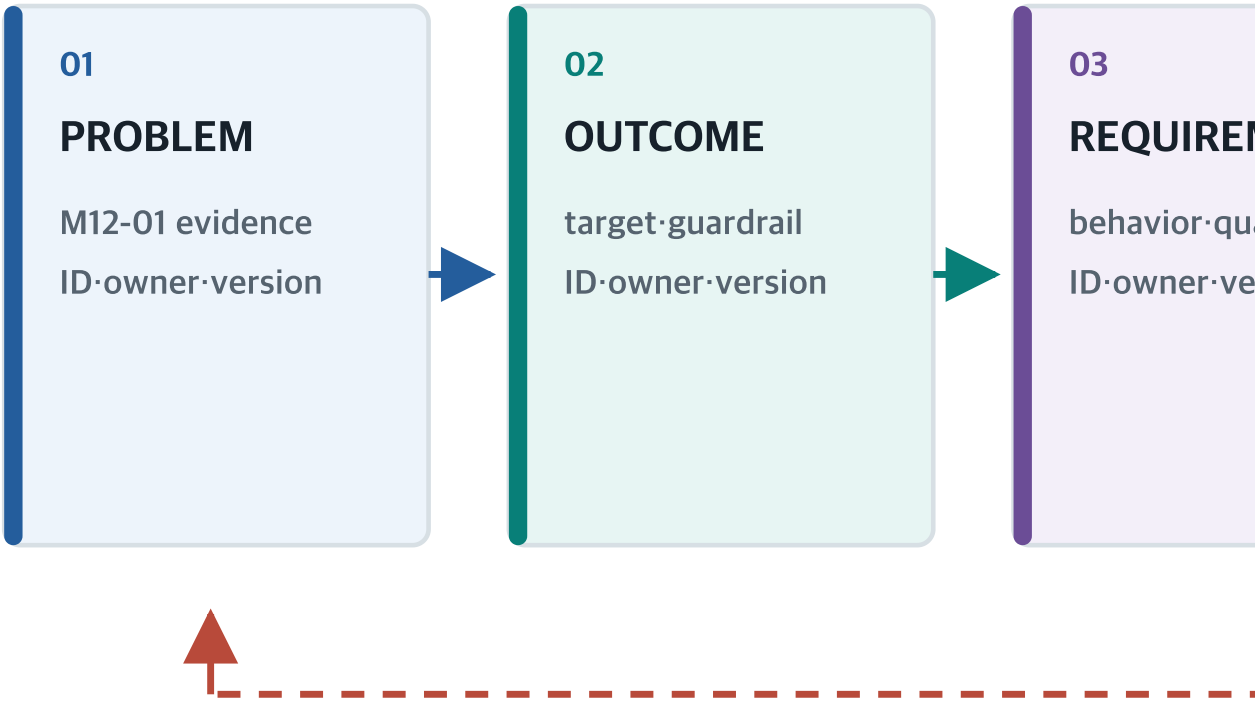
M12-02 · GIBALJA VISUAL MANUAL

기능·비기능 요구사항과 완료 기준 쓰기

한 문장 목표: `problem·outcome·guardrail·source → level·scope → functional behavior + measurable quality + constraint → observable acceptance + verification evidence + DoD → Requirement Readiness Gate → M12-03 handoff` 를 끊김 없이 연결합니다.

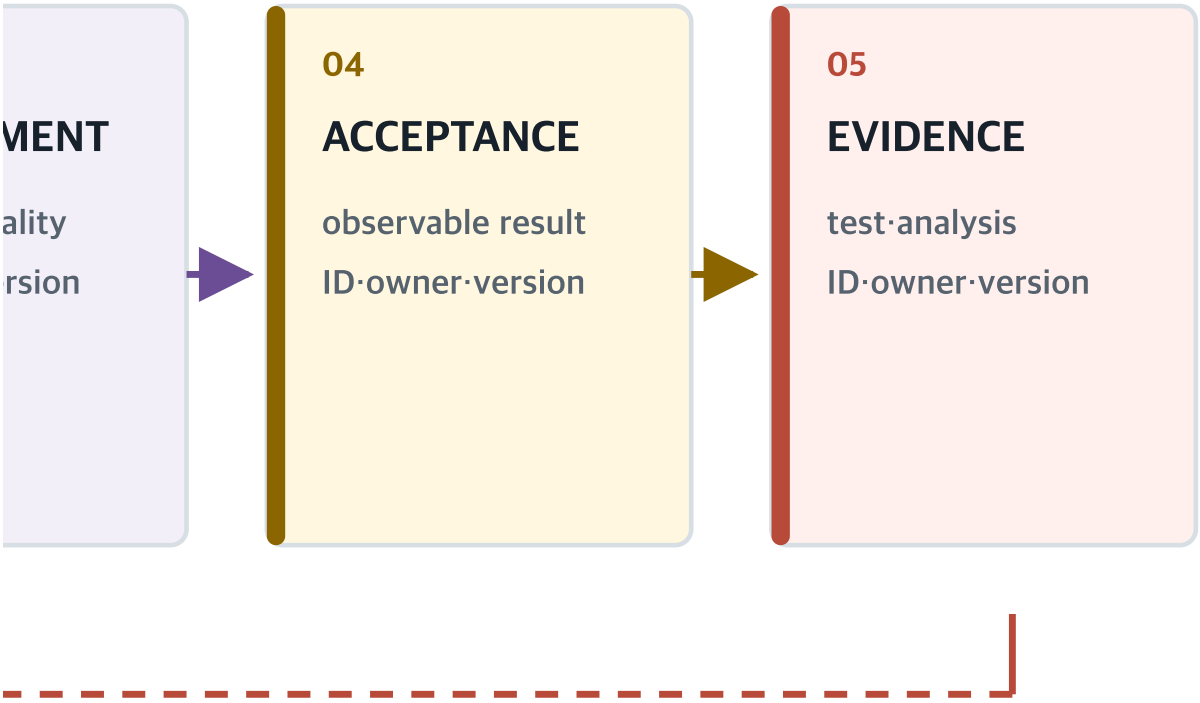
그림 1. 요구사항은 기능명 모음이 아니라 왜 필요한지부터 무엇으로 완료를 판정할지까지 이어지는 양방향 추적 사슬입니다.

기능명 하나가 아니라 존재 이유·관찰 기준·검증 방법·판정 근거가 양방향으



[illegible]

[illegible]로 연결됩니다.


MENT
ality
rsion
04
ACCEPTANCE
observable result
ID·owner·version
05
EVIDENCE
test·analysis
ID·owner·version

0. 한눈에 보는 요구사항 한 장

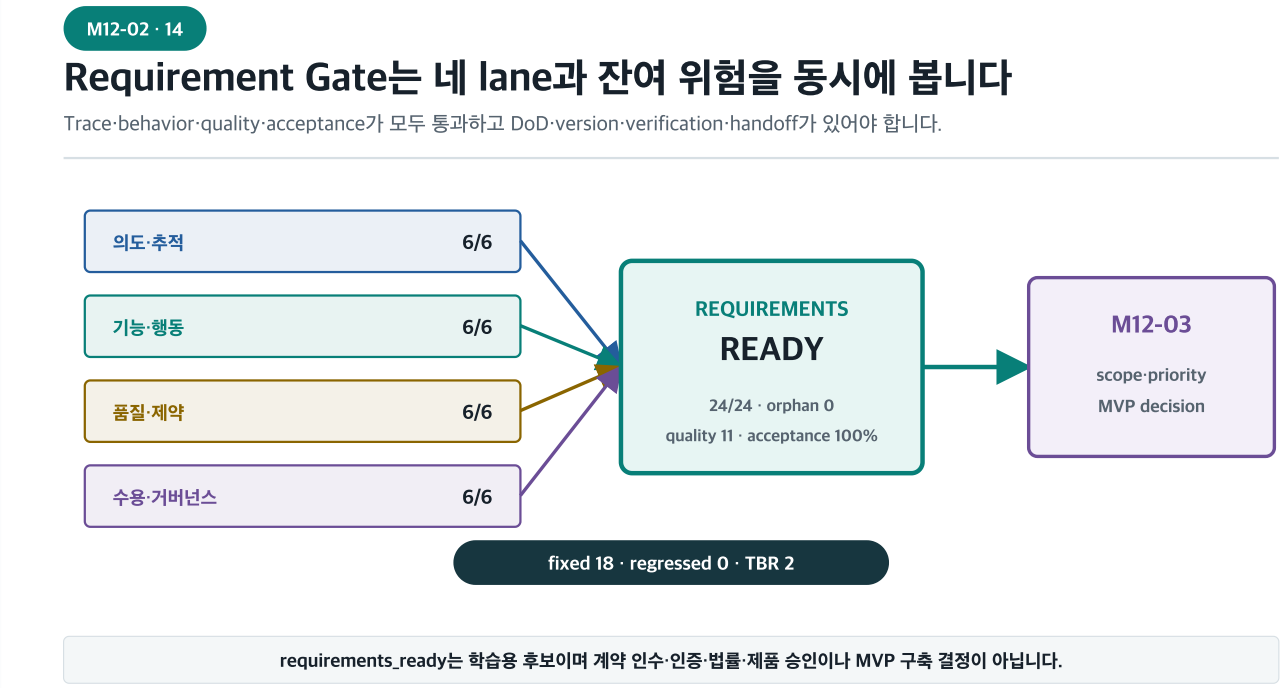
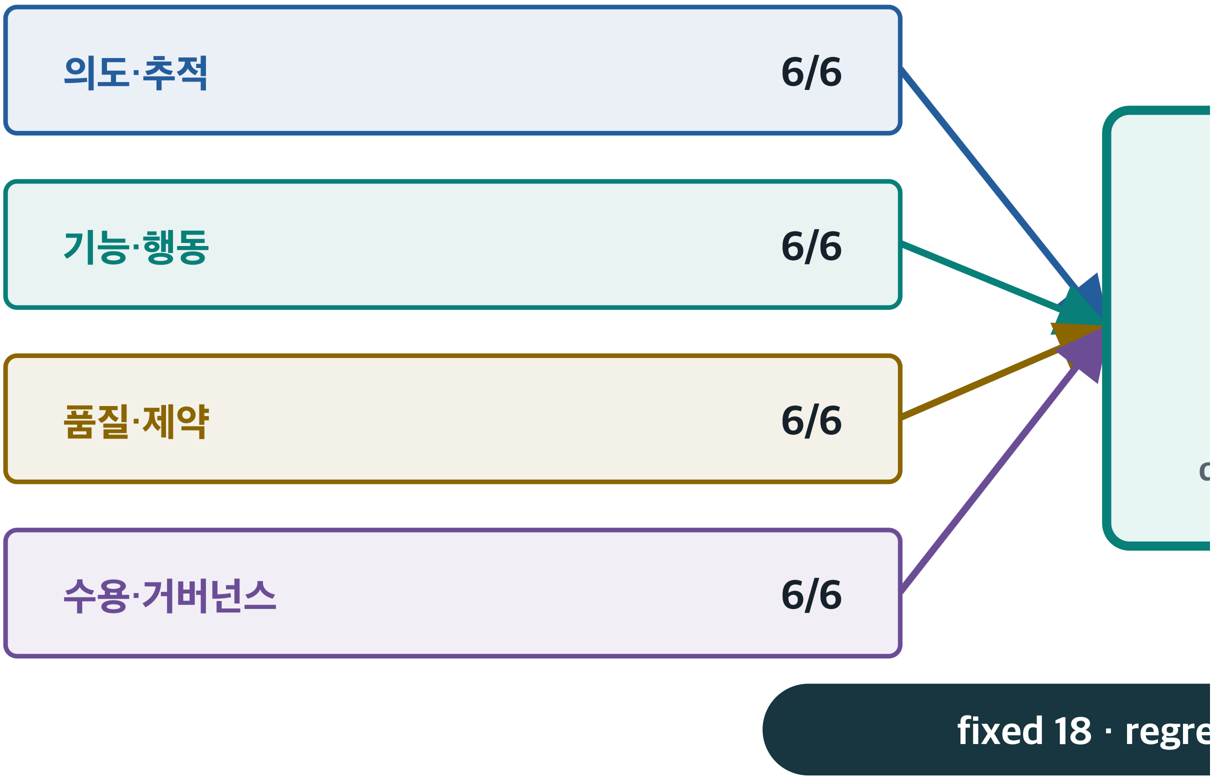
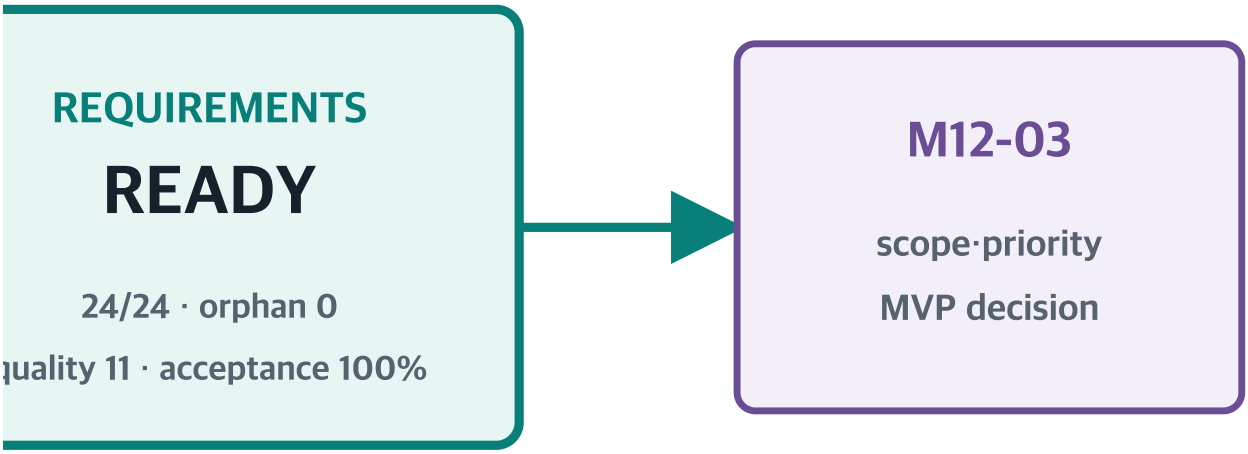


그림 14. 네 lane이 모두 6/6을 통과하고 orphan·vague·conflict가 0이며 acceptance 100%, 품질 요구 11개, TBR 2개 이하일 때만 다음 단계 후보가 됩니다.

Trace·behavior·quality·acceptance가 모두 통과하고 DoD·version·verifi



cation·handoff가 있어야 합니다.



ssed 0 · TBR 2

학습 순서	시간	남기는 증거
그림 14장	45분	trace·behavior·quality·acceptance 전체 지도
개념·합성 사례	105분	19 requirement·4 constraint
실습 스튜디오	90분	24 scenario·57 audit·6 regression
템플릿·셀프 테스트	60분	내 요구사항 후보·30문항 답

본문의 사용자·조직·요구사항·수치·비용·시스템은 모두 허구의 합성 학습 자료입니다. `requirements_ready`는 실제 계약 인수, 보안·접근성 인증, 법률·정책·제품 승인이나 MVP 구축 결정을 대신하지 않습니다.

0.1 먼저 기억할 열두 문장

```
요구사항은 문제와 결과에서 시작한다.
기능명은 요구사항이 아니다.
한 requirement에는 한 obligation만 둔다.
수준과 시스템 경계를 먼저 맞춘다.
기능은 actor·조건·행동·결과로 쓴다.
정상 흐름만 있으면 절반이 비어 있다.
데이터와 상태 전이는 함께 본다.
품질 형용사는 measure·threshold로 바꾼다.
접근성·보안·privacy·safety는 처음부터 요구한다.
Acceptance는 positive·negative·boundary·authorization을 덮는다.
완료에는 verification evidence와 DoD가 필요하다.
requirements_ready는 MVP 승인이나 인증이 아니다.
```

1. 이 PDF를 공부하는 방법

1.1 1회차 · 그림과 caption만 읽기 · 45분

각 그림에서 `입력 근거 → 요구 계약 → 판정 증거` 세 지점을 손으로 짚습니다. 그림마다 아래 한 문장을 채웁니다.

> 이 계약이 없으면 _____ 요구를 완료로 오해하고, _____ 사용자·품질·위험을 놓친다.

1.2 2회차 · 합성 사례 실습 · 90분

요구사항 실습 생성기를 실행합니다.

```
./02_Labs/G12_Product_Definition/L12-02_create-requirements-practice.sh
```

Version	Pass	Trace	Quality	Acceptance	Decision
feature-list-v1	6/24	15%	0	25%	blocked_feature_list
functional-only-v2	15/24	82%	3	67%	blocked_quality_acceptance
verified-requirements-v3	24/24	100%	11	100%	requirements_ready

1.3 3회차 · 내 requirement set 만들기 · 120분

기능·비기능 요구사항과 완료 기준 템플릿을 다음 순서로 채웁니다.

1. M12-01 handoff와 system boundary
2. Level·scope·actor·용어
3. Atomic functional behavior
4. Alternative·exception·recovery
5. Data·state·history 경계
6. Measurable quality·assurance
7. Constraint·dependency·TBR
8. Acceptance·verification·DoD·Gate

1.4 막힐 때

기능·비기능 요구사항 용어집 300에서 지금 장과 같은 번호의 20개만 읽습니다. 용어를 외운 뒤 반드시 합성 사례의 requirement ID 하나를 붙입니다.

---

2. 요구사항을 다시 정의하기

2.1 요구사항은 무엇인가

REQUIREMENT CONTRACT 승인된 문제·결과·위험·정책을 위해 특정 수준과 범위에서 시스템이나 서비스가 충족해야 할 하나의 행동 또는 측정 가능한 품질·제약을, source·rationale·owner·acceptance·verification·version과 함께 표현한 계약입니다.

