는 현재의 상태와 비교하기 위하여 사용되는 데이터이다.

test objective: An identified set of software features to be measured under specified conditions by comparing actual behavior with the required behavior described in the software documentation.

test objective(조사 목적): software문서에 묘사된 요구되는 행동과 실제 행동을 비교하는 것에 의해 하나의 지정된 software feature의 집합은 지정된 조건하에 측정된다.

test set architecture: The nested relationships between sets of test cases that directly reflect the hierarchic decomposition of the test objectives.

test set architecture: test objectives의 계층적인 분해를 직접적으로 반영하는 test cases의 sets사이의 이입되는(포개지는) 관계

test unit: A set of one or more computer program modules together with associated control data, (for example, tables), usage procedures, and operating procedures that satisfy the following conditions:

test unit: 아래의 조건을 만족하는 하나 또는 하나 이상의 computer program modules과 함께 연합된 제어 자료(e.g. tables), 사용 과정 및 절차

A test unit may occur at any level of the design hierarchy from a single module to a complete program.

하나의 Test unit은 하나의 module에서 완전한 프로그램까지 design 계층의 어떤 level에서도 일어날 수 있다.

Therefore, a test unit may be a module, a few modules, or a complete computer program along with associated data and procedures.

그러므로 test unit은 연합된 data와 절차들과 함께 하나의 module, 약간의 modules 또는 완전한 컴퓨터 프로그램일지도 모른다.

1) All modules are from a single computer program

모든 modules은 단 한 개의 컴퓨터 프로그램으로부터 있다.

2) At least one of the new or changed modules in the set has not completed the unit test

unit test가 완료되지 않은 적어도 1개의 새로운 또는 변화된 modules을 가지고 있다.

A test unit may contain one or more modules that have already been unit tested.

Test unit은 이미 unit test가 준비된 하나 또는 그 이상의 modules를 포함할지도 모른다.

3) The set of modules together with its associated data and procedures are the sole object of a testing process

modules의 집합은 그것이 연합된 data와 절차들과 함께 testing process의 단 하나의 object이다.

unit: See: test unit.

unit requirements documentation: Documentation that sets forth the functional, interface, performance, and design constraint requirements for the test unit.

unit requirements documentation: test unit을 위한 외적 기능, interface, 성능과 설계 제약 요구사항을 기록한 문서화 작업

**3. Unit Testing Activities**

This section specifies the activities involved in the unit testing process and describes the associated input, tasks, and output. The activities described are as follows:

이 섹션은 unit testing process가 포함된 활동을 지정(명기)하고 연합된 input, tasks, output을 기술한다. 그 활동의 기술은 다음과 같다.

1) Perform test planning phase

 test planning 단계를 실행한다.

a) Plan the general approach, resources, and schedule

일반적인 접근법, 자원, 그리고 일정을 계획하라

b) Determine features to be tested

test될 특징을 결정하라.

c) Refine the general plan

대체적인 계획안을 정의해라.

2) Acquire test set phase

test set 단계를 습득하라.

a) Design the set of tests

tests의 집합을 설계하라.

b) Implement the refined plan and design

정의한 계획과 설계를 구현하라.

3) Measure test unit phase

test unit phase를 측정하라.

a) Execute the test procedures

test 절차를 실행하라.

b) Check for termination

종단(종료)를 검사하라.

c) Evaluate the test effort and unit

test effort(공수)와 unit을 평가하라.

When more than one unit is to be unit tested (for example, all those associated with a software project), the Plan activity should address the total set of test units and should not be repeated for each test unit.

하나 이상의 unit가 unit 시험될 때(예를 들어 software project와 관련된 모든 것들) 시험 활동은 전체 test units의 집합을 언급해야 하고, 각 test unit를 반복해서는 안 된다.

The other activities must be performed at least once for each unit.

다른 활동은 각 unit을 위해 적어도 한번 수행되어야 한다.

Under normal conditions, these activities are sequentially initiated except for the Execute and Check cycle as illustrated in Fig 1.

일반적인 조건들 하에, 이 활동들은 Fig 1. 에 도시되는 것처럼 Execute와 Check cycle을 제외하고 순차적으로 시작한다.

 When performing any of the activities except Plan, improper performance of a preceding activity or external events (for example, schedule, requirements, or design changes) may result in the need to redo one or more of the preceding activities and then return to the one being performed.

Plan을 제외하고 활동들의 무엇이든지 실행할 때, 진행 활동 또는 외부의 사건(예를 들어 일정, 요구사항들, 또는 설계 변경)의 부적당한 실행은 하나 또는 그 이상의 활동들의 진행을 다시 하는 필요를 가져올지도 모른다. 게다가 실행되고 있는 하나로 돌아올지도 모른다.



Fig 1 – Unit Testing Activities

During the testing process, a test design specification and a test summary report must be developed. Other test documents may be developed.

testing 과정(process) 동안, test 설계 명세서와 test 요약 리포트는 자세히 기술해야 한다. 다른 test 문서들도 자세히 기술해야 할지도 모른다.

 All test documents must conform to the ANSI/IEEE Std 829-1983 [2]. In addition, all test documents must have identified authors and be dated.

모둔 test 문서는 반드시 ANSI/IEEE Std 829-1983 [2]를 따라야 한다. 게다가, 모든 test 문서는 반드시 저자의 신원은 확인되어야 하고 연대는 추정되어야 한다.

The test design specification will derive its information from the Determine, Refine, and Design activities. The test summary report will derive its information from all of the activities.

test 설계 명세서는 Determine, Refine, Design 활동들로부터 그 정보를 끌어낼 것이다. test 개요 리포트는 모든 활동들로부터 그것의 정보를 끌어낼 것이다.

**3.1 Plan the General Approach, Resources, and Schedule.**

일반적인 접근법, 자원들, 그리고 일정을 계획해라.

General unit test planning should occur during overall test planning and be recorded in the corresponding planning document.

일반적인 unit test planning은 전체의 test planning동안 일어나야만 하고 일치하는 계획문서에서 기록되어야 한다.

**3.1.1 Plan Inputs:** Inputs을 계획해라

1) Project plans: 프로젝트 계획들

2) Software requirements documentation: 소프트웨어 요구사항 문서

**3.1.2 Plan Tasks:** 계획 작업들

(1)*Specify a General Approach to Unit Testing.*

: Unit testing을 위해 일반적인 접근법(방법)을 열거하라.

Identify risk areas to be addressed by the testing. Specify constraints on characteristic determination (for example, features that must be tested), test design, or test implementation (for example, test sets that must be used).

testing에 의해 언급되는 위험 지역을 확인해라. characteristic determination(특질 있는 결정-

예를 들어 반드시 test된 특징), test 설계, 또는 test 구현(예를 들어, 반드시 사용된 test sets)의 제약을 열거(상술)하라.

Identify existing sources of input, output, and state data (for example, test files, production files, test data generators).

