생성형 AI 소프트웨어공학 · 2026학년도 2학기 · 4강

# AI 위임의 기술, 딸깍의 경계

무엇이 지시 한 번으로 되는가: 위임의 스펙트럼 · 네 가지 질문 · 경계 탐사 실험, 그리고 과제 1

대전대학교 컴퓨터공학과 · 박상돈

4강 · 수업 안내

# 지난 시간, 그리고 오늘

## 지난 시간 요약

생성형 AI는 다음에 올 글 조각을 확률로 고릅니다. 같은 지시에도 결과가 달라질 수 있어, 결과는 언제나 확인 대상입니다.

내가 비운 칸은 AI가 그럴듯하게 채웁니다. 역할 · 맥락 · 요구 · 형식 네 기둥으로 중요한 빈칸을 내가 채웁니다.

그리고 실험 A · B · C로 지시문이 결과를 바꾸는 것을 직접 확인했습니다. 그 기록이 과제 1입니다.

## 이번 주 성취 기준

- 어떤 작업이 지시 한 번으로 될지, 네 가지 질문으로 예측하고 근거를 말할 수 있다.
- 한 번에 안 되는 큰 작업을 확인 가능한 조각으로 쪼갤 수 있다.
- AI 결과물을 받아 그대로 쓸지, 고쳐 쓸지, 버릴지 판정할 수 있다.

4강 · 수업 안내

# 과제 1 확인: 지시문이 결과를 바꿨습니다

1 **실험 A(막연한 지시): 결과가 사람마다 제각각**
형태도, 파일 이름도, 90점의 학점도 다릅니다. 내가 정하지 않은 것을 AI가 확률로 채웠기 때문입니다.

2 **실험 B·C(기준·형식을 채운 지시): 결과가 수렴**
지시문이 좋아질수록 결과가 예측 가능해집니다. 3강의 결론이었습니다.

3 **그런데, 지시문을 다듬어도 한 번에 안 되는 일이 있었습니다**
수정 지시를 몇 번 하고서야 쓸 만해졌거나, 방향이 어긋난 경험. 오늘은 바로 그 얘기입니다.

? **오늘의 질문**
지시문이 완벽하면 모든 일이 한 번에 될까요? 안 됩니다. 어떤 일은 되고 어떤 일은 안 됩니다. 그 경계가 어디인지, 무엇이 경계를 가르는지, 오늘 실험으로 직접 잽니다.

01

PART 1

# 딸깍이란 무엇인가

요즘 말로 딸깍, 정식으로는 위임. 정의와 스펙트럼, 그리고 왜 이것이 이 과목의 관건인지.

# 딸깍 = 위임: 지시 한 번으로 끝나는 일

**딸깍** (정식 용어로는 '위임 · delegation') **지시 한 번과 승인 몇 번으로, 그대로 쓸 만한 결과를 받는 것**

### 지시는 한 번

같은 일을 다시 시키거나 고쳐 달라고 하면 그때부터는 반딸깍입니다.

### 확인은 항상

승인 화면 읽기, 실행, 코드 읽기까지가 딸깍의 일부입니다. 확인 없는 딸깍은 도박입니다.

### 결과는 그대로 사용

받아서 고쳐야 쓸 수 있다면 완전한 딸깍은 아닙니다.

**딸깍은 놀림말이 아니라 이 수업의 목표입니다**

잘 위임하면 같은 시간에 훨씬 많은 일을 합니다. 다만 무엇을 딸깍해도 되는지 아는 사람과 모르는 사람의 차이가 곧 실력의 차이입니다.