2.2 비슷해 보이지만 다른 문장

문장	실제 정체	빠진 것
AI 회의 정리	기능 이름	actor·condition·behavior·output
빠르고 안전해야 한다	품질 희망	context·measure·threshold·assurance
담당자를 확인할 수 있다	기능 방향	권한·trigger·상태·예외
Given invalid, Then error	수용 초안	정확한 입력·오류·보존 상태
테스트가 통과했다	부분 증거	requirement ID·환경·oracle·limitation
Done	상태 주장	DoD checklist·evidence·owner

2.3 이전 매뉴얼과의 역할 차이

매뉴얼	핵심 질문	M12-02와의 경계
M07-02	이미 선택한 요구를 어떤 작은 작업으로 나눌까	작업 slicing과 task DoD
M10-01	주어진 요구에서 어떤 test case를 만들까	requirement를 전제로 test derivation
M12-01	누구의 어떤 문제와 결과를 다룰까	문제·결과·guardrail source 제공
M12-02	무엇을 어떤 품질과 증거로 완료라 할까	requirement·acceptance·verification 계약
M12-03	어떤 범위를 먼저 MVP로 선택할까	readiness 후보의 scope·priority 결정

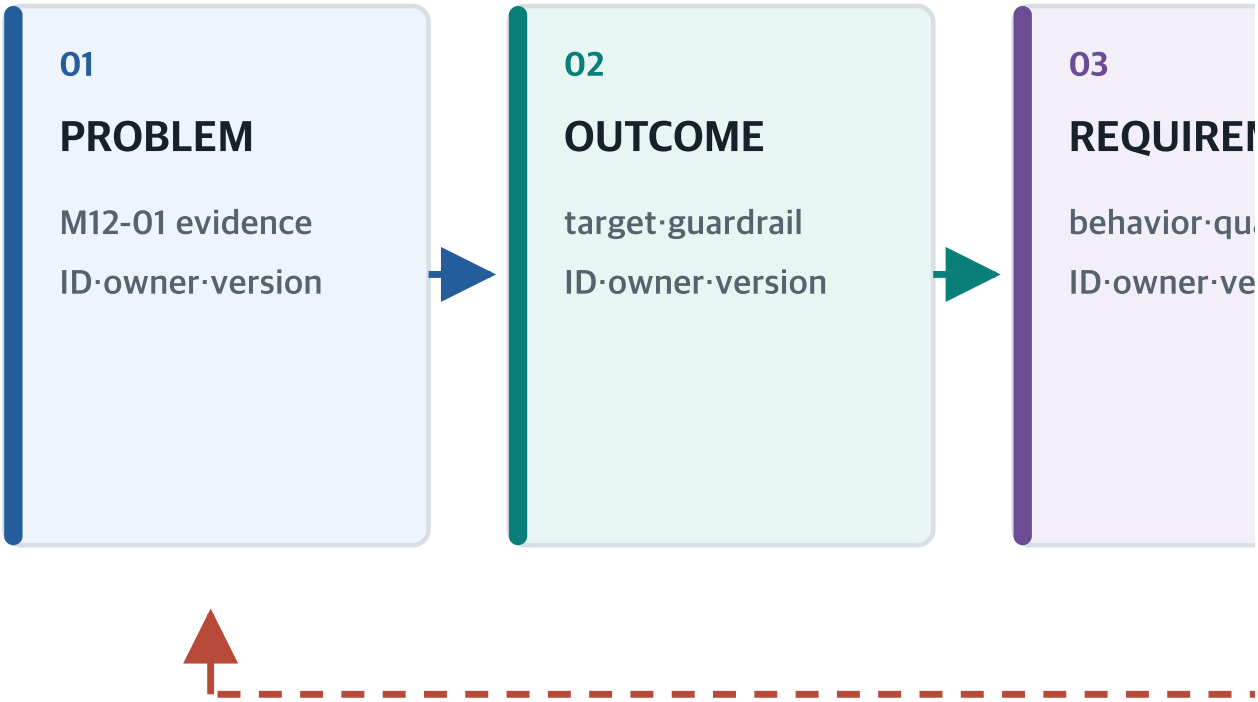
2.4 합성 사례의 최소 계약

```
M12-01 source + problem + desired outcome + baseline + target + guardrail
→ level + scope + actor + unique atomic requirement
→ normal + alternative + exception + recovery + data/state
→ quality context + measure + threshold + window + segment
→ constraint + dependency + TBR
→ acceptance(positive·negative·boundary·authorization)
→ verification(method·oracle·environment·artifact·limitation)
→ version + baseline + DoD + Requirement Readiness Gate
```

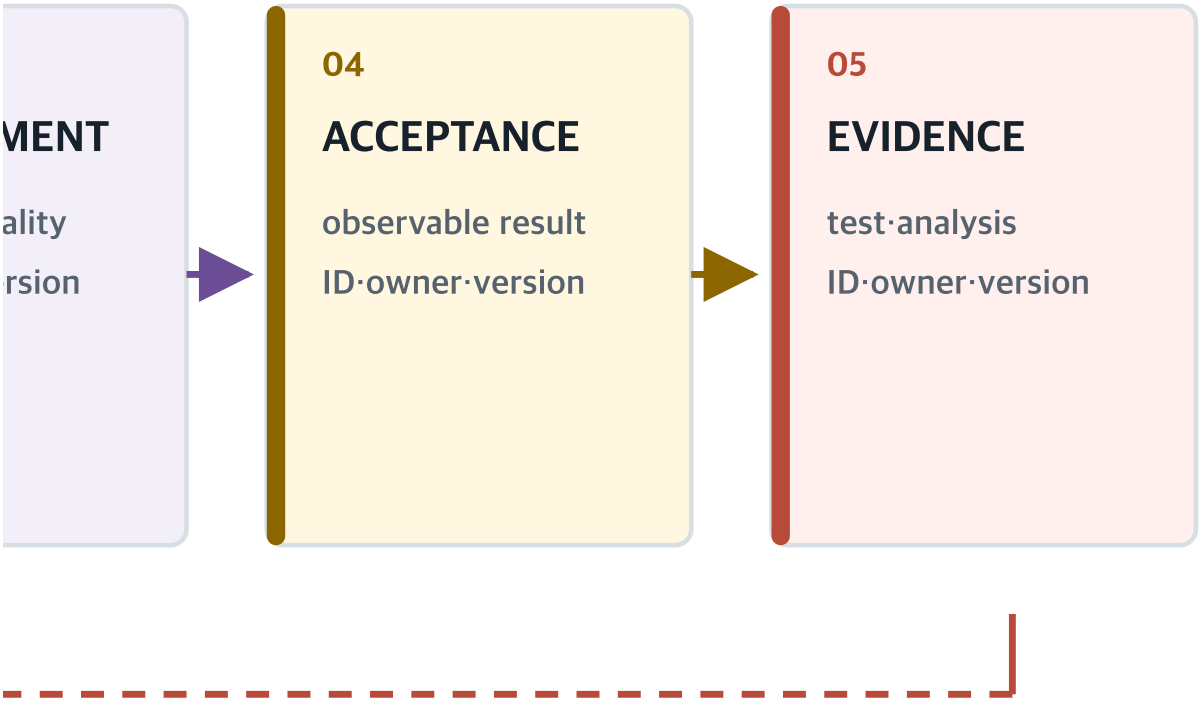
3. 문제에서 증거까지 양방향 추적하기

그림 1. 요구사항은 기능 아이디어가 아니라 문제·결과에서 acceptance·verification evidence까지 이어지는 versioned trace입니다.

기능명 하나가 아니라 존재 이유·관찰 기준·검증 방법·판정 근거가 양방향으



로 연결됩니다.


MENT
ality
rsion
04
ACCEPTANCE
observable result
ID·owner·version
05
EVIDENCE
test·analysis
ID·owner·version

3.1 그림을 합성 사례로 읽기

요구사항은 기능 아이디어가 아니라 문제·결과에서 acceptance·verification evidence까지 이어지는 versioned trace입니다.

요소	합성 사례	검토 계약
M12-01 source	운영 조정 담당자의 재입력과 완전 기록률 58%	문제·baseline·guardrail version
Requirement	FR-02 필수값 검증과 draft 보존	source·outcome·owner
Acceptance	필수값이 없으면 ready 거부·이유 표시·draft 보존	positive·negative·boundary
Evidence	재현 가능한 test와 결과 artifact	method·oracle·limitation

3.2 틀린 예와 고친 예

> 틀린 예: 기능명·형용사·정상 흐름만 적고 source와 완료 증거를 생략합니다.

> 고친 예: requirement ID마다 source·condition·observable result·acceptance·verification·owner·version을 연결합니다.

3.3 3분 연습

1. 내 requirement 후보 하나에 unique ID를 붙입니다.
2. 행동 또는 품질 obligation을 하나만 남깁니다.
3. source와 observable acceptance를 한 줄씩 연결합니다.
4. 실패·경계·권한 또는 measurement context 중 빠진 것을 하나 추가합니다.
5. 어떤 verification evidence를 누가 남길지 씁니다.

문장이 길어서 강한 것이 아닙니다. source·조건·의무·관찰 결과·검증 증거가 분리되어 다시 판정할 수 있을 때 강합니다.

4. Requirement 수준·범위·owner 맞추기

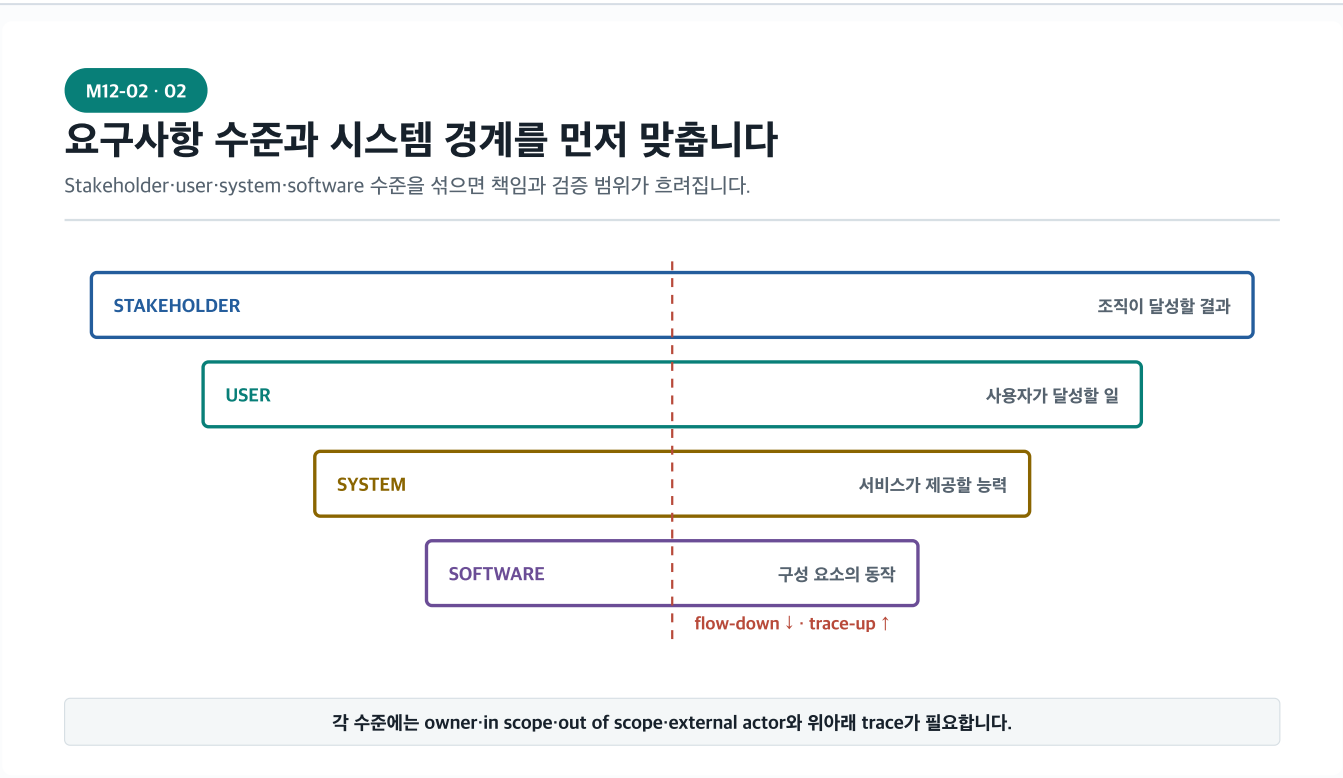
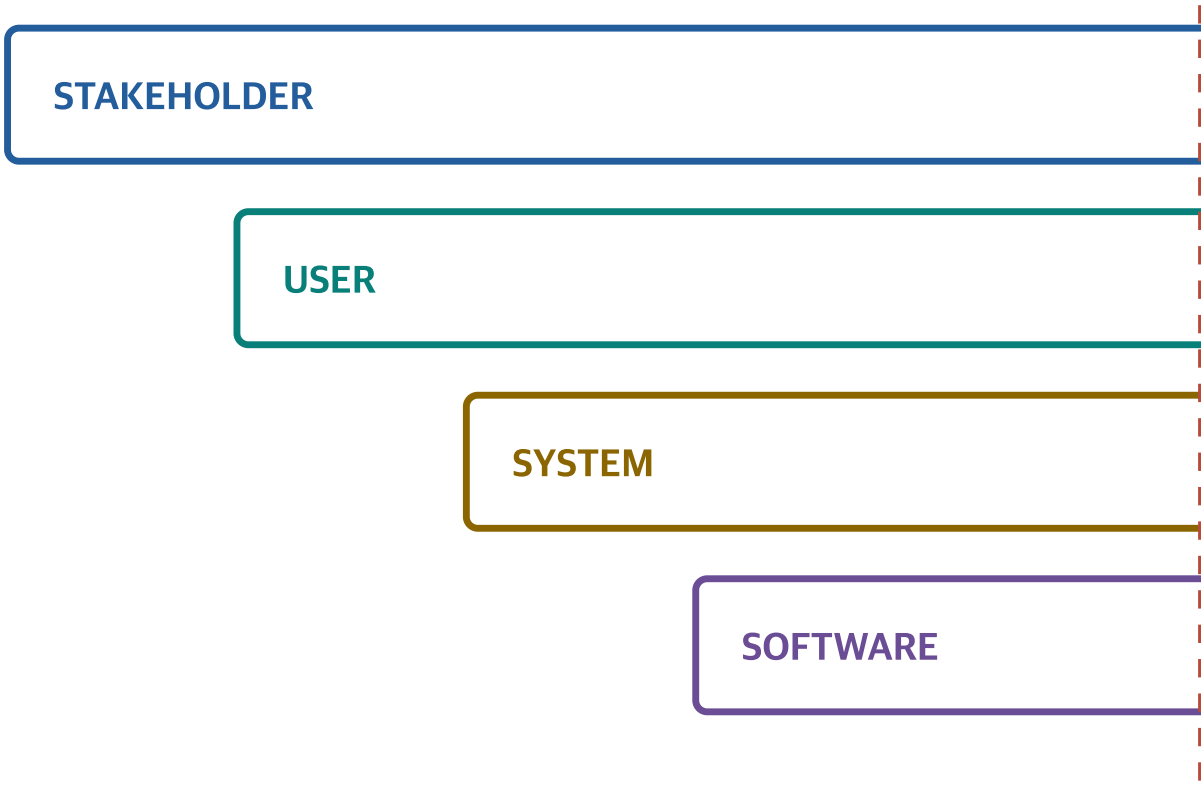


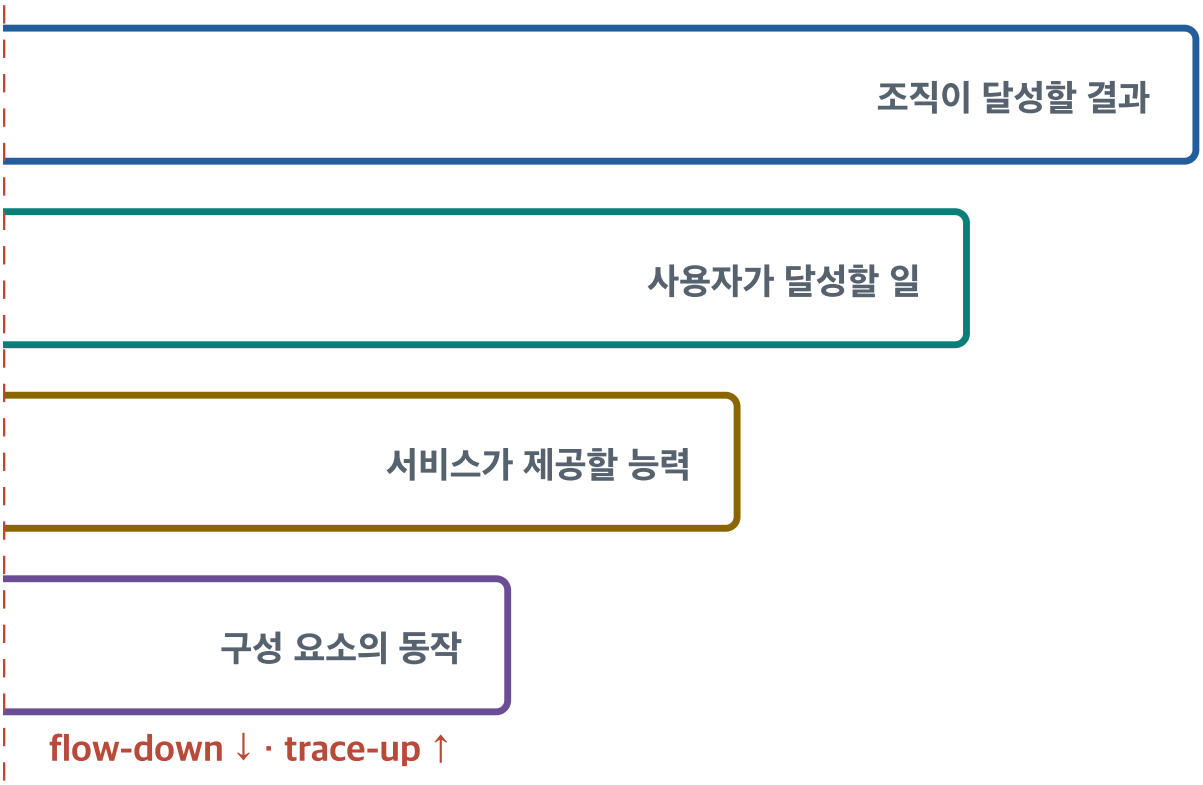
그림 2. 서로 다른 책임 수준을 한 목록에 섞지 않고 system boundary와 external actor를 먼저 고정합니다.

