People Leadership in Engineering

TPBOK Domain Guide 5 of 5. Task statements, guide chapter, worked scenario, questions and glossary.

International Insignia of Certified Technical Practitioners

22 October 2026

IICTP LIBRARY · VERSION 0.2 · PUBLIC DRAFT FOR COMMENT

People Leadership in Engineering

TPBOK Domain Guide 5. The task statements, the guide chapter, worked scenario, illustrative questions and glossary for one domain of the Standard.

International Insignia of Certified Technical Practitioners (IICTP)
22 October 2026

People Leadership in Engineering
IICTP Library · Version 0.2. Public draft for comment. 22 October 2026.

© 2026 International Insignia of Certified Technical Practitioners (IICTP).

Licensed under Creative Commons Attribution-NoDerivatives 4.0 International (CC BY-ND 4.0), https://creativecommons.org/licenses/by-nd/4.0/. You may copy and share this book unchanged, for any purpose, with credit to IICTP. Modified versions, including translations, need IICTP's written permission.

This book is drawn from *The Standard for Technical Practitioners* (TPBOK™), version 0.2.

TPBOK™, IICTP, ATP® and CTP® are marks of IICTP and are not licensed by the above. IICTP examination items are confidential and are not part of this book.

Status. This is a draft, published to be criticised. Nothing in it is final. IICTP is a new independent credentialing body and is not ISO/IEC 17024 accredited. Products and organisations named are cited as examples or sources; no endorsement is implied.

Contents

ABOUT THIS GUIDE **How to use this guide** 5

PART A · DOMAIN 5 **The task statements** 7

CHAPTER 5 **People Leadership in Engineering** 8

- 5.1 Scope 8
- 5.2 Levels 8
- 5.3 Principles at work 9
- 5.4 Knowledge by task 9
- 5.5 Anti-patterns 18
- 5.6 Worked scenario 19
- 5.7 Illustrative questions 22
- 5.8 Evidence of good practice 24
- 5.9 References and further reading 25

PART A · THE PRINCIPLES **The ten principles** 27

GLOSSARY **Terms used in this domain** 30

INDEX **All 56 task statements** 31

ABOUT THIS GUIDE

How to use this guide

This guide is Chapter 5 of *The Standard for Technical Practitioners* (TPBOK™), published on its own for people who work mainly in People Leadership in Engineering. It holds the domain's 10 task statements, the full guide chapter with its worked scenario and illustrative questions, the ten principles, and a glossary of the terms used.

In the ATP® examination this domain carries **12%** of the questions. In the CTP® examination it carries **10%**.

How to read it

- Read the task statements first. They say what a practitioner does.
- Use the chapter for the knowledge behind each task, then try the worked scenario and the questions.
- Task IDs such as `D2-T08` point to other domains. The task index at the back lists all 56.

The two credential levels

Every task statement carries a level tag. The tag says which exam may test it and how deeply.

Tag	Exam	What the candidate must show
A	ATP® (Associate)	Explain the concept, recognise it in a scenario, apply it correctly when the situation is ordinary.
C	CTP® (Certified)	Choose between options, justify the choice against trade-offs, and lead others through it when the situation is ambiguous.
A/C	Both	ATP tests the "explain and apply" part, CTP tests the "choose and justify" part.

A task tagged **C** only is not tested on the ATP exam.

Rules every exam item must follow

1. Every item traces to exactly one task statement ID (for example `D1-T06`). Items that cannot be traced are not used.
2. Items test judgement in a scenario wherever possible, not recall of a vendor's product names.
3. No item depends on a single vendor's tool. Tools may appear as examples in this text; the exam asks about the concept.
4. The share of items per domain matches the blueprint for that credential.

PART A · DOMAIN 5

The task statements

Blueprint weight: ATP 12%, CTP 10%. How engineers communicate, collaborate, grow and lead.

ID	Level	Task	Knowledge	Principles
D5-T01	A	Communicate technical information clearly in writing.	Design documents; PR and commit descriptions; incident updates; writing for the reader's decision.	P10
D5-T02	A	Collaborate within a delivery team.	Raising blockers early; pairing and mobbing; sharing context; respecting team agreements.	P1, P6
D5-T03	A	Give and receive code review feedback constructively.	Reviewing the change, not the person; specific, actionable comments; separating must-fix from preference; responding to feedback without defensiveness.	P4, P10
D5-T04	A	Act ethically as a practitioner.	Honesty about capability and status; confidentiality; protecting users and the public; conflicts of interest; the IICTP Code of Ethics.	P10
D5-T05	C	Lead a squad toward clear outcomes.	Setting goals; clarifying ownership; delegation; running meetings that reach decisions; balancing delivery with sustainability.	P3, P6
D5-T06	C	Mentor and develop other engineers.	Pairing as teaching; growth plans; feedback models; matching stretch work to the person.	P6
D5-T07	C	Align engineering careers with expectations.	Career ladders and levels; evidence for progression; individual-contributor and management tracks; fair, consistent assessment.	P10
D5-T08	C	Build psychological safety and a learning culture.	Psychological safety; generative vs pathological cultures; blameless reviews; how leaders respond to bad news.	P10
D5-T09	C	Shape team boundaries to manage cognitive load.	Team Topologies (stream-aligned, platform, enabling, complicated-subsystem teams; interaction modes); Conway's Law used deliberately.	P6
D5-T10	C	Work with stakeholders on scope and trade-offs.	Stakeholder mapping; negotiating scope, time and quality; saying no with options; reporting status honestly.	P3, P10

CHAPTER 5

People Leadership in Engineering

5.1 Scope

This domain is about the human side of engineering work: how practitioners communicate, collaborate, review each other's work, grow, lead and behave ethically. Leadership here is not the same as management. A senior engineer who leads a design, mentors colleagues or runs an incident is leading, whatever their title. The domain applies to both individual-contributor and management paths.