# 위임의 스펙트럼: 지난 3주를 올려 보면

| 완전 딸깍 | 반딸깍 | 쪼개면 딸깍 | 사람 몫 |
|---|---|---|---|
| **예: temp.py 변환기 (2강)** | **예: 학점 변환기 (3강)** | **예: 할 일 관리 (오늘)** | **예: 무엇을 만들지 정하기** |
| 지시 1번 · 수정 0번<br>그대로 사용 | 지시 1번 + 수정 1~2번<br>차이를 말로 만들어 재지시 | 사람이 조각으로 나누면<br>조각마다 딸깍 | 위임 불가<br>판단과 책임의 영역 |

지시 한 번이면 끝 → 사람이 생각해야 하는 일

위임 가능성은 켜짐·꺼짐이 아니라 스펙트럼입니다. 오늘 실습에서 이 네 칸을 전부 직접 밟아 봅니다.

# 이 과목의 관건: 경계를 판단하는 눈

**예전 실력:** 코드를 빠르고 정확하게 친다
**지금 실력:** **무엇을 시키고, 무엇을 쪼개고, 무엇을 직접 할지 판단한다**

### 안 될 일을 통째로 시키면

시간을 잃습니다

"쇼핑몰 만들어 줘" 그럴듯한 파일 더미를 받고, 확인하다 밤을 새우고, 결국 버립니다.

### 될 일을 직접 하면

기회를 잃습니다

구구단을 30분 동안 손으로 짜는 사이, 옆 사람은 딸깍 한 번으로 끝내고 다음 일을 합니다.

### 확인 없이 받으면

신뢰를 잃습니다

그럴듯한 오류가 통과됩니다. 1강에서 본 사고들이 이 길 끝에 있습니다.

그래서 오늘부터 매주 실습 끝에 '오늘 한 일이 어디까지 딸깍이었나'를 기록합니다. 이 기록이 쌓이면 경계 감각이 생깁니다.

02

PART 2

# 딸깍을 가르는 네 가지 질문

확인할 수 있는가 · 기준이 명확한가 · 흔한 일인가 · 대가가 작은가. 판정과 처방까지.

2부 · 네 가지 질문

# 질문 ①②: 확인할 수 있는가 · 기준이 명확한가

## 1 결과를 확인할 수 있는가

맞는지 내가 판정할 수 없으면, 아무리 잘 만들어도 쓸 수 없습니다. 2강의 '확인 가능한 크기' 원칙이 여기서 나왔습니다.

- ✓ 구구단 출력: 눈으로 보면 확인 끝
- ✗ 복잡한 이자 계산·암호화: 판정 자체가 어려움

확인 방법이 안 떠오르면 아직 딸깍 대상이 아닙니다

## 2 '됐다'의 기준이 명확한가

내가 비운 칸은 AI가 확률로 채웁니다(3강). 기준이 또렷할수록 결과가 내 의도에 수렴합니다.

- ✓ "1~45 중복 없이 6개, 오름차순"
- ✗ "예쁘게, 알아서 잘"

말로 어려운 형식·스타일은 좋은 예시 한두 개가 문장 열 개를 이깁니다

# 질문 ③④: 흔한 일인가 · 대가가 작은가

## 3 세상에 흔한 일인가

AI는 훈련 데이터에서 배웁니다. 수백만 번 만들어진 일은 잘하고, 처음 보는 일은 그럴듯하게 지어냅니다(환각).

- ✓ 로그인 화면, CSV 읽기, 정렬
- ✗ 우리 학교만의 규정, 어제 나온 라이브러리

흔치 않은 일에는 문서·예시 코드를 지시문에 붙여 줍니다

## 4 틀렸을 때 대가가 작은가

연습 코드는 틀려도 다시 시키면 그만입니다. 대가가 큰 일은 만들게 하더라도 사람 확인을 사이에 끼웁니다.

- ✓ 연습·과제 코드, 일회성 스크립트
- ✗ 결제, 개인정보, 실사용자 배포 (13주차)

대가가 클수록 '사람 확인 단계'가 설계의 일부가 됩니다

네 질문을 순서대로 물으면 30초 안에 판정이 끝납니다. 다음 장에서 판정 후의 행동을 정합니다.

