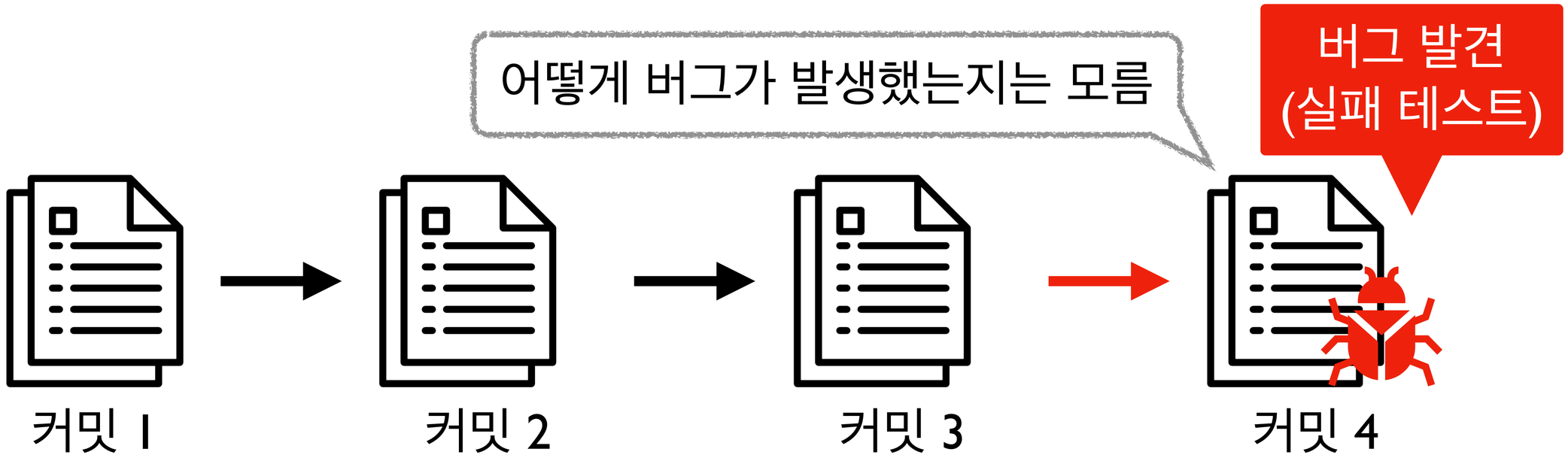
버그의 시발점을 찾아서

SIGPL 여름학교 2026
고려대학교 소프트웨어 분석 연구실
김동욱

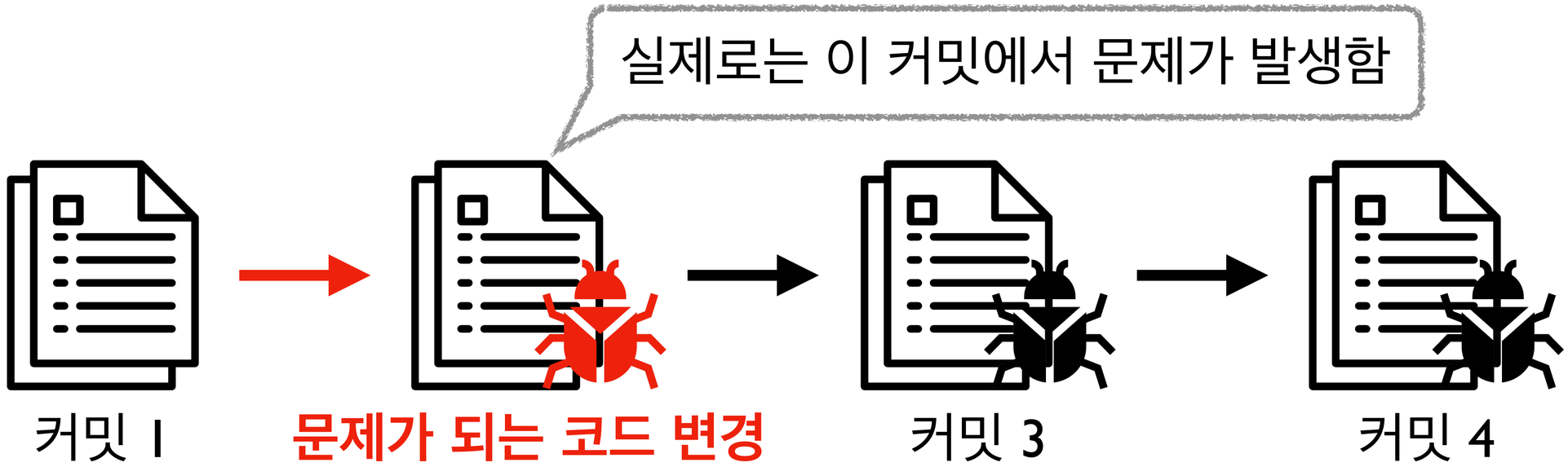
버그 유발 커밋

- 어떤 코드 변경이 버그를 만들었는지를 알 수 있다면?
- 진짜 원인을 파악할 수 있어 디버깅 과정에 큰 도움이 됨