In scope: written technical communication, collaboration in a team, code review conduct, professional ethics, leading a squad, mentoring, career ladders, psychological safety, team boundaries and cognitive load, and working with stakeholders.

Covered elsewhere:

- The mechanics of pull requests and review turnaround: Domain 2, `D2-T02`. This chapter covers how people give and receive feedback.
- Running an incident: Domain 4, `D4-T10`. This chapter covers the culture that makes blameless learning possible.
- Service boundaries as a technical choice: Domain 1, `D1-T05`. This chapter covers team boundaries, the other side of the same decision.
- Delivery metrics: Domain 2, `D2-T06`. This chapter covers how measurement affects people.

What this domain does not cover. Employment law, HR procedures, compensation and formal performance management are the responsibility of the employer and its HR function. Practitioners must know when an issue belongs there (harassment, discrimination, health and legal matters) and hand it on promptly, rather than handle it informally.

5.2 Levels

	ATP® (12% of 100 items)	CTP® (10% of 150 items)
Typical candidate	Team member who writes, reviews, collaborates and acts with integrity	Tech lead or engineering manager responsible for a squad's direction, growth, health and relationships
What items ask	"What is the most constructive response...", "What should the engineer do when...", "Which update is clearest..."	"What should the lead do first...", "How should the team be structured...", "What is the best response to this stakeholder or leader..."
Tasks tested	D5-T01, T02, T03, T04	D5-T05, T06, T07, T08, T09, T10

5.3 Principles at work

- **P10 Act with integrity and keep it safe to speak up** is this domain's core principle. Honest status, honest feedback and honest reviews depend on people not being punished for bad news.
- **P6 Design for human limits**: people have limited attention, energy and capacity to learn. Team size, scope and on-call load are design decisions with human consequences.
- **P3 Separate the "what" from the "how"**: leaders set direction and context. The people doing the work decide how, and are trusted to.
- **P1 Close the loop**: feedback, retrospectives and one-to-ones are feedback loops for people and teams, and they work only if they are frequent and acted on.

Deming (1986) argued that most problems in an organisation come from the system that management controls, not from individual workers. He estimated the share at around 94%, and the eighth of his 14 points for management was to "drive out fear". When something goes wrong, a leader's first question is what in the system allowed it.

5.4 Knowledge by task

D5-T01 Communicate technical information clearly in writing A

What the practitioner does. Writes so that the reader can understand, decide or act with the least effort.

Core knowledge.

- **Write for the reader's decision.** Before writing, ask who will read this and what they need to do with it. A design document for reviewers, an incident update for customer support and a PR description for a teammate need different content.

- **Lead with the point.** State the conclusion, request or decision first, then the supporting detail (the "pyramid principle", Minto). Busy readers may stop after the first paragraph.
- **Common forms:**
  - *PR and commit descriptions*: what changed, why, how it was tested, what to look at.
  - *Design documents*: problem, constraints, options considered, proposal, risks, open questions.
  - *Incident updates*: what is affected, what is being done, when the next update will come.
  - *ADRs* (`D2-T08`).
- **Plain language.** Short sentences; define terms and acronyms on first use; concrete numbers over vague adjectives ("p95 latency rose from 200 ms to 1.4 s", not "it got really slow").
- **Know the limits of your knowledge.** Separate what is known, what is believed and what is unknown. Do not present guesses as facts.
- **Writing scales; meetings do not.** A good written proposal can be read by people in other time zones and later by people who join after the decision. Distributed teams depend on it.

ATP level. Choose the clearest version of an update or description; improve a vague message; include what the reader needs and leave out what they do not.

Common mistakes. Status updates that list activity instead of outcomes and risks. Design documents that hide the proposal on page six. Incident updates that stop when there is no news.

D5-T02 Collaborate within a delivery team A

What the practitioner does. Works in a way that helps the whole team deliver, not just their own tickets.

Core knowledge.

- **The team's goal over individual output.** Finishing the team's most important item, by helping, reviewing or unblocking, is usually worth more than starting a new item of your own (`D2-T05`).
- **Raise blockers early.** Say so as soon as you are stuck or a risk appears, with what you have tried. Struggling silently for days costs the team more than asking.
- **Share knowledge on purpose.**
  - *Pairing* (two people, one task) and *mobbing* or *ensemble* work (the whole team, one task) spread knowledge, catch problems early and reduce single points of failure.

  - Rotate who works on which areas so that no part of the system has a "bus factor" of one.
- **Team agreements.** Working hours and response expectations, review turnaround, Definition of Done, how decisions are made. Follow them, and propose changes openly in retrospectives rather than ignoring them.
- **Disagreement.** Disagree about ideas with evidence, in the open and early. Once the team decides, support the decision, even if you argued against it, unless it is unethical or unsafe. Revisit it later with new evidence.
- **Remote and hybrid work.** Default to written, searchable communication for decisions; make sure remote colleagues hear decisions at the same time as those in the room.

ATP level. Choose the most helpful action in a team situation; recognise when to ask for help; respond appropriately to disagreement.

Common mistakes. "My tickets are done" while the sprint goal fails. Being the only person who understands a component, and keeping it that way. Revisiting settled decisions in every meeting.

D5-T03 Give and receive code review feedback constructively A

What the practitioner does. Uses code review to improve the code and the team, and treats feedback, given and received, as being about the work.

Core knowledge.

- **What review is for.** Catching defects is only part of it. Studies of modern code review (Bacchelli and Bird, 2013) found knowledge transfer, shared awareness and alternative solutions to be major outcomes. A review is also a conversation between colleagues.
- **The standard.** Approve a change when it clearly improves the overall health of the code, even if it is not exactly how you would have written it. Block only for real problems, not personal preference (a principle in Google's published engineering practices).
- **Giving feedback:**
  - Comment on the code, not the person: "this function returns null when the list is empty", not "you forgot the empty case again".
  - Be specific and give reasons, ideally with a suggestion.
  - Mark how much each comment matters, for example *must fix* versus *suggestion* or *nit*, so the author knows what blocks merging.
  - Ask questions when unsure: "what happens if two requests arrive at once?"
  - Say what is good, too.
  - Move long debates to a conversation, then record the outcome in the PR.