요구사항 수준과 시스템 경계를 분리한다

Stakeholder·user·system·software 수준을 섞으면 책임과 검증 범위가 흐



려집니다.



4.1 그림을 합성 사례로 읽기

서로 다른 책임 수준을 한 목록에 섞지 않고 system boundary와 external actor를 먼저 고정합니다.

요소	합성 사례	검토 계약
Stakeholder	완전 기록으로 실행 누락을 줄인다	조직 결과·정책
User	담당·기한을 확인하고 수정 제안한다	사용자 과업·맥락
System	확인 상태와 version을 보존한다	외부 관찰 능력
Software	동시 version conflict를 감지한다	구성 요소 행동

4.2 틀린 예와 고친 예

> 틀린 예: 기능명·형용사·정상 흐름만 적고 source와 완료 증거를 생략합니다.

> 고친 예: requirement ID마다 source·condition·observable result·acceptance·verification·owner·version을 연결합니다.

4.3 3분 연습

1. 내 requirement 후보 하나에 unique ID를 붙입니다.
2. 행동 또는 품질 obligation을 하나만 남깁니다.
3. source와 observable acceptance를 한 줄씩 연결합니다.
4. 실패·경계·권한 또는 measurement context 중 빠진 것을 하나 추가합니다.
5. 어떤 verification evidence를 누가 남길지 씁니다.

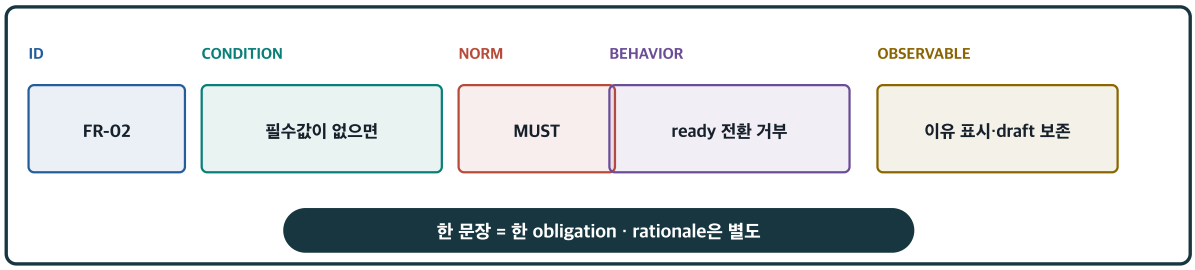
문장이 길어서 강한 것이 아닙니다. source·조건·의무·관찰 결과·검증 증거가 분리되어 다시 판정할 수 있을 때 강합니다.

5. 원자적 문장과 규범어 쓰기

M12-02 · 03

좋은 요구사항은 원자적이고 규범어의 뜻이 고정되어 있습니다

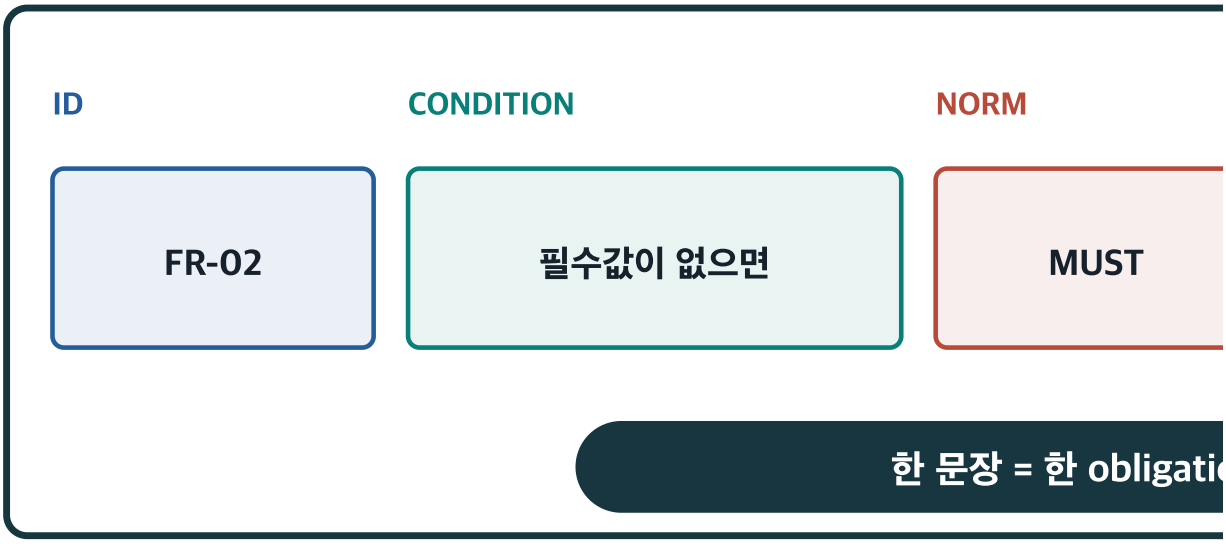
Unique ID·조건·MUST/MUST NOT/SHOULD/MAY·한 행동·관찰 결과를 분리합니다.



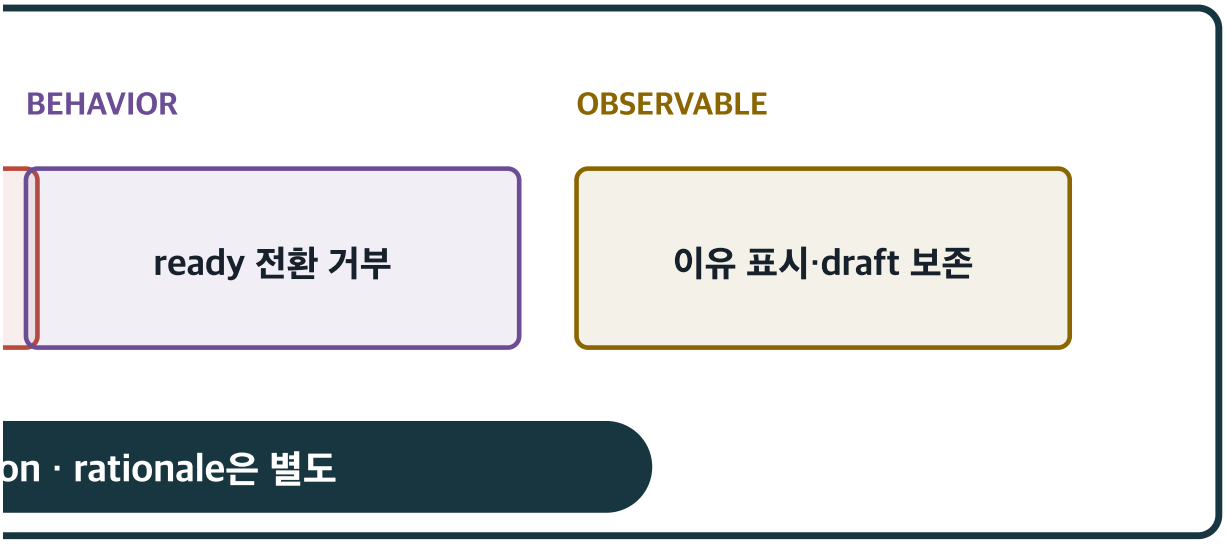
'빠르게·쉽게·적절히·and/or·등'은 측정값이나 별도 requirement로 교체합니다.

그림 3. 한 requirement에는 한 obligation만 두고 MUST·MUST NOT·SHOULD·MAY의 조직 의미를 고정합니다.

Unique ID·조건·MUST/MUST NOT/SHOULD/MAY·한 행동·관찰 결과를 분



[illegible]리합니다.


BEHAVIOR
OBSERVABLE
ready 전환 거부
이유 표시·draft 보존
on · rationale은 별도

5.1 그림을 합성 사례로 읽기

한 requirement에는 한 obligation만 두고 MUST·MUST NOT·SHOULD·MAY의 조직 의미를 고정합니다.

요소	합성 사례	검토 계약
나쁜 예	시스템은 빠르고 쉽게 입력·확인·export한다	복합·모호·측정 불가
고친 기능	FR-02 invalid 입력이면 ready 전환을 거부하여야 한다	조건·행동 1개
고친 결과	field-level 이유를 표시하고 draft를 보존하여야 한다	필요하면 별도 ID
고친 품질	50 active user에서 p95 ≤ 2초	context·measure·threshold

5.2 틀린 예와 고친 예

> 틀린 예: 기능명·형용사·정상 흐름만 적고 source와 완료 증거를 생략합니다.

> 고친 예: requirement ID마다 source·condition·observable result·acceptance·verification·owner·version을 연결합니다.

5.3 3분 연습

1. 내 requirement 후보 하나에 unique ID를 붙입니다.
2. 행동 또는 품질 obligation을 하나만 남깁니다.
3. source와 observable acceptance를 한 줄씩 연결합니다.
4. 실패·경계·권한 또는 measurement context 중 빠진 것을 하나 추가합니다.
5. 어떤 verification evidence를 누가 남길지 씁니다.

문장이 길어서 강한 것이 아닙니다. source·조건·의무·관찰 결과·검증 증거가 분리되어 다시 판정할 수 있을 때 강합니다.

6. 기능 요구사항을 행동 계약으로 쓰기

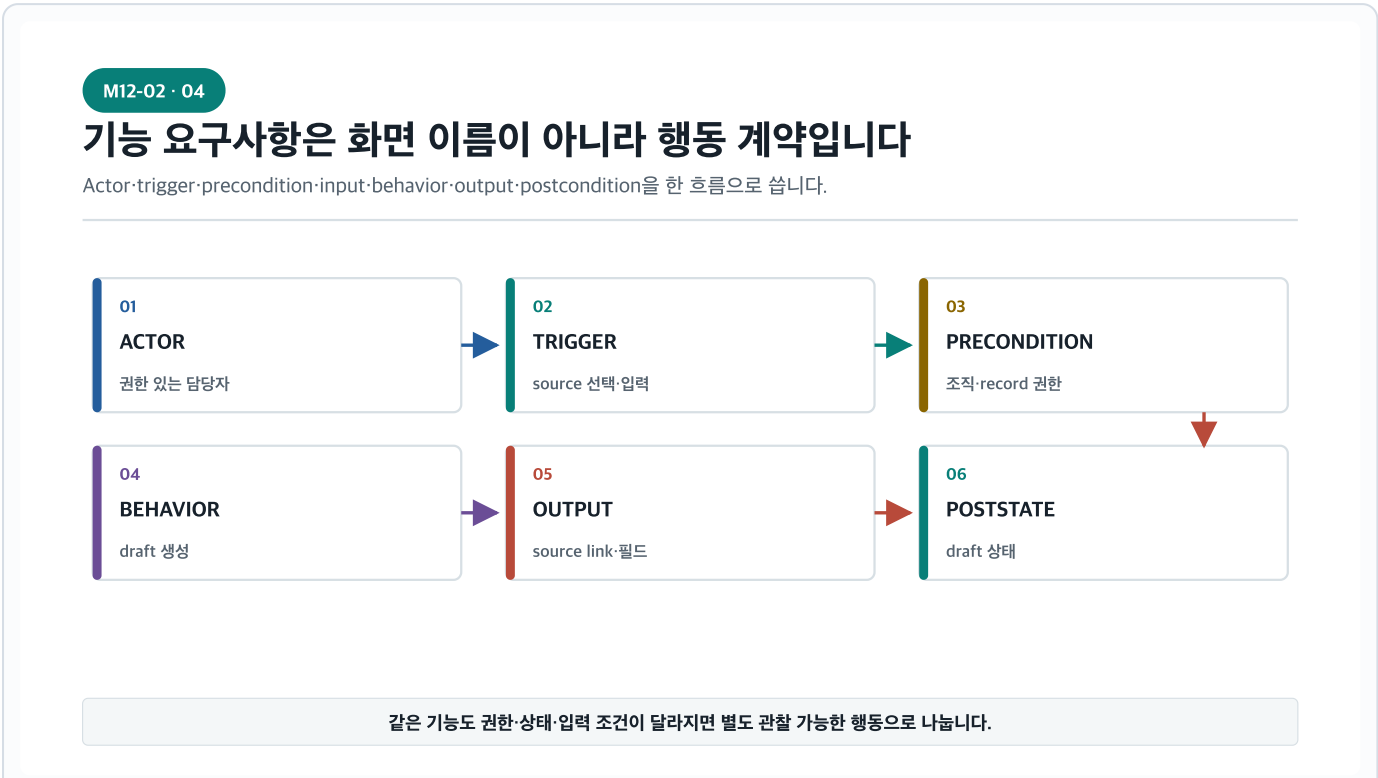


그림 4. 화면 이름이 아니라 actor가 어떤 조건에서 무엇을 입력하고 시스템이 어떤 상태·출력을 만드는지 씁니다.

Actor·trigger·precondition·input·behavior·output·postcondition을 한

01

ACTOR

권한 있는 담당자

02

TRIGGER

source 선택·입력

04

BEHAVIOR

draft 생성

05

OUTPUT

source link·필드

흐름으로 씁니다.

6.1 그림을 합성 사례로 읽기

화면 이름이 아니라 actor가 어떤 조건에서 무엇을 입력하고 시스템이 어떤 상태·출력을 만드는지 씁니다.

요소	합성 사례	검토 계약
Actor	권한 있는 운영 조정 담당자	역할·조직·record 권한
Trigger	회의 source 선택과 필드 입력	행동 시작 사건
Behavior	source link가 있는 draft 생성	한 obligation
Poststate	draft이며 ready가 아님	다음 허용 행동

6.2 틀린 예와 고친 예

> 틀린 예: 기능명·형용사·정상 흐름만 적고 source와 완료 증거를 생략합니다.

> 고친 예: requirement ID마다 source·condition·observable result·acceptance·verification·owner·version을 연결합니다.

6.3 3분 연습

1. 내 requirement 후보 하나에 unique ID를 붙입니다.
2. 행동 또는 품질 obligation을 하나만 남깁니다.
3. source와 observable acceptance를 한 줄씩 연결합니다.
4. 실패·경계·권한 또는 measurement context 중 빠진 것을 하나 추가합니다.
5. 어떤 verification evidence를 누가 남길지 씁니다.

문장이 길어서 강한 것이 아닙니다. source·조건·의무·관찰 결과·검증 증거가 분리되어 다시 판정할 수 있을 때 강합니다.

7. 대안·예외·오류·복구까지 쓰기

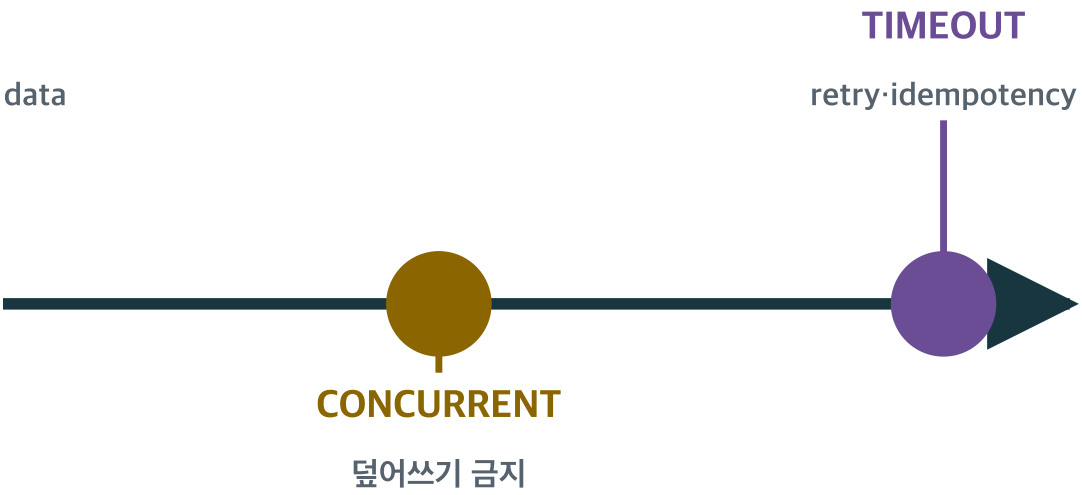
그림 5. 정상 성공 외에 invalid·empty·denied·timeout·concurrent update와 사용자가 다시 일할 수 있는 복구를 정의합니다.

[illegible]

Invalid·empty·denied·concurrent·timeout과 사용자에게 보이는 오류·복

ERROR ≠ END · 관

구를 함께 정의합니다.



찰·복구·재시도 경계

7.1 그림을 합성 사례로 읽기

정상 성공 외에 invalid·empty·denied·timeout·concurrent update와 사용자가 다시 일할 수 있는 복구를 정의합니다.

요소	합성 사례	검토 계약
Invalid	ready 거부·field 이유·draft 보존	입력 손실 0
Denied	server-side 기본 거부	허용 데이터 노출 0
Concurrent	version conflict·두 변경 보존	silent overwrite 금지
Timeout	retry와 idempotency 경계	중복 side effect 금지

7.2 틀린 예와 고친 예

> 틀린 예: 기능명·형용사·정상 흐름만 적고 source와 완료 증거를 생략합니다.

> 고친 예: requirement ID마다 source·condition·observable result·acceptance·verification·owner·version을 연결합니다.

