제 12 장 테스트 계획

12.1 개 요

테스트 계획은 테스트 관리 프로세스의 시작 활동으로 동적 테스트를 효과적이고 효율적으로 수행하기 위한 계획 수립을 목적으로 한다. 즉, 테스트 목적을 달성하기 위한 테스트 컨텍스트를 설정하고 다양한 측면에서 적절한 테스트 전략을 수립하고 테스트 수행을 위한 계획을 수립한다.

테스트 계획은 조직 테스트 프로세스의 결과물을 바탕으로 수립되며 동적 테스트 프로세스 수행을 위한 제반사항을 포함한다. 그림 12.1은 테스트 계획 활동이 조직 테스트 프로세스 및 동적 테스트 프로세스와 어떤 관계가 있는지를 보여 준다.

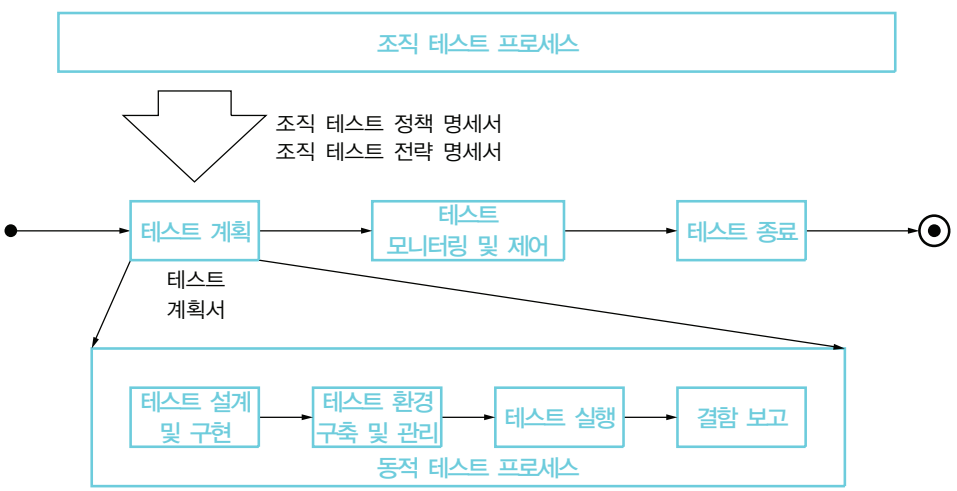

그림 12.1 테스트 계획

- 조직 테스트 정책 명세서 및 조직 테스트 전략 명세서를 참고하여 테스트 계획서를 작성한다. 예를 들어, 조직 테스트 정책 명세서에서 명시된 테스트 목적, 테스트 프로세스, 테스트 조직 및 역할 등을 바탕으로 현재 수행할 테스트에 맞추어 조정해 적용한다. 마찬가지로 조직 테스트 전략 명세서에서 명시된 위험 관리, 테스트 문서화, 형상 관리, 결함 관

리 등의 전략을 바탕으로 구체적인 테스트 계획을 수립한다.

- 테스트 계획서는 동적 테스트 프로세스 수행에 대한 구체적인 계획을 포함한다. 즉, 테스트 설계 및 구현 활동은 테스트 계획서에 정의된 테스트 설계 기법, 테스트 환경 요건, 테스트 데이터 요건 등을 바탕으로 수행된다. 재테스팅 및 리그레션 테스팅, 테스트 중단 및 재시작 조건 등은 테스트 실행 활동에서 반영된다.

- 테스트 계획서를 바탕으로 동적 테스트 프로세스가 수행되는지에 대한 모니터링 및 제어가 수행된다. 예를 들어, 테스트 메트릭을 활용하여 동적 테스트 프로세스의 수행 현황을 파악한다. 그리고 테스트 완료 기준을 바탕으로 수행된 테스트의 완료 여부를 평가하고 이를 테스트 종료 활동에서 작성하는 테스트 종료 보고서에 기록한다.

테스트 계획은 동적 테스트 프로세스 및 다른 테스트 관리 프로세스 활동에 대한 구체적인 방법과 수행 계획을 포함한다. 테스트 계획을 구성하는 상위 수준의 항목은 다음과 같다.

- **테스트 컨텍스트 명세**: 테스트 대상, 테스트 범위 등 테스트 수행의 배경이 되는 다양한 요소를 식별하고 명세한다.

- **위험 분석**: 테스트 프로젝트의 목적 달성에 방해가 될 수 있는 제품 및 프로젝트 유형의 위험 요소를 식별하고, 분석하여 조치 계획을 수립한다.

- **테스트 전략 수립**: 테스트 산출물, 테스트 설계 기법, 테스트 환경 요건 및 테스트 데이터 요건 등 테스트를 효과적이고 효율적으로 수행하기 위한 다양한 전략을 결정한다.

- **테스트 수행 계획 수립**: 테스트 조직/인력과 역할, 테스트 활동 및 일정 등의 수행 계획을 수립한다.

테스트 계획서는 테스트 계획 활동의 산출물로서 테스트 목적을 달성하기 위해 결정된 다양한 테스트 계획 사항을 포함한다. 표 12.1은 테스트 계획서를 구성하는 상위 수준의 항목을 보여 준다.

표 12.1 **테스트 계획 활동 산출물 요약**

산출물	주요 항목
테스트 계획서	• 테스트 컨텍스트 • 위험 분석 • 테스트 전략 • 테스트 수행 계획

12.2 테스트 컨텍스트 명세

구체적인 테스트 전략을 수립하기에 앞서서 테스트 수행의 배경이 되는 다양한 요소들을 테스트 컨텍스트로 식별하고 명세한다. 표 12.2는 테스트 컨텍스트를 구성하는 항목을 보여준다.

표 12.2 테스트 컨텍스트 항목

항목	설명
테스트 계획 유형	테스트 계획서가 목표로 하는 테스트 계획의 유형을 명시한다.
테스트 대상	테스트하고자 하는 대상을 식별하고 명확하게 기술한다.
테스트 범위	기능 및 비기능 등 테스트하고자 하는 피처들을 나열한다.
가정 및 제약 사항	표준, 정책, 전략, 일정, 비용, 인력 등 테스트를 수행할 때 반드시 고려/준수해야 할 사항을 파악한다.
이해관계자	개발팀, 마케팅팀 등 테스트 프로젝트의 결과에 영향을 받는 이해관계자를 파악하고 의사소통 방법을 결정한다.

12.2.1 테스트 계획 유형

테스트 계획서가 목표로 하는 테스트 계획의 유형을 명시한다. 테스트 계획서는 프로젝트 테스트 계획서와 개별 테스트 계획서로 분류될 수 있다. 표 12.3은 테스트 계획의 유형을 보여 준다.

표 12.3 테스트 계획의 유형

테스트 계획 유형		설명
프로젝트 테스트 계획		테스트 대상에 대한 전체 개별 테스트를 포함하는 종합적인 계획
개별 테스트 계획	레벨 테스트 계획	컴포넌트/통합/시스템/인수 등 레벨 테스트의 수행에 대한 계획
	유형 테스트 계획	기능 및 품질(성능/신뢰성/보안성 등) 등 유형 테스트의 수행에 대한 계획

프로젝트 테스트 계획은 테스트 대상에 대한 최상위 수준의 종합적인 테스트 계획이다. 그러므로 프로젝트 테스트 계획에는 테스트 대상 및 테스트 범위를 바탕으로 수립한 개별 테스트 계획이 포함된다.

12.2.2 테스트 대상

테스트 대상, 즉, 테스트를 수행함으로써 결함을 검출하거나 품질을 평가하고자 하는 대상을 명시한다. 테스트 대상은 시스템 전체가 될 수도 있고, 시스템을 구성하는 개별 요소가 될 수도 있다. 또는 개별 요소 간의 연결을 대상으로 해서 테스트를 수행할 수도 있다. 그리고 이러한 테스트 대상의 유형을 기준으로 테스트 레벨이 결정된다. 표 12.4는 각 레벨 테스트별로 테스트 대상의 예를 보여 준다.

표 12.4 테스트 레벨별 테스트 대상 예

테스트 레벨	테스트 대상 예
컴포넌트 테스트	시스템을 구성하는 서브 시스템, 컴포넌트, 클래스, 함수 등의 개별적인 요소
통합 테스트	시스템을 구성하는 각 요소 간의 연결로서 서브 시스템 간의 연결, 컴포넌트 간의 연결, 클래스 간의 연결, 함수 간의 연결 등
시스템 테스트	시스템 전체
인수 테스트	시스템 전체

테스트 대상이 명확하게 식별되도록 테스트 대상의 이름과 함께 버전, 그리고 테스트 대상이 접근할 수 있는 위치 등도 명시하도록 한다.

또한, 테스트 대상을 이해하고 테스트 범위 설정을 도울 수 있도록 테스트 대상의 미션이나 비즈니스 목표 등을 기술한다. 직접 기술하지 않고 관련 문서를 명시할 수도 있다. 예를 들어, 시스템 테스트에는 요구사항 명세서가 필요하고 통합 테스트에는 구조 설계 명세서가 필요하다. 인수 테스트에는 설치 가이드, 사용자 매뉴얼 등이 필요하다.

12.2.3 테스트 범위

테스트 범위는 테스트를 통해서 확인하고자 하는 테스트 대상의 기능/비기능 등의 특성을 의미한다. 테스트 범위에 포함된 기능 및 품질 특성에 초점을 두고 테스트 케이스 및 테스트 절차가 개발되고 테스트 환경이 준비된다.

각 테스트 레벨 별로 초점을 두는 테스트 범위가 다를 수 있다. 예를 들어, 표 12.5는 각 테스트 레벨별로 대표적인 테스트 범위를 보여 준다. 일반적으로 컴포넌트 테스트 및 통합 테스트에서는 기능에 초점을 둔다. 다만 높은 수준의 위험도를 가진 컴포넌트에 대해서는 추

가적으로 성능, 신뢰성 등의 비기능 특성을 테스트 범위에 포함할 수도 있다. 시스템 테스트 및 인수 테스트에는 기능 요구사항과 함께 요구사항 명세서에 정의된 비기능 요구사항도 테스트 범위에 포함된다.

표 12.5 테스트 레벨별 테스트 범위 예

테스트 레벨	테스트 범위
컴포넌트 테스트	• 각 컴포넌트의 기능적 요구사항 • 최고 수준의 위험도를 가지는 컴포넌트에 대해서는 성능과 신뢰성 등 포함
통합 테스트	• 컴포넌트 연결 간의 동작 • 최고 수준의 위험도를 가지는 컴포넌트 간 성능과 신뢰성 등 포함
시스템 테스트	• 시스템에 대한 기능 요구사항 전체 • 시스템에 대한 비기능 요구사항 전체
인수 테스트	• 시스템에 대한 기능 요구사항 전체 • 시스템에 대한 비기능 요구사항 전체

테스트 범위는 테스트하고자 하는 피처를 나열하는 방법으로 기술한다. 즉, 테스트 대상을 두고 테스트로 확인하고자 하는 개별적인 기능 및 품질 특성을 나열해 나가며 테스트 범위를 정의한다. 예를 들어, 표 12.6은 테스트 범위를 테스트에 포함될 피처를 이용하여 정의한 예를 보여 준다. 테스트 범위에 포함될 피처를 나열할 때는 피처의 유형도 함께 명시한다. 예를 들어, 침입 탐지, 침입 발생 알림, 알림 설정, 활성화 설정은 기능 유형의 피처가 되며, 탐지 판단 성능과 침입 발생 알림 성능은 성능 유형의 피처이다.