- **Receiving feedback:**
  - Assume good intent.
  - Separate the point from the tone.
  - Ask for clarification rather than defending.
  - Fix or reply to every comment.
  - If you disagree, explain why with evidence. Escalate to the team's agreed decision process only if needed.
- **Power and fairness.** Feedback from senior engineers carries extra weight. Seniors should be especially careful with tone, and should invite juniors to review their code too.

ATP level. Choose the most constructive review comment; respond appropriately to critical feedback; distinguish a blocking issue from a preference.

Common mistakes. Sarcasm and rhetorical questions in reviews. Blocking a PR over style a formatter could settle. Ignoring comments, or taking every comment as a personal attack.

D5-T04 Act ethically as a practitioner A

What the practitioner does. Acts honestly and responsibly toward users, the public, employers and colleagues, including when under pressure.

Core knowledge.

- **Codes of ethics.**
  - The ACM/IEEE-CS Software Engineering Code of Ethics and Professional Practice (1999) sets out eight principles: the public, client and employer, product, judgement, management, profession, colleagues, and self. The public interest comes first.
  - The ACM Code of Ethics and Professional Conduct (2018) and the IICTP Code of Ethics express the same core.
  - IICTP credential holders agree to the IICTP Code.
- **Honesty about work.** Report real status, including risks and bad news. Do not mark work as done or tested when it is not. Do not claim others' work or credentials you do not hold.
- **Protect users and data.** Access only the data your task needs. Do not look at personal data out of curiosity. Report suspected breaches promptly.
- **Raise concerns.** When asked to do something unsafe, deceptive or unlawful, say so clearly, in writing if needed, and escalate through the proper channels (lead, ethics or compliance contact, whistle-blowing procedure). Disagreeing is not disloyalty.
- **Conflicts of interest.** Declare them: a vendor you have ties to, reviewing a friend's promotion case, outside work that competes with your employer.

- **Competence.** Take on work you can do or can learn to do safely, and say when you cannot.
- **Fairness and respect.** Treat colleagues with respect regardless of background. Harassment and discrimination are reported to the appropriate function, not handled informally (see 5.1).

ATP level. Identify the ethical response in a scenario involving pressure on status, data access, or a conflict of interest.

Common mistakes. "Green" status reports for work that is red. Copying production personal data to a laptop "to debug faster". Staying silent about a known safety issue because someone senior approved the release.

D5-T05 Lead a squad toward clear outcomes C

What the practitioner does. Gives a team a clear direction and the conditions to reach it, then gets out of the way.

Core knowledge.

- **Outcomes over output.** Goals describe the change wanted, such as fewer failed bookings or a faster onboarding, not a list of features. Objectives and key results (OKRs; Grove, 1983; Doerr, 2018) are one common format. Keep goals few and visible.
- **Clarity of ownership.**
  - Each service, decision and recurring task has a named owner.
  - Responsibility frameworks such as RACI or DACI help where many people are involved.
  - Ambiguous ownership is a common cause of dropped work and conflict.
- **Delegation.**
  - Delegate outcomes with context and constraints, not step-by-step instructions (P3).
  - Delegation is a range, from "I decide and tell you", through "we decide together", to "you decide and inform me" (Appelo's seven levels).
  - Match the level to the person's experience and the decision's risk, and say which level applies.
- **Meetings that decide.** Every meeting has a purpose, the right people, and an output: a decision, an owner and a date, written down. Prefer written proposals and asynchronous review for anything that does not need live discussion.
- **Sustainable pace.** Overtime as a habit lowers quality and drives people away. Protect focus time, keep on-call load reasonable, and treat repeated crunch as a planning failure to fix.

- **The lead's own work.** A tech lead who keeps all the hard technical work becomes the bottleneck. Lead by reviewing, unblocking, pairing and making decisions visible, while staying close enough to the code to judge it.

CTP level. Turn vague direction into clear outcomes and ownership; choose an appropriate delegation level; fix a team whose meetings, ownership or pace are failing.

Common mistakes. Goals that are just a feature list. "Everyone owns it", so nobody does. The lead as the approval gate for every change.

D5-T06 Mentor and develop other engineers C

What the practitioner does. Helps other engineers grow in skill, judgement and confidence, deliberately and not only when asked.

Core knowledge.

- **Mentoring, coaching, sponsoring.**
  - *Mentoring* shares experience and advice.
  - *Coaching* asks questions that help someone find their own answer.
  - *Sponsoring* uses your influence to give someone opportunities and visibility, for example recommending them to lead a project.

  Underrepresented engineers are often mentored but less often sponsored.
- **Stages of skill.** The Dreyfus model of skill acquisition (novice, advanced beginner, competent, proficient, expert) explains why beginners need clear rules and examples, while experienced people need context and room. Adjust your approach to the person and the topic: someone can be an expert in one area and a novice in another.
- **Learning through work.**
  - *Stretch assignments*: slightly beyond current ability, with support.
  - *Pairing*, with the learner at the keyboard.
  - Reviewing the learner's designs before code exists.
  - Letting them run a meeting or an incident review with backup.
- **Feedback that helps.** Timely, specific and about behaviour and its effect. Models such as Situation–Behaviour–Impact (SBI) keep it factual: "In yesterday's design review (situation), you presented two options with their trade-offs (behaviour), and it let the team decide in ten minutes (impact)." Give positive feedback as specifically as corrective feedback.
- **Growth plans.** Agree a few concrete goals tied to the career ladder (`D5-T07`), the evidence that will show progress, and when to check in. Regular one-to-ones are the usual place for this (Grove, 1983).