7.3 3분 연습

1. 내 requirement 후보 하나에 unique ID를 붙입니다.
2. 행동 또는 품질 obligation을 하나만 남깁니다.
3. source와 observable acceptance를 한 줄씩 연결합니다.
4. 실패·경계·권한 또는 measurement context 중 빠진 것을 하나 추가합니다.
5. 어떤 verification evidence를 누가 남길지 씁니다.

문장이 길어서 강한 것이 아닙니다. source·조건·의무·관찰 결과·검증 증거가 분리되어 다시 판정할 수 있을 때 강합니다.

8. 데이터·validation·상태 전이 연결하기

그림 6. 필드·검증·권한·provenance·history·retention·export를 상태 전이와 같은 계약에서 봅니다.

제 [illegible]

Validation·authorization·provenance·history·retention·export 경계를 [illegible]

FIELD · type·require

HISTORY · actor·time·before/afte

상태 변화와 연결합니다.

ed·validation·source

er·reason · RETENTION · EXPORT

8.1 그림을 합성 사례로 읽기

필드·검증·권한·provenance·history·retention·export를 상태 전이와 같은 계약에서 봅니다.

요소	합성 사례	검토 계약
Field	decision·owner·due·source link	type·required·validation
State	draft→ready→pending→confirmed	허용 전이만
History	actor·time·before/after·reason	변경 100%
Boundary	metadata-only trace·retention·export	민감 원문 금지

8.2 틀린 예와 고친 예

> 틀린 예: 기능명·형용사·정상 흐름만 적고 source와 완료 증거를 생략합니다.

> 고친 예: requirement ID마다 source·condition·observable result·acceptance·verification·owner·version을 연결합니다.

8.3 3분 연습

1. 내 requirement 후보 하나에 unique ID를 붙입니다.
2. 행동 또는 품질 obligation을 하나만 남깁니다.
3. source와 observable acceptance를 한 줄씩 연결합니다.
4. 실패·경계·권한 또는 measurement context 중 빠진 것을 하나 추가합니다.
5. 어떤 verification evidence를 누가 남길지 씁니다.

문장이 길어서 강한 것이 아닙니다. source·조건·의무·관찰 결과·검증 증거가 분리되어 다시 판정할 수 있을 때 강합니다.

9. 품질 형용사를 측정 시나리오로 바꾸기

그림 7. '빠르게·안정적으로' 대신 어떤 조건에서 무엇을 재고 어느 값을 통과로 볼지 씁니다.

Context·stimulus·measure·threshold·window·load·segment·source를

01

CONTEXT

50 active users

02

STIMULUS

create·confirm·filter

04

THRESHOLD

≤ 2 seconds

05

WINDOW

30 min sustained loa

품질 형용사 → 측정

붙입니다.

03

MEASURE

server response p95

06

EVIDENCE

repeatable load test

형 가능한 scenario

9.1 그림을 합성 사례로 읽기

‘빠르게·안정적으로’ 대신 어떤 조건에서 무엇을 재고 어느 값을 통과로 볼지 씁니다.

요소	합성 사례	검토 계약
Context	50 concurrent active user·30분	환경·load
Stimulus	create·confirm·filter request	측정 대상
Measure	server response p95	집계 정의
Threshold	2초 이하	test·source·owner

9.2 틀린 예와 고친 예

> 틀린 예: 기능명·형용사·정상 흐름만 적고 source와 완료 증거를 생략합니다.

> 고친 예: requirement ID마다 source·condition·observable result·acceptance·verification·owner·version을 연결합니다.

9.3 3분 연습

1. 내 requirement 후보 하나에 unique ID를 붙입니다.
2. 행동 또는 품질 obligation을 하나만 남깁니다.
3. source와 observable acceptance를 한 줄씩 연결합니다.
4. 실패·경계·권한 또는 measurement context 중 빠진 것을 하나 추가합니다.
5. 어떤 verification evidence를 누가 남길지 씁니다.

문장이 길어서 강한 것이 아닙니다. source·조건·의무·관찰 결과·검증 증거가 분리되어 다시 판정할 수 있을 때 강합니다.

10. ISO 품질 지도로 누락 찾기

그림 8. 공통 품질 모델을 체크박스가 아니라 문제·risk에 맞는 누락 질문과 제외 근거를 찾는 지도처럼 씁니다.

ISO/IEC 25010:2023의 제품 품질 아홉 특성을 체크리스트가 아닌 질문 지

기능 적합성

성능

상호작용 능력

신

유지보수성

유

ISO/IEC 25010:202

|도처럼 사용합니다.

3 · 9 characteristics

10.1 그림을 합성 사례로 읽기

공통 품질 모델을 체크박스가 아니라 문제·risk에 맞는 누락 질문과 제외 근거를 찾는 지도처럼 씁니다.

요소	합성 사례	검토 계약
사용 품질	완전 기록률·재입력 시간·오배정률	QIU-01~03
제품 품질	성능·접근·보안·신뢰·복구·감사·비용	QR-01~08
선택 기준	문제·guardrail·failure impact	무관한 MUST 남발 금지
Review	특성·측정·threshold·verification	source와 version

10.2 틀린 예와 고친 예

> 틀린 예: 기능명·형용사·정상 흐름만 적고 source와 완료 증거를 생략합니다.

> 고친 예: requirement ID마다 source·condition·observable result·acceptance·verification·owner·version을 연결합니다.

10.3 3분 연습

1. 내 requirement 후보 하나에 unique ID를 붙입니다.
2. 행동 또는 품질 obligation을 하나만 남깁니다.
3. source와 observable acceptance를 한 줄씩 연결합니다.
4. 실패·경계·권한 또는 measurement context 중 빠진 것을 하나 추가합니다.
5. 어떤 verification evidence를 누가 남길지 씁니다.

문장이 길어서 강한 것이 아닙니다. source·조건·의무·관찰 결과·검증 증거가 분리되어 다시 판정할 수 있을 때 강합니다.

11. 접근성·보안·privacy·safety를 처음부터 요구하기

그림 9. 표준 이름만 붙이지 않고 적용 core flow·data·권한·사람 검토·검증 범위를 명시합니다.

접근성·보안·privacy·safety를 처음부터

핵심 flow·data·권한·피해 경계에 요구사항과 검증 방법을 처음부터 연결합

ACCESSIBILITY

SECURITY

PRIVACY

SAFETY

[illegible]니다.

WCAG 2.2 scope · keyboard · name/role/value

server authorization · least privilege · ASVS

minimize · purpose · retention · metadata-only

harm boundary · human review · stop/escalate

11.1 그림을 합성 사례로 읽기

표준 이름만 붙이지 않고 적용 core flow·data·권한·사람 검토·검증 범위를 명시합니다.

요소	합성 사례	검토 계약
Accessibility	WCAG 2.2 A·AA 적용 기준·keyboard·name/role/value	자동+사람+보조기술
Security	조직·record·action server authorization	ASVS·deny by default
Privacy	원문 logging 금지·metadata-only	최소화·retention
Safety	피해 경계·중단·escalation	인증 주장 아님

11.2 틀린 예와 고친 예

> 틀린 예: 기능명·형용사·정상 흐름만 적고 source와 완료 증거를 생략합니다.

> 고친 예: requirement ID마다 source·condition·observable result·acceptance·verification·owner·version을 연결합니다.

11.3 3분 연습

1. 내 requirement 후보 하나에 unique ID를 붙입니다.
2. 행동 또는 품질 obligation을 하나만 남깁니다.
3. source와 observable acceptance를 한 줄씩 연결합니다.
4. 실패·경계·권한 또는 measurement context 중 빠진 것을 하나 추가합니다.
5. 어떤 verification evidence를 누가 남길지 씁니다.

문장이 길어서 강한 것이 아닙니다. source·조건·의무·관찰 결과·검증 증거가 분리되어 다시 판정할 수 있을 때 강합니다.

12. Constraint·dependency·assumption·TBR 분리하기

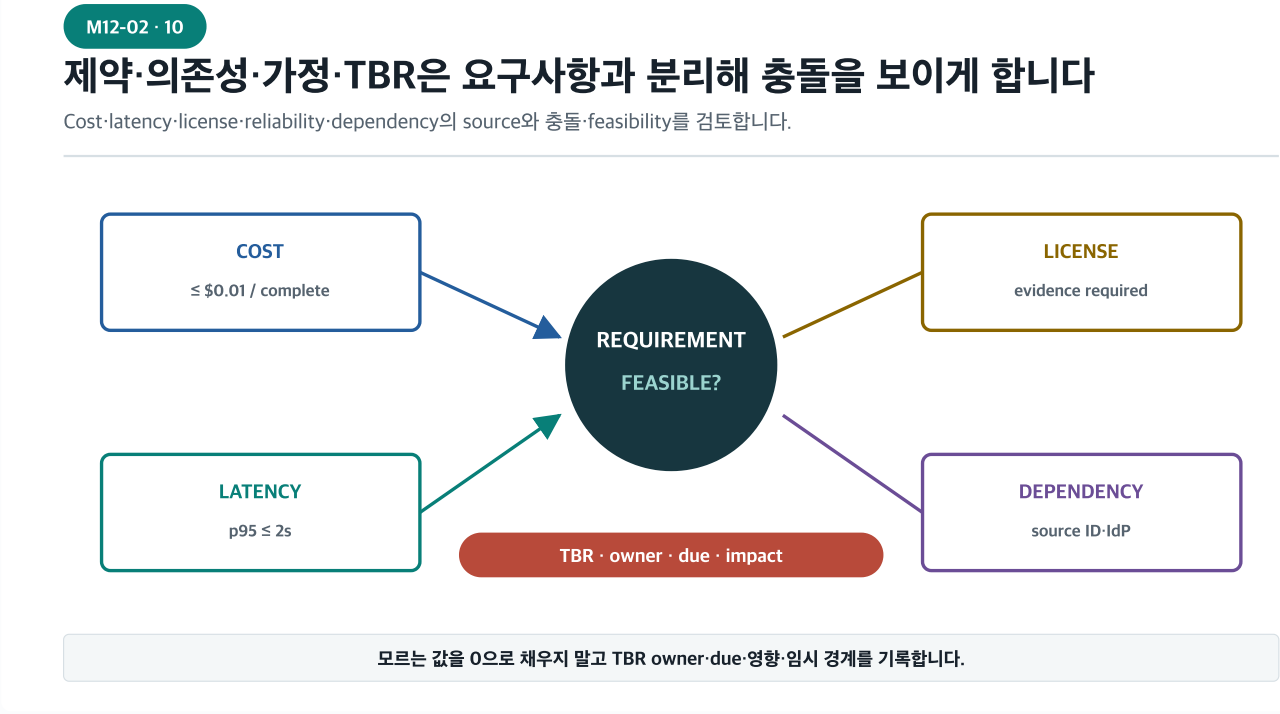
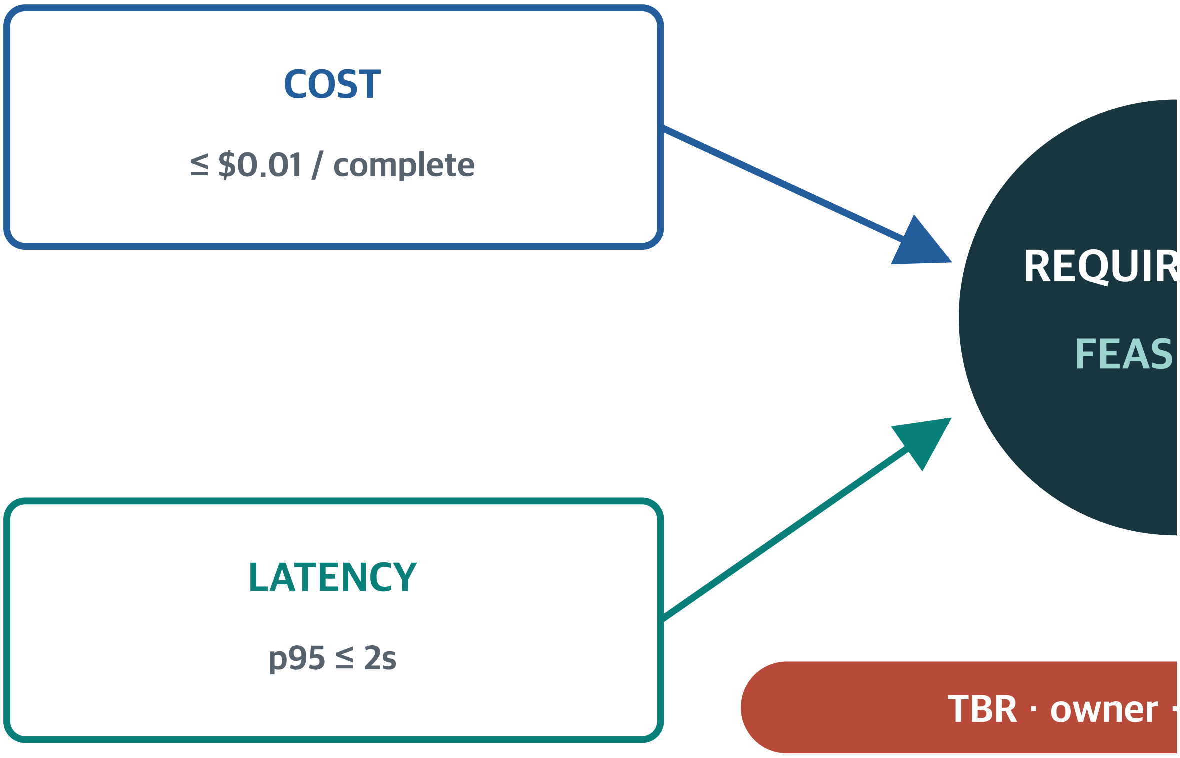
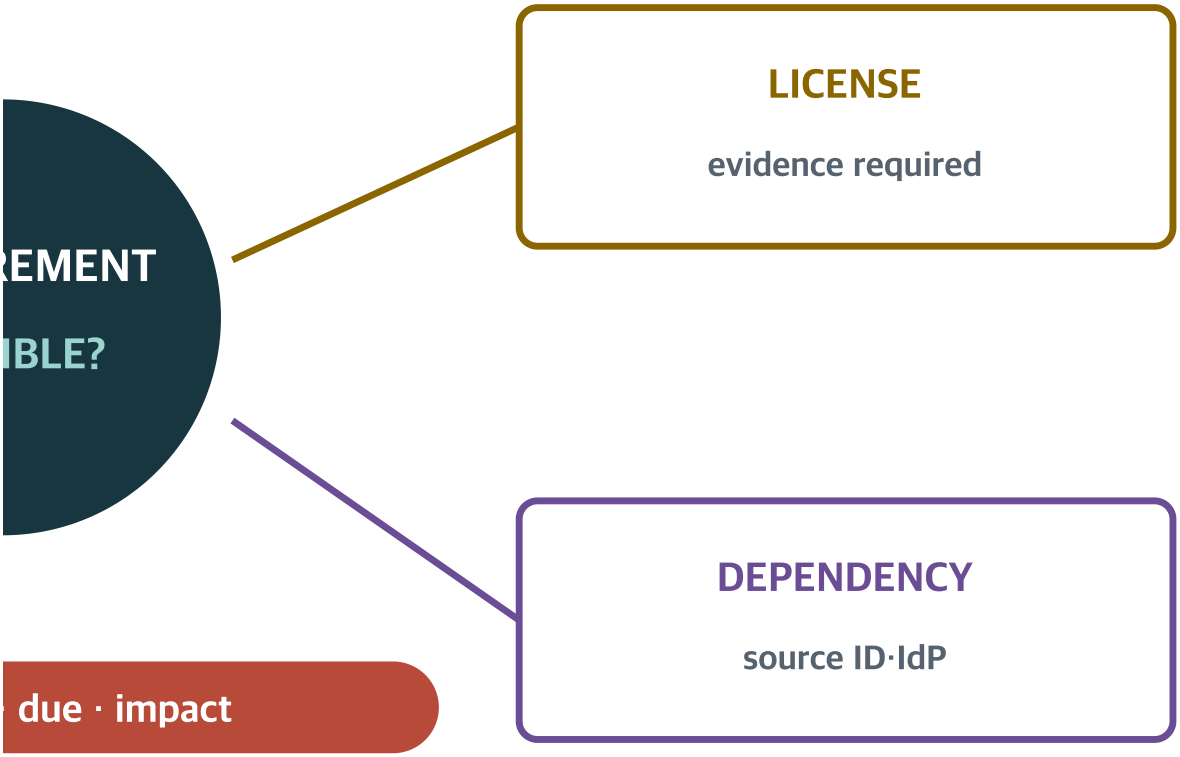


그림 10. 요구와 현실 조건을 한 문장에 숨기지 않고 source·conflict·feasibility·owner·due를 드러냅니다.

Cost·latency·license·reliability·dependency의 source와 충돌·feasibility



를 검토합니다.


LICENSE
evidence required
DEPENDENCY
source ID·IdP
due · impact

12.1 그림을 합성 사례로 읽기

요구와 현실 조건을 한 문장에 숨기지 않고 source·conflict·feasibility·owner·due를 드러냅니다.

요소	합성 사례	검토 계약
Cost	complete record당 USD 0.01 이하	월간 fully allocated
License	tool·library·model·data evidence	배포·학습·export 범위
Dependency	meeting source ID·organization IdP	availability·owner
TBR	미결정 2개 이하	owner·due·impact·임시 경계

12.2 틀린 예와 고친 예

> 틀린 예: 기능명·형용사·정상 흐름만 적고 source와 완료 증거를 생략합니다.

> 고친 예: requirement ID마다 source·condition·observable result·acceptance·verification·owner·version을 연결합니다.

12.3 3분 연습

1. 내 requirement 후보 하나에 unique ID를 붙입니다.
2. 행동 또는 품질 obligation을 하나만 남깁니다.
3. source와 observable acceptance를 한 줄씩 연결합니다.
4. 실패·경계·권한 또는 measurement context 중 빠진 것을 하나 추가합니다.
5. 어떤 verification evidence를 누가 남길지 씁니다.