입력과 출력 그리고 state data(예를 들어, test 파일들, 제작 파일들, test data generators(생성 프로그램-발생기))의 source들이 존재하는지 확인해라.

Identify general techniques for data validation.

자료 확인(비준)을 위한 일반적인 기술들을 확인해라

Identify general techniques to be used for output recording, collection, reduction, and validation.

기록, collection, 정리, 확인을 출력하기 위해 사용되는 일반적인 기술을 확인해라.

Describe provisions for application software that directly interfaces with the units to be tested.

test된 units을 직접적으로 인터페이스와 연결한 응용 소프트웨어를 위한 조항(규정)을 기술하라.

(2)*Specify Completeness Requirements.*

요구사항 완전성을 열거하라.

Identify the areas (for example, features, procedures, states, functions, data characteristics, instructions) to be covered by the unit test set and the degree of coverage required for each area.

unit test set과 각 area를 위한 요구된 적용범위의 정도(등급)에 의해 덮이는(에 포함되는 아니면 보호되는) areas(예를 들어, 특징들, 절차들, 상태들, functions, data characteristics, 명령어들) 를 확인해라.

When testing a unit during software development, every software feature must be covered by a test case or an approved exception.

소프트웨어 개발과정 동안 하나의 unit을 testing할 때, test case(선례) 또는 승인된 예외에 의해 모든 소프트웨어 특징은 반드시 보유해야 한다.

The same should hold during software maintenance for any unit testing.

마찬가지로 어떠한 unit testing이라도 소프트웨어 유지기간 중에 보유해야 한다.

When testing a unit implemented with a procedural language (for example, COBOL) during software development, every instruction that can be reached and executed must be covered by a test case or an approved exception, except for instructions contained in modules that have been separately unit tested.

소프트웨어 개발 중에 절차적인 언어(e.g. COBOL)로 구현된 unit을 testing할 때, modules (개별적으로(단독으로) test된 unit)을 포함하는 명령어들을 제외하고, test case 또는 승인된 예외에 의해 도달가능하고(=be reached) 실행될 수 있는 모든 명령어는 보호되어야 (포함되어야) 한다.

The same should hold during software maintenance for the testing of a unit implemented with a procedural language.

마찬가지로 절차상의 언어로 구현된 unit의 testing을 소프트웨어 유지보수 중에 (기억장치에)남겨둬야 한다.

(3)*Specify Termination Requirements.*

종료 요구사항을 열거하라

Specify the requirements for normal termination of the unit testing process.

unit testing process(처리단위)의 일반적인 종료 요구사항들을 열거하라.

Termination requirements must include satisfying the completeness requirements.

종료 요구사항들은 반드시 충분한 요구사항 완전성을 포함해야 한다.

Identify any conditions that could cause abnormal termination of the unit testing process (for example, detecting a major design fault, reaching a schedule deadline) and any notification procedures that apply.

어떤 조건 - 비정상적인 종료를 야기할 수 있는 unit testing process(예를 들어, 주요한 설계 결점을 찾거나, 일정 마감에 도달했는지)와 적용하는 어떤 통지절차들 - 을 확인해라(식별해라)

(4) *Determine Resource Requirements.*

resource 요구사항을 결정하라.

Estimate the resources required for test set acquisition, initial execution, and subsequent repetition of testing activities.

test set의 획득, 최초의 실행, 그리고 testing 활동들 다음의 반복에 요구된 자원을 측정하라.

Consider hardware, access time (for example, dedicated computer time), communications or system software, test tools, test files, and forms or other supplies.

하드웨어, 접근 시간(e.g. 바쳐진(?) 컴퓨터 시간), 통신수단들 또는 system software, test 도구들, test 파일들 그리고 다른 form(서식) 또는 다른 소모품들을 고려해라.

Also consider the need for unusually large volumes of forms and supplies.

또한 보통과는 달리(드문 방식으로) 많은 양의 서식과 소모품들의 필요성을 고려해라.

Identify resources needing preparation and the parties responsible.

준비와 책임 당사자들이 필요한 자원을 확인해라.

Make arrangements for these resources, including requests for resources that require significant lead time (for example, customized test tools).

이 자원들을 – 신청한 자원(중요한 lead time을 요구하는- e.g. customized test tools)을 포함한 – 준비(정렬)하라. (customize: 자기 취미에 맞도록 설정을 바꾸다)

Identify the parties responsible for unit testing and unit debugging.

unit testing과 unit debugging의 책임이 있는 당사자들을 확인해라.

Identify personnel requirements including skills, number, and duration.

능력(기술들), 동료, 존속기간을 포함하는 인사 요구사항을 확인해라.

(5)*Specify a General Schedule.*

Specify a schedule constrained by resource and test unit availability for all unit testing activity.

모든 unit testing 활동을 위한 자원과 test unit 유효성에 의해 강요 받는 일정을 확인해라.

**3.1.3 Plan Outputs**

(1) General unit test planning information (from 3.1.2 (1) through (5) inclusive)

정보를 계획하고 있는 전반적인 단위 시험

(2) Unit test general resource requests-if produced from 3.1.2 (4)

전반적인 자원 requests-if가 3.1.2에서 만들어 내었던 단위 시험

**3.2 Determine Features To Be Tested:** test된 특징들을 결정하라

**3.2.1 Determine Inputs:** 입력을 결정하라.

(1) Unit requirements documentation: unit 요구사항들 문서화

(2) Software architectural design documentation-if needed:

소프트웨어 구조 설계 문서화 – 필요하다면

**3.2.2 Determine Tasks:** task들을 결정해라.

(1)*Study the Functional Requirements.*

기능적 요구사항들을 연구해라.

Study each function described in the unit requirements documentation.

unit 요구 사항 문서에 기술된 각 기능에 대해 연구해라.

Ensure that each function has a unique identifier.

각 기능이 유일한 식별자를 가지는 것을 (보증한다).

When necessary, request clarification of the requirements.

필요하다면, 요구 사항들의 설명(해명)을 요청해라.

(2)*Identify Additional Requirements and Associated Procedures.*

추가적인 요구 사항 들과 관련된 절차들을 확인해라.

Identify requirements other than functions (For example, performance, attributes, or design constraints) associated with software characteristics that can be effectively tested at the unit level.

요구 사항들 이외의 unit level에서 효과적으로 test할 수 있는 소프트웨어 characteristics와 관련된 기능(e.g. 성능, 속성들 또는 설계 제약들)들을 확인해라.

Identify any usage or operating procedures associated only with the unit to be tested.

오직 test된 unit과 연관된 어떤 사용(법)또는 운용 절차들을 확인해라.

Ensure that each additional requirement and procedure has a unique identifier.

각 추가적인 요구 사항과 절차는 유일한 식별자를 가지는 것을 보증한다.

When necessary, request clarification of the requirements.

필요하다면，필요 조건들의 설명 요청해라．

(3)*Identify States of the Unit.*

unit의 상태를 확인해라.