표 12.6 테스트 범위 – 포함될 피처

피처 유형	피처 명
기능	침입 탐지
	침입 발생 알림
	알림 설정
	활성화 설정
성능	탐지 판단 성능
	침입 발생 알림 성능
호환성	적외선 카메라 호환성
	움직임 센서 호환성
	소리 센서 호환성
	Android 단말기 호환성

테스트 범위에서 제외될 피처도 제외하는 사유와 함께 명시적으로 나열할 필요가 있다. 제외 사유는 위험도가 매우 낮거나 기존에 충분히 테스트 된 부분이거나, 이해관계자와 제외하기로 합의되었거나 하는 것들이 될 수 있다. 그리고 제외된 피처는 다음 테스트 범위에 포함될 수 있다. 표 12.7은 테스트 범위에서 제외될 피처와 제외 사유에 대한 예를 보여 준다.

표 12.7 **테스트 범위 - 제외될 피처**

피처 유형	제외 피처	제외 사유
기능	알림 방법	기존에 충분한 테스트 수행됨
호환성	운영 콘솔 호환성	위험도가 매우 낮음
	iOS 단말기 호환성	이후에 테스트할 범위

12.2.4 가정 및 제약사항

테스트를 수행할 때 참고해야 할 가정 및 제약 사항을 기술한다. 테스트를 수행하면서 반드시 준수해야 하는 표준, 조직 테스트 정책 및 조직 테스트 전략, 테스트 프로젝트 고객과의 계약, 테스트 일정, 비용, 투입 인력, 자동화 도구 그리고 테스트 환경 등이 있다면 기술한다.

12.2.5 이해관계자

테스트와 관련된 이해관계자를 식별한다. 여기에는 테스트에 요구사항을 제시하거나 테스트 결과에 관심을 가지는 이해관계자가 포함된다. 예를 들어, 테스트 대상을 구축 및 운영하거나 직접 사용하는 개발팀, 마케팅팀, 운영지원팀 등은 이해관계자가 될 수 있다.

각 이해관계자가 어떤 관점에서 테스트 프로젝트에 관심을 가지는지, 즉, 각 이해관계자가 테스트 프로젝트에 어떤 요구가 있는지를 식별한다. 또한, 각 이해관계자와의 의사소통 방법을 정의한다. 테스트 계획서를 포함하여 테스트 현황 보고서, 테스트 종료 보고서 등의 테스트 관리 프로세스 산출물은 이해관계자의 의사소통에서 중요한 역할을 한다.

예를 들어, 테스트 계획을 이해관계자와 공유하고 승인을 받은 후에 테스트 설계 및 구현 활동을 시작하는 것이 바람직하다. 테스트 현황 보고서를 통해 정기적으로 이해관계자와 테스트 상황을 공유하고 최신 정보를 교환할 필요가 있다. 그리고 테스트가 종료된 후에는

테스트 프로세스의 성과를 테스트 종료 보고서에 기록하여 이해관계자와 공유하는 것이 바람직하다.

12.3 위험 분석

테스팅을 수행하면서 고려해야 하는 위험 요소를 식별한다. 만약 조직 테스트 전략에서 이미 식별된 위험 요소가 있다면 이들도 포함한다. 이어서 각 위험 요소에 대한 위험을 평가하고 그 결과에 따라 적절한 조치를 계획하고 실행한다. 위험 관리 즉 위험 분석, 위험 조치 수행, 그리고 위험 모니터링은 조직 테스트 전략에 명시된 위험 분석 방법을 따라야 한다.

테스트 프로젝트에서 발생할 수 있는 위험 요소를 식별하고 각 위험 요소에 대하여 발생 가능성과 영향도를 바탕으로 위험도를 산정하고 위험 수준을 결정한다. 조치가 필요하다고 판단되는 수준의 위험 요소에 대해서는 조치 작업 계획을 수립한다. 조치 작업은 위험 수준과 비용 등을 고려하여 위험 회피, 위험 완화, 위험 수용, 위험 전가에 해당하는 조치를 계획한다. 그뿐만 아니라 각 위험 요소에 대하여 위험이 실제로 발생하는 경우에 어떻게 할지에 대한 비상 계획(Contingency plan)을 수립한다.

- **위험 회피**(Risk Avoidance): 식별된 위험 요소가 발생할 가능성 또는 발생 영향을 제거하여 위험 요소 발생을 원천적으로 예방한다.
- **위험 완화**(Risk Mitigation): 위험 요소 제거가 불가능하거나 너무 많은 비용이 필요하다면 위험도를 감소시키는 방법을 계획한다. 즉 위험의 발생 가능성을 낮추거나 발생에 따른 영향도를 약화하는 방안을 수립한다.
- **위험 수용**(Risk Acceptance): 위험도가 매우 낮은 경우에는 위험에 대한 모니터링만 할 수도 있다. 그뿐만 아니라 위험 회피 또는 위험 완화를 하기에 너무 많은 비용이 소요되는 경우 위험을 수용할 수도 있다.
- **위험 전가**(Risk Transfer): 테스팅 조직이 위험에 대한 조치를 수행하기에 비용, 역량 등이 적절하지 않다면 위험을 타 조직에 전가할 수 있다. 예를 들어, 위험에 해당되는 기능 및 비기능 피처에 대한 테스팅을 아웃소싱하거나 정형적 방법, V&V 분석 등의 다른 V&V 방법으로 위험을 전가시킬 수 있다.

테스트 프로젝트를 수행할 때의 위험 요소는 소프트웨어 제품(Product) 위험과 프로젝트 위험으로 분류된다. 프로젝트 위험은 테스트 수행과 관련된 위험으로 조직, 인력, 예산, 일정 등과 관련되며, 제품 위험은 기능 및 비기능 측면에서 제품이 목표로 하는 수준의 품질을 제공하지 못할 위험을 의미한다.

12.3.1 프로젝트 위험

테스트 프로젝트를 수행하면서 테스트 프로젝트의 목적 달성에 방해가 될 수 있는 프로젝트 유형의 위험 요소를 식별하고, 분석하며 조치 계획을 수립한다. 표 12.8은 프로젝트 위험 관리 계획의 예를 보여 준다. 각 위험 요소에 대하여 L(Likelihood, 발생 가능성)과 I(Impact, 영향도)를 바탕으로 위험도(E, Expose)를 평가하고, 적절한 조치 계획을 수립한다.

표 12.8 **프로젝트 위험 관리 계획 예**

위험 요소	L	I	E	조치 계획(완화)
인력 부족	2	4	8	• 필요 일정을 보수적으로 추정 • 수립된 일정을 엄격히 준수 • 현황 보고를 빈번하게
기술 및 경험 부족	2	3	6	• 각 인력에 대한 훈련 계획을 포함 • 훈련 시간을 충분히 확보 • 미숙련 인력의 산출물에 대한 검토 및 개선 시간을 충분히 확보

일반적으로 테스트 프로젝트를 포함하여 소프트웨어 프로젝트의 위험 요소는 다음과 같은 유형이 있다.

- **조직 관련 위험**: 테스트 프로젝트의 목적과 범위 등 테스트 정책에 대한 명확한 합의가 있는가? 테스트 관리, 동적 테스트, 정적 테스트 등에 대한 명확한 프로세스가 정의되어 있는가?
- **인력 관련 위험**: 충분한 인력이 확보되었는가? 필요한 역량을 보유하고 있는가? 기존에 유사한 도메인에서 테스트 프로젝트를 수행한 경험이 있는가?
- **비용 관련 위험**: 테스트 프로젝트를 완료할 수 있는 충분한 예산이 확보되었는가? 비용 추정은 정확한가? 인력에 대한 교육 및 훈련 비용도 고려되고 있는가?

- **일정 관련 위험**: 충분한 시간이 확보되었는가? 일정이 현실적인가? 일정을 고려하여 테스트 범위 등이 조정될 수 있는가? 테스트 완료 예정일 조정이 가능한가?

12.3.2 제품 위험

제품 위험은 고객, 사용자 등 이해관계자의 기대를 충족하지 못할 가능성을 뜻한다. 예를 들어, 고객, 사용자가 요구한 중요한 기능이 구현에서 누락되거나 구현되었다 하더라도 고객, 사용자가 기대하는 방식으로 동작하지 않을 가능성이 있다면 제품 위험에 해당된다. 특히 고객, 사용자의 기대를 만족시키지 못하는 상황이 사용자 및 고객사의 큰 경제적 손실로 이어지거나 사용자를 다치게 하거나 심지어 사용자를 사망에 이르게 할 수 있는 상황도 제품 위험에 해당된다.

제품 유형의 위험이 식별되면 각 위험 분석 결과를 바탕으로 위험 조치를 도출하고 테스트 계획에 반영한다. 높은 수준의 위험을 유발할 수 있는 기능 및 비기능 요구사항을 테스트 범위에 반드시 포함해야 한다. 그리고 이러한 기능 및 비기능 피처는 강도 높은 테스트가 수행되도록 테스트 전략을 수립한다.

예를 들어, 컴포넌트 레벨부터 해당 기능 및 비기능 피처를 테스트할 수 있다. 더욱 강도 높은 테스트 설계 기법을 적용하여 많은 수의 테스트 케이스 및 테스트 절차를 개발하고 실행할 수도 있고 테스트 완료 기준을 더 높은 수준으로 설정하는 방법도 있다. 이러한 위험 분석을 바탕으로 테스트 계획을 수립하고 테스트 활동을 수행하는 방법에 대해서는 위험 기반 테스트를 참고하기 바란다.

12.4 테스트 전략 수립

지금까지 테스트 대상에 대한 테스트 범위를 식별하였다. 이어서 파악된 테스트 범위에 대한 테스팅을 효과적이고 효율적으로 수행하기 위한 테스트 전략을 수립해야 한다. 표 12.9는 테스트 계획을 수립할 때 결정될 테스트 전략 요소를 보여 준다.

표 12.9 테스트 전략 항목

항목	설명
개별 테스트	프로젝트 테스트 계획에는 프로젝트 테스트를 구성하는 개별 테스트를 명시한다.
테스트 산출물	테스트 활동을 통해서 작성할 산출물을 명시한다.
테스트 설계 기법	동적 테스트, 정적 테스트 등 적용할 테스트 설계 기법을 명시한다.
테스트 환경 요건	테스트 실행을 위한 테스트 환경 요소를 명시한다.
테스트 데이터 요건	테스트 실행 시 필요한 테스트 데이터를 명시한다.
재테스팅 및 리그레션 테스팅	재테스팅 및 리그레션 테스팅 방법을 명시한다.
테스팅 중단 및 재시작 조건	테스트 활동을 중단하거나 다시 시작하는 조건을 명시한다.
테스트 메트릭	테스트를 수행하면서 측정할 메트릭을 명시한다.
테스트 완료 기준	테스트의 완료 여부를 판단할 수 있는 기준을 명시한다.
조직 테스트 전략과의 차이점	조직 테스트 전략과의 차이점과 근거를 기술한다.