✔ 2부 · 네 가지 질문

# 판정과 처방: '아니오'는 고치면 됩니다

① 확인할 수 있는가 | ② 기준이 명확한가 | ③ 흔한 일인가 | ④ 대가가 작은가

## 네 개 모두 '예'

**딸깍하세요. 시키는 게 맞습니다.**

단, 확인까지가 딸깍입니다. 승인을 읽고, 실행하고, 코드를 읽어 설명할 수 있어야 위임 성공입니다.

## '아니오'가 있으면 → 처방

| | |
|---|---|
| 기준이 흐리면 | 기준을 적고, 예시를 보여 준다 |
| 확인이 어려우면 | 확인 가능한 크기로 쪼갠다 |
| 흔치 않은 일이면 | 문서·예시 코드를 붙여 준다 |
| 대가가 크면 | 사람 확인 단계를 끼워 넣는다 |

**경계는 고정이 아니라 움직입니다**

작년의 '아니오'가 올해의 '예'가 됩니다. 그래서 목록 암기는 무의미하고, 새 모델이 나올 때마다 예측하고 실험해서 경계를 다시 재는 습관이 진짜 실력입니다.

2부 · 네 가지 질문

# 딸깍의 함정 두 가지

## 그럴듯함의 함정

AI의 결과물은 항상 매끄럽습니다. 그 매끄러움이 확인을 생략하게 만듭니다. 말투는 근거가 아닙니다(3강). 결과가 그럴듯할수록 실행하고 읽어서 확인합니다. 딸깍과 확인은 한 세트입니다.

## 이해 없는 축적의 함정

확인 없는 딸깍을 이어 가면 설명 못 하는 코드가 쌓입니다. 어느 순간 아무도 프로그램을 이해하지 못하고, 그때부터는 고칠 수도 없습니다. 1강 규칙 ②가 딸깍 시대에 더 중요해지는 이유입니다(10주차).

딸깍의 속도는 확인과 이해가 받쳐 줄 때만 진짜 속도입니다. 빨리 받는 것보다, 받은 것을 내 것으로 만드는 쪽이 항상 먼저입니다.

2부 · 네 가지 질문

# 딸깍이 안 되는 일들이, 남은 학기의 목차입니다

**무엇을 만들지 정의한다 (5주차 · 요구공학)**
원하는 것은 내 머릿속에만 있습니다. AI는 읽지 못합니다.

**구조를 판단한다 (6주차 · 설계)**
구조마다 장단점을 재고 상황에 맞게 고르는 것은 판단의 문제입니다.

**'됐다'의 기준을 세운다 (8·9주차 · 인수 조건)**
통과와 불통과의 선을 긋는 것은 요구한 사람의 몫입니다.

**정말 되는지 검증한다 (9·10주차 · 테스트)**
그럴듯함 너머의 실제 결함을 찾아내는 눈입니다.

**품질·보안·윤리에 책임진다 (13주차)**
취약점, 개인정보, 라이선스. 최종 책임은 이름을 올린 사람이 집니다.

오늘의 네 질문은 이 다섯 주제의 압축판입니다. 남은 학기는 딸깍의 경계를 넓히는 기술과 경계 밖을 지키는 기술을 배우는 시간입니다.

03

PART 3

# 경계 탐사 실험

예측을 먼저 적고, 실험 네 개로 경계를 직접 재 봅니다. 오늘 기록이 곧 출석 인증입니다.

# 실험 방법: 예측 먼저, 실행은 그다음

notes4.md (예측표 · hello-sw 폴더에 만들기)

```
## 실험 1: 로또 번호 생성기
예측: O (완전 딸깍)
근거: 네 질문 모두 예
실제: (실행 후 기록)
배운 것: (실행 후 기록)

## 실험 2: 단위 변환기
예측: △ (반딸깍)
근거: 입력 방식 기준이 비어 있음
...
```

1 **실행 전에 예측부터 적는다**
notes4.md에 실험마다 예측(O·△·X)과 근거를 먼저. 적지 않은 예측은 예측이 아닙니다.

2 **실험마다 ZCode 'New Task'**
앞 실험의 대화가 남으면 다음 실험이 오염됩니다(3강 규칙).