If the unit requirements documentation specifies or implies multiple states (For example, inactive, ready to receive, processing) software, identify each state and each valid state transition.

만약 unit 요구 사항 문서가 복수의 상태 소프트웨어(e.g. 비활동, ready to receive, processing)를 명기하거나 의미하면, 각 상태와 각 유효한(타당한) 상태 변화를 확인해라.

Ensure that each state and state transition has a unique identifier.

각 주립이고 주립의 이행이 유일한 식별자를 가질 것을 확인해라．

When necessary, request clarification of the requirements.

필요하다면，필요 조건들의 설명을 요청해라．

(4)*Identify Input and Output Data Characteristics.*

data characteristics의 입력과 출력을 확인하라.

Identify the input and output data structures of the unit to be tested.

test된 unit의 입출력 자료 구조들을 확인하라.

For each structure, identify characteristics, such as arrival rates, formats, value ranges, and relationships between field values.

각 구조(=structure)를 위해 characteristics – arrival rates(도착 비율들), formats, 값의 범위, 그리고 field 값들 사이의 관계와 같은 – 을 확인하라.

For each characteristic, specify its valid ranges.

각 characteristic를 위해 그들의 유효한 범위를 지정하라.

Ensure that each characteristic has a unique identifier.

각 characteristic는 유일한 식별자를 가지는 것을 보증하라.

When necessary, request clarification of the requirements.

필요하다면，필요 조건들의 설명을 요청하라.

(5)*Select Elements to be Included in the Testing.*

testing에 포함시키게 되는 요소들을 선택하라.

Select the features to be tested. test된 특징들을 선택하라.

Select the associated procedures, associated states, associated state transitions, and associated data characteristics to be included in the testing.

관련된 절차들, 관련된 상태들，관련된 상태의 변화 그리고 관련된 data characteristics(testing에 포함시키게 되는)를 선택하라.

Invalid and valid input data must be selected.

무효하고 유효한 입력 자료들은 반드시 선택되어야 한다.

When complete testing is impractical, information regarding the expected use of the unit should be used to determine the selections.

완전한 시험이 비현실(비 실용)적일 때, 기대되는(예상되는) unit의 사용에 관한 정보는 선택들을 결정하기 위해 사용되어야 한다.

Identify the risk associated with unselected elements.

선택되지 않은 요소들과 관련된 위험을 확인하라.

Enter the selected features, procedures, states, state transitions, and data characteristics in the *Features* *to be Tested* section of the unit’s Test Design Specification.

unit의 Test 설계 명세서의 ‘test된 특징들’의 섹션에 선택된 특징들, 절차들, 상태들, 상태 변화, 그리고 characteristics를 넣어라.

**3.2.3 Determine Outputs**

(1) List of elements to be included in the testing (from 3.2.2 (5))

요소들(from 3.2.2)의 목록은 testing에 포함시키게 된다.

(2) Unit requirements clarification requests-if produced from 3.2.2 (1) through (4) inclusive

unit 요구 사항들 설명을 요청한다 – 만약 3.2.2의 (1)에서부터 (4)를 포함하여 만들어지면

**3.3 Refine the General Plan**

전반적인 계획을 세밀히 구별하라(상세히 논술하라)

**3.3.1 Refine Inputs:** 입력들을 상세히 논술하라.

(1) List of elements to be included in the testing (from 3.2.2 (5))

요소들(from 3.2.2)의 목록은 testing에 포함시키게 된다.

(2) General unit test planning information (from 3.1.2 (1) through (5) inclusive)

전반적인 단위 시험 계획 정보(3.1.2의 1에서 5까지 포함하여)

**3.3.2 Refine Tasks:** Tasks를 상세히 논술하라.

(1)*Refine the Approach:* 접근법을 상세히 논술하라.

Identify existing test cases and test procedures to be considered for use.

사용을 위해 고려되는 현재의 test cases들과 test 절차들을 확인하라.

Identify any special techniques to be used for data validation.

data 확인을 위해 사용되는 특별한 어떤 기술들을 확인하라.

Identify any special techniques to be used for output recording, collection, reduction, and validation.

출력 기록, collection, 정리, 그리고 확인을 위해 사용되는 어떤 특별한 기술을 확인하라.

Record the refined approach in the *Approach Refinements* section of the unit’s test design specification.

unit의 test 설계 명세서의 Approach Refinements 섹션에 상세히 논술된 접근법을 기록하라.

(2)*Specify Special Resource Requirements.*

특별한 자원 요구 사항들을 확인하라.

Identify any special resources needed to test the unit (for example, software that directly interfaces with the unit).

Unit (e.g. unit에 직접적으로 인터페이스로 접속하는 소프트웨어)을 test하기 위해 필요로 하는 어떤 특별한 자원을 확인하라.

Make preparations for the identified resources.

확인된 자원을 위한 사전 준비를 하다.

Record the special resource requirements in the *Approach Refinements* section of the unit’s test design specification.

unit의 test 설계 명세서의 Approach Refinements 섹션에 특별한 자원 요구 사항들을 기록하라.

(3) *Specify a Detailed Schedule.* – 상세한 계획을 지정하라.

Specify a schedule for the unit testing based on support software, special resource, and unit availability and integration schedules.

지원 소프트웨어, 특별한 자원, unit 유효성과 통합 일정에 근거하여 unit testing을 위한 일정을 지정하라.

Record the schedule in the *Approach Refinements* section of the unit’s test design specification.

unit의 test 설계 명세서의 Approach Refinements 섹션에 일정을 기록하라.

3.3.3 출력들을 정련해라

(1) 정보（완전히 3.3.2（1）（3）로부터 포함하게 되는）를 계획하고 있는 특정의 단위 시험

(2) 만일 3.3.2（2）에서 만들어 내게 되면 특별한 자원이 요청하는 단위 시험．

**3.3.3 Refine Outputs –** 출력들을 상세히 논술하라.

(1) Specific unit test planning information (from 3.3.2 (1) through (3) inclusive)

특정의 단위 시험을 계획하는 정보(3.3.2의 (1)에서 (3)을 포함하는)

(2) Unit test special resource requests - if produced from 3.3.2 (2).

특별한 자원이 요청하는 단위 시험 – 만약 3.3.2의 (2)로부터 만들어졌다면.

**3.4 Design the Set of Tests:** tests의 집합 설계

**3.4.1 Design Inputs:** 입력들을 설계하라.

(1) Unit requirements documentation: 단위 요구 사항 문서

(2) List of elements to be included in the testing (from 3.2.2 (5))

(3.2.2의 (5)로부터)testing에 포함시키게 되는 요소들의 목록

(3) Unit test planning information (from 3.1.2 (1) and (2) and 3.3.2 (1))

(3.1.2의 (1)과 3.3.2의 (1)로부터) 단위 시험을 계획하는 정보

(4) Unit design documentation: 단위 설계 문서

(5) Test specifications from previous testing - if available:

이전의 시험으로부터의 시험 명세서 – 만약 이용할 수 있다면

**3.4.2 Design Tasks:** 태스크들을 설계하라.

(1) *Design the Architecture of the Test Set. -* 시험 집합의 구조 설계를 설계하라.