이러한 테스트 전략 항목 대부분은 조직 테스트 전략 명세서에서 정의한 개별 테스트 수행 차원의 표준적인 전략이다. 만약 조직 테스트 전략 명세서와 내용이 동일하면 해당 문서를 참조한다. 그리고 조직 테스트 전략 명세서와 내용이 다른 항목은 "조직 테스트 전략과의 차이점"에 요약해서 기술한다.

12.4.1 개별 테스트

개별 테스트는 개별 테스트 계획이 아니라 프로젝트 테스트 계획의 하위 요소이다. 프로젝트 테스트 계획에서는 프로젝트 테스트를 구성하는 개별 테스트를 나열한다. 구체적으로 보면 프로젝트 테스트에 포함되어 수행될 레벨 테스트 및 유형 테스트를 나열하는 것이다.

개별 테스트는 테스트 컨텍스트에서 파악된 테스트 범위를 바탕으로 결정될 수 있다. 테스트 범위에 포함된 기능 유형 및 비기능 유형 요구사항에 대한 테스트를 효과적이고 효율적으로 수행하기 위하여 어떤 레벨의 테스트를 수행할 것인지 결정한다. 또한, 비기능 유형

의 요구사항에 따라서 성능, 신뢰성, 보안 등의 유형 테스트를 포함할 수도 있다.

그림 12.2는 프로젝트 테스트를 구성하는 다양한 개별 테스트의 예를 보여 준다. 레벨 테스트로는 컴포넌트, 통합, 시스템, 인수 테스트를 수행한다. 그리고 유형 테스트로는 성능 테스트, 신뢰성 테스트, 보안 테스트를 수행한다. 컴포넌트 테스트, 통합 테스트, 시스템 테스트는 기능에 초점을 두며, 인수 테스트는 기능과 보안, 신뢰성, 성능의 품질에 초점을 둔다. 성능 테스트와 신뢰성 테스트는 컴포넌트, 통합, 시스템 레벨에도 각각 수행된다. 그러나 보안 테스트는 시스템 수준에서만 수행된다.

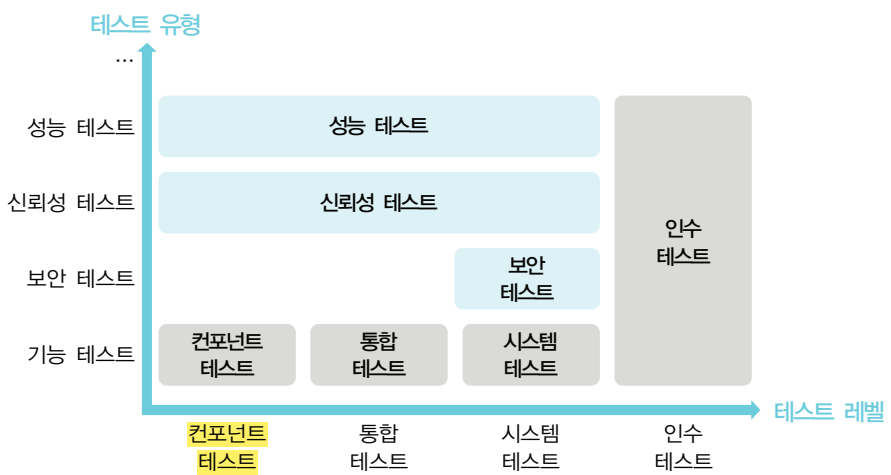

그림 12.2 프로젝트 테스트 계획에 포함된 개별 테스트 유형

테스트 전략은 각 개별 테스트마다 다를 수 있다. 예를 들어, 컴포넌트 테스트, 통합 테스트, 시스템 테스트, 성능 테스트별로 테스트 설계 기법, 테스트 환경, 테스트 완료 기준 등 테스트 전략이 달라질 수 있다. 그러므로 프로젝트 테스트 계획서와 별개로 개별 테스트별로 계획서를 작성한다. 예를 들어, 그림 12.3은 프로젝트 테스트 계획과 컴포넌트 테스트, 통합 테스트 등의 각 개별 테스트 계획의 관계를 보여 준다.

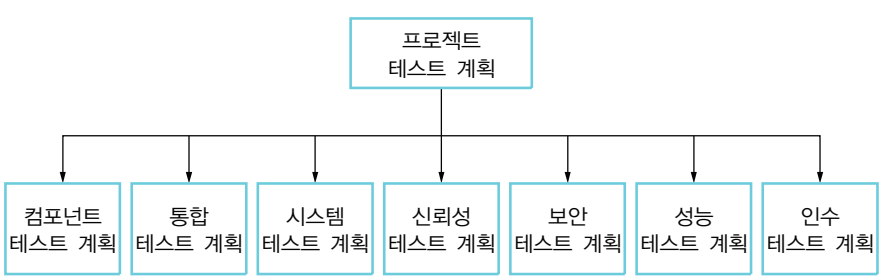

그림 12.3 프로젝트 테스트 계획과 개별 테스트 계획

개별 테스트 계획서에서 공통적인 내용은 프로젝트 테스트 계획서를 참조한다. 즉, 각 개별 테스트 계획에서 동일한 내용은 프로젝트 테스트 계획서에 기술하고 이를 참조하는 방식이다. 그리고 각 개별 테스트의 고유한 내용은 개별 테스트 계획서에서 기술한다.

여기서 나열되는 개별 테스트는 프로젝트 수준의 조직 테스트 전략과 일치해야 한다. 즉, 프로젝트 수준의 조직 테스트 전략에는 이미 수행할 개별 테스트가 제시되었으므로 이에 준해야 한다. 만약 조직 테스트 전략과 차이가 있다면 그 차이를 "조직 테스트 전략과의 차이점"에 기술한다.

12.4.2 테스트 산출물

테스트를 수행하면서 작성할 산출물을 정의한다. 프로젝트 수준의 조직 테스트 전략 명세서를 바탕으로 테스트를 수행하는 과정에서 산출되는 문서를 정의한다. 표 12.10과 표 12.11은 각각 테스트 관리 프로세스의 산출물과 동적 테스트 프로세스의 산출물을 보여 준다.

표 12.10 테스트 관리 프로세스 산출물

활동	산출물
테스트 계획	테스트 계획서
테스트 모니터링 및 제어	테스트 현황 보고서
테스트 종료	테스트 종료 보고서

표 12.11 동적 테스트 프로세스 산출물

활동	산출물
테스트 설계 및 구현	테스트 설계 명세서
	테스트 케이스 명세서
	테스트 절차 명세서
	테스트 환경 요건 명세서
	테스트 데이터 요건 명세서
테스트 환경 구축 및 관리	테스트 환경 준비 보고서
	테스트 데이터 준비 보고서
테스트 실행	테스트 실행 로그
결함 보고	결함 보고서 결함 추적 보고서

이러한 테스트 문서뿐만 아니라 테스트 입력과 출력, 그리고 테스트를 수행하기 위하여 개발된 자동화 도구도 테스트 산출물로 포함될 수 있다. 그리고 이러한 테스트 산출물이 언제 누구에게 제공될지에 대한 계획도 기술한다.

12.4.3 테스트 설계 기법

테스트 컨텍스트에서 명시된 테스트 대상과 피처를 바탕으로 테스트 설계 기법을 결정한다. 테스트 설계 기법은 정적 테스트 설계 기법과 동적 테스트 설계 기법으로 분류된다. 그림 12.4는 대표적인 정적 테스트 및 동적 테스트 방법을 보여 준다.

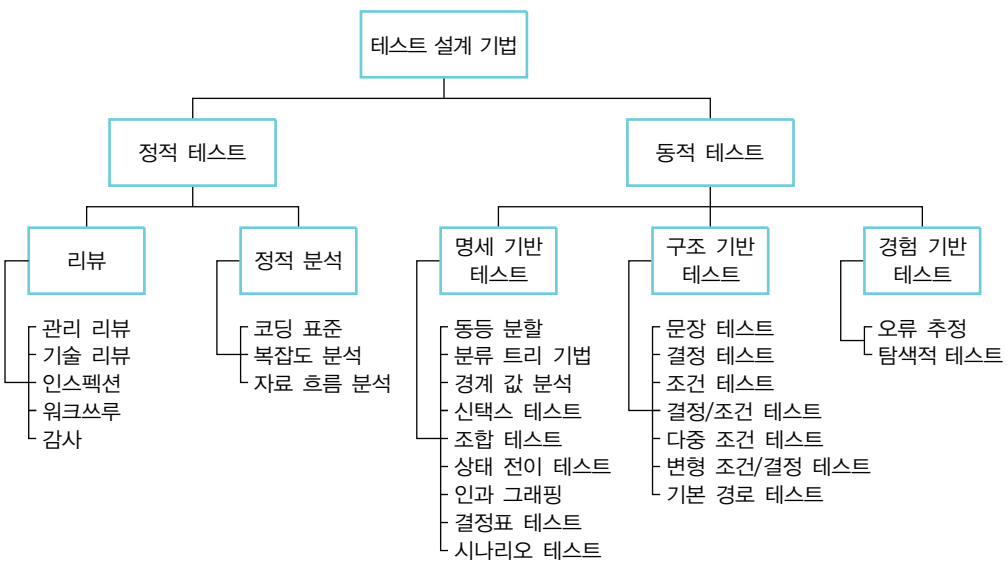

그림 12.4 테스트 설계 기법 분류

이렇게 다양한 테스트 설계 기법 중에서 높은 효과와 효율성을 달성할 수 있는 테스트 설계 기법을 적용해야 한다. 즉, 최대한 많은 수의 결함을 검출하면서 최소의 비용(테스트 케이스 및 테스트 절차의 수, 테스트 시간 등)이 소요되는 기법을 선택한다. 만약 이미 조직 테스트 전략 명세서에서 이러한 테스트 설계 기법이 선정되어 있다면 조직 테스트 전략 명세서의 테스트 설계 기법을 참조한다.

테스트 대상에 대하여 식별된 피처(기능 및 비기능 특성)를 효과적으로 테스트하기 위한 방법을 선정해야 한다. 예를 들어, 기능에 초점을 둔다면 기능의 동작에 영향을 줄 수 있는 다양한 값의 조합을 바탕으로 테스트를 수행할 필요가 있다. 반면에 성능에 초점을 둔다면 입

력값의 다양성보다는 입력 자체의 규모와 빈도를 바탕으로 테스트를 수행하는 것이 바람직하다.

테스트 계획 활동 중에는 테스트 설계 기법을 선정하고 핵심적인 전략을 결정한다. 여기서 테스트 설계 기법은 테스트 설계 및 구현 활동에서 테스트 케이스 개발을 지원할 정도로 구체화된다. 예를 들어, 테스트 계획 활동에서는 동등 분할 방법을 적용할지 아니면 경곗값 분석을 적용할지를 결정한다. 그리고 테스트 설계 및 구현 활동에서는 동등 분할 방법인 경우에 구체적으로 One-to-One 동등 분할을 할지 아니면 최소화 동등 분할을 할지를 구체적으로 결정하는 것이다. 경곗값 분석의 경우에는 2-value 경곗값 분석인지 3-value 경곗값 분석인지를 구체화한다.