문장이 길어서 강한 것이 아닙니다. source·조건·의무·관찰 결과·검증 증거가 분리되어 다시 판정할 수 있을 때 강합니다.

13. Acceptance criterion을 위험 공간으로 펼치기

그림 11. Given·When·Then 또는 동등한 형식으로 정상·실패·경계·권한 결과를 외부에서 관찰 가능하게 씁니다.

Positive·negative·boundary·authorization 결과를 Given·When·Then 또

AC

POSITIVE

정상 성공

Given·When·Then

외부에서 관찰

AC

NEGATIVE

invalid·denied

Given·When·Then

외부에서 관찰

Criterion coverage = requi

:는 동등한 형식으로 씁니다.

AC

BOUNDARY

0·1·max·time

Given·When·Then

외부에서 관찰

AC

AUTHORIZATION

role·record·action

Given·When·Then

외부에서 관찰

irement × risk × boundary

13.1 그림을 합성 사례로 읽기

Given·When·Then 또는 동등한 형식으로 정상·실패·경계·권한 결과를 외부에서 관찰 가능하게 씁니다.

요소	합성 사례	검토 계약
Positive	유효 필드면 draft 생성	성공 output·poststate
Negative	필수값 누락이면 ready 거부	이유·draft 보존
Boundary	0·1·max·due edge	정확한 값
Authorization	허용 role만 read·edit·export	server denial

13.2 틀린 예와 고친 예

> 틀린 예: 기능명·형용사·정상 흐름만 적고 source와 완료 증거를 생략합니다.

> 고친 예: requirement ID마다 source·condition·observable result·acceptance·verification·owner·version을 연결합니다.

13.3 3분 연습

1. 내 requirement 후보 하나에 unique ID를 붙입니다.
2. 행동 또는 품질 obligation을 하나만 남깁니다.
3. source와 observable acceptance를 한 줄씩 연결합니다.
4. 실패·경계·권한 또는 measurement context 중 빠진 것을 하나 추가합니다.
5. 어떤 verification evidence를 누가 남길지 씁니다.

문장이 길어서 강한 것이 아닙니다. source·조건·의무·관찰 결과·검증 증거가 분리되어 다시 판정할 수 있을 때 강합니다.

14. 검증 방법과 증거 계약하기

그림 12. Requirement 특성과 risk에 맞는 방법·oracle·환경·owner·artifact·limitation을 완료 기준에 연결합니다.

Inspection·analysis·demonstration·test를 requirement 특성과 risk에 맞

01

INSPECTION

문서·UI·구성 검토

owner·environment

artifact·result

02

ANALYSIS

수치·로그·모형 계산

owner·environment

artifact·result

method + oracle + e

합격

맞춰 선택합니다.

03

DEMONSTRATION

대표 flow 관찰

owner·environment

artifact·result

04

TEST

통제 조건·expected 비교

owner·environment

artifact·result

vidence + limitation

14.1 그림을 합성 사례로 읽기

Requirement 특성과 risk에 맞는 방법·oracle·환경·owner·artifact·limitation을 완료 기준에 연결합니다.

요소	합성 사례	검토 계약
Inspection	문장·UI·configuration·trace 검토	review artifact
Analysis	rate·p95·cost·availability 계산	source·denominator
Demonstration	대표 actor flow 관찰	screen·result
Test	통제 조건에서 expected/actual 비교	repeatable result

14.2 틀린 예와 고친 예

> 틀린 예: 기능명·형용사·정상 흐름만 적고 source와 완료 증거를 생략합니다.

> 고친 예: requirement ID마다 source·condition·observable result·acceptance·verification·owner·version을 연결합니다.

14.3 3분 연습

1. 내 requirement 후보 하나에 unique ID를 붙입니다.
2. 행동 또는 품질 obligation을 하나만 남깁니다.
3. source와 observable acceptance를 한 줄씩 연결합니다.
4. 실패·경계·권한 또는 measurement context 중 빠진 것을 하나 추가합니다.
5. 어떤 verification evidence를 누가 남길지 씁니다.

문장이 길어서 강한 것이 아닙니다. source·조건·의무·관찰 결과·검증 증거가 분리되어 다시 판정할 수 있을 때 강합니다.

15. 문장과 set 전체 품질을 함께 검토하기

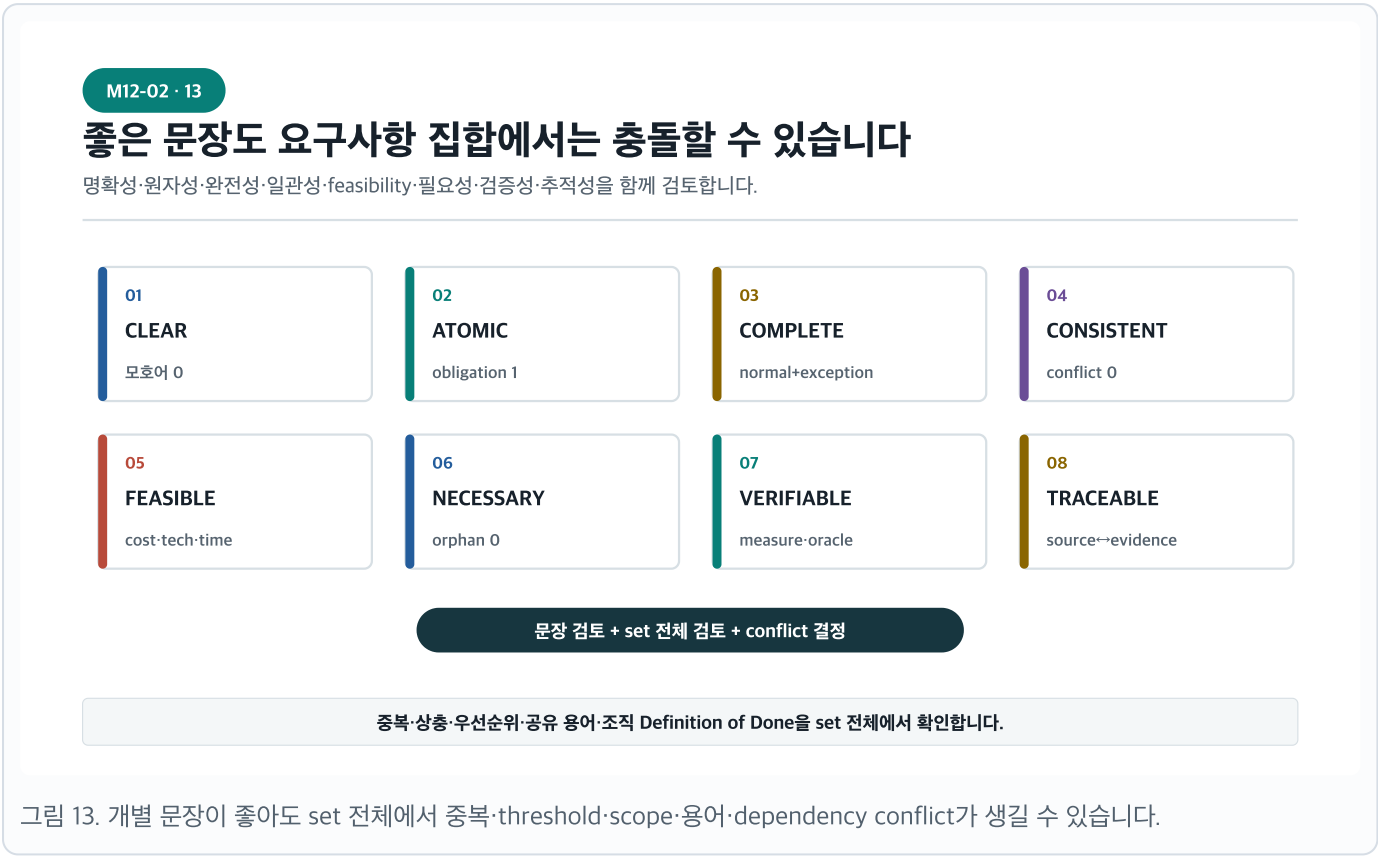


그림 13. 개별 문장이 좋아도 set 전체에서 중복·threshold·scope·용어·dependency conflict가 생길 수 있습니다.

명확성·원자성·완전성·일관성·feasibility·필요성·검증성·추적성을 함께 검토

01

CLEAR

모호어 0

02

ATOMIC

obligation 1

05

FEASIBLE

cost·tech·time

06

NECESSARY

orphan 0

문장 검토 + set 전체

합니다.

03

COMPLETE

normal+exception

04

CONSISTENT

conflict 0

07

VERIFIABLE

measure·oracle

08

TRACEABLE

source↔evidence

검토 + conflict 결정

15.1 그림을 합성 사례로 읽기

개별 문장이 좋아도 set 전체에서 중복·threshold·scope·용어·dependency conflict가 생길 수 있습니다.

요소	합성 사례	검토 계약
문장	clear·atomic·complete·verifiable	vague·and/or·etc 0
집합	consistent·necessary·feasible	duplicate·conflict 0
추적	source↔requirement↔evidence	orphan 0
결정	finding·owner·action·closure	waiver·residual risk

15.2 틀린 예와 고친 예

> 틀린 예: 기능명·형용사·정상 흐름만 적고 source와 완료 증거를 생략합니다.

> 고친 예: requirement ID마다 source·condition·observable result·acceptance·verification·owner·version을 연결합니다.

15.3 3분 연습

1. 내 requirement 후보 하나에 unique ID를 붙입니다.
2. 행동 또는 품질 obligation을 하나만 남깁니다.
3. source와 observable acceptance를 한 줄씩 연결합니다.
4. 실패·경계·권한 또는 measurement context 중 빠진 것을 하나 추가합니다.
5. 어떤 verification evidence를 누가 남길지 씁니다.

문장이 길어서 강한 것이 아닙니다. source·조건·의무·관찰 결과·검증 증거가 분리되어 다시 판정할 수 있을 때 강합니다.

16. Requirement Readiness Gate와 M12-03 handoff

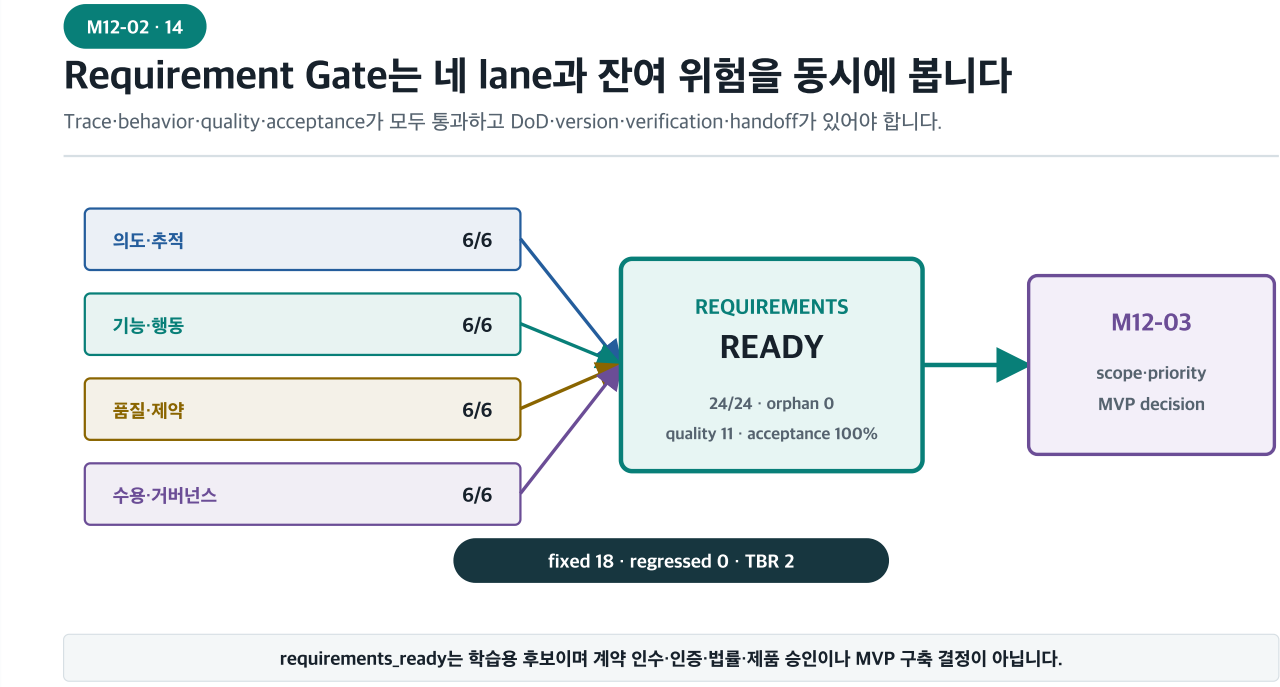
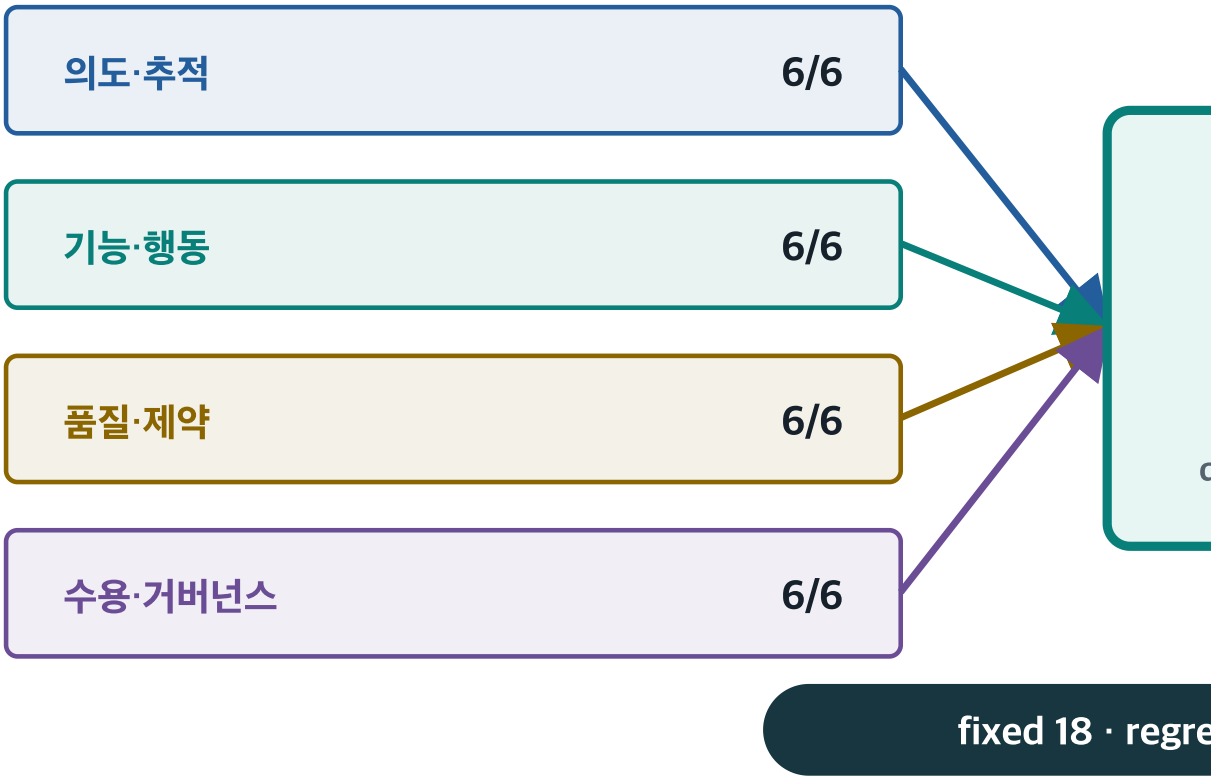


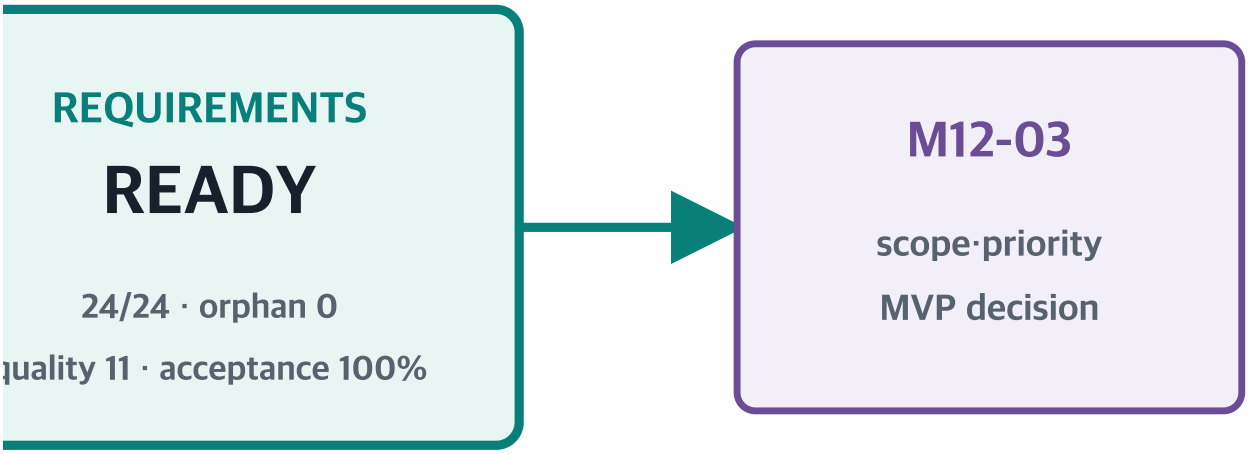
그림 14. 24개 scenario와 12개 control, trace·quality·acceptance·DoD·TBR를 동시에 보고 다음 단계 후보를 판정합니다.