Based on the features to be tested and the conditions specified or implied by the selected associated elements (for example, procedures, state transitions, data characteristics), design a hierarchically decomposed set of test objectives so that each lowest-level objective can be directly tested by a few test cases.

시험된 특징들과 선택된 연관된 요소들(e.g. 절차들, 상태 변화, data characteristics)에 의해 지정되거나 포함된 조건들에 근거하여 시험 계층적으로 분석된 목적들의 집합을 설계하면 각 최하의 수준 목적은 약간의 시험 사례에 의해 직접적으로 시험될 수 있다.

Select appropriate existing test cases. 적절한 현재의 시험 사례를 선택하라.

Associate groups of test-case identifiers with the lowest-level objectives.

최하 수준 목적들로 시험 사례 식별자들의 집단을 결합시켜라.

Record the hierarchy of objectives and associated test case identifiers in the *Test Identification* section of the unit’s test design specification.

목적들의 계층과 연관된 시험 사례 식별자들을 단위 시험 설계 명세서의 Test identification 섹션에 기록하라.

(2) *Obtain Explicit Test Procedures as Required.*

필요에 따라 명백한 시험 절차들을 획득하라.

A combination of the unit requirements documentation, test planning information, and test-case specifications may implicitly specify the unit test procedures and therefore minimize the need for explicit specification.

단위 요구사항 문서, 시험 계획 정보, 그리고 시험 항목 규정의 조합은 내재적으로 단위 시험 절차들을 지정할지도 모르고, 그 결과 명백한 명세서의 필요성을 최소화한다.

Select existing test procedures that can be modified or used without modification.

수정될 수 있거나 수정 없이 사용될 수 있는 현재의 시험 절차들을 선택하라.

Specify any additional procedures needed either in a supplementary section in the unit’s test design specification or in a separate procedure specification document.

단위 시험 설계 명세서의 보충하는 섹션 또는 독립된 절차 규정 문서의 어느 하나라도 필요 되는 추가적인 시험 절차들을 규정한다(명시한다)

Either choice must be in accordance with the information required by ANSI/IEEE Std 829-1983 [2].

어느 하나를 선택하더라도 ANSI/IEEE Std 829-1983 [2].에 의해 요구된 정보에 따라야 한다.

When the correlation between test cases and procedures is not readily apparent, develop a table relating them and include it in the unit’s test design specification.

시험 사례들과 절차들 사이의 상호작용이 쉽사리 명백하지 않을 때, 그들과 관련된 일람표(표)를 개발하고 단위 시험 설계 명세서에 포함시켜라.

(3) *Obtain the Test Case Specifications.* – 시험 사례 명세서(규정)들을 얻어라．

Specify the new test cases. - 새로운 시험 사례들을 지정해라．

Existing specifications may be referenced. 현재의 명세서(규정)들은 참조사항을 붙이게 될지도 모른다．

Record the specifications directly or by reference in either a supplementary section of the unit’s test design specification or in a separate document.

직접적으로 또는 단위 시험 설계명세서의 보충하는 섹션이나 독립된 문서의 어느 한쪽의 참조에 의해 명세서(규정)들을 기록하라.

Either choice must be in accordance with the information required by ANSI/IEEE Std 829-1983 [2].

어느 한 쪽의 선택은 ANSI/IEEE Std 829-1983 [2]에 의해 요구된 정보와 일치하여야 한다.

(4) *Augment, as Required, the Set of Test-Case Specifications Based on Design Information.*

필요에 따라서, 설계 정보에 근거한 시험 사례 명세서들의 집합을 증가시켜라.

Based on information about the unit’s design, update as required the test set architecture in accordance with 3.4.2 (1).

단위 설계에 관한 정보에 근거하여 필요하다면 3.4.2 (1)과 일치하는 시험 집합 구조를 갱신하라.

Consider the characteristics of selected algorithms and internal data structures.

선택된 알고리즘과 내부의 데이터 구조들의 characteristics를 고려하라.

Identify control flows and changes to internal data that must be recorded.

제어 흐름들과 반드시 기록되어야 하는 내부의 데이터 변화들을 확인해라.

Anticipate special recording difficulties that might arise, for example, from a need to trace control flow in complex algorithms or from a need to trace changes in internal data structures (for example, stacks or trees).

예를 들어 복잡한 알고리즘의 제어 흐름을 추적하는 필요성으로부터 또는 내부의 데이터 구조(e.g. stack이나 트리)들의 변화들을 추적하는 필요성으로부터 일어날지도 모르는 특별한 난제들을 기록하는 것을 예기하라.

When necessary, request enhancement of the unit design (for example, a formatted data structure dump capability) to increase the test-ability of the unit

필요하다면, 단위의 시험 능력의 상승을 위해 단위 설계(e.g. 포맷된 데이터 구조의 덤프 능력)의 향상을 요청하라.

Based on information in the unit’s design, specify any newly identified test cases and complete any partial test case specifications in accordance with 3.4.2 (3).

단위 설계의 정보에 근거하여 어떤 새롭게 지정된 시험 사례를 지정하고 3.4.2 (3)에 따르는 부분적인 시험 사례 명세서들을 완료하라.

(5) *Complete the Test Design Specification.* - 시험 디자인 규정을 완료해라

Complete the test design specification for the unit in accordance with ANSI/IEEE Std 829-1983 [2].

ANSI/IEEE Std 829-1983 [ 2 ]와 일치하는 단위를 위해 시험 설계 규정(명세서)들을 완료해라

**3.4.3 Design Outputs** 출력들을 설계하라.

(1) Unit test design specification (from 3.4.2 (5)) - 단위 시험 설계 규정（3.4.2 (5)로부터）

(2) Separate test procedure specifications - if produced from 3.4.2 (2)

만약 3.4.2（2）에서 만들어 내게 되면 시험 절차 규정들을 분리하라.

(3) Separate test-case specifications if produced from 3.4.2 (3) or (4)

만약 3.4.2의 (3) 또는 (4)로부터 만들어졌다면 시험 사례 규정들을 분리하라.

(4) Unit design enhancement requests - if produced from 3.4.2 (4)

만약 3.4.2의 (4)에서 만들어 졌다면 단위 설계 향상을 요청하라.

**3.5 Implement the Refined Plan and Design -** 정확한 계획과 설계를 구현해라.

**3.5.1 Implement Inputs –** 입력들을 구현하라.

(1) Unit test planning information (from 3.1.2 (1), (4), and (5) and 3.3.2 (1) through (3) inclusive)

단위 시험 계획 정보 (3.1.2 (1), (4), (5)와 3.3.2 (1)에서 (3)을 포함한 것으로부터)

(2) Test-case specifications in the unit test design specification or separate documents (from 3.4.2 (3) and (4)

단위 시험 설계 명세서들 또는 독립된 문서들(3.4.2의 (3), (4)로부터)의 시험 사례 규정들.