테스트 설계 기법 선정은 테스트 설계 기법 자체의 특성도 고려할 필요가 있다. 어떤 테스트 설계 기법은 입력 변수 간의 제약사항을 고려하는 방법일 것이고 어떤 설계 기법은 입력 변수 간의 제약사항을 고려하지 않는 방법일 수 있다. 그러므로 테스트 대상의 특성에 맞는 테스트 설계 기법을 선택해야 한다. 그림 12.5는 명세 기반 테스트 설계 기법을 선택하는 방법을 보여 준다.

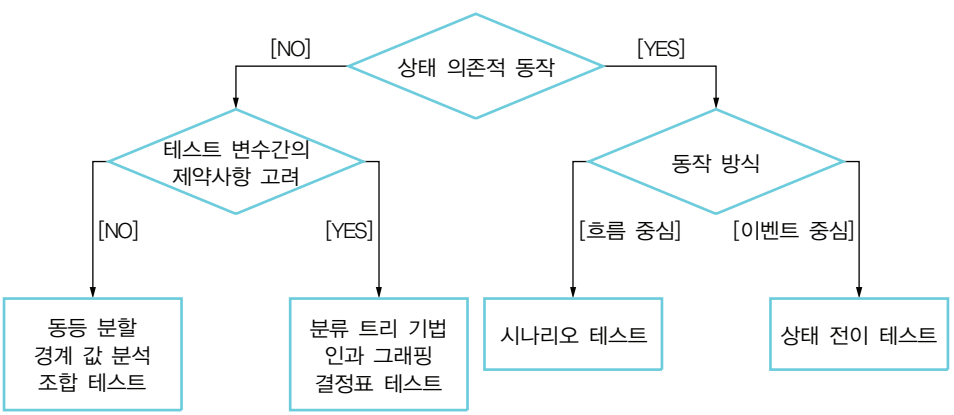

그림 12.5 **명세 기반 테스트 설계 기법 선택**

- **상태 의존적 동작 여부**: 테스트할 동작이 상태 의존적인지 아닌지에 따라서 테스트 설계 기법을 선정할 수 있다. 테스트 대상의 동작이 현재의 입력과 더불어 과거의 입력에도 영향을 받는 경우에 상태 의존적이라고 한다. 테스트 대상의 동작이 상태 의존적인 경우에는 시나리오 테스트 및 상태 전이 테스트가 적용된다. 반면에 동등 분할, 경곗값 분석, 분류 트리 기법 등은 상태 의존적이지 않은 동작을 테스트할 때 적용한다.

- **동작 방식**: 상태 의존적 동작은 동작 방식에 따라서 시나리오 테스트 또는 상태 전이 테스트 방법을 선택한다. 테스트 대상이 흐름 중심(Flow-based) 동작을 가지면 시나리오

테스트를 적용하고 이벤트 중심(Event-driven) 동작을 가지면 상태 전이 테스트를 적용한다.

- **테스트 변수 간의 제약사항 고려 여부**: 테스트하고자 하는 동작이 독립적이면, 동작에 영향을 주는 변수 간의 제약사항 고려 여부에 따라서 테스트 설계 기법을 결정한다. 변수 간의 제약사항을 고려하지 않는 경우에는 동등 분할 및 경곗값 분석, 조합 테스팅을 적용하고 제약사항을 고려하는 경우에는 분류 트리 기법, 인과 그래핑, 결정표 테스트를 적용한다.

12.4.4 테스트 환경 요건

테스트를 수행하기 위해 필요한 환경을 식별한다. 테스트 환경은 테스트 대상을 실행하기 위한 모든 환경으로 하드웨어, 운영 체제를 포함한 시스템 소프트웨어, 외부 연동 시스템, 공존하는 응용 소프트웨어, 테스트 도구 등을 포함한다.

테스트 계획을 수립할 때는 테스트 대상의 실행 환경을 식별한다. 하드웨어, 시스템 소프트웨어, 외부 연동 시스템 등 테스트 환경 유형 별로 필요한 환경 항목을 나열한다. 표 12.12는 테스트 계획 활동에서 식별한 테스트 환경의 예를 보여 준다.

표 12.12 **테스트 환경 예**

환경 항목 유형	환경 항목	설명
하드웨어	HP 서버	웹 애플리케이션 서버
시스템 소프트웨어	Oracle WebLogic 서버	웹 애플리케이션 서버
외부 연동 시스템	BMS	빌딩관리 시스템
테스트 도구	JMeter	성능 테스트 도구

그리고 각 환경 항목에 요건을 정의한다. 예를 들어, 하드웨어는 CPU 사양, CPU 성능, 메모리 및 디스크 크기 등 요건을 결정한다. 만약 테스트 계획을 수립하는 시점에 이러한 요건이 구체화되기 어려우면 테스트 설계 및 구현 활동에서 구체화하고 테스트 환경 요건 명세서에 기록한다.

개별 테스트에 따라서 필요한 테스트 환경은 달라질 수 있다. 예를 들어, 표 12.13은 컴포넌트 테스트, 통합 테스트, 시스템 테스트, 인수 테스트에 따른 일반적인 테스트 환경을 보여 준다. 기본적으로 시스템이 동작하는 실제 환경과 최대한 유사한 환경에서 테스트를 수

행하는 것이 중요하다. 이는 테스트 환경과 실제 동작 환경의 차이가 크면 클수록 테스트 환경에서는 통과되었던 테스트 케이스가 실제 동작 환경에서는 실패할 수 있기 때문이다. 표 12.13에서 볼 수 있듯이 테스트의 수준이 높아질수록 테스트 환경이 실제 동작 환경에 근접해진다.

표 12.13 테스트 레벨별 테스트 환경

환경 항목 유형	단위 테스트	통합 테스트	시스템 테스트	인수 테스트
하드웨어	개발자	개발자	시스템 테스트 전용	실제 시스템과 동일함
시스템 소프트웨어				
외부 연동 시스템	고려안 됨	고려될 수 있음	시뮬레이션 또는 실제	실제
공존 응용 소프트웨어	고려안 됨	고려 안 됨	고려될 수 있음	고려함
테스트 도구	테스트 실행 지원 도구			

일반적으로 컴포넌트 테스트와 통합 테스트는 개발자 환경에서 테스트를 실행하고, 시스템 테스트와 인수 테스트는 테스트만을 위한 환경을 구성한다. 컴포넌트 테스트와 통합 테스트는 일반적으로 기능 테스트에 초점을 두며, 시스템 테스트와 인수 테스트는 추가적으로 성능, 신뢰도, 안전성, 가용성 등과 같은 비기능적인 요구사항에 대한 테스트도 수행된다. 시스템 테스트와 인수 테스트에서 테스트 환경을 추가로 구축하는 이유는 성능, 신뢰도, 안전성 등의 비기능적인 품질은 테스트 대상 시스템뿐만 아니라 하드웨어, 시스템 소프트웨어, 주변 시스템, 공존 소프트웨어 등을 포함한 운영 환경에 많은 영향을 받기 때문이다.

그러나 테스트 환경을 실제 운영 환경과 유사하게 구성하려면 비용, 구현 가능성 측면에서 어려움이 따른다. 실제 운영 환경과 유사한 하드웨어, 입/출력 장치, 시스템 소프트웨어를 추가적으로 구매하고 환경을 구성해야 한다. 또한, 대상 시스템과 연동되는 외부 시스템이 있더라도 해당 시스템과 실제로 연동하는 것은 그 시스템을 불안하게 할지 모르는 위험성이 따른다.

12.4.5 테스트 데이터 요건

테스트 케이스가 실행되기 위해서는 테스트 환경과 더불어 테스트 데이터가 필요하다. 표 12.14는 테스트 데이터의 유형을 보여 준다.

표 12.14 테스트 데이터 예

테스트 데이터 항목 유형	설명
테스트 대상 입력	테스트 절차 실행 시 테스트 대상에 입력되는 데이터
테스트 대상 상태	테스트 케이스의 선행 조건을 충족시키기 위하여 테스트 대상에 대하여 사용되는 데이터
테스트 환경 항목 상태	테스트 케이스의 선행 조건을 충족시키기 위하여 테스트 환경 항목에 대하여 사용되는 데이터

- **테스트 대상 입력 유형**: 테스트 케이스는 테스트 대상에 입력되는 데이터가 필요하다. 입력 데이터의 규모가 크고 복잡한 구조를 가지고 있는 경우에는 해당 데이터가 테스트 케이스와 별개로 파일 또는 데이터베이스에 정의되고 이를 참조할 수 있다.

- **테스트 대상 상태 유형**: 테스트 대상에 입력값을 전달하기에 앞서 테스트 대상이 특정 상태에 도달해야 할 경우가 있다. 이러한 상태는 각 테스트 케이스의 선행 조건에 명시된다. 따라서 테스트 대상을 테스트 케이스의 선행 조건에 명시된 상태로 설정하기 위한 테스트 데이터가 필요하다.

- **테스트 환경 항목 유형**: 테스트 케이스의 선행 조건에는 테스트 대상에 대한 상태뿐만 아니라 테스트 환경 항목에 대한 상태도 명시된다. 그러므로 각 테스트 환경 항목을 테스트 케이스의 선행 조건에서 요구하는 상태로 설정하기 위한 테스트 데이터도 필요하다.

테스트 계획을 수립할 때는 이러한 3가지 유형의 테스트 데이터를 식별한다. 각 테스트 데이터에 대한 이름과 역할을 간략하게 설명한다. 그리고 각 테스트 데이터에 대한 구체적인 요건을 식별한다. 예를 들어, 테스트 데이터의 규모, 저장 위치, 접근 권한 등을 결정한다. 만약 테스트 계획을 수립하는 시점에 이러한 요건을 구체화하기 어렵다면 테스트 설계 및 구현 활동에서 결정한다.

12.4.6 재테스팅 및 리그레션 테스팅

동적 테스트 프로세스의 수행과 더불어 재테스팅 및 리그레션 테스팅에 대한 전략을 수립한다. 즉 재테스팅과 리그레션 테스팅 각각에 대하여 어떤 조건에서 시작을 할지, 어떤 방법을 적용할지를 기술한다.

12.4.6.1 재테스팅

테스팅을 통해서 검출된 결함이 올바르게 해결되었는지 여부에 앞서 결함을 검출한 테스트 절차(테스트 케이스)를 활용해서 다시 테스팅을 수행해야 한다. 이 재테스팅은 검출된 결함의 해결 여부가 검증될 때까지 반복적으로 수행될 수 있다. 예를 들어, 1차 재테스팅에서 결함이 올바르게 해결되지 않았다는 것이 확인되면 개발자는 다시 해당 결함을 해결하기 위하여 디버깅과 소스 코드 수정을 수행할 것이다. 이 수정된 소스 코드를 대상으로 2차 재테스팅을 수행하게 된다.

이처럼 반복적으로 재테스팅이 수행되어야 하므로 재테스팅을 효율적으로 수행하기 위한 전략이 필요하다. 우선, 재테스팅은 기존 결함을 검출하였던 테스트 절차를 다시 수행하는 것이므로 테스트 절차를 자동적으로 수행하는 전략을 적용할 수 있다. 즉, 각 테스트 절차를 자동화 도구를 통해서 실행시킬 수 있도록 테스트 스크립트(Script)를 개발하여 실행하는 것이다. 한편, 테스트 스크립트 개발은 많은 시간이 필요하므로 Record&Playback 도구를 사용하여 테스트 스크립트를 자동 생성하는 방법을 이용할 수도 있다.

