## 고장이 나지 않으면 수정하지 않기

*이름은 일반적으로 "~하기"이다.*

*의도: 차이를 만들 시스템의 일부분을 위해 여러분의 리엔지니어링 노력을 아끼자*

*의도는 패턴의 에센스를 표현한다.*

### 문제

*문제는 간단한 질문으로 바꿔 말할 수 있다. 때로는 맥락을 명시적으로 설명한다.*

레거시 시스템의 어떤 부분을 리엔지니어링 할 것인가?

*이 문제는 다음과 같은 이유로 어렵다.*

*포스에 대해서 논의 한다. 여기서 문제의 어려움과 흥미로운 관점에 대해 논의 한다. 또한 해결에 대한 주요한 부분에 대하 다룬다.*

- 레거시 시스템은 크고 복잡하다.
- 전체 재작성은 비싸고 리스크가 크다.

*그러나 이 문제를 해결할 수 있는 이유는 다음과 같다.*

- 리엔지니어링을 분명한 목표를 가지고 진행한다.

### 해결

*해결에는 때때로 패턴을 적용하기 위한 단계별 레시피를 포함한다.*

계획된 변경에 더 이상 대응할 수 없는 "고장난" 부분만 수정한다.

### 트레이드오프

*각 패턴은 긍정적인 면과 부정적인 면의 트레이드오프를 포함한다.*

***장점*** 당신분 문제의 핵심 경로가 아닌 것들을 고치느라 시간을 낭비하지 않는다.

***단점*** 당장 중요해 보이지 않는 수정을 미루면 장기적으로 더 큰 비용이 들 수 있다.

***어려움*** 무엇이 "고장"인지 판단은 어려울 수 있다..

*패턴 적용의 현실적 사례를 포함할 수 있다.*

### *근거*

레거시 시스템에는 보기 싫지만 잘 작동하고 유지보수에도 큰 부담이 되지 않는 부분들이 있을 수 있다. 이러한 구성 요소들을 격리하고 래핑할 수 있다면, 굳이 교체할 필요가 없을 수도 있다.

해결이 왜 타당한지 근거를 설명한다.

### 알려진 용도

이 패턴의 잘 문서화된 사례를 몇 가지 제시한다.

데이비스는 "소프트웨어 개발의 201가지 원칙"에서 이 내용을 다룬다.

### 관련 패턴

관련된 패턴은 대안으로 제안할 수 있다. 다른 패턴은 논리적인 후속 조치로 제안할 수 있다.

증상이 아닌 문제 수정하기

### 다음 단계

"가장 가치 있는 것 먼저 하기"에서 시작하기를 고려한다.