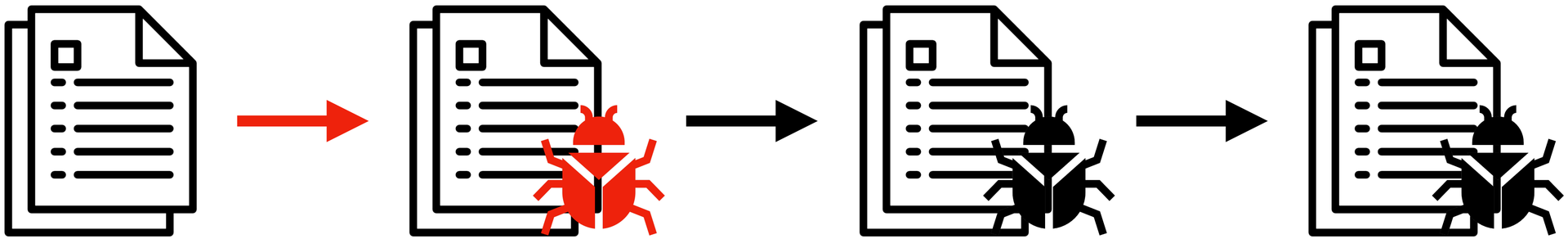
버그 유발 커밋

- 어떤 코드 변경이 버그를 만들었는지를 알 수 있다면?
- 진짜 원인을 파악할 수 있어 디버깅 과정에 큰 도움이 됨



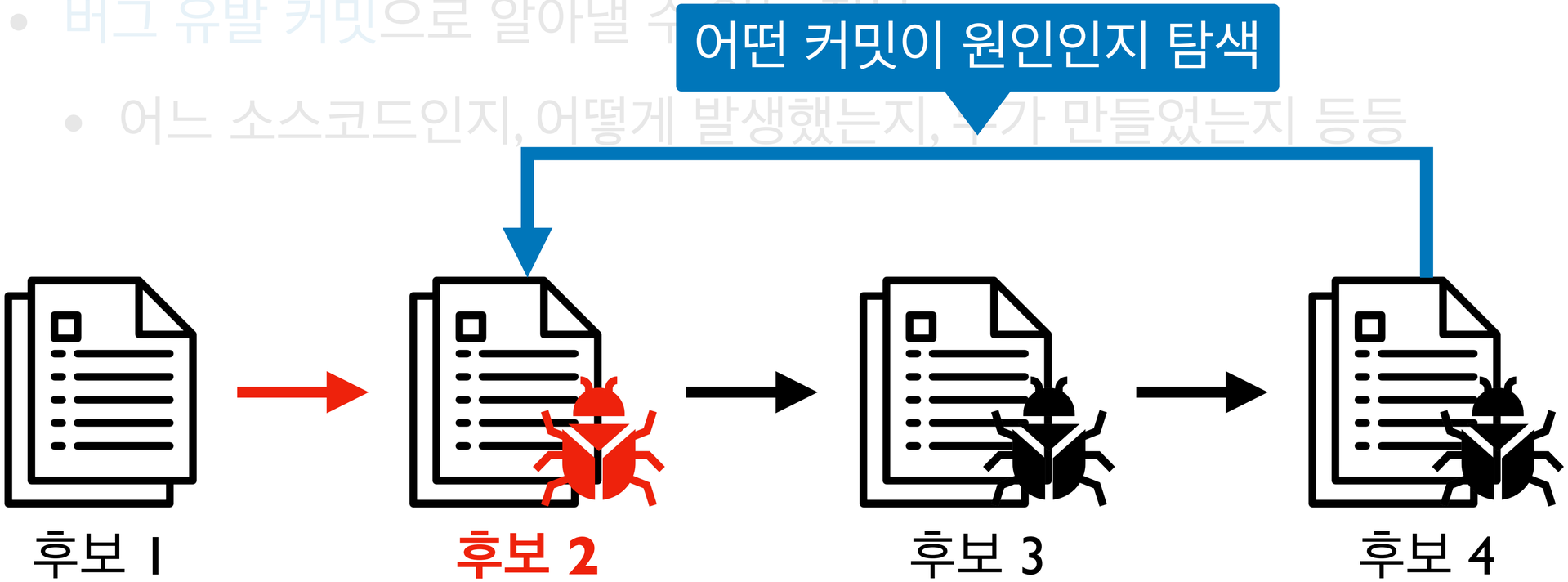
버그 유발 커밋

- Bug-Inducing Commit (BIC): 새로운 버그를 발생시키는 최초의 커밋
- 버그 유발 커밋으로 알아낼 수 있는 정보:
  - 어느 소스코드인지, 어떻게 발생했는지, 누가 만들었는지 등등



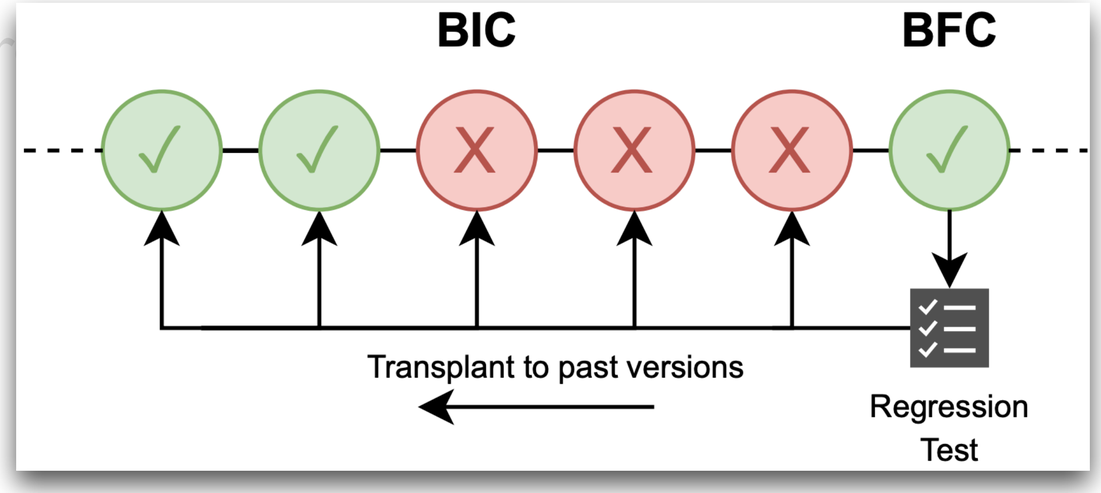
목표: 버그 유발 커밋 찾아내기

- Bug-Inducing Commit (BIC): 새로운 버그를 발생시키는 최초의 커밋
- 버그 유발 커밋으로 알아낼 수 있는 정보
  - 어느 소스코드인지, 어떻게 발생했는지, 누가 만들었는지 등등