재테스팅은 컴포넌트 테스트, 통합 테스트, 시스템 테스트 등 여러 레벨에서, 그리고 성능 테스트, 신뢰성 테스트 등 여러 유형에서 수행될 수 있다. 만약 시스템 테스트를 수행하는 과정에서 검출 결함의 수정에 대한 재테스팅을 수행하는 경우라면, 먼저 수정이 발생한 개별 컴포넌트에 대하여 컴포넌트 테스트를 수행하고, 수정이 발생한 컴포넌트와 다른 컴포넌트 간의 통합 테스트를 수행한 후에 시스템 테스트에서 재테스팅을 수행한다. 마찬가지로 어떤 변경이 성능에 영향을 미친다면 성능 테스트에서도 재테스팅을 수행할 필요가 있다.

또한, 변경의 유형과 가용한 시간을 고려하여 재테스팅이 수행되는 개별 테스팅을 선정한다. 예를 들어, 컴포넌트에 발생한 변경이 컴포넌트 내부로 한정되고 컴포넌트 외부에 영향을 미치지 않는다면 통합 테스트의 재테스팅을 생략할 수 있다. 또는 재테스팅을 수행하는 결함의 심각도에 따라 선택적으로 재테스팅을 수행하는 전략을 수립할 수도 있다. 예를

들어, 낮은 심각도의 결함에 대해서는 컴포넌트 테스트와 통합 테스트에서는 생략하고 시스템 테스트에서 재테스팅을 수행할 수 있다.

12.4.6.2 리그레션 테스팅

리그레션 테스팅은 소프트웨어 변경이 발생한 후에 수행되는 테스트로 소프트웨어에 가해진 변경이 의도하지 않은 결함을 만들지 않았으며 시스템이 기존의 요구사항을 충족함을 검증하기 위하여 수행한다.

리그레션 테스팅은 특정 레벨에 한정되지 않고 컴포넌트 테스트, 통합 테스트, 그리고 시스템 테스트의 여러 레벨에서 수행될 수 있다. 즉, 변경이 발생한 컴포넌트를 대상으로 리그레션 테스팅을 수행하는 것뿐만 아니라 이 컴포넌트와 연결된 다른 컴포넌트와의 통합 테스트를 수행하고 변경된 컴포넌트와 관련된 기능 요구사항을 포함한 성능, 신뢰성 등의 품질에 대하여 시스템 테스트를 수행할 수도 있다. 그러므로 리그레션 테스팅 방법은 개별 테스트에 대한 계획서보다는 프로젝트 테스트 계획서에 기술한다.

주어진 시간과 비용에 따라 리그레션 테스팅을 전체 레벨에서 수행하지 않을 수도 있다. 리그레션 테스팅을 전체 레벨에서 수행하지 않는다면 테스팅을 수행할 테스트 레벨을 결정해야 한다. 예를 들어, 컴포넌트 레벨에서는 리그레션 테스팅을 수행한 후에 통합 테스트의 리그레션 테스팅은 생략하고 바로 시스템 테스트에서 리그레션 테스팅을 수행할 수 있다.

리그레션 테스팅을 수행할 테스트 레벨 결정은 소프트웨어에 가해진 변경의 유형과 그 영향에 대한 추정을 바탕으로 한다. 예를 들어, 컴포넌트 내부의 알고리즘 변경으로 그 영향이 컴포넌트 내부로만 한정되고 컴포넌트 간의 연결에 대한 영향이 적다면 컴포넌트 테스트와 시스템 테스트에서 리그레션 테스팅을 수행하고 통합 테스트에서는 생략할 수 있다. 또한, 변경의 유형이 기능에 한정되고 이로 인한 성능에 대한 영향이 미약하다면 성능 테스트의 리그레션 테스팅은 생략될 수 있다.

또한, 리그레션 테스팅을 수행할 각 개별 테스트에서 어떤 리그레션 테스팅 전략을 적용할지도 결정해야 한다. 표 12.15는 대표적인 리그레션 테스팅 전략을 보여 준다. 발생한 변경의 유형, 변경에 따른 영향 범위의 크기, 그리고 가용한 시간과 비용 등을 고려하여 적절한 전략을 선택한다.

표 12.15 리그레션 테스팅 전략

리그레션 테스팅 전략	설명
Retest-all 전략	기존에 개발된 모든 테스트 절차 수행
선택적 리그레션 테스트 전략	기존의 테스트 절차 중에서 일부를 선택하여 수행
테스트 최소화 전략	중복된 테스트 절차를 제거하여 테스트 절차의 수를 최소화
테스트 우선순위화 전략	우선순위가 높은 테스트 절차를 선정하여 수행

재테스팅과 마찬가지로 리그레션 테스팅도 기존 테스트 절차의 반복적인 수행이 필요하므로 테스트 절차를 자동화할 수 있는 도구를 활용하여 리그레션 테스팅의 효율성을 높이는 방법도 고려한다.

12.4.7 테스팅 중단 및 재시작 조건

테스트 계획에 따라 수행 중인 테스트 활동의 수행을 중단하거나 중단된 테스트 활동을 다시 시작할 수 있는 조건을 기술한다. 기본적으로 각 테스트 활동의 원활한 수행을 방해할 수 있는 상황을 중단 조건으로 설정할 수 있다. 그리고 테스트가 중단된 원인이 해결되어 테스팅을 다시 시작할 때 수행해야 하는 테스트를 명시한다. 표 12.16은 테스트 중단 및 재시작 조건의 예를 보여 준다.

표 12.16 테스트 중단 및 재시작 조건 예

중단 조건	재시작 조건
과도한 비율(예 50% 이상)로 테스트가 실패할 때 해당 테스트 중단	관련 결함 해결 후에 해당 테스트 전체 재시작
기본 시나리오 테스트 실패 시 대안 시나리오 테스트 중단	기본 시나리오 관련 결함 해결 후에 해당 유스케이스의 모든 시나리오 테스트 재시작
테스트 환경 문제로 테스트 실행 실패 시 동일 테스트 환경을 사용하는 모든 테스트 중단	테스트 환경 문제 해결 후에 동일 테스트 환경을 사용하는 모든 테스트 재시작

12.4.8 테스트 메트릭

테스트 활동 수행과 결과를 정량적으로 판단하는 데 사용할 측정 메트릭을 결정한다. 즉, 테스트 계획, 테스트 설계 및 구현, 테스트 환경 구축 및 관리, 테스트 실행, 그리고 결함 보고 활동 별로 해당 활동의 수행 상황을 파악할 수 있는 메트릭을 결정한다.

테스트 메트릭은 테스트 진척도를 파악할 때 지표로 이용될 수 있다. 즉, 테스트 현황 보고서에서 각 테스트 활동의 진척을 보고할 때 다양한 테스트 메트릭의 값을 제시한다. 예를 들어, 테스트 설계 및 구현 활동의 진척도는 개발된 테스트 케이스의 수 및 개발된 테스트 절차의 수를 통해서 파악할 수 있으며, 테스트 실행 활동의 진척도는 실행된 테스트 케이스 및 테스트 절차의 수를 통해서 파악할 수 있다.

그뿐만 아니라 테스트 메트릭은 테스트 완료 기준으로도 사용되어 테스트 종료 보고서에 테스트 완료 여부에 대한 평가 데이터로 활용된다. 예를 들어, 테스트 완료 기준으로 통과된 테스트 케이스 및 절차의 비율 또는 검출되었지만 아직 종결되지 않은 결함 수 등이 사용될 수 있다.

12.4.8.1 테스트 계획 활동 메트릭

테스트 계획 활동은 테스트 목적에 따라서 테스트 대상을 선정하고 각 테스트 대상별로 테스트 범위 즉 피처를 식별한다. 따라서 테스트 계획에서 식별된 테스트 대상 및 피처의 수를 통해서 테스트 활동에 대한 진척도를 파악할 수 있다. 표 12.17은 테스트 계획 활동에서 측정될 수 있는 메트릭을 보여 준다.

표 12.17 테스트 계획 활동 메트릭

메트릭	설명
테스트 대상 수	계획된 테스트 대상의 수
피처 수	계획된 전체 피처의 수

12.4.8.2 테스트 설계 및 구현 활동 메트릭

테스트 설계 및 구현 활동은 테스트 계획에서 식별된 테스트 대상별로 유사한 피처를 그룹화하여 피처 집합을 결정한다. 그리고 각 피처 집합별로 테스트 케이스와 테스트 절차를 개

발하며 테스트 환경 요건과 테스트 데이터 요건을 결정한다. 표 12.18은 테스트 설계 및 구현 활동에서 측정될 수 있는 메트릭 중에서 테스트 설계 관련 메트릭을 보여 준다.

표 12.18 테스트 설계 및 구현 활동 메트릭 - 테스트 설계 관련 메트릭

메트릭	설명
피처 집합 수	각 테스트 대상별 피처 집합의 수
테스트 케이스 수	개발된 전체 테스트 케이스의 수
테스트 절차 수	개발된 전체 테스트 절차의 수
테스트 환경 항목 수	식별된 테스트 환경 항목의 수
테스트 데이터 수	식별된 테스트 데이터의 수

개발된 테스트 케이스 및 테스트 절차의 적합성, 즉, 테스트 케이스 및 테스트 절차가 목표로 하는 범위를 충분히 반영하였는지를 커버리지(Coverage)를 통해서 측정할 수 있다. 즉, 개발된 테스트 케이스 및 테스트 절차를 통해서 주어진 요구사항, 설계, 그리고 코드가 어느 정도 확인되었는지 측정할 수 있다. 표 12.19는 테스트 설계 및 구현 활동에 대한 커버리지 관련 메트릭을 보여 준다.

표 12.19 테스트 설계 및 구현 활동 메트릭 - 커버리지 관련 메트릭

메트릭	설명
요구사항 커버리지	$\frac{\text{개발된 전체 테스트 케이스(절차)로 확인되는 요구사항 요소 수}}{\text{전체 요구사항 요소 수}}$
설계 커버리지	$\frac{\text{개발된 전체 테스트 케이스(절차)로 확인되는 설계 요소 수}}{\text{전체 설계 요소 수}}$
코드 커버리지	$\frac{\text{개발된 전체 테스트 케이스(절차)로 확인되는 코드 요소 수}}{\text{전체 코드요소 수}}$

커버리지(Coverage)는 테스트 케이스 및 테스트 절차로 확인이 되는 테스트 베이시스(Basis)의 비중으로, 얼마나 충분한 테스트를 하는지 보여주는 정량적인 지표이다. 그림 12.6에서 볼 수 있듯이 테스트 커버리지는 테스트 베이시스의 유형에 따라서 요구사항 커버리지, 설계 커버리지, 코드 커버리지로 분류된다.

