9장. 모듈성: 분리와 재조합을 위한 기준

모듈성

- 시스템의 컴포넌트를 분리하고 재조합할 수 있는 특성
- 종종 유연성과 다양한 사용이라는 이점을 제공

모듈성의 특징

- 단순한 수준에서 모듈은 프로그램에 포함될 수 있는 명령어와 데이터의 집합

코드가 덜 복잡한 것이 왜 중요한가요?

- 복잡성은 소프트웨어의 소유 비용을 증가 시킨다.
    - 정의상 변경하기가 어려움

TDD의 교훈: 테스트가 어렵다면 설계도 문제다

- 설계 간 테스트를 하지 않을 경우 품질을 나타내는 객관적 척도가 없다.
- 그렇다고 TDD 가 자동으로 훌륭한 설계를 만들어 낸다는 뜻은 아니다.

TDD 로 모듈성 강화

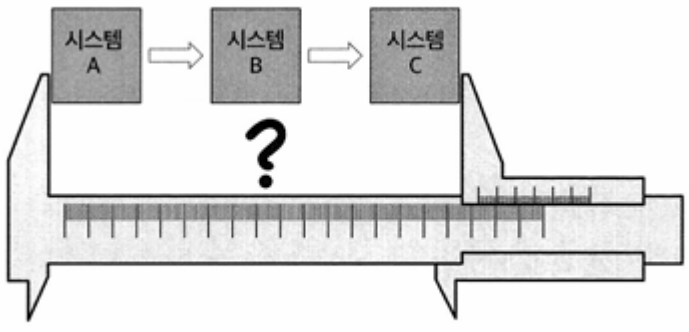

그림 9.2 결합된 시스템을 테스트하기

- 위 그림과 같이 시스템이 크고 복잡할 수록 결과의 가변성 역시 커지기 때문에 그다지 도움이 되지 않음
    - 물론 통합테스트 적인 측면에선 도움이 되지만 시스템의 상세한 통제를 제어하기 어려움

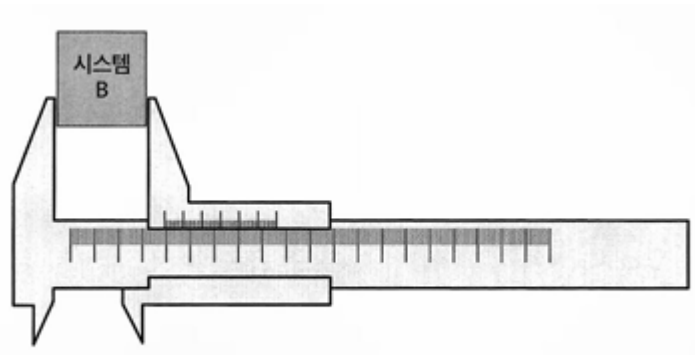

그림 9.3 모듈을 테스트하기

- 모듈화가 잘되어 있으면 캘리퍼스의 범위를 작게 잡아 자세히 테스트 가능

REST API 로 모듈성을 강화하자

- 대규모 코드 기반에서 가장 흔히 볼 수 있는 문제 중 하나는 경계의 양쪽에서 알려진 것과 노출된 것 사이에 차이를 무시한 결과다.
- 서비스의 진입점은 해당 서비스를 사용하는 소비자가 최악의 형태로 남용하지 못하게 제한하는 작은 방어벽이 되어야 한다.
    - 일종의 포트와 어댑터 모델이다.

모듈의 규모는 크고 작음이 없다.

- 모듈성은 모든 규모에서 중요하다.
- 각 클래스, 메서드 또는 함수는 간단하고 읽기 쉬워야 하며, 적절한 경우 독립적으로 이해할 수 있는 더 작은 하위 모듈로 구성되어야 한다.
    - TDD 가 이러한 세분화된 코드를 장려하는 데 도움이 된다.
- 일부 사람들은 이런 설계 스타일을 비판한다.
    - 표면적이 큰 코드를 이해하기 훨씬 더 어렵다는 형태로 나타낸다.
    - 저자는 잘못된 설계로 인한 테스트 부족으로 인해 코드의 표면적이 가려지는게 더 이해하기 어렵다고 말하고 있음

정리

- 모듈성은 소프트웨어가 미래에 어떻게 작동해야 마땅한지에 대한 전지전능한 관점이 없을 때 우리가 진전할 수 있는 능력의 초석이다.
- 모듈성은 시스템의 복잡성을 관리할 때 고려할 필요가 있는 그 밖의 아이디어, 즉 추상화, 관심사 분리, 결합도, 응집성과 같은 아이디어와 밀접하게 연결되어 있다

[요약]

1. 모듈성의 정의와 중요성

모듈성(modularity)은 **시스템의 컴포넌트를 분리하고 재조합할 수 있는 특성**으로, 유연성과 다양한 사용이라는 이점을 제공합니다. 이는 복잡성 관리의 핵심 도구입니다.