Requirement Gate는 "구현됐다"가 아

Trace·behavior·quality·acceptance가 모두 통과하고 DoD·version·verifi



cation·handoff가 있어야 합니다.



ssed 0 · TBR 2

16.1 그림을 합성 사례로 읽기

24개 scenario와 12개 control, trace·quality·acceptance·DoD·TBR를 동시에 보고 다음 단계 후보를 판정합니다.

요소	합성 사례	검토 계약
Baseline	feature-list-v1 6/24	blocked_feature_list
Partial	functional-only-v2 15/24	blocked_quality_acceptance
Candidate	verified-requirements-v3 24/24	requirements_ready
Handoff	19 requirements·4 constraints·TBR 2	M12-03 scope·priority

16.2 틀린 예와 고친 예

> 틀린 예: 기능명·형용사·정상 흐름만 적고 source와 완료 증거를 생략합니다.

> 고친 예: requirement ID마다 source·condition·observable result·acceptance·verification·owner·version을 연결합니다.

16.3 3분 연습

1. 내 requirement 후보 하나에 unique ID를 붙입니다.
2. 행동 또는 품질 obligation을 하나만 남깁니다.
3. source와 observable acceptance를 한 줄씩 연결합니다.
4. 실패·경계·권한 또는 measurement context 중 빠진 것을 하나 추가합니다.
5. 어떤 verification evidence를 누가 남길지 씁니다.

문장이 길어서 강한 것이 아닙니다. source·조건·의무·관찰 결과·검증 증거가 분리되어 다시 판정할 수 있을 때 강합니다.

17. 공식 프레임워크를 실무 계약으로 연결하기

표준은 요구사항을 대신 써 주지 않습니다. 공통 언어·누락 질문·검증 경계를 제공하고, 실제 적용 범위와 최신 version은 owner가 다시 확인합니다.

공식 자료	이 매뉴얼에서 쓰는 지점	과도한 주장 방지
ISO/IEC/IEEE 29148:2018	요구공학 process·정보 항목·좋은 requirement 품질	현재판 확인과 조직 tailoring 필요
ISO/IEC 25010:2023	제품 품질 아홉 특성	전부 MUST가 아님
ISO/IEC 25019:2023	실제 사용 맥락의 quality-in-use	제품 품질과 구분
ISO/IEC 25030:2019	quality requirement framework	context·stakeholder need 연결
ISO/IEC 25040:2024	quality evaluation framework	평가 계획·evidence 필요
RFC 2119·8174	MUST·SHOULD·MAY 의미	규범어를 대문자로 쓸 때 적용
Scrum Guide 2020	Definition of Done	특정 criterion을 대체하지 않음
WCAG 2.2	웹 접근성 success criteria	범위·level·사람 검토 필요
OWASP ASVS 5.0.0	웹 보안 verification requirement	인증서가 아니라 검증 기준
NIST SSDF v1.1	개발 생명주기의 보안 관행	제품 보안 보증 전체를 대체하지 않음

17.1 Normative keyword 사용 규칙

```
MUST       정의된 범위에서 반드시 충족
MUST NOT   정의된 행동을 절대 허용하지 않음
SHOULD     강한 권고, 예외에는 근거·owner·승인 필요
MAY        허용되는 선택, 구현 의무 아님
```

RFC 8174는 이 규범 의미가 대문자로 표시될 때 적용됨을 명확히 합니다. 한국어 요구사항에서는 `하여야 한다·해서는 안 된다`와 내부 규범어 표를 함께 version 관리합니다.

18. 실습 스튜디오를 화면으로 읽기

18.1 데스크톱 · 전체 계약과 24개 시나리오

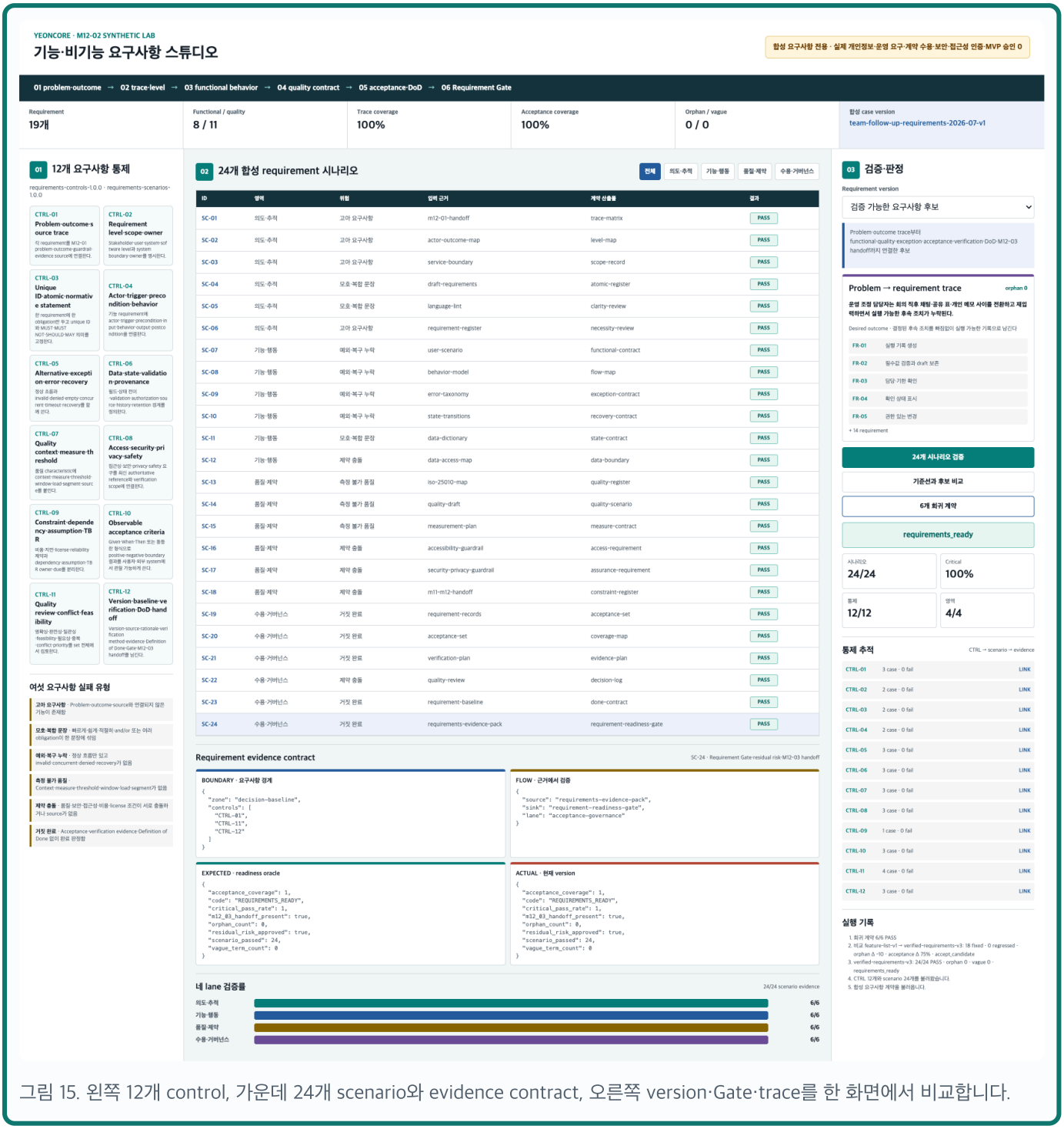


그림 15. 왼쪽 12개 control, 가운데 24개 scenario와 evidence contract, 오른쪽 version·Gate·trace를 한 화면에서 비교합니다.

18.2 Scenario evidence detail

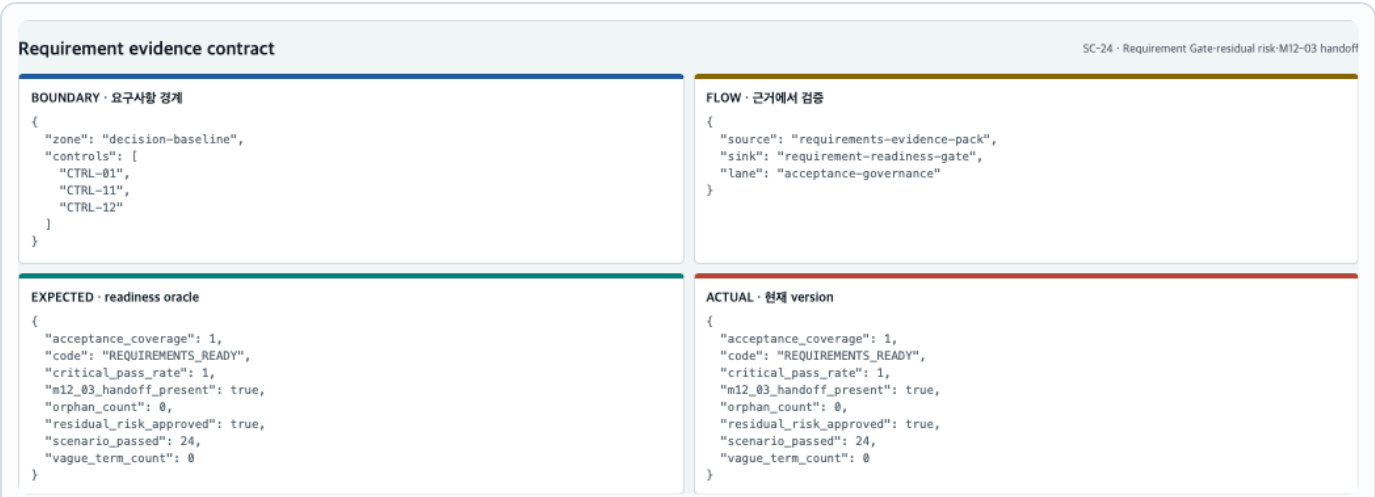


그림 16. Boundary·flow·expected·actual 네 칸이 같아야 scenario가 통과합니다. PASS 배지만 보지 말고 실제 field를 비교합니다.

18.3 모바일 · 통제·시나리오·Gate 세 구간

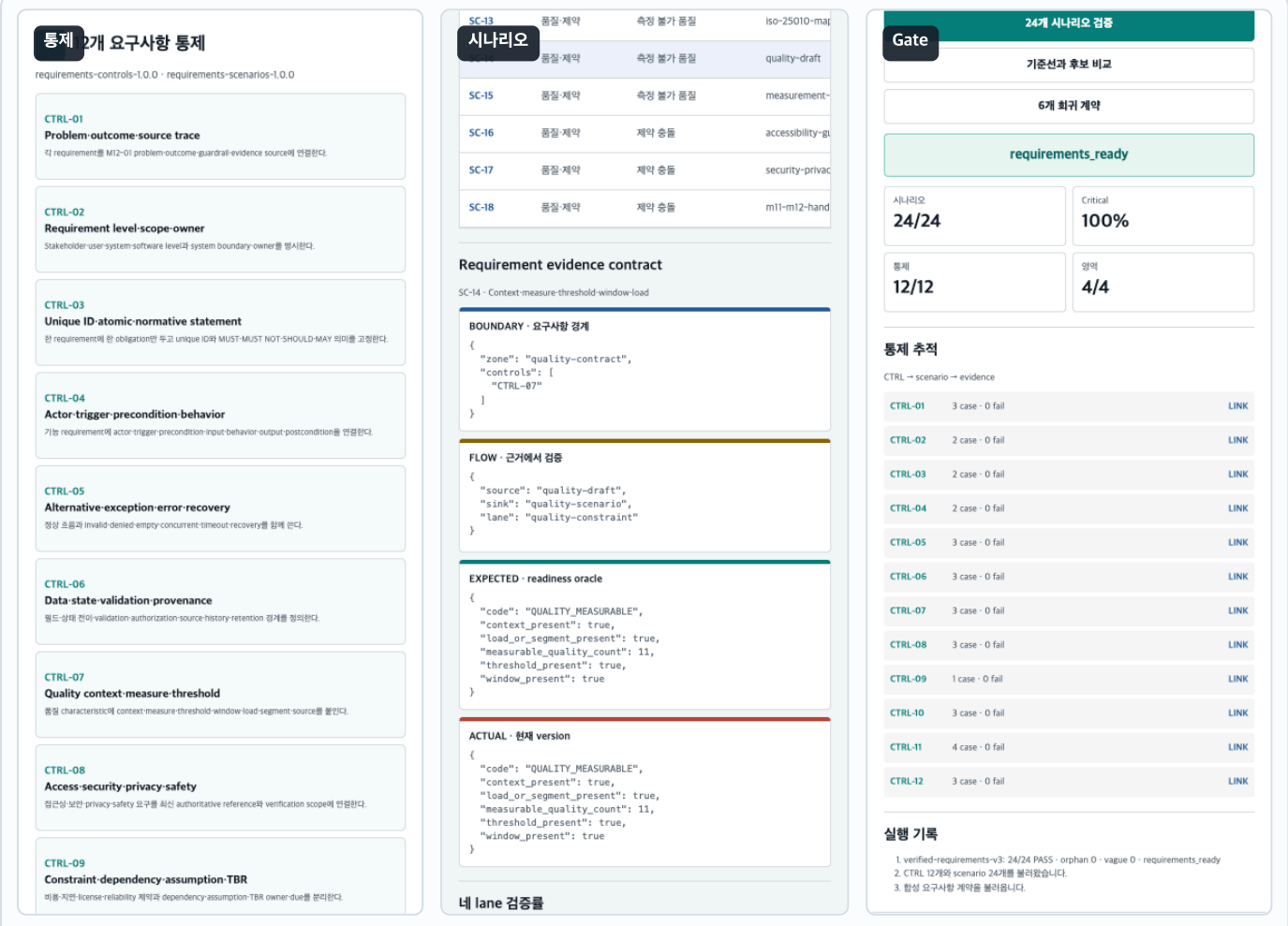


그림 17. 모바일에서도 control·lane filter·evidence·Gate가 순서대로 읽히며 page-level 가로 넘침은 없습니다.

18.4 화면에서 꼭 해 볼 다섯 행동

1. `feature-list-v1`을 실행하고 18개 FAIL의 expected·actual 차이를 봅니다.
2. `functional-only-v2`에서 품질·수용·DoD 9개 FAIL이 남는 이유를 읽습니다.
3. 네 lane filter를 눌러 각각 6개 scenario인지 확인합니다.
4. `SC-14`, `SC-16`, `SC-20`, `SC-24`의 evidence contract를 엽니다.
5. 기준선 비교의 `fixed 18·regressed 0`과 회귀 `6/6 PASS`를 확인합니다.

---

19. 열두 control로 요구사항을 검수하기

Control은 문서 장식이 아니라 어떤 실패를 막고 무엇을 evidence로 남길지 정한 검토 계약입니다.

CTRL-01 · Problem·outcome·source trace

각 requirement를 M12-01 problem·outcome·guardrail·evidence source에 연결한다.

- 최소 evidence: `Outcome-to-requirement trace`
- 작성 질문: 이 control이 요구하는 ID·source·condition·owner는 어디에 있는가?
- 실패 신호: 기능명·형용사·정상 흐름·PASS 배지만 있고 실제 계약 field가 없는가?
- 교정 행동: obligation을 나누고 acceptance·verification·limitation을 연결합니다.
- 보호 질문: 빠지면 어떤 사용자·품질·권한·데이터·운영 위험이 커지는가?
- 변경 질문: 새 evidence가 나오면 어떤 requirement와 test version을 함께 올리는가?
- M12-03 전달: scope·priority 판단에 필요한 trace·constraint·TBR를 표시합니다.

CTRL-02 · Requirement level·scope·owner

Stakeholder·user·system·software level과 system boundary·owner를 명시한다.

- 최소 evidence: `Requirement level map`
- 작성 질문: 이 control이 요구하는 ID·source·condition·owner는 어디에 있는가?
- 실패 신호: 기능명·형용사·정상 흐름·PASS 배지만 있고 실제 계약 field가 없는가?
- 교정 행동: obligation을 나누고 acceptance·verification·limitation을 연결합니다.
- 보호 질문: 빠지면 어떤 사용자·품질·권한·데이터·운영 위험이 커지는가?
- 변경 질문: 새 evidence가 나오면 어떤 requirement와 test version을 함께 올리는가?
- M12-03 전달: scope·priority 판단에 필요한 trace·constraint·TBR를 표시합니다.

CTRL-03 · Unique ID·atomic·normative statement

한 requirement에 한 obligation만 두고 unique ID와 MUST·MUST NOT·SHOULD·MAY 의미를 고정한다.

- 최소 evidence: `Atomic requirement register`
- 작성 질문: 이 control이 요구하는 ID·source·condition·owner는 어디에 있는가?
- 실패 신호: 기능명·형용사·정상 흐름·PASS 배지만 있고 실제 계약 field가 없는가?
- 교정 행동: obligation을 나누고 acceptance·verification·limitation을 연결합니다.
- 보호 질문: 빠지면 어떤 사용자·품질·권한·데이터·운영 위험이 커지는가?
- 변경 질문: 새 evidence가 나오면 어떤 requirement와 test version을 함께 올리는가?
- M12-03 전달: scope·priority 판단에 필요한 trace·constraint·TBR를 표시합니다.

CTRL-04 · Actor·trigger·precondition·behavior

기능 requirement에 actor·trigger·precondition·input·behavior·output·postcondition을 연결한다.

- 최소 evidence: `Functional behavior contract`
- 작성 질문: 이 control이 요구하는 ID·source·condition·owner는 어디에 있는가?
- 실패 신호: 기능명·형용사·정상 흐름·PASS 배지만 있고 실제 계약 field가 없는가?
- 교정 행동: obligation을 나누고 acceptance·verification·limitation을 연결합니다.
- 보호 질문: 빠지면 어떤 사용자·품질·권한·데이터·운영 위험이 커지는가?
- 변경 질문: 새 evidence가 나오면 어떤 requirement와 test version을 함께 올리는가?
- M12-03 전달: scope·priority 판단에 필요한 trace·constraint·TBR를 표시합니다.

CTRL-05 · Alternative·exception·error·recovery

정상 흐름과 invalid·denied·empty·concurrent·timeout·recovery를 함께 쓴다.

