

Module 3 — 가설 시뮬레이션: 숫자보다 먼저 '못 믿을 조건'을 정한다

시뮬레이터를 만드는 것은 반나절이면 됩니다. 어려운 것은 그 숫자를 회의에서 근거로 쓸 수 있게 만드는 일입니다. 이 프로젝트에서 판정 신뢰가 깨진 사고는 전부 시뮬레이터 바깥에서 났습니다 — 상태 플래그 하나를 근거로 삼았을 때였습니다.

모듈	난이도	학습 시간	선행 학습
Module 3	심화	120분	Module 1

학습 목표 (Learning Objectives)

이 모듈을 마치면 수강자는 다음을 할 수 있습니다.

- ✓ 기능 가설을 정량 지표로 환산하고 **임계값을 착수 전에** 확정할 수 있다
- ✓ 몬테카를로 시뮬레이션으로 실패율·절감 공수를 추정하고 **오타락 비용까지** 계산할 수 있다
- ✓ 판정을 참/거짓 2값이 아니라 **기각 / 기각 근거 없음 / 판정 불가 3값**으로 설계할 수 있다
- ✓ "연결됨" 같은 **상태 플래그를 근거로 삼으면 안 되는 이유**를 실제 사례로 설명할 수 있다
- ✓ 시뮬레이션이 **기각에 강하고 채택에 약하다**는 한계를 결과와 함께 노출할 수 있다

실무 배경 — 어떤 문제에서 출발했나

- **상황:** 검증되지 않은 가설이 그대로 개발로 넘어갔습니다. 분기당 착수 검토 기능은 20여 건이고, 틀렸다는 사실은 출시 후에 알게 됩니다. 롤백된 기능에 들어간 공수는 회수되지 않습니다.
- **1차 해결과 그 한계:** 착수 전에 가설을 걸러내는 시뮬레이터(`/simulator`)를 만들었습니다. 여기까지는 쉬웠습니다. 문제는 그 다음입니다 — "이 숫자를 왜 믿어야 하는가"에 답할 수 없으면 회의에서 아무것도 바꾸지 못합니다.
- **진짜 문제:** 같은 프로젝트에서 **판정을 잘못 믿을 뻔한 사고**가 실제로 났습니다. Jira 연동 상태를 확인할 때 응답의 `offline: false` 를 보고 "연결됐다"고 판단할 뻔했는데, **토큰이 틀려도 그 값은 false 로 나옵니다(\$5)**. 시뮬레이터의 숫자든 연동의 상태든, **관측치가 아니라 플래그를 근거로 삼는 순간 판정은 무너집니다.**

💡 **이 모듈의 핵심 교훈:** 시뮬레이션의 값어치는 정밀도가 아니라 **반증 가능성**에 있습니다. "어떤 결과가 나오면 이 숫자를 못 믿는가"를 먼저 적으세요.

§2

실무 배경 — 어떤 문제에서 출발했나 (계속)

3.1 프롬프팅 프레임워크 적용 — R-C-T-C-F

요소	실제 입력값
Role	데이터 기반 의사결정을 하는 PO. 숫자를 만드는 사람이 아니라 숫자를 의심하는 사람
Context	분기당 기능 가설 20여 건, 과거 롤백률 34%. 강의 중 학습자가 자기 조직 숫자를 넣고 직접 돌린다
Task	가설별 착수 여부 판정과 절감 공수 추정 파이프라인 구현
Constraints	결정론(시드 고정) · 오탈락 비용을 결과에 항상 노출 · 판정 불가를 미완성과 구분 · 외부 스크립트 실행 기본 차단
Format	지표 표 + 실패율 분포 히스토그램 + <code>notes</code> (한계와 주의사항)

Constraints 의 두 번째 줄이 결과를 갈랐습니다.

"오탈락 비용을 결과에 **항상** 노출한다"

3.1 프롬프팅 프레임워크 적용 — R-C-T-C-F (계속)

이 문장이 없으면 시뮬레이터는 "검출률을 올리면 실패율이 내려간다"는 좋은 뉴스만 보여 줍니다. 좋은 뉴스만 보여 주는 도구는 의사결정을 돕는 게 아니라 이미 내린 결정을 정당화합니다.