(3) Software data structure descriptions - 소프트웨어 데이터 구조 설명서

(4) Test support resources – 시험 지원 자원들

(5) Test items – 시험 데이터 항목들

(6) Test data from previous testing activities - if available

이전의 시험 활동들로부터의 시험 자료 – 만약 이용가능 하다면

(7) Test tools from previous testing activities - if available

이전의 시험 활동들로부터의 시험 도구들 – 만약 이용가능 하다면

**3.5.2 Implement Tasks -** Task들을 구현하라.

(1) *Obtain and Verify Test Data.* - 테스트 데이터를 얻고 검증하라.

Obtain a copy of existing test data to be modified or used without modification.

현재의 수정된 시험 데이터 또는 수정 없이 사용된 사본을 얻어라.

Generate any new data required. – 요구된(필요한) 어떤 새로운 자료들을 생성해라

Include additional data necessary to ensure data consistency and integrity.

데이터의 일관성 및 무결성을 보증하기 위해 필요한 추가적인 데이터를 포함해라.

Verify all data (including those to be used as is) against the software data structure specifications.

소프트웨어 데이터 구조 규정들에 반대하는 모든 데이터(그들을 대용하는 것을 포함한)를 검증하라.

When the correlation between test cases and data sets is not readily apparent, develop a table to record this correlation and include it in the unit’s test design specification.

시험 사례들과 데이터 집합들 사이의 상관관계가 쉽사리 명백하지 않을 때, 이 상관관계를 기록하는 테이블을 개발하고 단위 시험 설계 명세서에 그것을 포함하라.

(2) *Obtain Special Resources.* ­- 특별한 자원들을 얻어라

Obtain the test support resources specified in 3.3.2 (2).

3.3.2（2）에 지정된 시험 지원 자원들을 얻어라

(3) *Obtain Test Items.* - 시험 데이터 항목들을 얻어라．

Collect test items including available manuals, operating system procedures, control data (for example, tables), and computer programs

이용 가능한 매뉴얼, 운영 시스템 절차들, 제어 데이터(예를 들어, 표) 그리고 컴퓨터 프로그램을 포함한 시험 데이터 항목들을 모아라

\*Obtain software identified during test planning that directly interfaces with the test unit.

시험 계획(직접적으로 인터페이스로 접속하는 시험 단위)을 식별(확인)하는 동안 소프트웨어를 얻어라.

When testing a unit implemented with a procedural language, ensure that execution trace information will be available to evaluate satisfaction of the code-based completeness requirements.

절차적 언어로 구현되는 단위를 시험할 때, 실행 추적 정보는 코드기반 완전성의 만족을 평가하기 위해 이용가능 할 것이라는 것을 확보해야 한다.

Record the identifier of each item in the *Summary* section of the unit’s test summary report.

단위 시험 요약 보고서의 Summary 섹션에 각 데이터 항목의 식별자를 기록해라.

**3.5.3 Implement Outputs** 출력들을 구현하라.

(1) Verified test data (from 3.5.2 (1)) – 검증된 시험 데이터 (3.5.2의 (1)로부터)

(2) Test support resources (from 3.5.2 (2)) – 시험 지원 자원들 (3.5.2 (2)로부터)

(3) Configuration of test items (from 3.5.2 (3)) – 시험 데이터 항목들의 구성(3.5.2 (3)로부터)

(4) Initial summary information (from 3.5.2 (3)) – 최초의 요약 정보(3.5.2의 (3)로부터)

**3.6 Execute the Test Procedures** – 시험 절차들을 실행하라

**3.6.1 Execute Inputs** – 입력들을 실행하라.

(1) Verified test data (from 3.5.2 (1)) – 검증된 시험 데이터(3.5.2의 (1)로부터)

(2) Test support resources (from 3.5.2 (2)) – 시험 지원 자원들(3.5.2의 (2)로부터)

(3) Configuration of test items (from 3.5.2 (3)) – 시험 데이터 항목의 구성(3.5.2의 (3)로부터)

(4) Test-case specifications (from 3.4.2 (3) and (4)) – 시험 사례 명세서들(3.4.2의 (3)과 (4)로부터)

(5) Test procedure specifications (from 3.4.2 (2))-if produced

시험 절차 명세서들(3.4.2의 (2)로부터) – 만약 만들어졌다면

(6) Failure analysis results (from debugging process)-if produced

실패 분석 결과들(디버깅 프로세스로부터) – 만약 만들어졌다면

**3.6.2 Execute Tasks** – 태스크들을 실행하라.

(1) *Run Tests.* 시험들을 실행하라

Set up the test environment. – 시험 환경을 설정하라.

Run the test set. – 시험 집합을 실행하라.

Record all test incidents in the *Summary of Results* section of the unit’s test summary report.

단위 시험 요약 보고서의 Summary of Results 섹션에 시험 시험에서 일어난 일을 기록하라.

(2) *Determine Results.* – 결과들을 결정하라.

For each test case, determine if the unit passed or failed based on required result specifications in the case descriptions.

만약 단위가 사례 설명서의 요구된 결과 명세서에 근거하여 통과되거나 실패하게 되면 각 시험 사례를 결정하라.

Record pass or fail results in the *Summary of Results* section of the unit’s test summary report.

단위 시험 요약 보고서의 Summary of Results 섹션에 통과하고 실패한 결과들을 기록하라.

Record resource consumption data in the *Summary of Activities* section of the report.

보고서의 Summary of Activities에 자원 소모 데이터를 기록하라.

When testing a unit implemented with a procedural language, collect execution trace summary information and attach it to the report.

절차적 언어로 구현된 단위를 시험할 때, 실행 추적 요약 정보를 모으고 그것을 보고서에 붙여라.

For each failure, have the failure analyzed and record the fault information in the Summary of Results section of the test summary report.

각 실패들은, 실패를 분석하고, 시험 요약 보고서의 Summary of Results섹션에 기록하라.

Then select the applicable case and perform the associated actions.

그리고 나서 적용할 수 있는 사례를 선택하라 그리고 관련된 동작들을 실행하라.

*Case 1: A Fault in a Test Specification or Test Data.*

시험 명세서 또는 시험 데이터의 결점

Correct the fault, record the fault correction in the *Summary of Activities* section of the test summary report, and rerun the tests that failed.

결점을 정정하고, 장애 교정을 시험 요약 보고서의 Summary of Activities에 기록하고, 실패했던 시험들을 재 실행하라.

*Case 2: A Fault in Test Procedure Execution.* – 시험 절차 실행의 결점.

Rerun the incorrectly executed procedures.

부정확하게(틀리게) 실행된 절차들을 재실행하라.