- 최소 evidence: `Exception and recovery map`
- 작성 질문: 이 control이 요구하는 ID·source·condition·owner는 어디에 있는가?
- 실패 신호: 기능명·형용사·정상 흐름·PASS 배지만 있고 실제 계약 field가 없는가?
- 교정 행동: obligation을 나누고 acceptance·verification·limitation을 연결합니다.
- 보호 질문: 빠지면 어떤 사용자·품질·권한·데이터·운영 위험이 커지는가?
- 변경 질문: 새 evidence가 나오면 어떤 requirement와 test version을 함께 올리는가?
- M12-03 전달: scope·priority 판단에 필요한 trace·constraint·TBR를 표시합니다.

CTRL-06 · Data·state·validation·provenance

필드·상태 전이·validation·authorization·source·history·retention 경계를 정의한다.

- 최소 evidence: `Data and state contract`
- 작성 질문: 이 control이 요구하는 ID·source·condition·owner는 어디에 있는가?
- 실패 신호: 기능명·형용사·정상 흐름·PASS 배지만 있고 실제 계약 field가 없는가?
- 교정 행동: obligation을 나누고 acceptance·verification·limitation을 연결합니다.
- 보호 질문: 빠지면 어떤 사용자·품질·권한·데이터·운영 위험이 커지는가?
- 변경 질문: 새 evidence가 나오면 어떤 requirement와 test version을 함께 올리는가?
- M12-03 전달: scope·priority 판단에 필요한 trace·constraint·TBR를 표시합니다.

CTRL-07 · Quality context·measure·threshold

품질 characteristic에 context·measure·threshold·window·load·segment·source를 붙인다.

- 최소 evidence: `Quality attribute scenario`
- 작성 질문: 이 control이 요구하는 ID·source·condition·owner는 어디에 있는가?
- 실패 신호: 기능명·형용사·정상 흐름·PASS 배지만 있고 실제 계약 field가 없는가?
- 교정 행동: obligation을 나누고 acceptance·verification·limitation을 연결합니다.
- 보호 질문: 빠지면 어떤 사용자·품질·권한·데이터·운영 위험이 커지는가?
- 변경 질문: 새 evidence가 나오면 어떤 requirement와 test version을 함께 올리는가?
- M12-03 전달: scope·priority 판단에 필요한 trace·constraint·TBR를 표시합니다.

CTRL-08 · Access·security·privacy·safety

접근성·보안·privacy·safety 요구를 최신 authoritative reference와 verification scope에 연결한다.

- 최소 evidence: `Assurance requirement register`
- 작성 질문: 이 control이 요구하는 ID·source·condition·owner는 어디에 있는가?
- 실패 신호: 기능명·형용사·정상 흐름·PASS 배지만 있고 실제 계약 field가 없는가?
- 교정 행동: obligation을 나누고 acceptance·verification·limitation을 연결합니다.
- 보호 질문: 빠지면 어떤 사용자·품질·권한·데이터·운영 위험이 커지는가?
- 변경 질문: 새 evidence가 나오면 어떤 requirement와 test version을 함께 올리는가?
- M12-03 전달: scope·priority 판단에 필요한 trace·constraint·TBR를 표시합니다.

CTRL-09 · Constraint·dependency·assumption·TBR

비용·지연·license·reliability 제약과 dependency·assumption·TBR owner·due를 분리한다.

- 최소 evidence: `Constraint and TBR register`
- 작성 질문: 이 control이 요구하는 ID·source·condition·owner는 어디에 있는가?

- 실패 신호: 기능명·형용사·정상 흐름·PASS 배지만 있고 실제 계약 field가 없는가?
- 교정 행동: obligation을 나누고 acceptance·verification·limitation을 연결합니다.
- 보호 질문: 빠지면 어떤 사용자·품질·권한·데이터·운영 위험이 커지는가?
- 변경 질문: 새 evidence가 나오면 어떤 requirement와 test version을 함께 올리는가?
- M12-03 전달: scope·priority 판단에 필요한 trace·constraint·TBR를 표시합니다.

CTRL-10 · Observable acceptance criteria

Given·When·Then 또는 동등한 형식으로 positive·negative·boundary 결과를 사용자·외부 system에서 관찰 가능하게 쓴다.

- 최소 evidence: `Acceptance criteria set`
- 작성 질문: 이 control이 요구하는 ID·source·condition·owner는 어디에 있는가?
- 실패 신호: 기능명·형용사·정상 흐름·PASS 배지만 있고 실제 계약 field가 없는가?
- 교정 행동: obligation을 나누고 acceptance·verification·limitation을 연결합니다.
- 보호 질문: 빠지면 어떤 사용자·품질·권한·데이터·운영 위험이 커지는가?
- 변경 질문: 새 evidence가 나오면 어떤 requirement와 test version을 함께 올리는가?
- M12-03 전달: scope·priority 판단에 필요한 trace·constraint·TBR를 표시합니다.

CTRL-11 · Quality review·conflict·feasibility

명확성·완전성·일관성·feasibility·필요성·중복·conflict·priority를 set 전체에서 검토한다.

- 최소 evidence: `Requirements quality review`
- 작성 질문: 이 control이 요구하는 ID·source·condition·owner는 어디에 있는가?
- 실패 신호: 기능명·형용사·정상 흐름·PASS 배지만 있고 실제 계약 field가 없는가?
- 교정 행동: obligation을 나누고 acceptance·verification·limitation을 연결합니다.
- 보호 질문: 빠지면 어떤 사용자·품질·권한·데이터·운영 위험이 커지는가?
- 변경 질문: 새 evidence가 나오면 어떤 requirement와 test version을 함께 올리는가?
- M12-03 전달: scope·priority 판단에 필요한 trace·constraint·TBR를 표시합니다.

CTRL-12 · Version·baseline·verification·DoD·handoff

Version·source·rationale·verification method·evidence·Definition of Done·Gate·M12-03 handoff를 남긴다.

- 최소 evidence: `Requirement Readiness Gate`
- 작성 질문: 이 control이 요구하는 ID·source·condition·owner는 어디에 있는가?
- 실패 신호: 기능명·형용사·정상 흐름·PASS 배지만 있고 실제 계약 field가 없는가?

- 교정 행동: obligation을 나누고 acceptance·verification·limitation을 연결합니다.
- 보호 질문: 빠지면 어떤 사용자·품질·권한·데이터·운영 위험이 커지는가?
- 변경 질문: 새 evidence가 나오면 어떤 requirement와 test version을 함께 올리는가?
- M12-03 전달: scope·priority 판단에 필요한 trace·constraint·TBR를 표시합니다.

20. 24개 scenario를 네 lane으로 검증하기

각 scenario는 구현 테스트 하나가 아니라 requirement set의 계약 빈칸을 찾는 검토 질문입니다. Candidate는 24/24, critical 100%, control·lane coverage 100%를 만족해야 합니다.

Lane A · 의도·추적

M12-01 source에서 requirement level·scope·원자성·필요성까지 연결합니다.

SC-01 · Problem·outcome·guardrail에서 requirement 추적

- Lane: `intent-trace` · Risk: `orphan-requirement` · Priority: `P1`
- Control: `CTRL-01` · Trust zone: `problem-outcome`
- Evidence path: `m12-01-handoff → trace-matrix`
- Expected contract:
  - ☐ `code` = `INTENT_TRACE_COMPLETE`
  - ☐ `problem_linked` = `True`
  - ☐ `outcome_linked` = `True`
  - ☐ `guardrail_linked` = `True`
  - ☐ `source_version_present` = `True`
- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-02 · Stakeholder·user·system·software level 구분

- Lane: `intent-trace` · Risk: `orphan-requirement` · Priority: `P1`
- Control: `CTRL-02` · Trust zone: `problem-outcome`
- Evidence path: `actor-outcome-map → level-map`
- Expected contract:

    - [ ] `code` = `REQUIREMENT_LEVELS_MAPPED`
    - [ ] `stakeholder_level` = `True`
    - [ ] `user_level` = `True`
    - [ ] `system_level` = `True`
    - [ ] `software_level` = `True`
    - [ ] `flowdown_visible` = `True`

- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-03 · System boundary·external actor·owner·out of scope

- Lane: `intent-trace` · Risk: `orphan-requirement` · Priority: `P1`
- Control: `CTRL-02` · Trust zone: `problem-outcome`
- Evidence path: `service-boundary → scope-record`
- Expected contract:
    - [ ] `code` = `SCOPE_OWNER_DEFINED`
    - [ ] `system_boundary_present` = `True`
    - [ ] `external_actors_present` = `True`
    - [ ] `owner_present` = `True`
    - [ ] `out_of_scope_present` = `True`

- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-04 · Unique ID·한 obligation·normative keyword

- Lane: `intent-trace` · Risk: `ambiguity-compound` · Priority: `P0`
- Control: `CTRL-03` · Trust zone: `requirement-specification`
- Evidence path: `draft-requirements → atomic-register`
- Expected contract:
    - [ ] `code` = `ATOMIC_REQUIREMENTS`
    - [ ] `unique_ids` = `True`
    - [ ] `one_obligation_each` = `True`

- [ ] `normative_terms_defined` = `True`
- [ ] `rationale_separate` = `True`

- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-05 · Vague term·pronoun·and/or·etc 제거

- Lane: `intent-trace` · Risk: `ambiguity-compound` · Priority: `P1`
- Control: `CTRL-03, CTRL-11` · Trust zone: `requirement-specification`
- Evidence path: `language-lint → clarity-review`
- Expected contract:
  - [ ] `code` = `AMBIGUITY_REMOVED`
  - [ ] `vague_term_count` = `0`
  - [ ] `indefinite_pronouns` = `0`
  - [ ] `and_or_count` = `0`
  - [ ] `etc_count` = `0`

- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-06 · Requirement마다 source·rationale·owner·necessity

- Lane: `intent-trace` · Risk: `orphan-requirement` · Priority: `P1`
- Control: `CTRL-01, CTRL-11` · Trust zone: `problem-outcome`
- Evidence path: `requirement-register → necessity-review`
- Expected contract:
  - [ ] `code` = `REQUIREMENTS_NECESSARY`
  - [ ] `orphan_count` = `0`
  - [ ] `rationale_complete` = `True`
  - [ ] `owner_complete` = `True`
  - [ ] `source_complete` = `True`

- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?

- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

Lane B · 기능·행동

Actor·정상·대안·예외·복구·데이터·상태 경계를 검증합니다.

SC-07 · Actor·trigger·precondition·input·behavior·output

- Lane: `functional-behavior` · Risk: `missing-exception` · Priority: `P1`
- Control: `CTRL-04` · Trust zone: `requirement-specification`
- Evidence path: `user-scenario → functional-contract`
- Expected contract:
  - ☐ `code` = `FUNCTIONAL_CONTRACT_COMPLETE`
  - ☐ `actor_present` = `True`
  - ☐ `trigger_present` = `True`
  - ☐ `precondition_present` = `True`
  - ☐ `input_output_present` = `True`
  - ☐ `postcondition_present` = `True`
- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-08 · 정상 flow와 alternative flow

- Lane: `functional-behavior` · Risk: `missing-exception` · Priority: `P1`
- Control: `CTRL-04, CTRL-05` · Trust zone: `requirement-specification`
- Evidence path: `behavior-model → flow-map`
- Expected contract:
  - ☐ `code` = `ALTERNATIVE_FLOWS_DEFINED`
  - ☐ `happy_path_present` = `True`
  - ☐ `decline_present` = `True`
  - ☐ `correction_present` = `True`
  - ☐ `cancel_present` = `True`
- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?

- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-09 · Invalid·empty·denied·timeout 처리

- Lane: `functional-behavior` · Risk: `missing-exception` · Priority: `P1`
- Control: `CTRL-05` · Trust zone: `requirement-specification`
- Evidence path: `error-taxonomy → exception-contract`
- Expected contract:
  - ☐ `code` = `ERROR_CONTRACT_COMPLETE`
  - ☐ `invalid_present` = `True`
  - ☐ `empty_present` = `True`
  - ☐ `denied_present` = `True`
  - ☐ `timeout_present` = `True`
  - ☐ `observable_error_present` = `True`
- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-10 · Concurrent update·idempotency·retry·recovery

- Lane: `functional-behavior` · Risk: `missing-exception` · Priority: `P0`
- Control: `CTRL-05, CTRL-06` · Trust zone: `requirement-specification`
- Evidence path: `state-transitions → recovery-contract`
- Expected contract:
  - ☐ `code` = `CONCURRENCY_RECOVERY_DEFINED`
  - ☐ `silent_overwrite_forbidden` = `True`
  - ☐ `version_check_present` = `True`
  - ☐ `idempotency_present` = `True`
  - ☐ `retry_boundary_present` = `True`
  - ☐ `recovery_present` = `True`
- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-11 · Field·validation·state transition·history

- Lane: `functional-behavior` · Risk: `ambiguity-compound` · Priority: `P1`
- Control: `CTRL-06` · Trust zone: `requirement-specification`
- Evidence path: `data-dictionary → state-contract`
- Expected contract:
  - ☐ `code` = `DATA_STATE_CONTRACT_COMPLETE`
  - ☐ `required_fields_present` = `True`
  - ☐ `validation_rules_present` = `True`
  - ☐ `state_transitions_present` = `True`
  - ☐ `history_present` = `True`
- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-12 · Authorization·source·retention 경계

- Lane: `functional-behavior` · Risk: `constraint-conflict` · Priority: `P0`
- Control: `CTRL-06, CTRL-08` · Trust zone: `requirement-specification`
- Evidence path: `data-access-map → data-boundary`
- Expected contract:
  - ☐ `code` = `DATA_BOUNDARY_DEFINED`
  - ☐ `server_authorization_present` = `True`
  - ☐ `source_provenance_present` = `True`
  - ☐ `retention_present` = `True`
  - ☐ `export_boundary_present` = `True`
- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

Lane C · 품질·제약

품질 모델·measurement·접근성·보안·privacy·constraint를 검증합니다.

SC-13 · Product quality characteristic coverage

- Lane: `quality-constraint` · Risk: `unmeasurable-quality` · Priority: `P1`
- Control: `CTRL-07` · Trust zone: `quality-contract`
- Evidence path: `iso-25010-map → quality-register`
- Expected contract:
  - ☐ `code` = `QUALITY_MODEL_COVERED`
  - ☐ `functional_suitability` = `True`
  - ☐ `performance_efficiency` = `True`
  - ☐ `interaction_capability` = `True`
  - ☐ `reliability` = `True`
  - ☐ `security` = `True`
  - ☐ `maintainability_or_flexibility` = `True`
- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-14 · Context·measure·threshold·window·load

- Lane: `quality-constraint` · Risk: `unmeasurable-quality` · Priority: `P1`
- Control: `CTRL-07` · Trust zone: `quality-contract`
- Evidence path: `quality-draft → quality-scenario`
- Expected contract:
  - ☐ `code` = `QUALITY_MEASURABLE`
  - ☐ `measurable_quality_count` = `11`
  - ☐ `context_present` = `True`
  - ☐ `threshold_present` = `True`
  - ☐ `window_present` = `True`
  - ☐ `load_or_segment_present` = `True`
- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-15 · Percentile·denominator·segment·measurement source

- Lane: `quality-constraint` · Risk: `unmeasurable-quality` · Priority: `P1`
- Control: `CTRL-07` · Trust zone: `quality-contract`
- Evidence path: `measurement-plan → measure-contract`
- Expected contract:
  - ☐ `code` = `MEASUREMENT_CONTRACT_COMPLETE`
  - ☐ `percentile_when_needed` = `True`
  - ☐ `denominator_present` = `True`
  - ☐ `segment_present` = `True`
  - ☐ `measurement_source_present` = `True`
  - ☐ `owner_present` = `True`
- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-16 · WCAG·keyboard·name role value·human evaluation

- Lane: `quality-constraint` · Risk: `constraint-conflict` · Priority: `P0`
- Control: `CTRL-08` · Trust zone: `quality-contract`
- Evidence path: `accessibility-guardrail → access-requirement`
- Expected contract:
  - ☐ `code` = `ACCESS_REQUIREMENT_VERIFIABLE`
  - ☐ `wcag_22_scope_present` = `True`
  - ☐ `keyboard_present` = `True`
  - ☐ `name_role_value_present` = `True`
  - ☐ `human_evaluation_present` = `True`
  - ☐ `no_group_worse_linked` = `True`
- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-17 · Security·privacy·least privilege·safe logging

- Lane: `quality-constraint` · Risk: `constraint-conflict` · Priority: `P0`

- Control: `CTRL-08` · Trust zone: `quality-contract`
- Evidence path: `security-privacy-guardrail → assurance-requirement`
- Expected contract:
  - ☐ `code` = `ASSURANCE_REQUIREMENTS_DEFINED`
  - ☐ `server_authorization_present` = `True`
  - ☐ `least_privilege_present` = `True`
  - ☐ `data_minimization_present` = `True`
  - ☐ `raw_content_logging_forbidden` = `True`
  - ☐ `asvs_ssdf_reference_present` = `True`
- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-18 · Cost·latency·license·reliability·dependency·TBR

- Lane: `quality-constraint` · Risk: `constraint-conflict` · Priority: `P0`
- Control: `CTRL-09` · Trust zone: `quality-contract`
- Evidence path: `m11-m12-handoff → constraint-register`
- Expected contract:
  - ☐ `code` = `CONSTRAINTS_GOVERNED`
  - ☐ `cost_present` = `True`
  - ☐ `latency_present` = `True`
  - ☐ `license_present` = `True`
  - ☐ `reliability_present` = `True`
  - ☐ `dependency_present` = `True`
  - ☐ `open_tbr_count` = `2`
- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

Lane D · 수용·거버넌스

Acceptance coverage·verification·review·baseline·DoD·handoff를 검증합니다.