3.2 아키텍처 — 몬테카를로 + 3값 판정

[입력: 기능 수 · 기존 실패율 · 검출률 · 오탈락률 · 기능당 공수 · 시행 횟수]

|

▼

[시드 PRNG (mulberry32)] ← Math.random() 금지. 화면과 교안의 숫자가 달라지면 안 됨

|

▼

[몬테카를로 N회] 기능마다: 실패 가설인가? → 걸러졌는가?

|

└─ 착수분의 실패율 (개선된 실패율)

└─ 걸러낸 실패 가설 × 공수 (절감)

└─ 오탈락 × 공수 (손실) ← 이걸 빼면 순이익이 부풀려짐

|

▼

[결과] 지표 표 + 분포 10버킷 + notes(한계)

|

└─ 실패 시: 예외를 던지지 않고 MOCK 폴백 + fallbackReason 노출

§3

3.2 아키텍처 — 몬테카를로 + 3값 판정 (계속)

§4

구현 과정 (Step by Step)

단계	작업	소요	산출물
1	가설을 정량 지표·임계값으로 환산	20분	<code>module-3-hypotheses.csv</code>
2	시드 PRNG 로 결정론 확보	20분	<code>createRng()</code>
3	오탈락 비용을 지표·notes 양쪽에 노출	30분	<code>runHypothesisPipeline()</code>
4	판정을 3값으로 설계	30분	<code>resolvePracticeBindings()</code>
5	실패 폴백과 실행 경로 가드	20분	<code>runSimulation()</code> · <code>resolveScript()</code>

4.1 결정론은 취향이 아니라 요구사항이다

강의 중에 같은 입력으로 두 번 돌렸는데 숫자가 다르면, 학습자는 그 도구를 다시 열지 않습니다. 그래서 `Math.random()` 을 쓰지 않고 시드 PRNG 를 씁니다.

```
/** mulberry32 - 32비트 시드 하나로 재현 가능한 균등분포를 냅니다. */
export function createRng(seed: number): () => number {
  let state = seed >>> 0;
  return () => {
    state = (state + 0x6d2b79f5) >>> 0;
    let t = state;
    t = Math.imul(t ^ (t >>> 15), t | 1);
    t ^= t + Math.imul(t ^ (t >>> 7), t | 61);
    return ((t ^ (t >>> 14)) >>> 0) / 4294967296;
  };
}
```

4.1 결정론은 취향이 아니라 요구사항이다 (계속)

결정론이 만든 부수 효과가 하나 더 있습니다. 시뮬레이터 설정의 목(mock) 출력을 손으로 적지 않고 파이프라인을 실제로 돌려서 만듭니다 (`prisma/seed.ts`). 목 데이터가 실제 계산과 어긋나는 사고가 구조적으로 불가능해집니다.

4.2 오탈락 — 좋은 뉴스만 보여 주지 않기

한 판의 규칙은 두 줄입니다. 실패 가설은 검출률로, 성공 가설은 오탈락률로 걸러집니다.

```
const willFail = rng() < input.baselineFailureRate;  
const rejected = rng() < (willFail ? input.screeningRecall : input.falseRejectRate);
```

검출률만 올리면 실패율은 계속 내려갑니다. 그래서 **오탈락으로 날린 공수를 결과에 항상 함께 적습니다.**

오탈락으로 날린 공수는 약 N man-day 입니다.
검출률만 올리고 오탈락을 안 보면 순이익이 과대평가됩니다.

이 문장이 사라지지 않도록 테스트로 고정했습니다.

4.2 오탈락 — 좋은 뉴스만 보여 주지 않기 (계속)

```
test('오탈락 비용을 주의사항에 반드시 적는다', () => {  
  const output = runHypothesisPipeline(base, HYPOTHESIS_SCHEMA.seed);  
  assert.ok(  
    output.notes.some((note) => note.includes('오탈락')),  
    '오탈락 비용을 숨기면 순이익이 과대평가됩니다',  
  );  
});
```

 **일반화:** 지표의 좋은 쪽과 나쁜 쪽은 **같은 화면에** 있어야 합니다.

나쁜 쪽을 상세 페이지로 밀면 아무도 안 봅니다.

4.3 판정은 2값이 아니라 3값이다

Module 2 실습 바인딩(Tool A~D 완성 여부)을 판정하면서 배운 것인데, 시뮬레이션 결과 해석과 정확히 같은 구조입니다.

판정	의미	잘못 뭉개면
READY	기준을 만족함	—
PENDING	기준을 만족하지 못함 (근거 있음)	—
UNKNOWN	조회 자체가 실패해 판정할 수 없음	이미 완성됐던 도구가 되돌아간 것처럼 보임

```
if (jira.offline) {  
  return { ...binding, status: 'UNKNOWN', doneCount: 0, totalCount: 0, jiraUrl, confluenceUrl,  
    note: 'Atlassian 자격증명이 없어 완성 여부를 판정하지 못했습니다.' };  
}
```

4.3 판정은 2값이 아니라 3값이다 (계속)

시뮬레이션도 같습니다. 통과한 가설은 "성공한다"가 아니라 "지금 기각할 근거가 없다" 입니다. 세 번째 값을 없애고 2값으로 접으면, 모르는 것이 아는 것으로 둔갑합니다.