기존 기술

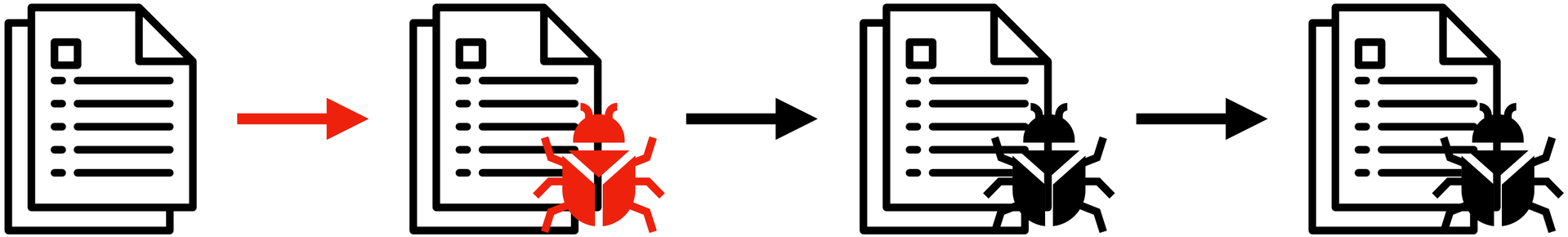
- Dynamic Approach (e.g., ***git bisect***)
  - 과거 프로그램을 직접 실행해 테스트가 실패하는 커밋을 탐색함
  - **장점**: BIC를 정확하게 찾을 수 있음
  - **단점**: 빌드/테스트 호환성, 실행 비용
- Static Appr

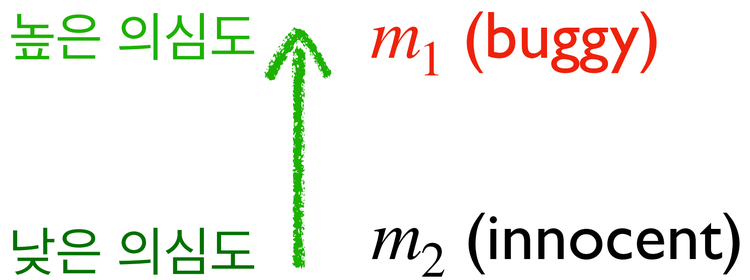


기존 기술

- Dynamic Approach
- Static Approach
  - 현재 테스트 실행 정보에 기반해 예전 커밋들을 탐색함
  - 테스트 실행 정보 (FL) + 프로그램의 구문/의미 변화