*Case 3: A Fault in the Test Environment (for example, system software).*

시험 환경의 결점(예를 들어, 시스템 소프트웨어)

Either have the environment corrected, record the fault correction in the *Summary of Activities* section of the test summary report, and rerun the tests that failed OR prepare for abnormal termination by documenting the reason for not correcting the environment in the *Summary*

*of Activities* section of the test summary report and proceed to check for termination (that is, proceed to activity 3.7).

시험 요약 보고서의 Summary of Activities 섹션에 장애 교정을 기록하고, 실패한 시험들을 재실행하거나 시험 요약 보고서의 Summary of Activities에 환경을 정정하지 않은 이유를 문서로 증명한 것에 의해 비정상적인 종료를 준비하고, 종료에 대한 점검을 속행한다는 것 중 하나로 환경은 정정된다. (이것은, proceed to activity 3.7)

*Case 4: A Fault in the Unit Implementation.* – 단위 구현의 결점

Either have the unit corrected, record the fault correction in the *Summary of Activities* section of the test summary report, and rerun all tests OR prepare for abnormal termination by documenting the reason for not correcting the unit in the *Summary of Activities* section of the test summary report and proceed to check for termination (that is, proceed to activity 3.7).

시험 요약 보고서의 Summary of Activities에 장애 교정을 기록하고, 모든 시험들을 재실행하거나

시험 요약 보고서의 Summary of Activities에 환경을 정정하지 않은 이유를 문서로 증명한 것에 의해 비정상적인 종료를 준비하고, 종료에 대한 점검을 속행한다는 것 중 하나로 단위는 정정된다. (즉, activity 3.7까지 진행하라)

*Case 5: A Fault in the Unit Design.* – 단위 설계의 결점

Either have the design and unit corrected, modify the test specification and data as appropriate, record the fault correction in the *Summary of Activities* section of the test summary report, and rerun all tests OR prepare for abnormal termination by documenting the reason for not correcting the design in the *Summary of Activities* section of the test summary report and proceed to check fortermination (that is, proceed to activity 3.7).

시험 명세서와 데이터를 적절하게 수정하고, 시험 요약 보고서의 Summary of Activities 섹션에 장애 교정을 기록하고, 시험 요약 보고서의 Summary of Activities에 환경을 정정하지 않은 이유를 문서로 증명한 것에 의해 비정상적인 종료를 준비하고, 종료에 대한 점검을 속행한다는 것 중 하나로 설계와 단위는 정정된다. (이것은, proceed to activity 3.7)

NOTE - The cycle of Execute and Check Tasks must be repeated until a termination condition defined in 3.1.2 (3) is satisfied (See Fig 3).

참고사항 - 사이클의 실행 및 검사 태스크들은 3.1.2의 (3)에서 만족한(Fig. 3을 보라) 정의된 종료 조건이 될 때까지 반복해야 합니다.

Control flow within the Execute activity itself is pictured in Fig 2.

Execute 활동내의 제어 흐름 그 자체는 Fig 2에서 그려진다.



Fig 3 Control Flow Within the Check Activity

**3.6.3 Execute Outputs** – 출력들을 실행하라.

(1) Execution information logged in the test summary report including test outcomes, test incident descriptions, failure analysis results, fault correction activities, uncorrected fault reasons, resource consumption data and, for procedural language implementations, trace summary information (from3.6.2 (1) and (2))

실행 정보는 시험 결과들, 시험 사건 설명서들, 실패 분석 결과들, 장애 교정 활동들, 정정되지 않은 장애 이유들, 자원 소모 데이터 그리고 절차적 언어 실현들, 추적 요약 정보를 포함한 시험 요약 보고서에 기록된다. (3.6.2의 (1)과 (2)로부터)

(2) Revised test specifications - if produced from 3.6.2 (2)

수정된 시험 규정들 - 만약 3.6.2 (2) 에서 만들어 내게 되면

(3) Revised test data - if produced from 3.6.2 (2)

수정된 테스트 데이터 - 만약 3.6.2 (2) 에서 만들어 내게 되면

**3.7 Check for Termination** – 종료를 검사하라.

**3.7.1 Check Inputs –** 입력들을 검사하라.

(1) Completeness and termination requirements (from 3.1.2 (2) and (3))

완전성과 종료 요구사항들（3.1.2 (2)와 (3)로부터）

(2) Execution information (from 3.6.2 (1) and (2))

실행 정보（3.6.2 (1)과 (2)로부터）

(3) Test specifications (from 3.4.2 (1) through (3) inclusive) - if required

시험 규정들(3.4.2 (1)에서 (3)을 포함한 것으로부터) – 만약 필요하다면

(4) Software data structure descriptions - if required

소프트웨어 데이터 구조 기술들 – 만약 필요하다면

**3.7.2 Check Tasks**  - 태스크들을 검사해라.

(1) *Check for Normal Termination of the Testing Process.*

testing process의 정규(normal) 종료를 검사하라.

Determine the need for additional tests based on completeness requirements or concerns raised by the failure history.

요구사항들의 완전성 또는 실패 역사에 의해 올라간 관심에 근거하여 필요한 추가적인 시험들을 결정하라.

For procedural language implementations, analyze the execution trace summary information (for example, variable, flow).

절차적 언어 구현을 위해, 실행 추적 요약 정보를 분석하라(예를 들어, 변수, 흐름)

If additional tests are *not* needed, then record normal termination in the *Summary of Activities* section of the test summary report and proceed to evaluate the test effort and unit (that is, proceed to activity 3.8).

만약 추가적인 시험들이 필요하지 않다면, 시험 요약 보고서의 Summary of Activities 섹션의 정규 종료를 기록하고, 시험 공수와 단위의 평가를 속행한다. (즉, activity 3.8까지 진행하라)

(2) *Check for Abnormal Termination of the Testing Process.*

testing process의 비정상적인 종료를 검사하라

If an abnormal termination condition is satisfied (for example, uncorrected major fault, out of time) then ensure that the specific situation causing termination is documented in the *Summary of Activities* section of the test summary report together with the unfinished testing and any uncorrected faults.

만약 비정상의 종료 조건이 만족된다면(예를 들어, 교정되지 않은 주요한 결함(장애), 시간 초과)

종료를 일으키는 특정한 상황은 시험 요약 보고서의 Summary of Activities에 상세히 기록된다는 (종료되지 않은 시험과 어떤 정정되지 않은 결점들과 함께)것을 확인하다.

Then proceed to evaluate the test effort and unit (that is, proceed to activity 3.8).

그리고 나서 시험 노력과 단위의 평가를 계속한다. (즉, activity 3.8까지 진행하라)

(3) *Supplement the Test Set.* – 시험 집합의 보충(추가)

When additional tests are needed and the abnormal termination conditions are not satisfied, supplement the test set by following steps (a) through (e).

추가의 시험들이 필요하고 비정상의 종료 조건들이 만족되지 않을 때, 다음의 (a)에서 (e)까지 단계들에 의해 시험 집합(class)들을 보충하라.