4.4 실패해도 던지지 않는다

강의 중에 시뮬레이터가 500 을 뱉는 것보다, "지금은 저장된 예시 결과를 보여주는 중"이라고 화면에 쓰는 편이 낫습니다. 단 왜 폴백했는지를 반드시 함께 적습니다.

```
return {
  /* ... */
  source: 'MOCK',
  output: config.mockOutputData,
  fallbackReason:
    `${config.scriptPath}` 는 내장 파이프라인이 아닙니다. ` +
    '외부 스크립트를 실제로 돌리려면 SIMULATOR_ALLOW_SCRIPT_EXEC=true 로 두고 scripts/ 아래에 두세요.',
};
```

이유 없는 폴백은 Module 1 §4.3 의 빈 템플릿과 같은 사고입니다 — 화면은 채워져 있고 내용은 가짜인데, 그 사실이 어디에도 안 적혀 있습니다.

4.5 시뮬레이터가 원격 코드 실행 통로가 되지 않게

관리자 화면에서 `scriptPath` 를 아무 값으로나 저장할 수 있다면, API 하나가 임의 파일 실행 경로가 됩니다. 내장 레지스트리만 실행하는 것이 기본값이고, 환경변수를 켜 경우에도 프로젝트 안 경로·허용 확장자만 통과합니다.

```
export function resolveScript(scriptPath: string): string | null {
  const root = path.join(process.cwd(), 'scripts');
  const resolved = path.resolve(root, scriptPath.replace(/^scripts[\\\/]/, ''));
  if (!resolved.startsWith(`${root}${path.sep}`)) return null;
  if (!/\. (py|js|mjs)$/.test(resolved)) return null;
  return resolved;
}
```

5.1 파이프라인 자체

지표	Before	After	측정 근거
같은 입력 재현성	보장 없음	항상 동일	<code>tests/simulator.test.ts</code> 결정론 3건
오탈락 비용 노출	없음	지표 + <code>notes</code> 양쪽	테스트로 고정
실행 실패 시 화면	500	MOCK 폴백 + 이유 표기	<code>runSimulation()</code>
임의 스크립트 실행	가능	화이트리스트 + 경로 가드	실행 경로 가드 테스트 3건

5.2 판정 신뢰 — 2026-08-10 실측에서 실제로 있었던 일

여기까지의 연동은 코드와 테스트로만 존재했습니다. 배포본에 자격증명을 넣고 실제 Atlassian 에 붙여 본 기록입니다(04 §9).

① `offline: false` 를 근거로 삼을 뻔했습니다.

응답에 `offline: false` 가 찍히면 연결된 것처럼 보입니다. 아닙니다 — 토큰이 틀려도 이 값은 **false** 로 나옵니다. 배너만 보고 "연동 완료"라고 판정하면 그 뒤의 모든 판정이 빈 결과 위에서 돌아갑니다.

판정 근거를 플래그에서 관측치로 바꿨습니다.

근거로 삼은 것	신뢰 가능	이유
<code>offline: false</code>	✗	자격증명 유효성과 무관하게 false
<code>recentIssues</code> 10건 · <code>recentIssuesError: null</code>	✓	실제로 읽어 온 데이터
<code>LECTWEB-12</code> · <code>LECTWEB-13</code> 신규 생성	✓	실제로 쓴 결과

5.2 판정 실패 — 2026-08-10 실측에서 실제로 있었던 일 (계속)

② **Tool A~D** 판정이 처음으로 진짜 값을 냈습니다.

자격증명이 없던 동안에는 네 도구가 전부 **UNKNOWN** 이었습니다.

도구	판정	완료
Tool A	PENDING	0/1
Tool B	READY	1/1
Tool C	READY	1/1
Tool D	READY	1/1

Tool A 만 **PENDING** 인 것은 버그가 아니라 **진행 상황을 정확히 읽은 것**입니다. 2값 판정이었다면 **UNKNOWN** 시절의 네 건이 전부 "미완성"으로 보였을 것이고, 자격증명을 넣은 순간 세 건이 갑자기 완성된 것처럼 보였을 겁니다.

확인 순서 — 다음에도 이 순서로 합니다.

5.2 판정 실패 — 2026-08-10 실측에서 실제로 있었던 일 (계속)

1. `INTEGRATION_DRY_RUN=true` 인 채로 자격증명만 넣고 읽기부터
2. `recentIssues` 로 실제 읽기 검증 — `offline: false` 만으로는 부족
3. `false` 로 바꿔 쓰기 1건만
4. 확인 후 즉시 `true` 로 복구

⚠ 이 세션의 네트워크 정책이 `awesomest.atlassian.net` 과 `*.vercel.app` 을 차단하므로 위 실측은 배포본에서 수행했고, 이 문서에서 재확인하지 않았습니다. 숫자는 2026-08-10 관측치입니다. 결정론·오탐락·폴백·경로 가드는 로컬 테스트가 계속 고정합니다.