- **복잡성 관리:** 현대 소프트웨어 시스템은 방대하고 정교하여 인간의 능력을 뛰어넘기 때문에, 모듈성은 시스템을 더 작고 이해하기 쉬운 조각으로 나누어 개발자가 다른 부분에 대해 걱정하지 않고 당면한 작업에 집중할 수 있게 합니다.
- **이상적인 모듈:** 모듈 내에 위치한 코드는 유용한 작업을 수행하기 위해 시스템의 다른 부분을 요구하더라도, 시스템의 다른 부분의 컨텍스트를 벗어나 **독립된 단위로 쉽게 이해할 수 있을 만큼 충분히 짧아야 합니다**.
- **설계의 중요성:** 모듈성을 위한 설계는 프로그래밍 언어의 구문을 아는 것과는 분명히 다른 유형의 기술입니다. 이는 평생을 들여도 결코 완벽해질 수 없는 기술이지만, **뛰어난 프로그래머와 그렇지 못한 프로그래머를 구분하는 진정한 기술**입니다.

2. 모듈성과 테스트 가능성 (Testing & Design)

모듈성은 테스트 가능성을 지탱하고 향상하며, **테스트는 설계를 이끄는 방식**으로 모듈성을 장려합니다.

- **측정 지점:** 시스템의 무결성을 손상시키지 않으면서도 시스템의 동작을 검사할 수 있는 많은 **측정 지점**이 있도록 시스템을 설계해야 합니다. 측정 지점은 특정 상태에서 코드를 얼마나 쉽게 설정하고 동작 결과를 얼마나 쉽게 관찰하고 평가할 수 있는지를 정의합니다.
- **결정론과 정밀도:** 모듈성은 우리가 측정할 수 있는 대상을 더 잘 제어하고 더 정밀하게 측정할 수 있게 합니다. 특정 테스트나 평가의 경우, 테스트 대상 소프트웨어의 버전이 동일하다면 매번 **동일한 결과가 나올 것으로 예상해야** 합니다 (결정론적이어야 함).
- **프랙탈 성격:** 테스트 가능성은 프랙탈이며, **전체 엔터프라이즈 시스템 수준과 코드 몇 줄이라는 좁은 초점 모두에서** 관찰하고 도구로서 사용할 수 있습니다. 테스트 가능성은 모듈화된 설계를 장려하여 궁극적으로 더 읽기 쉬운 코드를 만듭니다.

3. 모듈성과 아키텍처적 적용

모듈성은 시스템 설계 및 배포 방식에 근본적인 영향을 미칩니다.

- **서비스와 봉합 지점:** 서비스는 코드의 다른 부분으로부터 세부 사항을 숨기는 구획이며, 모듈 간의 통신은 모듈 내부의 통신보다 **더 보호되어야 합니다**. 모듈과 모듈이 맞닿아 있는 경계(봉합 지점)는 정보 은닉을 달성하는 데 매우 중요합니다.
- **배포 파이프라인의 범위:** 배포 파이프라인은 릴리스 가능성을 결정하는 메커니즘이며, 그 결과물이 독립적으로 배포 가능하다는 사실은 평가 범위가 **'독립적으로 배포 가능한 소프트웨어 단위'**여야 함을 의미합니다.
- **마이크로서비스:** 마이크로서비스는 **모듈화가 매우 잘되어 있기 때문에** 릴리스 전에 다른 서비스와 함께 테스트할 필요가 없습니다. 이는 조직적인 의존성을 최소화하고 팀 간 조정 비용 없이 고품질로 신속히 작업할 수 있는 **조직의 확장성을 위한 패턴**입니다.

4. 고성과 조직과 모듈성

조직적인 관점에서 모듈성은 개발의 수직 확장에 필수적입니다.

- **인력 추가의 한계:** 주어진 컴퓨터 시스템에는 인력 추가에 심각한 한계가 있으며, 더 많은 인원을 추가하면 속도가 느려집니다. 그 이유는 **결합도** 때문입니다.
- **조직의 자율성:** 모듈식 접근 방식은 팀이 훨씬 더 독립적으로 작업할 수 있게 하여, 모듈형 조직은 전통적인 조직 구조보다 훨씬 더 유연하고 확장성이 뛰어나며 효율적입니다. 고성과 팀은 다른 그룹의 허가나 조율 없이 팀 내에서 의사결정을 내릴 수 있습니다. 이는 팀이 **정보적으로 결합이 분리**되어 있다는 뜻입니다.

모듈성은 시스템의 복잡성을 방어하는 데 필요한 도구 모음 중 첫 번째이며, 다른 아이디어(응집성, 관심사 분리 등)와 함께 시스템의 품질을 높이는 데 기여합니다.

비유: 모듈성은 복잡한 기계를 레고 블록으로 만드는 것과 같습니다. 레고 블록(모듈)은 크기가 작고, 정의된 연결점(인터페이스)을 가지며, 각각이 독립적으로 완벽하게 기능합니다. 기계를 만들 때 블록 하나하나를 이해하는 것은 쉽고, 나중에 고장 나거나 개선이 필요할 때 전체 기계를 부수지 않고 문제가 되는 블록만 교체(변경)할 수 있습니다. 이는 전체 시스템의 복잡성을 관리하고 유연하게 발전시키는 핵심적인 방법입니다