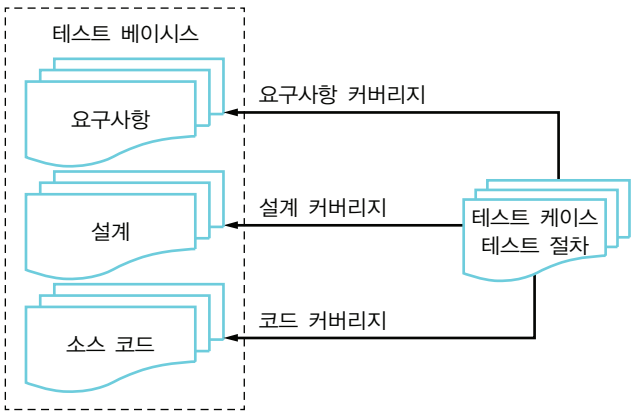

그림 12.6 테스트 커버리지 개념

요구사항 커버리지, 설계 커버리지 그리고 코드 커버리지는 구체적으로 확인할 요소에 따라서 세분화된다. 예를 들어, 요구사항 커버리지에는 테스트 케이스 및 테스트 절차가 전체 유스케이스 중에서 확인한 것이 얼마나 되는지를 보여 주는 유스케이스 커버리지가 있다. 그림 12.7은 각 테스트 커버리지 유형별로 구체적인 커버리지 기준을 보여 준다.

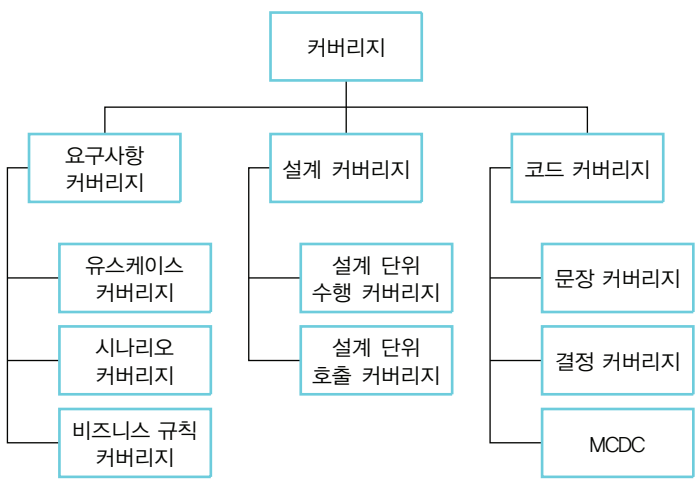

그림 12.7 테스트 커버리지 유형

요구사항 커버리지는 테스트 케이스 및 테스트 절차가 제시된 요구사항을 어느 정도 확인하고 있는지를 의미한다. 그림 12.8은 요구사항 커버리지의 개념을 보여 준다. 각 요구사항 요소에 대한 테스트 케이스 및 테스트 절차 개발 여부에 대한 추적성을 바탕으로 요구사항 커버리지를 계산할 수 있다. 요구사항 요소 R2의 경우에는 테스트 케이스 및 테스트 절차로 추적되지 않으므로 이는 요구사항 커버리지 100%를 달성하지 못하고 있다.

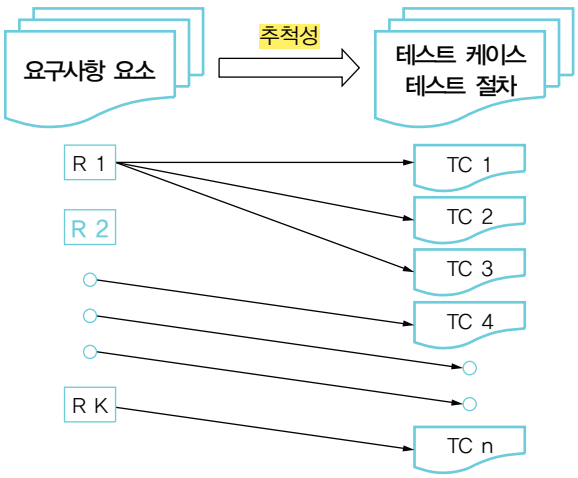

그림 12.8 요구사항 커버리지 개념

요구사항 커버리지는 요구사항의 세부 요소가 무엇인지에 따라서 구체적으로 정의될 수 있다. 표 12.20은 요구사항 요소로서 유스케이스, 시나리오, 비즈니스 규칙 등이 있을 때 정의되는 요구사항 커버리지를 보여 준다.

표 12.20 요구사항 커버리지 유형

요구사항 커버리지	설명
유스케이스 커버리지	$\frac{\text{1회 이상 확인된 유스케이스의 수}}{\text{전체 유스케이스의 수}}$
시나리오 커버리지	$\frac{\text{1회 이상 확인된 시나리오의 수}}{\text{전체 시나리오의 수}}$
비즈니스 규칙 커버리지	$\frac{\text{1회 이상 확인된 비즈니스 규칙의 수}}{\text{전체 비즈니스 규칙의 수}}$

설계 커버리지는 구조 설계, 상세 설계 등의 설계 활동 결과물에 대해서 테스트 케이스 및 테스트 절차가 얼마나 많이 확인할 수 있는지를 의미한다. 설계 커버리지는 기본적으로 개별적인 설계 요소에 대한 확인과 설계 요소 간의 연결에 대한 확인을 기준으로 결정된다. 예를 들어, 그림 12.9는 설계 결과로서 5개의 설계 단위(예 컴포넌트, 클래스 또는 함수)와 7개의 연결(설계 단위 간의 호출 등)을 보여 준다.

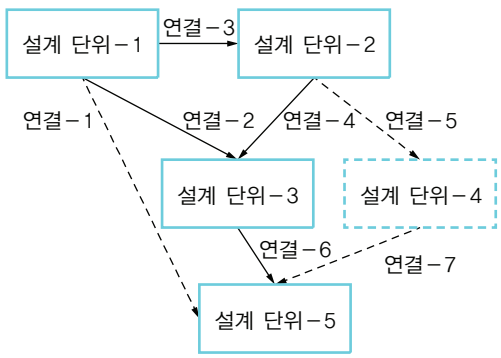

그림 12.9 **설계 커버리지 개념**

설계 단위에 대한 수행 커버리지는 전체 설계 단위 중에서 몇 개가 확인되었는지를 의미한다. 예를 들어, 그림에서 점선으로 표시된 설계 단위-4는 테스트 케이스 및 테스트 절차에 의해서 실행되지 않는 설계 단위를 의미한다. 따라서 설계 단위에 대한 수행 커버리지는 80%($\frac{4}{5}$)가 된다.

설계 커버리지의 두 번째 유형은 설계 단위 간 호출에 대한 커버리지로 전체 호출 중에서 테스트 케이스 및 테스트 절차를 통해서 확인이 될 수 있는 호출의 수를 의미한다. 예를 들어, 그림에서 점선으로 표시된 연결-1, 연결-5, 연결-7은 테스트 케이스 및 테스트 절차를 통해서 확인이 되지 않는 것을 의미한다. 그러므로 설계 단위 호출 커버리지는 57%($\frac{4}{7}$)가 된다.

설계 커버리지는 구조 설계 및 상세 설계의 결과물을 바탕으로 분석된다. 구조 설계와 상세 설계는 설계 단위가 상이하므로 앞에서 설명한 설계 커버리지는 설계 단위에 따라서 구체화된다. 즉, 설계 단위가 함수인 경우에는 함수를 기준으로 설계 커버리지가 정의되며, 설계 단위가 클래스 또는 컴포넌트인 경우에는 클래스 또는 컴포넌트를 기준으로 설계 커버리지가 정의된다.

예를 들어, 상세 설계에서 소프트웨어의 설계 단위로 함수들이 있고 함수 간의 호출 관계로 설계 결과가 정의된다면 함수를 기준으로 설계 커버리지를 정의한다. 표 12.21은 설계 단위가 함수인 경우의 설계 커버리지를 보여 준다.

표 12.21 **설계 커버리지 유형 – 함수 대상**

유형	설계 커버리지	설명
수행	함수 수행 커버리지	$\frac{\text{1회 이상 확인된 함수의 수}}{\text{전체 함수의 수}}$
호출	함수 호출 커버리지	$\frac{\text{1회 이상 확인된 함수 호출의 수}}{\text{전체 함수 호출의 수}}$

클래스가 설계 단위인 경우에는 각 클래스의 수준과 클래스가 가지는 연산의 수준에서 각각 수행 커버리지 및 호출 커버리지를 정의한다. 표 12.22는 클래스가 설계 단위인 경우에 설계 커버리지를 보여 준다.

표 12.22 설계 커버리지 유형 – 클래스 대상

유형	설계 커버리지	설명
수행	클래스 수행 커버리지	1회 이상 확인된 클래스의 수 / 전체 클래스의 수
	클래스 연산 수행 커버리지	1회 이상 확인된 연산의 수 / 전체 연산의 수
호출	클래스 호출 커버리지	1회 이상 확인된 클래스 호출의 수 / 전체 클래스 호출의 수
	클래스 연산 호출 커버리지	1회 이상 확인된 연산 호출의 수 / 전체 연산 호출의 수

컴포넌트가 설계 단위인 경우에는 각 컴포넌트의 수준과 컴포넌트가 가지는 인터페이스 및 인터페이스를 구성하는 연산의 수준에서 각각 수행 커버리지 및 호출 커버리지를 정의한다. 표 12.23은 컴포넌트가 설계 단위인 경우에 설계 커버리지를 보여 준다.

표 12.23 설계 커버리지 유형 – 컴포넌트 대상

유형	설계 커버리지	설명
수행	컴포넌트 수행 커버리지	1회 이상 수행된 컴포넌트의 수 / 전체 컴포넌트의 수
	인터페이스 수행 커버리지	1회 이상 확인된 인터페이스의 수 / 전체 인터페이스의 수
	인터페이스 연산 수행 커버리지	1회 이상 수행된 인터페이스 연산의 수 / 전체 인터페이스 연산의 수
호출	컴포넌트 호출 커버리지	1회 이상 수행된 컴포넌트 호출의 수 / 전체 컴포넌트 호출의 수
	인터페이스 호출 커버리지	1회 이상 수행된 인터페이스 호출의 수 / 전체 인터페이스 호출의 수
	인터페이스 연산 호출 커버리지	1회 이상 수행된 인터페이스 연산 호출의 수 / 전체 인터페이스 연산 호출의 수

코드 커버리지는 테스트 대상의 소스 코드 요소를 테스트 케이스 또는 테스트 절차가 얼마나 확인하는가를 의미한다. 소스 코드의 요소로는 문장, 결정, 조건 등이 있으며 각 요소별로 코드 커버리지의 유형이 정의된다. 표 12.24는 가장 대표적인 코드 커버리지의 유형을 보여 준다.

표 12.24 코드 커버리지 유형

코드 커버리지	설명
문장 커버리지	1회 이상 확인된 문장들의 수 / 전체 실행 가능한 문장의 수
결정 커버리지	1회 이상 확인된 결정 결과의 수 / 전체 프로그램 결정 결과의 수
MCDC 커버리지	1회 이상 확인된 MCDC를 만족하는 조건의 수 / 전체 조건과의 수

12.4.8.3 테스트 환경 구축 및 관리 활동 메트릭

테스트 환경 구축 및 관리 활동에서는 테스트 설계 및 구현 활동에서 결정된 테스트 환경 및 테스트 데이터를 준비한다. 따라서 계획된 테스트 환경과 테스트 데이터에 대한 준비 상황을 메트릭으로 정의할 수 있다. 표 12.25는 테스트 환경 구축 및 관리 활동에서 사용할 수 있는 메트릭을 보여 준다.