3 **도구는 수업 공용 ZCode · GLM-5.3**
왼쪽 아래 'Connected' 확인. 'Connect'로 바뀌었으면 질문하세요.

4 **막히면**
1302·Rate limit은 잠깐 대기 후 재전송, 정 안 되면 Codex로 진행해도 됩니다.

예측이 틀려도 됩니다. 오히려 틀린 예측이 오늘 제일 값진 수확입니다. 경계 감각이 고쳐지는 순간이니까요.

# 예측 연습: 실험 ①을 네 질문에 넣어 보면

**실험 ① 지시문 (그대로 사용)**

"lotto.py 파일을 만들어 줘. 1부터 45 사이에서 중복 없이 숫자 6개를 뽑아 오름차순으로 출력해. 실행할 때마다 다른 번호가 나와야 해."

| | | |
|---|---|---|
| ① 확인할 수 있는가 | 실행해서 6개·범위·중복·정렬을 눈으로 보면 됨 | 예 |
| ② 기준이 명확한가 | 방금 그 네 가지가 지시문에 전부 적혀 있음 | 예 |
| ③ 흔한 일인가 | 무작위 뽑기는 프로그래밍 연습 문제의 단골 | 예 |
| ④ 대가가 작은가 | 연습 코드. 틀리면 다시 시키면 됨 | 예 |

네 개 모두 '예' → 예측은 O, 완전 딸깍. notes4.md에 이 예측과 근거를 적고, 다음 장에서 실행으로 확인합니다. 이 순서를 오늘 네 번 반복합니다.

# 실험 ① 로또 생성기: 완전 딸깍 확인

VS Code 터미널 (또는 ZCode 아래 터미널)

```
python lotto.py
[4, 12, 23, 31, 38, 44]
python lotto.py
[2, 9, 17, 26, 33, 41]
python lotto.py
[7, 11, 20, 29, 35, 45]
# 매번 다른 6개 · 1~45 · 중복 없음 · 오름차순
```

1. **예측을 적었는지 확인**
   notes4.md에 예측 O와 근거가 있어야 시작.
2. **New Task → 지시문 붙여넣기 → Allow**
   lotto.py를 만들겠다는 내용을 훑어보고 허용.
3. **세 번 실행해서 규칙 확인**
   6개 · 1~45 · 중복 없음 · 오름차순, 매번 다른 번호.
4. **코드를 열어 설명 점검**
   열 줄 안팎. 한 줄 한 줄 설명할 수 있으면 통과.
5. **notes4.md에 '실제' 기록**
   수정 지시 0번이면 완전 딸깍 성공. 예측 적중 여부도.

통과 기준: 세 번 실행 모두 규칙 충족 + 수정 지시 0번 + 코드 설명 가능. 끝난 사람은 기다리지 말고 바로 실험 ②로.

확인 조건(6개·범위·정렬)을 지시문에 미리 적어 둔 것에 주목하세요. 검증을 지시문에 담으면 처음부터 그렇게 나옵니다.

# 실험 ② 단위 변환기: 빈칸을 남긴 지시문

**실험 ② 지시문 (마지막 문장이 일부러 비어 있습니다)**

"converter.py 파일을 만들어 줘. 길이는 cm·m·km·inch, 무게는 g·kg·lb, 온도는 섭씨·화씨를 서로 바꾸는 변환기야. 입력을 어떻게 받을지는 네가 정해."

**예측: AI가 대신 정하게 될 것들**

메뉴 방식일까, 한 줄 입력일까
반올림은 몇 자리까지 할까
없는 단위를 넣으면 어떻게 될까
**→ 질문 ②가 '아니오'이므로 예측은 △**

1. **실행 → AI가 정한 결정 3개 이상 찾기**
   내가 비운 칸을 AI가 어떻게 채웠는지 notes4.md에 적습니다.
2. **수정 지시 한 번 보내기**
   "결과는 소수점 둘째 자리까지, 없는 단위를 넣으면 안내문을 출력해 줘"
3. **다시 실행해 확인**
   고쳐졌으면 반딸깍 성공. 수정 지시 횟수를 기록.