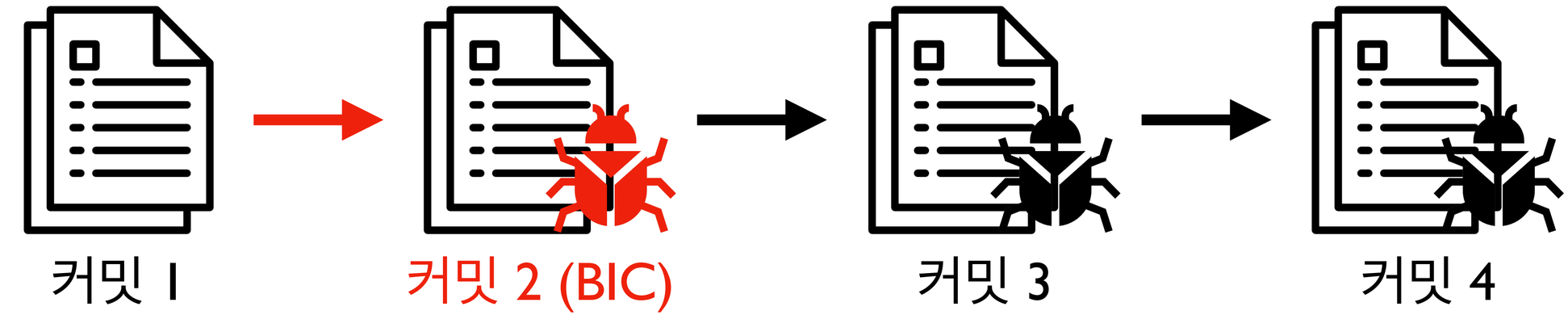
기존 기술

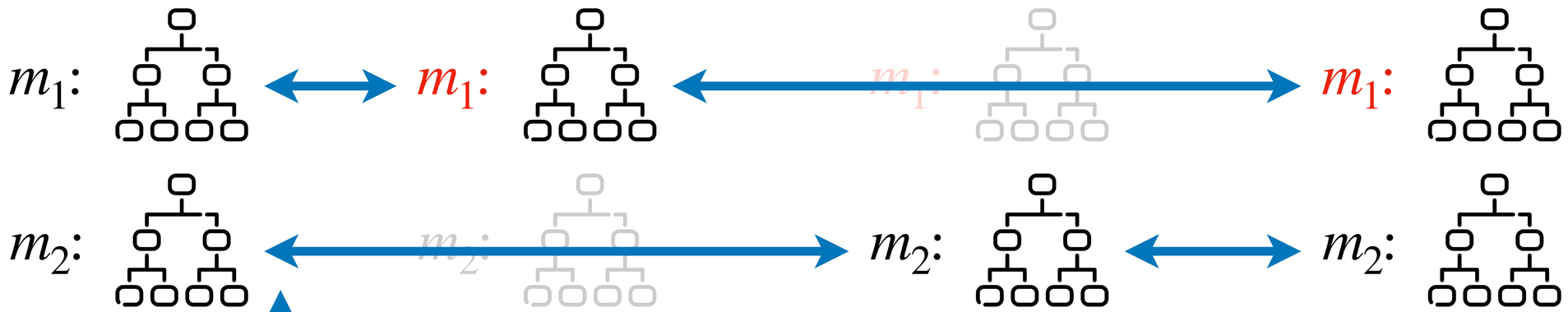




1. 버그와 연관된 의심 메서드들 탐색

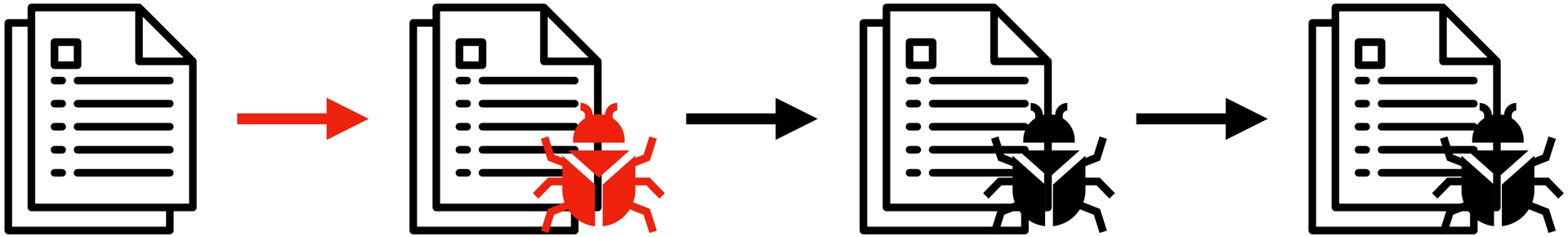
기존 기술





2. 커밋들의 코드 변경을 분석

문제 상황



커밋 1

수정한 메서드:

…

커밋 2 (BIC)

수정한 메서드:

$m_1, \ldots$

커밋 3

수정한 메서드:

$m_2, \ldots$

커밋 4

수정한 메서드:

$m_1, m_2, \ldots$

탐색 공간이 지나치게 넓음 (프로젝트 규모 × 커밋 수)

=> 비용이 매우 비쌈 or 효과적이지 못함

아이디어

- 모든 구문/의미 변화를 **효율적으로** 저장하고 갱신하는 히스토리 지식 베이스
- 를 활용하여 버그 유발 커밋을 **효과적으로** 탐색하기