CTP level. Plan the development of a described engineer; choose between mentoring, coaching and sponsoring in a situation; give feedback in a form that helps.

Common mistakes. Taking over the keyboard whenever a junior struggles. Feedback saved up for the annual review. Assigning only the work someone already does well.

D5-T07 Align engineering careers with expectations C

What the practitioner does. Makes expectations for each level clear, and makes growth and promotion decisions fair and based on evidence.

Core knowledge.

- **Career ladders.** A written description of what is expected at each level. Usually organised by dimensions such as technical skill, scope and impact, collaboration and leadership. Scope grows with level: own a task, a feature, a system, a group of systems, an organisation's technical direction.
- **Dual track.** Senior engineers can grow as individual contributors (staff, principal) without moving into management, and management is a different job, not a promotion. Both tracks need equal recognition, or the best engineers become reluctant managers.
- **Evidence over impressions.**
  - Promotion cases rest on concrete examples of work at the next level's scope, collected over time.
  - Visibility, confidence and being liked are not evidence.
  - Calibration across teams keeps standards consistent.
- **Bias.** Known risks include:
  - *recency bias*: only the last few months count;
  - *similarity bias*: favouring people like oneself;
  - unequal access to high-visibility work.

  Structured criteria, written evidence and fair allocation of opportunities reduce them.
- **Measuring engineers.** Single numbers (lines of code, tickets closed, story points) are poor and easily gamed measures of an engineer's contribution (see D2-T06). The SPACE framework (Forsgren et al., 2021) argues for several dimensions together: satisfaction and well-being, performance, activity, communication and collaboration, and efficiency and flow.
- **Honest conversations.** Tell people where they stand and what the gap is. A surprise at promotion time is a failure of the lead.

CTP level. Explain the expectations of a level and the gap to the next; assess a described case against a ladder; spot unfair or poorly evidenced decisions.

Common mistakes. No written ladder, so promotion depends on who argues loudest. Promoting the best engineer into management as a reward. Measuring engineers by commit counts.

D5-T08 Build psychological safety and a learning culture C

What the practitioner does. Creates a team where people can admit mistakes, ask questions and challenge ideas without fear, so problems surface early.

Core knowledge.

- **Psychological safety (Edmondson, 1999).** A shared belief in a team that it is safe to take interpersonal risks: to ask, to admit an error, to disagree. Edmondson found it associated with team learning behaviour, and Google's internal study of team effectiveness (Project Aristotle) reported it as the most important of the dynamics it studied.
- **It is not the same as comfort.** Safe teams have high standards and frank disagreement. What they lack is punishment for honesty.
- **Organisational culture (Westrum, 2004).** Cultures differ in how they handle information:
  - *Pathological* (power-oriented): messengers are punished.
  - *Bureaucratic* (rule-oriented): messengers are ignored.
  - *Generative* (performance-oriented): messengers are trained, failures lead to inquiry, and new ideas are welcomed.

  The DORA research found generative culture associated with better software delivery and organisational performance.
- **What leaders do.**
  - Respond to bad news with thanks and curiosity.
  - Admit their own mistakes and uncertainty.
  - Ask quieter people for their view.
  - Make it explicit that questions are expected.
  - Run blameless reviews (`D4-T10`).
  - Never use errors disclosed in a retrospective against someone later.
- **Protecting the team upward.** When a senior leader asks "who did this?", the lead answers with the system view and the actions being taken, and takes responsibility for the team. This is one of the moments that decides whether people will speak up next time.
- **Signals to watch.** Who talks in retrospectives; whether incidents are reported promptly; whether juniors ask questions in public channels; whether status reports are ever red.

CTP level. Diagnose a low-safety team from described behaviour; choose the leader's best response to a mistake or to blame from above; design team practices that build safety.

Common mistakes. Declaring "this is a safe space" while reacting badly to bad news. Blameless reviews that end in quiet consequences. Confusing safety with avoiding disagreement.

D5-T09 Shape team boundaries to manage cognitive load C

What the practitioner does. Designs what each team owns and how teams interact so that every team can understand and deliver its part well.

Core knowledge.

- **Cognitive load.** The amount of mental effort a task demands. Cognitive load theory (Sweller, 1988) distinguishes:
  - *intrinsic* load, the difficulty of the problem itself;
  - *extraneous* load, effort spent on the environment (tooling, process, unclear ownership);
  - *germane* load, effort that builds understanding.

  A team's capacity is limited. Reduce extraneous load first.
- **Team size and communication.** Communication paths grow as n(n−1)/2: 5 people have 10 paths, 10 people have 45. Small, long-lived teams of roughly 5 to 9 people are a common guideline. Adding people to a late project can make it later (Brooks, 1975).
- **Team Topologies (Skelton and Pais, 2019).** Four team types:
  - *stream-aligned*: owns a flow of change for a product, service or user journey, end to end; most teams should be this type;
  - *platform*: provides internal services that reduce other teams' load;
  - *enabling*: helps other teams gain capabilities, temporarily;
  - *complicated-subsystem*: owns a part needing deep specialist knowledge.

  Three interaction modes: *collaboration* (working closely for a period), *X-as-a-service* (consuming something with minimal coordination), and *facilitating* (one team helping another learn).
- **Conway's Law, used deliberately.** Team boundaries and system boundaries tend to converge (`D1-T05`). Design them together: if two teams must coordinate on every change, either the system boundary or the team boundary is in the wrong place.
- **Signs of overload.** Too many services or domains per team; constant context switching; on-call shifts dominated by unfamiliar systems; long onboarding; work waiting on another team's backlog.

CTP level. Assess a described team's load and propose changes to ownership, team type or interaction mode; recognise when growth requires splitting a team and where to split it.