**반딸깍을 다루는 기술**
기대와 결과의 차이를 문장으로 만드는 것. 그리고 그 확인 조건을 다음부터는 지시문에 미리 담는 것. 이 둘이 되면 △짜리 일을 사실상 딸깍처럼 씁니다.

# 실험 ③ 할 일 관리, 한 번에: 동작 ≠ 딸깍 성공

**실험 ③ 지시문 (한 번에 통째로)**

"todo.py 파일을 만들어 줘. 할 일 추가, 목록 보기, 완료 표시, 삭제가 되고, 프로그램을 껐다 켜도 할 일이 남아 있어야 해."

**예측 먼저**

기능 4개 + 파일 저장 = 확인 항목이 많다 (① 흔들림)
저장 방식·명령 체계가 비어 있다 (② 아니오)
**→ 예측: △ 또는 X. 어디서 막힐지도 적어 보기**

1. **실행: 아마 동작합니다**
   GLM-5.3은 이 정도를 곧잘 만듭니다. 그런데 성공일까요?
2. **AI가 정한 결정 세어 보기**
   저장 형식, 명령 체계, 번호 매기기, 오류 처리. 대여섯 개가 나옵니다.
3. **설명 점검 + 시간 비교**
   코드 백 줄 안팎. 다 설명할 수 있나요? 확인 시간이 생성 시간보다 깁니다.

**'동작한다'와 '딸깍 성공'은 다릅니다**

성공 기준은 그대로 쓸 수 있고, 설명할 수 있는 것까지. 동작하지만 이해 못 하는 백 줄은 다음 주의 빚입니다. todo.py는 이대로 두고, 다음 장에서 쪼개서 다시 갑니다.

# 실험 ③ 쪼개서 다시: 딸깍 세 번으로

1. "todo2.py를 만들어 줘. 실행 중에 할 일 추가와 목록 보기만 되면 돼. 저장은 아직 없이."
   → 실행해서 추가·목록 확인
2. "todo2.py에 파일 저장을 추가해 줘. 껐다 켜도 할 일이 유지되게."
   → 껐다 켜서 유지 확인
3. "todo2.py에 완료 표시와 삭제 기능을 추가해 줘."
   → 완료·삭제 확인

실험마다 New Task로 새 대화. 파일 이름을 todo2로 해서 아까 것과 비교합니다.

## 한 방 vs 쪼개기

| | 한 방에 | 쪼개서 |
|---|---|---|
| 단계마다 확인 | 어려움 | 가능 |
| 전부 설명 가능 | 부담 큼 | 가능 |
| 중간에 방향 수정 | 불가 | 가능 |
| 총 걸린 시간 | 비슷 | 비슷 |

**통제력이 완전히 다릅니다.**

쪼개면 경계 밖 작업이 딸깍 세 번으로 바뀝니다. 무엇을 기준으로 어떻게 쪼개는지는 7주차에서 본격적으로 다룹니다. 오늘은 '쪼개면 달라진다'까지 몸으로 확인하면 됩니다.

# 실험 ④ 쇼핑몰: 예측만 하고, 실행하지 않습니다

**실험 ④ 지시문 (오늘은 보내지 않습니다)**

"회원가입과 로그인, 장바구니가 되는 쇼핑몰 웹사이트를 만들어 줘."

| | | |
|---|---|---|
| ① 확인할 수 있는가 | 파일 수십 개 · 화면 수십 개. 다 볼 방법이 없음 | 아니오 |
| ② 기준이 명확한가 | 결제는? 재고는? 회원 정보는 어디까지? 전부 빈칸 | 아니오 |
| ③ 흔한 일인가 | 쇼핑몰은 흔하지만 이 크기를 한 번에 만드는 건 다른 문제 | 절반 |
| ④ 대가가 작은가 | 회원 정보 = 개인정보. 틀리면 대가가 큼 | 아니오 |

실행하지 않는 이유가 곧 교훈: 확인 못 할 크기는 시키지 않는다(2강 원칙) + 받아도 설명 불가 + 공용 계정 한도를 혼자 잡아먹는 일.