(a) Update the test set architecture in accordance with 3.4.2 (1) and obtain additional test-case specifications in accordance with 3.4.2 (3).

3.4.2의 (1)과 일치하는 시험 집합 구조를 갱신하고, 3.4.2의 (3)과 일치하는 추가의 시험-사례 규정들을 얻어라.

(b) Modify the test procedure specifications in accordance with 3.4.2 (2) as required.

필요에 따라서 3.4.2의 (1)과 일치하는 시험 절차 규정들을 수정하라.

(c) Obtain additional test data in accordance with 3.5.2 (1).

3.5.2의 (1)과 일치하는 추가의 시험 데이터를 얻어라

(d) Record the addition in the *Summary of Activities* section of the test summary report.

시험 요약 보고서의 Summary of Activities 섹션에 추가를 기록하라.

(e) Execute the additional tests (that is, return to activity 3.6).

추가적인 시험들을 실행하라. (즉, activity 3.6으로 돌아가라)

**3.7.3 Check Outputs** – 출력들을 검사하라.

(1) Check information logged in the test summary report including the termination conditions and any test case addition activities (from 3.7.2 (1) through (3) inclusive)

종료 조건들과 어떤 시험 사례 추가 활동들 (3.7.2의 (1)에서 (3)을 포함하는 것으로부터)을 포함한 시험 요약 보고서의 기록된 정보를 검사하라

(2) Additional or revised test specifications - if produced from 3.7.2 (3)

추가되거나 수정되었던 시험 규정들 – 만약 3.7.2 (3)에서 만들어 지면

(3) Additional test data - if produced from 3.7.2 (3)

추가의 테스트 데이터 – 만약 3.7.2 (3)에서 만들어 지면

**3.8 Evaluate the Test Effort and Unit**

시험 공수와 단위를 평가하라.

**3.8.1 Evaluate Inputs –** 출력들을 평가하라.

(1) Unit Test Design Specification (from 3.4.2 (5)

단위 시험 설계 규정들(3.4.2의 (5)로부터)

(2) Execution information (from 3.6.2 (1) and (2))

실행 정보(3.6.2의 (1)과 (2)로부터)

(3) Checking information (from 3.7.2 (1) through (3) inclusive)

검사 정보(3.7.2의 (1)에서 (3)을 포함한 것으로부터)

(4) Separate test-case specifications (from 3.4.2 (3) and (4)) - if produced

독립된 시험-사례 규정들(3.4.2의 (3)과 (4)로부터) – 만약 만들어졌다면

**3.8.2 Evaluate Tasks** – 태스크들을 평가하라.

(1) *Describe Testing Status.* – 시험 상태를 설명하라.

Record variances from test plans and test specifications in the *Variances* section of the test summary report.

시험 요약 보고서의 Variances 섹션에 시험 계획들과 시험 규정들로부터 변화들을 기록하라.

Specify the reason for each variance.

각 변화의 이유를 상술하라.

For abnormal termination, identify areas insufficiently covered by the testing and record reasons in the *Comprehensiveness Assessment* section of the test summary report.

비정상의 종료는, 시험하는 것과 시험 요약 보고서의 Comprehensiveness Assessment 섹션에 이유들을 기록하는 것으로 불충분하게 포함되는 범위들을 확인하라

Identify unresolved test incidents and the reasons for a lack of resolution in the *Summary of Results* section of the test summary report.

미해결의 시험 사건을 확인하고 시험 요약 보고서의 Summary of Results 섹션에 해결의 부족을 논리적으로 생각하라.

(2) *Describe Unit’s Status.* - 단위의 상태를 기술해라．

Record differences revealed by testing between the unit and its requirements documentation in the *Variances* section of the test summary report.

시험 요약 보고서의 Variances 섹션에 단위들 요구사항들의 문서화와 단위 사이의 시험에 의해 밝혀진 차이들을 기록하라.

Evaluate the unit design and implementation against requirements based on test results and detected fault information.

시험 결과들과 발견된 결점 정보에 근거한 요구사항들에 반대하여 단위 설계와 구현을 평가하라.

Record evaluation information in the *Evaluation* section of the test summary report.

시험 요약보고서의 Evaluation 섹션에 평가 정보를 기록하라.

(3) *Complete the Test Summary Report.*

시험 요약 보고서를 완성시켜라.

Complete the test summary report for the unit in accordance with ANSI/IEEE Std 829-1983 [2].

ANSI/IEEE Std 829-1983 [2]와 일치하는 단위를 위한 시험 요약 보고서를 완성시켜라.

(4) *Ensure Preservation of Testing Products.*

시험 산출물들의 보존을 확인하라.

Ensure that the testing products are collected, organized, and stored for reference and reuse.

시험 산출물들을 재사용성과 참조를 위해 수집하고 조직하고 저장하는 것을 확인하라.

These products include the test design specification, separate test-case specifications, separate test procedure specifications, test data, test data generation procedures, test drivers and stubs, and the test summary report.

이 산출물들은 시험 설계 명세서, 독립된 시험-사례 명세서, 독립된 시험 절차 명세서들, 시험 데이터, 시험 데이터 생성 절차들, 시험 드라이버들과 스터브들, 그리고 시험 요약 보고서를 포함한다.

**3.8.3 Evaluate Outputs -** 출력들을 평가하라

(1) Complete test summary report (from 3.8.2 (3))

완전한 시험 요약 보고서(3.8.2의 (3)으로부터)

(2) Complete, stored collection of testing products (from 3.8.2 (4))

완전한, 시험 산출물들의 저장된 집합 (3.8.2의 (4)로부터)

**관계 용어**

**phase: 단계**

컴퓨터에서 업무를 처리할 경우 한 번에 주기억 장치에서 꺼내어 실행하는 프로그램의 처리 단계. 예를 들면, 5단계로 된 프로그램이란 주기억 장치의 인출, 실행을 5회 반복하여 처리를 완료하도록 한 프로그램을 말한다.

**activity: 활동**

문서의 전송과 같이 응용 프로세스에 의해 주어진 작업의 단위를 구성하는 한 단위의 데이터를 처리하는 것. 활동에는 개시와 종료가 있으며, 그 사이에는 소(小)동기점, 대(大)동기점이 존재한다.


**task: 태스크**

**[컴]** ①운영 체계의 모드 중 배치 모드와 다중 프로그래밍 모드에서 실행되는 프로그램을 구분하기 위해 도입된 개념으로, 다중 프로그래밍된 상태에서의 루틴 또는 컴퓨터의 수행 내용. 다중 프로그래밍 능력을 갖는 시스템에서 프로그램에 대한 순차 처리를 하기 위해 작업 단계라는 개념이 도입되었다. 따라서 작업은 둘 이상의 작업 단계로 구분되고 작업 단계 각각은 둘 이상의 태스크로 나뉜다. 태스크와 작업 단계의 또 다른 차이점은 전자의 경우 시스템 자원을 사용하기 위해서 동시에 경쟁이 가능하지만 후자의 경우는 순서적으로 시스템 자원을 사용할 수 있다.
②컴퓨터 시스템 내 활동의 기본적 단위. 이것은 외부적인 특성만 명시되고 내부적 구조와 운용은 명시되지 않는다. 태스크의 외부적 특성은 연구나 통제 받고 있는 작업의 내용에 따른다. 즉, 이것들은 입력과 출력의 매개 변수나 변수, 자원의 요구 사항, 수행 시간 등을 포함한다.