Common mistakes. One team owning everything nobody else wants. Splitting teams by technical layer (frontend team, backend team, database team), which creates hand-offs on every feature. Reorganising often without changing system boundaries.

D5-T10 Work with stakeholders on scope and trade-offs C

What the practitioner does. Builds working relationships with the people who depend on or affect the team's work, and negotiates scope, time and quality honestly.

Core knowledge.

- **Know your stakeholders.** List who is affected and who has influence. A power–interest grid helps decide whom to manage closely, keep satisfied, keep informed, or monitor. Understand what each one actually needs; it is often different from what they first ask for.
- **The trade-off triangle.** With fixed people, scope, time and quality pull against each other. When a date is fixed, scope is the usual lever. Quality is the dangerous lever, because cutting it creates debt and incidents that cost more time later (D2-T09).
- **Say no with options.** Instead of a flat no, show the choices and their costs. For example: "We can launch on the 15th with A and B, or on the 29th with A, B and C. Doing all three by the 15th would mean skipping the load test, and that is how last year's outage happened. Which matters more?" This keeps the decision with the person who owns the outcome (P3) and the information honest (P10).
- **Forecast honestly.** Use ranges and probabilities from real data (D2-T05), not single-date promises. Update forecasts as soon as they change. Bad news delivered early is a planning input; delivered late, it is a crisis.
- **Regular, predictable communication.** A short, consistent update (outcomes, risks, decisions needed) builds more trust than occasional long reports. Show working software, not slides (D2-T03).
- **Managing up.** Bring problems with options and a recommendation. Make your manager aware of risks before they hear about them elsewhere.

CTP level. Respond to a stakeholder demand that cannot be met as stated; present trade-offs in a way that leads to a decision; plan stakeholder communication for a piece of work.

Common mistakes. Agreeing to an impossible date to avoid conflict, then missing it. Cutting testing silently to meet a date. Status reports that turn red the week before launch.

5.5 Anti-patterns

Anti-pattern	Symptom	Why it hurts	Remedy	Task
Hero culture	One person fixes every crisis and reviews every change	Bottleneck; burnout; nobody else learns; risk if they leave	Pairing, rotation, shared on-call, documented knowledge; reward teaching, not rescuing	D5-T02, D5-T05
Blame and punish	Mistakes lead to public criticism or quiet consequences	Problems are hidden until they are big	Blameless reviews; leaders model admitting error; protect the team upward	D5-T08
Feature factory	Success measured by features shipped; no one checks outcomes	Effort without impact; disengaged team	Outcome-based goals; review what changed for users	D5-T05
Watermelon status	Reports green outside, red inside	Decisions made on false information; late surprises	Make honest red status safe and expected; forecast with ranges	D5-T04, D5-T10
Reluctant manager	Best engineer promoted into management as a reward	Lose a strong engineer, gain an unhappy manager	Dual-track ladder with equal recognition	D5-T07
Everything team	One team owns many unrelated services and all of their on-call	High cognitive load; slow delivery; burnout	Re-draw ownership; platform services; split along stream boundaries	D5-T09
Drive-by review	Harsh or vague comments; style wars	Authors stop asking for review; knowledge stops flowing	Comment norms; marked severity; formatter for style	D5-T03
Yes to everything	Every request accepted with the same date	Missed commitments; silent quality cuts	Trade-off conversations with options	D5-T10

5.6 Worked scenario

Situation (CTP level). After a reorganisation, two squads of five have been merged into one team of ten under a newly appointed tech lead, Priya. Priya was previously a senior engineer on one of the two squads. The team now owns nine services, including both former squads' services, and shares one on-call rotation. Within the first month:

- **Ravi**, the most experienced engineer, reviews nearly every pull request and is the only person who understands the payment reconciliation service. Reviews wait for him, and he is working late most nights.
- **Retrospectives** are quiet. Most comments come from Ravi and one other senior. The three newest engineers rarely speak.
- **A director** publicly asked "who broke checkout?" in a company channel after a short incident caused by a junior engineer's change. The engineer has been subdued since.

- **The product manager** has promised a large customer a new invoicing feature by the end of next month. The team's own estimate is about twice that.
- **Two engineers** have asked Priya what they need to do to be promoted. The company's ladder is a one-page list of adjectives.

Priya is asked by her manager how she plans to "get the team back on track".

Step 1: Respond to the director (D5-T08, D5-T04). This comes first, because the whole team saw it. In the same channel, Priya replies with the system view: the change passed review and all automated checks, so the gap was in what the checks covered, and the team is adding a test for that case and a canary step for checkout changes. She takes responsibility as the lead. Privately, she:

- thanks the junior engineer for helping with the fix;
- tells them the review will look at the system, not at them;
- asks her own manager to raise with the director the effect of public blame on reporting.

At the next retrospective she describes a mistake of her own from the previous year and what the team learned from it.

Step 2: Reduce the team's load (D5-T09, P6). Ten people means 45 communication paths, compared with 20 across the two former squads of five. Together with nine services and one shared rotation, the load is too high. Priya maps the services by the user journeys they serve and proposes two stream-aligned squads of five again:

- an **invoicing and payments** squad;
- an **ordering** squad.

Each owns the services for its journey and has its own on-call rotation. Two shared internal tools that both squads use, but neither wants to own, are proposed for the existing platform team to provide as a service. Priya writes this up with the reasoning and asks her manager to approve it. The decision is recorded in an ADR, because it changes ownership (`D2-T08`).

Step 3: Remove the single point of failure (D5-T02, D5-T05, D5-T06). Priya meets Ravi one-to-one. She acknowledges the load he has carried, and asks what he would like to spend more time on. He wants to lead the design of the new invoicing feature. They agree:

- Review is shared. Any two engineers can approve most changes, and Ravi is required only for reconciliation logic, for the next two months.
- Ravi pairs with two engineers on reconciliation each week, with the less experienced person at the keyboard, and they write a short runbook together.