SC-19 · Given·When·Then observable outcome

- Lane: `acceptance-governance` · Risk: `false-completion` · Priority: `P1`
- Control: `CTRL-10` · Trust zone: `acceptance-evidence`
- Evidence path: `requirement-records → acceptance-set`
- Expected contract:
  - ☐ `code` = `ACCEPTANCE_OBSERVABLE`
  - ☐ `given_context_present` = `True`
  - ☐ `when_event_present` = `True`
  - ☐ `then_observable_present` = `True`
  - ☐ `implementation_detail_avoided` = `True`
- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-20 · Positive·negative·boundary·authorization criteria

- Lane: `acceptance-governance` · Risk: `false-completion` · Priority: `P1`
- Control: `CTRL-10` · Trust zone: `acceptance-evidence`
- Evidence path: `acceptance-set → coverage-map`
- Expected contract:
  - ☐ `code` = `ACCEPTANCE_COVERAGE_COMPLETE`
  - ☐ `acceptance_coverage` = `1.0`
  - ☐ `positive_present` = `True`
  - ☐ `negative_present` = `True`
  - ☐ `boundary_present` = `True`
  - ☐ `authorization_present` = `True`
- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-21 · Verification method·environment·evidence·owner

- Lane: `acceptance-governance` · Risk: `false-completion` · Priority: `P1`
- Control: `CTRL-10, CTRL-12` · Trust zone: `acceptance-evidence`

- Evidence path: `verification-plan → evidence-plan`
- Expected contract:
  - ☐ `code` = `VERIFICATION_EVIDENCE_PLANNED`
  - ☐ `method_present` = `True`
  - ☐ `environment_present` = `True`
  - ☐ `evidence_present` = `True`
  - ☐ `owner_present` = `True`
  - ☐ `reproducible` = `True`
- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-22 · Conflict·duplicate·feasibility·priority negotiation

- Lane: `acceptance-governance` · Risk: `constraint-conflict` · Priority: `P1`
- Control: `CTRL-11` · Trust zone: `decision-baseline`
- Evidence path: `quality-review → decision-log`
- Expected contract:
  - ☐ `code` = `REQUIREMENT_SET_CONSISTENT`
  - ☐ `conflict_count` = `0`
  - ☐ `duplicate_count` = `0`
  - ☐ `feasibility_reviewed` = `True`
  - ☐ `priority_present` = `True`
  - ☐ `decision_owner_present` = `True`
- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-23 · Version·baseline·change impact·Definition of Done

- Lane: `acceptance-governance` · Risk: `false-completion` · Priority: `P1`
- Control: `CTRL-12` · Trust zone: `decision-baseline`
- Evidence path: `requirement-baseline → done-contract`
- Expected contract:

- [ ] `code` = `BASELINE_DOD_DEFINED`
- [ ] `version_present` = `True`
- [ ] `baseline_present` = `True`
- [ ] `change_impact_present` = `True`
- [ ] `definition_of_done_present` = `True`
- [ ] `organizational_floor_present` = `True`

- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

SC-24 · Requirement Gate·residual risk·M12-03 handoff

- Lane: `acceptance-governance` · Risk: `false-completion` · Priority: `P0`
- Control: `CTRL-01, CTRL-11, CTRL-12` · Trust zone: `decision-baseline`
- Evidence path: `requirements-evidence-pack → requirement-readiness-gate`
- Expected contract:
  - [ ] `code` = `REQUIREMENTS_READY`
  - [ ] `scenario_passed` = `24`
  - [ ] `critical_pass_rate` = `1.0`
  - [ ] `orphan_count` = `0`
  - [ ] `vague_term_count` = `0`
  - [ ] `acceptance_coverage` = `1.0`
  - [ ] `m12_03_handoff_present` = `True`
  - [ ] `residual_risk_approved` = `True`

- 확인: expected와 actual이 같은가, 다르면 어느 source·requirement·test를 고칠 것인가?
- 증거: source·rationale·owner·version·verification·limitation 여섯 필드를 남겼는가?
- 회고: 통과 이유와 아직 남은 residual risk를 각각 한 문장으로 씁니다.

21. 합성 requirement set 19개 읽기

아래 문장은 완성품이 아니라 학습용 후보입니다. 각 행을 원래 problem·outcome과 acceptance evidence까지 역추적합니다.

21.1 기능 요구사항 8개

ID	제목	Priority	Verification	Source
FR-01	실행 기록 생성	MUST	demonstration	M12-01 problem·journey
FR-02	필수값 검증과 draft 보존	MUST	test	M12-01 completeness baseline
FR-03	담당·기한 확인	MUST	demonstration	M12-01 largest uncertainty
FR-04	확인 상태 표시	MUST	inspection	M12-01 beneficiary outcome
FR-05	권한 있는 변경	MUST	test	M12-01 recovery journey
FR-06	조회와 filter	SHOULD	demonstration	M12-01 current alternatives
FR-07	동시 변경 충돌 복구	MUST	test	M12-01 quality guardrail
FR-08	접근 가능한 export	MAY	test	M12-01 access·handoff

21.2 품질·사용 품질 요구사항 11개

ID	제목	Priority	Verification	Outcome
QIU-01	완전 기록률	MUST	analysis	완전 기록률 58%→85%
QIU-02	재입력 시간	MUST	analysis	재입력 14분→5분
QIU-03	오배정률	MUST	analysis	정확성 유지
QR-01	응답 성능	MUST	test	시간 guardrail
QR-02	접근성	MUST	inspection+test	지원 필요 사용자 no worse
QR-03	Privacy 최소화	MUST	inspection+test	민감정보 사고 0
QR-04	Authorization	MUST	test	무단 접근 방지
QR-05	Availability	MUST	analysis	업무 지속
QR-06	Recovery	MUST	test	기록 복구
QR-07	Auditability	MUST	inspection+analysis	변경 설명 가능
QR-08	Cost efficiency	SHOULD	analysis	지속 가능한 단위 비용

21.3 Constraint 4개

ID	제목	합성 경계
C-01	License	사용 tool·library·model·data는 배포·학습·export 방식에 허용되는 정확한 license·contract evidence를 가져야 한다.
C-02	Meeting time	확인 flow는 회의 자체를 3분 초과 연장하도록 요구하지 않는다.
C-03	Sensitive data	실제 적용 전 data classification·retention·access owner를 승인한다.
C-04	Dependency	회의 source identifier와 조직 identity provider availability에 의존한다.

21.4 FR-02를 끝까지 추적한 예

```
Source: M12-01 completeness baseline 58%
Outcome: 누락 감소·완전 기록률 85%
FR-02: 필수값 invalid이면 ready 거부 + 이유 표시 + draft 보존
Acceptance negative: owner가 없을 때 ready 상태가 되지 않음
Acceptance boundary: due가 허용 경계와 같은 값일 때 명시된 결과
Verification: repeatable test
Evidence: version·input·expected·actual·result·limitation
DoD: trace·exception·test·access review·change log complete
```

22. Definition of Done과 Requirement Gate

22.1 특정 acceptance와 공통 DoD를 분리합니다

구분	질문	합성 예
Requirement acceptance	FR-02의 결과가 맞나	ready 거부·이유·draft 보존
Quality acceptance	QR-01 임곗값을 지키나	50 users·30분·p95 ≤2초
Product-specific Done	이 변경에 필요한 위험 검토를 했나	authorization·access·history
Organizational DoD	모든 증분의 공통 floor를 지켰나	review·test·security·docs·evidence

22.2 Gate 수치

```
scenario 24/24              critical 100%
lane 4/4                    control 12/12
requirement 19              quality measurable 11
trace 100%                  acceptance 100%
orphan 0                    vague 0
atomicity violation 0       conflict 0
open TBR ≤ 2                DoD complete
fixed 18                    regressed 0
```

22.3 판정의 안전 경계

`requirements_ready`는 이 합성 학습 계약이 다음 scope·priority 논의에 충분하다는 뜻입니다. 실제 계약 인수, 보안·접근성 certification, 법률·정책 승인, production readiness, 제품 투자나 MVP 구축 승인이 아닙니다.

---

23. M12-03 MVP 범위·우선순위로 넘기기

23.1 넘기는 묶음

1. Requirement baseline version과 change log
2. Problem·outcome·guardrail·source trace
3. Functional 8·quality 11·constraint 4
4. Positive·negative·boundary·authorization acceptance
5. Verification method·artifact·limitation
6. Dependency·TBR owner·due·impact
7. Out-of-scope·assumption·residual risk
8. Definition of Done와 Gate decision record

23.2 넘기지 않는 결정

- 모든 MUST가 MVP 첫 release에 들어간다는 결정
- `requirements_ready`가 시장 수요를 증명했다는 주장
- WCAG·ASVS를 적었으므로 인증됐다는 주장

- 합성 비용·성능 숫자를 production 약속으로 쓰는 결정
- 잔여 위험 owner 승인 없이 scope를 확정하는 결정

---

24. 셀프 테스트 30

먼저 답을 가리고 60초 안에 말합니다. 답에는 합성 requirement ID 또는 scenario ID 하나를 붙입니다.

Q01. 기능 목록과 요구사항 집합의 가장 큰 차이는 무엇인가요?

모범 답안 보기
요구사항 집합은 각 항목의 문제·결과 source, 범위, 조건, 행동·품질, 수용 기준, 검증 방법, owner와 version을 연결합니다. 기능 목록은 보통 이름과 아이디어만 있어 존재 이유와 완료 판정이 약합니다.

Q02. Functional requirement와 quality requirement를 같은 문장에 섞지 않는 이유는 무엇인가요?

모범 답안 보기
행동의 존재 여부와 행동의 품질 임곗값은 검증 조건과 변경 이유가 다릅니다. 분리해야 한쪽만 충족된 상태를 발견하고 trade-off를 관리할 수 있습니다.

Q03. Requirement level 네 가지를 쓰세요.

모범 답안 보기
Stakeholder, user, system, software level입니다. 각 수준을 flow-down하고 하위에서 상위 결과로 trace-up합니다.

Q04. Orphan requirement란 무엇인가요?

모범 답안 보기
M12-01 problem·outcome·guardrail·source 또는 후속 설계·검증 증거와 연결되지 않은 요구입니다.

Q05. 원자적 요구사항이 필요한 이유는 무엇인가요?

모범 답안 보기

한 문장에 한 obligation만 있어야 부분 통과를 숨기지 않고 우선순위·owner·변경·시험을 정확히 연결할 수 있습니다.

Q06. MUST와 SHOULD의 차이는 무엇인가요?

모범 답안 보기

MUST는 정의된 범위에서 반드시 충족해야 합니다. SHOULD는 강한 권고지만 명시적 근거와 승인된 예외가 가능하며 예외를 기록해야 합니다.

Q07. 요구사항에서 피해야 할 모호어 네 가지를 쓰세요.

모범 답안 보기

빠르게, 쉽게, 적절히, 가능한 한, 최대한, 안정적으로, 사용자 친화적, 등, and/or 중 네 가지 이상입니다.

Q08. 기능 행동 계약의 일곱 요소를 쓰세요.

모범 답안 보기

Actor, trigger, precondition, input, behavior, output, postcondition입니다.

Q09. 정상 흐름 외에 반드시 고려할 실패 조건 네 가지는 무엇인가요?

모범 답안 보기

Invalid, empty, denied, timeout, concurrent update 중 네 가지이며 오류 표시·데이터 보존·복구도 함께 씁니다.

Q10. Silent overwrite를 왜 금지해야 하나요?

모범 답안 보기

동시 변경에서 한 사용자의 의도와 기록을 알림 없이 잃게 해 데이터 무결성과 책임 추적을 깨기 때문입니다.

Q11. 상태 전이 화살표에 필요한 정보는 무엇인가요?

모범 답안 보기

Actor, trigger, precondition 또는 guard, validation, authorization, poststate, failure·recovery, audit evidence입니다.

Q12. 비기능 요구사항을 '빠르게'라고만 쓰면 왜 검증할 수 없나요?

모범 답안 보기

Context·measure·threshold·window·load·segment가 없어 어떤 환경에서 무엇을 재고 어느 값이면 통과인지 알 수 없기 때문입니다.

Q13. 품질 속성 시나리오의 최소 여섯 요소를 쓰세요.

모범 답안 보기

Context, stimulus, measure, threshold, window, load 또는 segment입니다. Source와 owner도 붙입니다.

Q14. ISO/IEC 25010:2023의 아홉 제품 품질 특성은 무엇인가요?

모범 답안 보기

기능 적합성, 성능 효율성, 호환성, 상호작용 능력, 신뢰성, 보안, 유지보수성, 유연성, 안전성입니다.

Q15. 모든 품질 특성을 MUST로 만들면 안 되는 이유는 무엇인가요?

모범 답안 보기

문제·risk·사용 맥락과 무관한 요구가 늘어 conflict와 비용이 커집니다. 필요한 특성과 제외 근거를 추적해야 합니다.

Q16. 접근성 요구에 WCAG 2.2만 적으면 왜 부족한가요?

모범 답안 보기

적용할 core flow와 success criteria, level, keyboard, name·role·value, 자동·사람·보조기술 검증 범위가 없으면 실제 완료를 판정할 수 없습니다.

Q17. 보안 요구에서 client-side 검사만으로 부족한 이유는 무엇인가요?

모범 답안 보기

요청자는 client를 우회할 수 있으므로 조직·record·action별 authorization을 server-side에서 deny by default로 확인해야 합니다.

Q18. Privacy 요구의 핵심 두 원칙을 쓰세요.

모범 답안 보기

목적 제한과 데이터 최소화입니다. Retention·deletion·access·logging boundary도 검증 가능하게 씁니다.

Q19. TBR에 반드시 붙여야 할 세 필드는 무엇인가요?

모범 답안 보기

Owner, due date, unresolved impact입니다. 그때까지 사용할 temporary boundary도 기록하면 좋습니다.

Q20. Constraint와 quality requirement의 차이는 무엇인가요?

모범 답안 보기

Quality requirement는 제품이나 사용의 바람직한 품질 반응과 임곗값을 정하고, constraint는 설계·운영·계약 선택이 넘지 말아야 할 경계를 정합니다.

Q21. Given·When·Then은 각각 무엇을 뜻하나요?

모범 답안 보기

Given은 초기 맥락·권한·상태, When은 행동을 시작시키는 사건·입력, Then은 외부에서 관찰 가능한 결과입니다.

Q22. Acceptance coverage의 네 방향을 쓰세요.

모범 답안 보기

Positive, negative, boundary, authorization입니다. 중요 risk와 user segment도 연결합니다.

Q23. Acceptance criterion과 Definition of Done의 차이는 무엇인가요?

모범 답안 보기

Acceptance criterion은 특정 requirement의 관찰 가능한 결과이고 DoD는 작업·증분 전체에 공통으로 적용하는 품질·검증·증거 기준입니다.

Q24. Verification과 validation의 차이는 무엇인가요?

모범 답안 보기

Verification은 명시된 요구대로 만들었는지 확인하고 validation은 의도한 사용자 필요와 사용 맥락을 충족하는지 확인합니다.

Q25. 네 가지 대표 verification method를 쓰세요.

모범 답안 보기

Inspection, analysis, demonstration, test입니다.

Q26. 좋은 검증 증거의 최소 필드는 무엇인가요?

모범 답안 보기

Requirement ID, method, environment·version, expected, actual, result, owner, artifact link, limitation입니다.

Q27. 요구 문장 각각은 좋아도 set 전체 검토가 필요한 이유는 무엇인가요?

모범 답안 보기

중복·threshold·scope·용어·priority·dependency conflict는 여러 요구를 함께 봐야 발견할 수 있기 때문입니다.

Q28. M12-02 Gate의 합성 후보 핵심 수치를 쓰세요.

모범 답안 보기

24/24 scenario, critical 100%, lane·control coverage 100%, requirement 19, quality 11, orphan·vague·atomicity·conflict 0, acceptance 100%, open TBR 2 이하입니다.

Q29. `requirements_ready`가 의미하지 않는 것을 세 가지 쓰세요.

모범 답안 보기
계약 인수, 보안 인증, 접근성 인증, 법률·정책 승인, 제품 승인, MVP 구축 결정 중 세 가지 이상입니다.

Q30. M12-03에 넘기는 최소 묶음은 무엇인가요?

모범 답안 보기
Versioned requirement set, outcome trace, functional·quality requirements, constraints·dependencies, acceptance·verification evidence, DoD, TBR·residual risk, out-of-scope와 decision record입니다.

25. 공식 자료와 추가 학습

- ISO/IEC/IEEE 29148:2018 Requirements engineering
- ISO/IEC 25010:2023 Product quality model
- ISO/IEC 25019:2023 Quality-in-use model
- ISO/IEC 25030:2019 Quality requirements framework
- ISO/IEC 25040:2024 Quality evaluation framework
- RFC 2119 Key words for use in RFCs
- RFC 8174 Ambiguity of Uppercase vs Lowercase
- Scrum Guide 2020
- W3C WCAG 2.2
- OWASP ASVS 5.0.0
- NIST SP 800-218 SSDF v1.1
- Cucumber Gherkin reference
- IREB CPRE Glossary
- NASA Systems Engineering Handbook appendices
- GOV.UK Writing user stories

ISO 페이지의 판본·확인 상태와 웹 표준·보안 기준의 최신 version은 실제 적용 직전에 다시 확인합니다. 이 PDF의 reviewed date는 2026-07-16입니다.

26. 60초 최종 설명

M12-01의 문제와 결과를 source로 삼아 수준·범위·owner를 맞춘다. 기능은 actor·조건·행동·결과와 예외·복구로 나누고, 품질은 context·measure·threshold·window·segment로 쓴다. 접근성·보안·privacy·safety와 constraint·TBR를 처음부터 연결한다. 각 requirement에 positive·negative·boundary·authorization acceptance와 verification evidence를 붙이고, set 전체 conflict·DoD·version을 검토한다. 24개 scenario와 Gate가 통과하면 M12-03의 MVP scope·priority 후보로 넘기되, 이것을 계약 인수·인증·제품 승인으로 오해하지 않는다.