그럼 쇼핑몰은 영영 못 맡기나? 조각으로는 갑니다. 요구를 정하고(5주) 구조를 나누고(6주) 조각마다 맡기고(7주) 검증하는(9주) 여정이 남은 학기입니다.

# 오늘 발견한 패턴 세 가지

### 완전 딸깍은 생각보다 좁다

네 질문이 모두 '예'일 때만. 로또는 됐지만, 빈칸 하나 있던 변환기부터 벌써 △였습니다.

### 반딸깍의 기술이 절반이다

차이를 문장으로 만들어 재지시하고, 확인 조건을 지시문에 미리 담기. 이게 되면 △를 딸깍처럼 씁니다.

### 쪼개기가 경계를 넓힌다

경계 밖이던 할 일 관리가 세 조각으로는 조각마다 딸깍이 됐습니다.

**예측표 채점: 틀린 예측에 동그라미**

맞은 예측보다 틀린 예측이 값집니다. 그 동그라미가 오늘 갱신된 여러분의 경계 감각이고, 다음 실습부터 그만큼 더 정확해집니다.

# 기록 올리기: 예측표와 실험 파일 push

VS Code 터미널 (hello-sw 안)

```
git add .           # 새 파일 전부
git commit -m "add clickability experiments"
git push
```

1. **notes4.md 완성**
   실험 4개 각각 예측·근거·실제·배운 것. 실험 ④의 실제 칸은 '실행 안 함'.

2. **파일 다섯 개 확인**
   lotto.py · converter.py · todo.py · todo2.py · notes4.md

3. **add · commit · push**
   지난주와 같은 세 단계. 브라우저에서 저장소 새로고침으로 확인.

4. **LMS 4주차 게시판에 저장소 주소 제출**
   여기까지가 오늘 출석 인증입니다.

push 전 습관: Ctrl+Shift+F 전체 검색으로 key, password 검색. 오늘은 키를 만진 적이 없으니 아무것도 안 나와야 정상입니다.

다섯 파일이 저장소에 보이면 완료. 예측표는 다음 주 요구공학 실습에서 다시 꺼내 씁니다.

✔ 4강 · 정리

# 오늘의 세 문장

1. 딸깍(위임)의 경계는 네 질문이 가릅니다. 확인할 수 있는가 · 기준이 명확한가 · 흔한 일인가 · 대가가 작은가.

2. 경계 밖 작업은 쪼개서 경계 안으로 가져옵니다. 한 번에 안 되던 할 일 관리가 세 조각으로는 됐습니다.

3. 경계는 모델과 함께 움직입니다. 목록 암기 대신, 예측하고 실험해서 다시 재는 습관이 실력입니다.

예고 · 5강

# 다음 주: 요구공학, 무엇을 만들지 정하기

### 말로 정하는 기술
머릿속의 '원하는 것'을, 남과 AI가 오해 없이 만들 수 있는 문장으로 바꿉니다. 오늘의 질문 ②를 체계로 만드는 주입니다.

### 팀 프로젝트 시작
팀 구성과 주제 선정을 합니다. 만들고 싶은 작은 서비스 아이디어를 하나 구상해 오세요.

### 요구가 곧 딸깍의 입력
요구가 흐리면 그 뒤의 모든 딸깍이 흔들립니다. 생명주기 여정의 첫 정거장입니다.

오늘 수업은 여기까지입니다. 예측표 제출 확인하고 가세요. 수고하셨습니다!

### 잊지 마세요

notes4.md와 실험 파일 push, 저장소 주소 LMS 4주차 제출.

준비물: 노트북과 오늘 환경 그대로, 그리고 팀 프로젝트 아이디어 한 개.