- Ravi leads the invoicing design. Priya explicitly delegates the technical decision to him ("you decide, and keep me informed"), with the constraint that the design must be reviewed by the squad.
- Priya stops reviewing routine PRs herself, so she is no longer a second bottleneck.

Step 4: Rebuild the conversation (D5-T08, D5-T03).

- **Retrospectives:** start with a few minutes of silent written input, so everyone contributes before the most confident voices set the direction. Priya asks the newest engineers first and acts visibly on at least one of their points each time.
- **Review norms:** the team agrees to mark each comment as blocking or not, to explain the reason, and to leave style to the formatter.
- **Psychological safety pulse:** after a month, Priya runs an anonymous three-question check. More engineers report that they are comfortable raising problems than in the first week. It is still not good enough, and she says so openly.

Step 5: The invoicing date (D5-T10, D2-T05). Priya looks at the team's historical throughput. At the current rate, delivering the full scope by the promised date is unlikely. She meets the product manager before the customer asks for an update, and brings options rather than refusals:

Option	Date	What it means
A	The promised date	Invoice generation and download only. Automated reminders and multi-currency follow four weeks later.
B	Four weeks later	The full scope.
C	The promised date, full scope	Skip load testing and the security review of the payment integration. Priya recommends against this, citing the previous year's incident.

The product manager takes option A to the customer. The customer mainly needed invoices for their own month-end close. The reduced first release is accepted.

Step 6: Make careers clear (D5-T07, D5-T06). Priya works with the other leads in her department to turn the one-page list into a ladder with dimensions (technical skill, scope, collaboration, leadership) and concrete examples of each at each level, and offers it to HR for adoption. For the two engineers asking about promotion, she:

1. writes down where each one stands against the draft ladder, with examples;
2. agrees one or two goals each that would show the next level's scope (for one, leading the reconciliation runbook and on-call handover; for the other, owning the invoicing API end to end);
3. books monthly check-ins.

She is honest with both that promotion decisions are made in calibration with other leads, not by her alone.

Step 7: Report to her manager (D5-T01). Priya sends a one-page written update:

- what she found;
- the decisions she has made;
- the decisions she needs from her manager (the team split and the platform handover);
- the risks;
- how she will know it is working: review wait times, on-call pages per person, retrospective participation, the pulse check, and delivery against the agreed invoicing scope.

What a weaker answer looks like.

- "Ask everyone to work harder until the invoicing deadline." This burns out the team, cuts quality silently, and does nothing about the hero bottleneck.
- "Stay quiet about the director's comment to avoid conflict." This teaches the team that mistakes lead to public blame.
- "Make Ravi the team's manager, since he knows the most." This loses the team's strongest engineer to a job he has not asked for and leaves the knowledge concentrated.

Mini-scenario (ATP level). A developer receives this review comment on their pull request: "This is wrong. Did you even test it?" They feel embarrassed and want to reply defensively. *What would a constructive version of the comment look like, and how should the developer respond?*

A constructive comment states the problem, its effect and a suggestion: "**Blocking:** when `items` is empty this returns `null`, and the caller on line 42 will throw. Could you return an empty list and add a test for that case?"

The developer should:

- separate the substance from the tone;
- ask what specifically fails if it is unclear;
- fix the issue and add the test;
- reply when done.

If the tone is a pattern, raise it calmly and privately with the reviewer or the lead, not in the PR thread (D5-T03).

5.7 Illustrative questions

These show the style of item for this domain. They are written for this guide and are not taken from the exam bank.

Q1. `D5-T03` · **ATP** Which review comment is most constructive?

- A. “Why would anyone write it like this?”
- B. “Wrong.”
- C. “Blocking: this query runs inside the loop, so a page of 100 items makes 100 database calls. Could you fetch them in one query before the loop?”
- D. “I would have done this differently, but whatever.”

Answer: C. It is specific, explains the impact, marks its severity and suggests a fix.

- **A** and **B** attack without informing.
- **D** is dismissive and gives the author nothing to act on.

Q2. `D5-T04` · **ATP** The day before a customer demo, a manager asks a developer to mark a feature as “Done” on the board so it appears in the demo report. The feature works on the developer’s machine, but its tests are failing in the pipeline. What should the developer do?

- A. Mark it Done, since it works locally
- B. Mark it Done and fix the tests after the demo
- C. Explain that it does not yet meet the Definition of Done, report its actual state and what remains, and suggest how it could be shown honestly in the demo (for example, as work in progress)
- D. Refuse to discuss it

Answer: C. Honest status is an ethical duty and protects the decisions others will make from the report.

- **A** and **B** misreport the work.
- **D** gives up the chance to find a workable alternative.

Q3. `D5-T08` · **CTP** After a short outage caused by a configuration change, a senior director asks the tech lead in a public channel, “Who made this change?” What is the best response?

- A. Name the engineer so the director has a direct answer
- B. Say nothing publicly and hope the question is forgotten
- C. Respond publicly with what happened at the system level (the change passed the existing checks, which did not cover this case), what the team is changing, and that the lead is accountable for the follow-up; then discuss the effect of public blame with the director privately
- D. Reply that the outage was the director’s fault for setting the deadline

Answer: C. It answers the legitimate question about cause and action, keeps the focus on the system, and protects the conditions for future honest reporting.

- **A** punishes disclosure.
- **B** leaves the question and the fear unaddressed.
- **D** replaces one form of blame with another.

Q4. `D5-T09` · **CTP** A team of eight owns eleven services spanning three unrelated user journeys, plus a shared build tool every team depends on. On-call is exhausting, delivery is slow, and new joiners take months to become productive. Which change is most likely to help?