표 12.25 테스트 환경 구축 및 관리 활동 메트릭

메트릭	설명
테스트 환경 구축률	구축된 테스트 환경 항목의 수 / 전체 테스트 환경 항목의 수
테스트 데이터 준비율	준비된 테스트 데이터의 수 / 전체 테스트 데이터의 수

12.4.8.4 테스트 실행 활동 메트릭

테스트 실행 활동에서는 개발된 테스트 케이스 및 테스트 절차를 실행하고 예상 결과와 실제 결과를 비교하고 테스트 실행 결과를 기록한다. 따라서 테스트 실행 활동에서는 테스트 케이스 및 테스트 절차 실행과 그 결과를 바탕으로 한 메트릭을 정의할 수 있다. 표 12.26은 테스트 실행 활동에 대한 대표적인 메트릭을 보여 준다.

표 12.26 테스트 실행 활동 메트릭 - 테스트 실행 관련 메트릭

메트릭	설명
실행된 테스트 케이스(테스트 절차) 수	실행이 성공한 테스트 케이스(테스트 절차) 수
통과된 테스트 케이스(테스트 절차) 수	실행되어 통과된 테스트 케이스(테스트 절차) 수
실패 테스트 케이스(테스트 절차) 수	실행되어 실패한 테스트 케이스(테스트 절차) 수

개발된 테스트 케이스(테스트 절차)의 수를 기준으로 비율도 정의될 수 있다. 즉, 실행된 테스트 케이스(테스트 절차) 비율은 개발된 테스트 케이스(테스트 절차) 중에서 실행에 성공한 테스트 케이스(테스트 절차)의 비중으로 정의된다.

테스트 실행 활동 메트릭의 두 번째 유형으로는 커버리지 관련 메트릭이 있다. 커버리지 관련 메트릭은 실행된 테스트 케이스(테스트 절차)를 통해서 달성될 수 있는 요구사항 커버리지, 설계 커버리지, 코드 커버리지를 의미한다. 표 12.27은 테스트 실행 활동에 대한 커버리지 관련 메트릭의 예를 보여 준다.

표 12.27 테스트 실행 활동 메트릭 - 커버리지 관련 메트릭

메트릭	설명
실행된 테스트 케이스(테스트 절차)에 따른 요구사항 커버리지	실행된 테스트 케이스(테스트 절차)에 의해서 확인된 요구사항 수 / 전체 요구사항 수
통과된 테스트 케이스(테스트 절차)에 따른 요구사항 커버리지	통과된 테스트 케이스(테스트 절차)에 의해서 확인된 요구사항 수 / 전체 요구사항 수
실행된 테스트 케이스(테스트 절차)에 따른 설계 커버리지	실행된 테스트 케이스(테스트 절차)에 의해서 확인된 설계요소 수 / 전체 설계요소 수
통과된 테스트 케이스(테스트 절차)에 따른 설계 커버리지	통과된 테스트 케이스(테스트 절차)에 의해서 확인된 설계 요소 수 / 전체 설계 요소 수
실행된 테스트 케이스(테스트 절차)에 따른 코드 커버리지	실행된 테스트 케이스(테스트 절차)에 의해서 확인된 코드 요소 수 / 전체 코드 요소 수
통과된 테스트 케이스(테스트 절차)에 따른 코드 커버리지	통과된 테스트 케이스(테스트 절차)에 의해서 확인된 코드 요소 수 / 전체 코드 요소 수

12.4.8.5 결함 보고 활동 메트릭

결함 보고 활동은 테스트 실행 결과를 분석하여 식별된 결함을 기록한다. 표 12.28은 결함 보고 활동에 대한 메트릭을 보여 준다.

표 12.28 결함 보고 활동 메트릭

메트릭	설명
검출 결함 수	실행된 테스트 에서 검출된 결함의 개수
검출 결함 밀도	검출된 결함 개수 / 대상 코드 행수(KLOC)
상태별 결함 수	결함 생명 주기의 각 상태별 결함의 개수
결함 나이	결함이 보고되고 종결될 때까지 걸린 시간

또한, 보고 기간 내에 측정된 각 메트릭 값을 전체 기간으로 누적한 수치도 결함 보고 활동에서 메트릭으로 사용할 수 있다. 누적 검출 결함 수는 전체 기간에 걸쳐서 식별된 모든 결함의 수를 의미한다. 마찬가지로 상태별 누적 결함 수는 전체 기간에 걸쳐서 식별된 모든 결함을 각 상태(Open, Review, Assigned 등)별로 구한 수치이다.

그리고 이러한 메트릭을 시스템 전체가 아니라 개별 테스트 대상별로 구할 수도 있다. 예를 들어, 시스템을 구성하는 각 컴포넌트별 검출 결함 수, 컴포넌트별 검출 결함 밀도 등을 구할 수가 있다.

12.4.9 테스트 완료 기준

테스트 완료 기준은 테스트 완료 여부를 판단할 수 있는 객관적인 기준을 의미한다. 즉, 테스트 대상에 대하여 충분한 테스트를 수행하였다고 판단할 수 있는 기준을 의미한다. 예를 들어, 테스트 완료 기준을 강하게 하면 좀 더 엄격한 수준으로 테스팅을 수행한다는 의미이며, 그만큼 테스트 대상에 대하여 높은 수준의 품질을 기대할 수 있다. 이처럼 테스트 대상에 대하여 기대하는 품질 수준에 따라서 적절한 테스트 완료 기준을 설정할 수 있다.

동적 테스트 프로세스 활동은 설정된 테스트 완료 기준을 고려한다. 예를 들어, 높은 수준의 테스트 완료 기준이라면 이 기준을 달성하기 위하여 더 많은 테스트 케이스 및 테스트 절차가 개발되고 이를 효율적으로 실행하기 위한 자동화 도구가 필요하다.

테스트가 종료되었을 때 해당 테스트의 수행 결과가 설정된 테스트 완료 기준을 충족하였

는지 평가한다. 즉, 테스트 종료 활동에서는 테스트 완료 기준에 따라서 수행된 테스트 활동을 평가하고 그 결과를 기록한다.

12.4.9.1 기본 유형의 테스트 완료 기준

기본적인 테스트 완료 기준은 테스트 케이스(테스트 절차) 유형, 테스트 커버리지 유형, 그리고 결함 유형으로 구분된다.

❑ 테스트 케이스(테스트 절차) 기반 방법

테스트 대상과 관련된 테스트 케이스 및 테스트 절차 중에서 어느 정도가 통과되었는지를 기준으로 삼는다. 예를 들어, "전체 테스트 케이스(테스트 절차) 중에서 일정 비율 이상이 통과되어야 한다." 또는 "특정 테스트 케이스(테스트 절차)들은 통과되어야 한다." 등이 기준으로 사용될 수 있다.

특히, 컴포넌트 테스트에서는 테스트 대상 모듈의 중요도에 따라서 테스트 케이스(테스트 절차) 통과 비율이 달라질 수 있고, 마찬가지로 통합 테스트에서도 중요한 기능 또는 비기능적 요소를 다루는 통합 시나리오는 좀 더 높은 테스트 케이스(테스트 절차) 통과 비율로 테스트 완료 여부를 판단한다.

❑ 테스트 커버리지 기반 방법

이 유형의 완료 기준에서는 테스트 케이스를 도출하는 기준 문서(즉 테스트 베이시스)의 내용이 얼마나 다루어졌는지를 기준으로 한다. 표 12.29는 시스템 테스트, 통합 테스트, 컴포넌트 테스트별 커버리지 기준을 보여 준다.

표 12.29 테스트 레벨별 테스트 커버리지 기준

테스트 레벨	테스트 커버리지	설명
시스템 테스트	요구사항 커버리지	요구사항 명세서에 명시된 각 요구사항의 테스트 여부
통합 테스트	설계 커버리지	설계 명세서를 바탕으로 설계 단위에 대한 수행 및 설계 단위 간의 호출에 대한 테스트 여부
컴포넌트 테스트	코드 커버리지	모듈의 소스 코드에 대한 테스트 여부

시스템 테스트를 수행할 때는 테스트 커버리지의 기준이 되는 요구사항 명세서의 내용을 얼마나 다루는지(Cover) 측면에서 테스트 완료 기준을 설정할 수 있다. 예를 들어, "전체

요구사항을 모두 다루어야 한다." 또는 "중요도가 높은 특정 요구사항을 다루어야 한다." 등이 테스트 완료 기준이 될 수 있다.

통합 테스트를 수행할 때는 설계 명세서에 명시된 컴포넌트, 클래스, 함수를 비롯한 설계 단위의 실행(설계 단위 수행 커버리지)과 설계 단위 간의 호출(설계 단위 호출 커버리지)을 얼마나 테스트하였는지를 테스트 완료 기준으로 설정할 수 있다.

컴포넌트 테스트를 수행할 때는 테스트 대상에 대한 소스 코드를 바탕으로 테스트 완료 기준을 정의할 수도 있다. 예를 들어, "95%의 문장 커버리지를 충족시켜야 한다." 또는 "90%의 결정 커버리지를 충족시켜야 한다."가 모듈에 대한 테스트 완료 기준이 될 수 있다.

❑ 결함 기반 방법

이 기준에서는 발견된 결함 정보를 바탕으로 테스트 완료 기준을 설정할 수도 있다. 예를 들면, "발견된 결함의 수가 일정 개수 이하이어야 한다." 또는 "심각하거나 중요한 결함은 존재하지 않아야 한다." 등이 테스트 완료 기준이 될 수 있다.

이러한 세 가지 유형은 단독으로뿐만 아니라 조합해서도 사용할 수 있다. 표 12.30은 컴포넌트 테스트 완료 기준의 예로 세 가지 유형의 기준을 함께 사용한 예를 보여 준다.

표 12.30 **컴포넌트의 테스트 완료 기준 예**

기준 유형	기준 예
테스트 케이스(테스트 절차) 기반 기준	• 90%의 테스트 케이스(테스트 절차)가 통과되어야 한다. • TCS-10 번과 TCS-20 번 테스트 케이스는 통과되어야 한다.
테스트 커버리지 기반 기준	• 95%의 문장 커버리지가 충족되어야 한다.
결함 기반 기준	• 2개 이하의 결함만이 허용된다. • 심각한 결함이 존재하지 않아야 한다.

12.4.9.2 분석 유형의 테스트 완료 기준

기본 유형의 테스트 완료 기준은 테스트 분석 및 설계 활동에서 생성된 테스트 케이스(테스트 절차)와 테스트 실행 활동에서 발견된 결함 정보에 바탕을 둔다. 따라서 테스트 설계 및 구현 활동과 테스트 실행 활동 그리고 결함 보고 활동의 결과물을 바탕으로 테스트 완료 기준의 충족 여부를 판단하는 것이 가능하다.

테스트 대상 및 테스트 결과에 대한 분석을 바탕으로 테스트 완료 기준을 설정할 수도 있다. 이러한 분석적 테스트 완료 기준으로는 신뢰도 예측 모델 방법, 결함 탐침 방법, 복수 테스트팀 방법이 있다. 이러한 방법은 동적 테스트 활동에 추가적인 노력이 필요하므로 시스템 테스트 및 인수 테스트 등 주요 의사 결정을 하는 경우에 테스트 완료 기준으로 사용될 수 있다.

❑ 신뢰도 예측 모델 기반 방법

테스트를 완료하는 시점은 테스트가 완료된 테스트 대상이 충분한 품질을 확보했는지 고려해야 한다. 일반적으로 테스트를 수행하면 할수록 많은 수의 결함을 검출하고 제거할 수 있으므로 높은 신뢰도를 달성할 수 있다. 만약 테스트 대상의 신뢰도가 만족스러운 수준이면 해당 시점에 테스트를 종료할 수 있다.