배운 것 / 다시 한다면 바꿀 것

-  **잘한 것:** 오탈락 비용을 테스트로 고정한 것. 지표를 지우려면 테스트를 먼저 지워야 하고, 그건 눈에 띄니다.
-  **잘한 것:** 판정에 세 번째 값을 둔 것. "모른다"를 표현할 수 없는 시스템은 모르는 것을 아는 것처럼 말합니다.
-  **다시 한다면:** 연동 확인 항목에 `offline` 을 넣지 않았을 겁니다. 신뢰할 수 없는 플래그를 화면에 띄워 두면 언젠가 누군가 그걸 근거로 씁니다.
-  **다시 한다면:** 시뮬레이터를 만들기 전에 "어떤 결과가 나오면 이 숫자를 못 믿는가" 를 먼저 적었을 겁니다. 그 목록이 곧 테스트가 됩니다.

산출물 링크 (Deliverables)

구분	URL
교안 PDF/슬라이드	/downloads/module-3-simulation.pdf
슬라이드 (HTML)	/downloads/module-3-simulation.html
실습용 데이터 세트	/downloads/module-3-hypotheses.csv
시뮬레이터 실행	/simulator
GitHub 저장소	https://github.com/ursulax/action
파이프라인 구현	<code>lecture-web/src/lib/simulator-pipeline.ts</code>
결정론·가드 테스트	<code>lecture-web/tests/simulator.test.ts</code>
연동 실측 기록	<code>docs/4-product-system/04-product3-course-web.md</code>
관련 PR	https://github.com/ursulax/action/pull/45

실습 과제 (Action Item)

1. 자기 팀의 최근 1년 롤백률을 구하세요. 없으면 체감값을 쓰되 "체감값"이라고 적으세요 (Module 1 §4.4 — 출처 없는 숫자는 출처 없다고 적힌 채로 두는 것이 낫습니다).
1. `/simulator` 에 넣고 검출률을 **0.5 / 0.7 / 0.9** 로 바꿔 절감 공수를 비교하세요. 그다음 오탈락률을 0.05 → 0.20 으로 올려 순이익이 뒤집히는 지점을 찾습니다.
1. "이 숫자를 못 믿을 조건" 3개를 먼저 적고, 각각을 판정하는 방법을 옆에 쓰세요. `` 못 믿을 조건 판정 방법
2. 시드가 안 고정됐다 두 번 돌려 같은 값인지 본다
3. 오탈락률이 0 이다 입력값을 직접 확인한다
4. 표본이 과거 1분기뿐 기간을 늘려 재실행한다 ``
1. 본인 시스템의 판정 로직에서 2값으로 접혀 있는 곳을 하나 찾아 3값으로 고치세요. "조회 실패"가 "실패"로 뭉개져 있는 자리가 대체로 있습니다.

실습 과제 (Action Item) (계속)

제출물: 시뮬레이션 결과 + 못 믿을 조건 3개와 판정 방법 + 3값 판정 diff 자가 점검 기준:

- ✓ 오탈락 비용을 계산에 넣었는가
- ✓ 임계값을 **결과를 보기 전에** 정했는가
- ✓ 판정 불가를 미충족과 구분해 표현하는가
- ✓ 판정 근거가 상태 플래그가 아니라 실제 관측치인가

Q&A / 자주 막히는 지점

증상	원인	해결
돌릴 때마다 숫자가 달라진다	<code>Math.random()</code> 또는 시드 미고정	시드 PRNG 로 교체. 두 번 돌려 같은지 테스트로 고정
절감 공수가 비현실적으로 크다	오타락률을 0 으로 뒀다	실제 값을 넣고 순이익을 다시 본다
통과한 가설이 실패했다	시뮬레이션은 채택을 보증하지 않는다	통과 = "기각할 근거가 없다"로 읽는다
연동이 됐는데 데이터가 안 보인다	<code>offline: false</code> 를 근거로 삼았다	실제 조회 결과 건수로 판정한다
자격증명을 넣으니 판정이 확 바뀐다	조회 실패를 미완성으로 뭉갬다	<code>UNKNOWN</code> 을 별도 값으로 둔다
시뮬레이터가 500 을 뱉는다	실패 시 예외를 던진다	목으로 풀백하되 이유를 화면에 적는다
목 데이터가 실제 계산과 다르다	목을 손으로 적었다	결정론 파이프라인을 실제로 돌려 생성한다
관리자 화면에서 아무 스크립트나 지정된다	실행 경로 검증이 없다	화이트리스트 + 프로젝트 내부 경로·확장자 가드