- A. Hire four more engineers into the same team
- B. Split ownership along user journeys into smaller stream-aligned teams, and move the shared build tool to a platform team that offers it as a service
- C. Add a weekly coordination meeting for all eleven services
- D. Ask the team to document every service in more detail before changing anything

Answer: B. It reduces each team's cognitive load and aligns ownership with flows of change.

- **A** raises communication overhead without reducing scope.
- **C** adds coordination load.
- **D** is useful, but leaves the structural overload in place.

Q5. `D5-T10` · **CTP** A sales director demands that a feature be delivered in three weeks for a customer. The team's forecast, based on its historical throughput, is six to eight weeks for the full scope. What should the tech lead do?

- A. Agree to three weeks to keep the relationship positive, and ask the team to work weekends
- B. Refuse, since the forecast says six weeks
- C. Find out what the customer most needs by the date, then present options, such as a smaller first release in three weeks, the full scope in six to eight, and the risks of anything else, so the director can decide
- D. Agree, then skip testing to save time

Answer: C. It turns a demand into an informed choice about scope.

- **A** and **D** make promises the team cannot keep, at a cost to people or quality.
- **B** ends the conversation without exploring what could be delivered.

5.8 Evidence of good practice

What a reviewer or assessor would expect to find in a team that does this domain well:

- Written design documents, ADRs and status updates that state the point first and are kept where everyone can find them.
- Review norms that are written down and followed. Reviews come from many people, not one gatekeeper, and comments mark their severity.
- No component that only one person understands. Pairing, rotation and runbooks are visible.
- Everyone, not only the senior members, contributes in retrospectives. Incidents and mistakes are reported promptly, and reviews are blameless in practice.
- Clear outcome goals and named owners for services and decisions. Team size and scope are within what the team can handle, and on-call load is tracked.
- A written career ladder with examples. Each engineer has a growth plan and regular one-to-ones. Promotion decisions are calibrated and evidence-based.
- Stakeholders receive regular, honest updates with forecasts as ranges, and trade-offs are decided openly.
- Engineers can say what the code of ethics asks of them and how to raise a concern.

5.9 References and further reading

Foundational research

- Bacchelli, A., & Bird, C. (2013). Expectations, outcomes, and challenges of modern code review. *Proceedings of the 35th International Conference on Software Engineering (ICSE 2013).*
- Brooks, F. P. (1975). *The Mythical Man-Month.* Addison-Wesley.
- Conway, M. E. (1968). How do committees invent? *Datamation*, 14(4), 28–31.
- Dreyfus, S. E., & Dreyfus, H. L. (1980). *A Five-Stage Model of the Mental Activities Involved in Directed Skill Acquisition.* University of California, Berkeley, Operations Research Center report.
- Edmondson, A. (1999). Psychological safety and learning behavior in work teams. *Administrative Science Quarterly*, 44(2), 350–383.
- Forsgren, N., Storey, M.-A., Maddila, C., Zimmermann, T., Houck, B., & Butler, J. (2021). The SPACE of developer productivity. *ACM Queue*, 19(1).
- Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2), 257–285.
- Westrum, R. (2004). A typology of organisational cultures. *Quality and Safety in Health Care*, 13(suppl. 2), ii22–ii27.

Ethics

- ACM (2018). *ACM Code of Ethics and Professional Conduct.*

- ACM/IEEE-CS Joint Task Force (1999). *Software Engineering Code of Ethics and Professional Practice.*
- IICTP. *Code of Ethics* (published on the IICTP website).

Practitioner books

- Appelo, J. (2011). *Management 3.0: Leading Agile Developers, Developing Agile Leaders.* Addison-Wesley.
- Deming, W. E. (1986). *Out of the Crisis.* MIT Press.
- Doerr, J. (2018). *Measure What Matters.* Portfolio.
- Fournier, C. (2017). *The Manager's Path.* O'Reilly.
- Grove, A. S. (1983). *High Output Management.* Random House.
- Larson, W. (2019). *An Elegant Puzzle: Systems of Engineering Management.* Stripe Press.
- Minto, B. (2009). *The Pyramid Principle: Logic in Writing and Thinking.* Pearson.
- Reilly, T. (2022). *The Staff Engineer's Path.* O'Reilly.
- Senge, P. M. (1990). *The Fifth Discipline.* Doubleday.
- Skelton, M., & Pais, M. (2019). *Team Topologies.* IT Revolution.

Industry practice

- Google. *Google Engineering Practices Documentation: The Standard of Code Review.* (Published online.)
- DORA / Google Cloud. *Accelerate State of DevOps Reports*: findings on generative culture and well-being (2022 and 2023 reports in the research archive).

Standards crosswalk

- IEEE Computer Society (2024). *SWEBOK Guide v4.0*: Software Engineering Professional Practice knowledge area.

PART A · THE PRINCIPLES

The ten principles

These ten principles sit underneath every domain. They are drawn from what PMI, Scrum, Kanban, IEEE SWEBOK, TOGAF and the DORA research programme agree on once their vocabulary is removed (see Appendix C2), plus two that the AI era has made urgent (P8, P9). Exam scenarios often test a principle through a task: an item on `D2-T05` (flow) is usually also an item on `P2` (small batches).

P1. Close the loop

Work in short cycles that end with evidence: a test result, a user's reaction, a production metric. Decide the next step from that evidence, not from the plan alone. *In practice:* a design is checked by a prototype before it is committed to; a sprint ends with working software that someone has looked at; an incident ends with a change, not just a report. *Roots:* Deming's PDCA cycle; Scrum's transparency, inspection and adaptation; DORA's emphasis on fast feedback.

P2. Keep batches small

Smaller pieces of work move faster, fail smaller and are easier to review. Time spent waiting grows with the amount of work in progress. *In practice:* small pull requests, frequent integration to the main branch, work items that fit in a few days, limits on work in progress. *Roots:* Little's Law ($L = \lambda W$); Kanban; trunk-based development in the DORA research.