테스트 대상 특히 시스템에 대한 신뢰도는 실제 사용자들이 장시간 동안 시스템을 사용하는 과정에서 수집된 데이터로 확인할 수 있으며, 시스템을 테스트하는 과정에서는 신뢰도에 대한 예측만 가능하다. 그러므로 테스트 대상이 기대하는 신뢰도를 달성할 수 있는지 예측을 해야 한다.

신뢰도 예측 모델 기반 방법은 테스트 대상에 대해서 신뢰도를 예측하고, 목표로 삼은 수준 이상으로 신뢰도 달성이 예상되면 테스트가 완료되었다고 판단하는 방법이다. 기존에 수행된 유사한 다른 시스템에 대한 테스트의 수행 결과 또는 현재 테스트 중인 시스템에 대한 테스트 정보를 바탕으로 신뢰도에 대한 예측을 할 수 있다. 신뢰도 예측 모델을 적용할 때는 예측 모델에서 가정하는 상황이 현재 시스템에서 유효한지를 유의해야 한다. 예를 들어, 어떤 신뢰도 예측 모델에서 결함은 임의의 시간에 임의로 발생하고, 모든 결함은 신뢰도에 동일하게 영향을 미치며, 수정(Fix) 시간이 무시되기도 한다.

❑ 결함 탐침 기반 방법

결함 탐침 기반 방법은 테스트를 시작하기 전에 미리 다양한 유형의 결함을 시스템에 인위적으로 삽입해 놓는다(Seed). 테스트에서 발견된 결함 중에서 인위적으로 삽입된 결함의 비율을 구한다. 이 비율을 이용하여 시스템에 남아 있는 발견되지 않은 결함의 수를 예측하고, 이 수가 일정 수준 이하이면 테스트가 완료되었다고 판단한다.

결함 탐침 기반 방법에서는 삽입된 결함 중에서 발견된 결함의 비율이 시스템에 원래 존재하는 결함 중에서 테스트를 통해서 발견된 결함의 비율과 동일하다는 가정을 하고 있다. 그

러나 삽입되는 결함은 테스터의 과거의 경험에 바탕을 두므로 편향될 가능성이 있으며, 새로운 유형의 시스템을 테스트할 때 위와 같이 균형적인 결함을 만드는 것은 어려울 수 있다.

❑ 복수 테스트팀 기반 방법

이 방법은 독립적인 두 개의 테스트팀이 동일한 시스템을 테스트하여 각각 발견한 결함의 수를 바탕으로 전체 결함의 수를 예측한다. 테스트팀 A와 B가 테스트를 수행하여 발견한 결함의 수를 각각 Na와 Nb라고 한다. 그리고 두 팀에서 공통적으로 발견된 결함의 수를 Nab라고 한다. Nab는 두 팀에서 발견한 결함을 비교하여 구할 수 있다.

복수 테스트팀 기반 방법으로 표 12.31의 값을 예측할 수 있다. 즉, 시스템에 존재하는 총 결함의 수, 발견되지 못한 결함의 수를 알 수 있다.

표 12.31 **복수 테스트팀 방법의 예측값**

N	시스템에 존재하는 전체 결함의 수	
Pa	테스트팀 A에 의해서 발견된 결함의 비율	Na / N
Pb	테스트팀 B에 의해서 발견된 결함의 비율	Nb / N
Pab	테스트팀 A와 B가 공통적으로 발견한 결함의 비율	Pa×Pb Nab / N
N(a)(b)	테스트팀 A 또는 B가 발견하지 못한 결함의 수	
P(a)(b)	테스트팀 A 또는 B가 발견하지 못한 결함의 비율	

만약 테스트팀의 독립성과 결함의 독립성을 가정한다면 다음의 식이 성립된다.

번호	식
(1)	$Pab = Pa \times Pb = \frac{Nab}{N}$
(2)	$Pa = \frac{Nab}{Nb}$
(3)	$Pb = \frac{Nab}{Na}$
(4)	$P(a)(b) = (1 - Pa) \times (1 - Pb)$

위의 식 (1), (2), (3), (4)를 바탕으로 아래와 같이 N, Pa, Pb, P(a)(b), N(a)(b)를 계산할 수 있다.

$$N = \frac{Na \times Nb}{Nab}$$
$$Pa = \frac{Nab}{Nb}$$
$$Pb = \frac{Nab}{Na}$$
$$P(a)(b) = \frac{(Na - Nab) \times (Nb - Nab)}{Na \times Nb}$$
$$N(a)(b) = \frac{(Na - Nab) \times (Nb - Nab)}{Nab}$$

예를 들어, 만약 테스트팀 A가 80개의 결함을 발견하고 테스트팀 B가 90개의 결함을 발견하였으며, 조사를 통하여 두 팀에서 공통적으로 발견된 결함이 72개라고 하면, 즉, Na = 80, Nb = 90, Nab = 72라고 하면

$$N = \frac{80 \times 90}{72} = 100$$
$$N(a)(b) = \frac{(80 - 72) \times (90 - 72)}{72} = 2$$
$$P(a)(b) = \frac{(80 - 72) \times (90 - 72)}{80 \times 90} = 0.02$$

를 구할 수 있다. 이 결과에 따르면 전체 결함 중에서 2%만이 테스트를 통하여 검출되지 않았다고 예상된다. 만약, 테스트 완료 기준에서 미발견 결함이 2% 허용된다면 이 시점에서 테스트가 완료되었다고 판단할 수 있다.

이 방법은 동일한 시스템에 대해서 두 개의 테스트팀을 투입해야 하므로 테스트 비용이 많이 소요된다. 또한, 각 팀이 발견하는 결함이 서로 독립적임을 가정하지만, 두 팀이 사용하는 테스트 전략과 테스트 경험에 따라서 결함 발견의 독립성이 보장되기 어려울 수 있다.

12.4.9.3 개별 테스트의 테스트 완료 기준

테스트 완료 기준은 수행되는 각 개별 테스트별로 구체적으로 설정한다. 예를 들어, 컴포넌트 테스트는 코드 커버리지가 테스트 완료 기준에 포함될 수 있다. 시스템 테스트는 요구사항 커버리지 및 미해결 결함의 수가 테스트 완료 기준에 포함될 수도 있다. 인수 테스트는 신뢰성 예측 모형을 이용한 신뢰도도 포함될 수 있다. 표 12.32는 레벨 테스트 및 성능

유형 테스트에 대한 테스트 완료 기준의 예를 보여 준다.

표 12.32 개별 테스트별 테스트 완료 기준 예

테스트 레벨	테스트 완료 기준
컴포넌트 테스트	• 각 컴포넌트는 모든 테스트 절차가 통과되어야 한다. • 각 컴포넌트는 90% 이상의 문장 커버리지와 80% 이상의 결정 커버리지를 만족해야 한다. • 전체 컴포넌트 중에서 90% 이상의 컴포넌트가 위의 커버리지 조건을 충족해야 한다.
통합 테스트	• 컴포넌트 수행 커버리지는 95% 이상이어야 한다. • 컴포넌트 호출 커버리지는 80% 이상이어야 한다.
시스템 테스트	• 모든 테스트 절차가 통과되어야 한다. • 유스케이스 시나리오 커버리지는 100%가 되어야 한다. • 결함 탐침 방법을 사용하여 예상된 미검출 결함의 수는 5개 이하이어야 한다.
성능 테스트	• 경미한 수준의 결함을 제외한 모든 테스트 절차가 통과되어야 한다.
인수 테스트	• 모든 테스트 절차가 통과되어야 한다. • 신뢰도 예측에 따른 신뢰도가 99% 이상이어야 한다.

12.4.10 조직 테스트 전략과의 차이점

이 절에는 수립된 테스트 계획과 조직 테스트 전략의 차이점과 함께 다른 전략을 수립한 근거를 기술한다. 테스트 전략 대부분의 항목은 조직 테스트 전략 명세서에 정의된다. 그러므로 조직 테스트 전략 명세서와 동일한 내용은 단순히 참조로 기술하면 충분하다. 따라서 조직 테스트 전략 명세와 차이 나는 사항을 테스트 계획서에 기술한다.

예를 들어, 조직 테스트 전략 명세서의 “수행 개별 테스트”에는 보안 테스트가 명시되어 있지만 금번 테스트 프로젝트의 보안과 관련해서 기존에 이미 충분히 테스트 되었거나 심각한 수준의 위험이 없다면 보안 테스트를 별도로 수행하지 않고 시스템 테스트의 범위에 보안을 포함할 수 있다.

참고로 표 12.33은 테스트 계획서의 테스트 전략 항목에 해당되는 조직 테스트 명세서의 항목을 보여 준다. “테스팅 중단 및 재시작 조건”은 테스트 계획서에는 기술되지만, 조직 테스트 명세서에는 명시되지 않는다.

표 12.33 테스트 계획서의 테스트 전략과 조직 테스트 전략 명세서 비교

테스트 계획서 – 테스트 전략	조직 테스트 전략 명세서
개별 테스트	프로젝트 수준 – 수행 개별 테스트
테스트 산출물	개별 테스트 수준 – 테스트 문서화
테스트 설계 기법	개별 테스트 수준 – 테스트 설계 기법
테스트 환경 요건	개별 테스트 수준 – 테스트 환경 및 테스트 데이터
테스트 데이터 요건	개별 테스트 수준 – 테스트 환경 및 테스트 데이터
재테스팅 및 리그레션 테스팅	개별 테스트 수준 – 재테스팅 및 리그레션 테스팅
테스팅 중단 및 재시작 조건	해당 없음
테스트 메트릭	개별 테스트 수준 – 테스트 메트릭
테스트 완료 기준	개별 테스트 수준 – 테스트 완료 기준

12.6 산출물 요약

12.6.1 테스트 계획서

표 12.38은 테스트 계획서 산출물의 구성 항목을 보여 준다. 테스트 계획서는 상위 수준에서 테스트 컨텍스트, 위험 분석, 테스트 전략, 테스트 수행 계획을 기술한다. 테스트 컨텍스트는 테스트 계획 유형, 테스트 대상, 테스트 범위, 가정 및 제약 사항, 그리고 이해관계자를 기술한다. 위험 분석은 프로젝트 위험과 제품 위험을 기술한다. 테스트 전략은 수행할 개별 테스트, 테스트 산출물, 테스트 설계 기법 등을 기술하며, 테스트 수행 계획은 조직, 일정, 의사소통 방법을 기술한다.

표 12.38 테스트 계획서 구성

테스트 컨텍스트	
테스트 계획 유형	
테스트 대상	
테스트 범위	
가정 및 제약 사항	
이해관계자	
위험 분석	
프로젝트 위험	
제품 위험	
테스트 전략	
개별 테스트	
테스트 산출물	
테스트 설계 기법	
테스트 환경 요건	
테스트 데이터 요건	
재테스팅 및 리그레션 테스팅	
테스팅 중단 및 재시작 기준	
테스트 메트릭	
테스트 완료 기준	
조직 테스트 전략과의 차이점	
테스트 수행 계획	
테스트 조직/인력 및 역할	
테스트 활동 및 일정	
의사소통	