**walkthrough: 검토회**

설계서나 프로그램의 오류를 탁상에서 발견하기 위한 회의. 상세 설계서나 프로그램 설계서는 처음부터 프로그램 논리나 부호화의 내용까지를 대상으로 하여 그 내용에 탈락이나 모순, 기타 잘못된 것이 있는지의 여부를 심의한다. 검토의 목적은 개발 담당자가 서로 검사함으로써 문제점을 뽑아내고 동시에 다른 사람이 알 수 있는 프로그램을 작성하는 것으로, 공정 관리나 개인의 능력을 평가하는 자리는 아니다.

**symbolic execution: 기호 실행**

①프로그램 단위 내의 입력 변수에 문자 값보다 기호 값을 할당시키는 확인 기법. 입력에 대한 기호 값을 식 내의 연산 수에 전달함으로써 프로그램이 분석될 수 있다. 그 결과로 나온 기호식은 모든 중간 계산과 판단이 항상 기호 입력에 대해서 표시될 수 있도록 간소화된다.
②입력 데이터의 실제 값보다 기호들을 사용하여 프로그램 수행이 모방되고, 프로그램 출력은 이러한 기호들을 포함한 논리적/수학적 수식들로 표현되는 검증 기법.

**facility: 설비**

①운용 체계상에서 특정한 목적을 충족시킬 수 있도록 설계된 것.
②데이터 처리 시스템의 사용이 어느 정도 편리하게 될 수 있는가를 나타내는 척도로서 시스템의 성능과 장치의 전체적인 생산성을 측정하는 주요한 요인.
③통신 서비스를 제공하기 위해 사용되는 모든 것으로서 통신 경로의 총칭.

**module: 모듈**

①잘 정의된 한 가지 일을 수행하는 프로그램의 논리적인 일부분. 주 프로그램은 논리적으로 몇 개의 모듈로 나뉠 수 있다. 모듈은 여러 프로그램 작성자에 의해 나뉘어 작성되는 성질을 지닌다. 그러므로 모듈이 서로 모여 하나의 완전한 프로그램으로 만들어질 수 있다.
②큰 시스템 중 비교적 독립적인 한 부분.

**requirement: 요구 사항**

①어떤 문제를 해결하거나 특정의 목적을 위하여 사용자가 필요로 하는 조건이나 능력.
②계약, 표준, 명세 또는 다른 형식으로 제시된 문서에 적합하여 시스템이나 시스템 구성 요소가 갖추어져야 할 조건이나 능력. 요구 사항들은 시스템이나 시스템 구성 요소의 후속 개발 단계의 자료가 된다.

**production: 제작**

소프트웨어의 소재 정보를 작성하거나 수집하는 작업. 작업 수행자가 저작권자인 경우에는 ‘製作’이라고 쓴다. 수탁 업무 등으로 인해 소프트웨어의 소재 정보를 작성, 수집하는 사람 측에 완성품에 대한 저작권이 없는 경우에는 ‘制作’이라고 쓴다.

**generator: 발생기**

특정한 입력 조건이나 개략적인 부호로부터 특정 목적의 프로그램을 만들어 내는 프로그램. 주로 매크로 명령어의 확장에 의한 방식이 많다. 보고서 프로그램 작성기(RPG), 정렬 프로그램 발생기 등이 있다.

**validation: 확인**

지정된 목적의 사용을 위한 특정 요구사항을 충족하는 객관적 증거에 대한 검사 및 규정에 의한 확인. 참고 가. 설계 및 개발 단계에서 확인은 사용자 요구에 대한 부합성을 판단하기 위 해 제품을 검사하는 프로세스를 말한다. 나. 확인은 일반적으로 정의된 운영 조건하에서 최종 제품을 대상으로 수행된 다. 확인은 그 이전 단계에서도 필요할 수 있다.다. "확인된(Validated)"이란 단어는 상응되는 상태를 나타내기 위해 사용된다. 라. 사용 목적이 다양하다면 다중 확인이 수행될 수 있다.

**recording: 기록**

화상 신호를 하드 복사로 변환시키는 것. 기록하는 방법에는 감열 기록, 통전 감열 기록(通電感熱記錄), 잉크 분사 기록, 정전 기록, 방전 기록 등이 있으며, 팩스나 인쇄기에 사용된다.

**completeness: 완전성**

모든 정의된 객체가 존재하는 정도.

**hold: 보유**

정보를 다른 장치에 전송한 후에도 그 정보를 원래의 저장 장치에 계속 보관하게 하는 기능.

**notification: 통지**

전송 관리 망 내의 피 관리 시스템이 자체적으로 발생한 상황을 관리시스템으로 보고하기 위하여 스스로 발생하는 일련의 절차.

**identifier: 식별자**

①데이터 항목에 이름을 부여하여 일시적으로 규정하거나, 그 데이터의 어떤 특성을 표시하기 위해서 사용하는 기호 또는 기호의 집합.

②이름을 붙이거나, 지시하거나, 위치를 나타내는 데 사용하는 부호. 데이터 구조, 데이터 항목, 프로그램 위치 등에 관련된다.
③자료 항목을 식별하거나 이름 붙이는 데, 또는 그 데이터의 특정 성질을 나타내는 데 사용하는 문자나 문자의 집합.

**inactive: 비활동**

컴퓨터의 메모리 내에 로드 되었지만 아직 실행되고 있지 않은 작업의 성질

**table: 표**

여러 가지 표나 일람표를 신속하게 참조할 수 있도록 적당한 형태로 자료를 모아 놓은 것. 자료는 연속적인 기억 장치 영역에 저장되거나 행과 열의 배열에 기록된다. 행과 열이 주어지면 특정한 정보가 있는 곳을 알게 된다. 또한 각 항목의 일부로 저장되어 있는 키를 지정함으로써 원하는 항목을 꺼낼 수도 있다. 보통 항목 표는 배열로 구현되나 2진 트리(binary tree), 해시표(hash table), 비트리(B-tree) 등으로 구성되는 경우도 있다.

**fault correction: 장애 교정**

**stub: 스터브(전파)**

고주파 회로에서 임피던스 매칭이나 신호의 선별적 필터링을 위하여 신호선외에 부가적으로 장치되는 조그만 선로.

동작 주파수의 파장에 대한 스터브의 길이 비율과, 선단의 단락, 개방 등 터미네이션 상태에 따라 유도성, 용량성 또는 순 저항성으로 동작한다.