P3. Separate the "what" from the "how"

Someone owns the outcome and its value; the people doing the work own how it is built. Both are accountable, for different things. *In practice:* a product owner orders the backlog; engineers decide the design; neither silently overrides the other. *Roots:* Scrum accountabilities; PMI sponsor and team roles; Dijkstra's separation of concerns.

P4. Make “done” explicit and checkable

Quality is agreed before work starts and checked by something other than the author’s opinion, ideally by automation. *In practice:* acceptance criteria on every work item; a definition of done; tests and checks that block a merge when they fail. *Roots:* Scrum’s Definition of Done; Kanban’s explicit policies; Deming’s “build quality in”.

P5. Retire risk early

Do the riskiest, least understood part first, while it is still cheap to change course. *In practice:* spikes and prototypes; threat modelling during design, not after release; canary releases before full rollout. *Roots:* Royce (1970), “do it twice”; Brooks (1986) on prototyping; progressive delivery.

P6. Design for human limits

People and teams have limited attention. The structure of a system tends to copy the structure of the organisation that builds it. *In practice:* teams sized so that everyone can talk to everyone; module and service boundaries chosen with team ownership in mind; on-call loads that people can sustain. *Roots:* Brooks’s Law and the n(n−1)/2 communication paths; Conway (1968); Team Topologies.

P7. Make change cheap

Requirements, technology and teams will change. Prefer decisions that are easy to reverse, and invest in the ones that are not. *In practice:* information hiding behind stable interfaces; decisions recorded with their reasons; feature flags that separate deploying from releasing. *Roots:* Parnas (1972); the Agile Manifesto; architecture decision records.

P8. Verify what you did not write

Code, text and data produced by AI tools, libraries, or other teams are inputs to be checked, not facts to be trusted. The person who ships it is responsible for it. *In practice:* AI-generated code goes through the same tests and review as human code; dependencies are scanned and pinned; model outputs are validated against a schema before use. *Roots:* Hoare (1981) on simplicity and verifiability; software supply-chain security practice; the OWASP Top 10 for LLM Applications.

P9. Contain failure

Assume parts of the system will fail and limit how far a failure can spread. Recovering fast matters more than never failing. *In practice:* least privilege; timeouts and circuit breakers; isolated failure domains; practised recovery; error budgets. *Roots:* Saltzer and Schroeder (1975); Armstrong (2003); Google SRE; DORA's recovery-time metric.

P10. Act with integrity and keep it safe to speak up

Be honest about what you know, what you built and what went wrong. Build teams where raising a problem is rewarded, not punished. *In practice:* blameless post-incident reviews; honest estimates and status; protecting user data; declaring conflicts of interest. *Roots:* ACM/IEEE-CS Software Engineering Code of Ethics; Westrum's generative culture as used in DORA; Deming's "drive out fear".

GLOSSARY

Terms used in this domain

Bus factor

The number of people who would have to leave before a part of the system could no longer be maintained.

Career ladder

A written description of expectations at each level of an engineering career, used for growth and promotion.

Cognitive load

The mental effort a task demands; a team's capacity for it is limited.

Dual track

A career structure in which engineers can progress as individual contributors or as managers, with equal recognition.

Generative culture

A performance-oriented culture (Westrum) in which information flows freely, messengers are welcomed and failures lead to inquiry.

Psychological safety

A shared belief that a team is safe for interpersonal risk-taking: asking, admitting mistakes, disagreeing.

Sponsorship

Using one's influence to give someone opportunities and visibility, as distinct from mentoring (advice) or coaching (questions).

Stream-aligned team

A team that owns a flow of change for a product, service or user journey, end to end (Team Topologies).

INDEX

All 56 task statements

The tasks of this guide's domain are shaded. **A** is tested in the ATP®, **C** in the CTP®.

D1-T01	A/C	Choose an interaction style for communication between components.
D1-T02	A	Design and consume HTTP APIs correctly.
D1-T03	A	Decompose code into modules that hide their design decisions.
D1-T04	A	Apply common design patterns where they fit, and avoid them where they do not.
D1-T05	C	Define service and module boundaries for a system and the teams that own it.
D1-T06	C	Design reliable event-driven flows.
D1-T07	A/C	Design and operate caching tiers.
D1-T08	C	Reason about consistency and availability trade-offs in distributed data.
D1-T09	C	Isolate failure domains so a fault in one part does not take down the rest.
D1-T10	A/C	Scale a system to meet demand.
D1-T11	C	Evolve a legacy system incrementally.
D2-T01	A	Use version control to collaborate safely.
D2-T02	A	Raise and review pull requests.
D2-T03	A	Work within an iterative delivery cycle.
D2-T04	A	Write and maintain work items others can act on.
D2-T05	A/C	Manage the flow of work.
D2-T06	C	Measure software delivery performance without turning measures into targets.
D2-T07	C	Plan and coordinate releases across several teams.
D2-T08	C	Record significant technical decisions.
D2-T09	C	Identify, quantify and pay down technical debt.
D2-T10	C	Tailor the delivery approach to the situation.
D3-T01	A	Explain how large language models behave, at the level needed to use them safely.
D3-T02	A	Use AI developer tools within policy.
D3-T03	A	Verify AI-generated code and content before relying on it.
D3-T04	A	Write prompts that are clear, bounded and repeatable.
D3-T05	C	Design retrieval-augmented generation (RAG) systems.
D3-T06	C	Choose and operate a vector store.
D3-T07	C	Evaluate AI features before and after release.

D3-T08	C	Secure AI systems.
D3-T09	C	Integrate AI into production services reliably and affordably.
D3-T10	C	Govern AI use responsibly.
D3-T11	A/C	Explain how machine learning models are trained and evaluated.
D3-T12	C	Choose the right approach for an AI problem.
D3-T13	C	Prepare and govern training data.
D3-T14	C	Fine-tune and adapt pretrained models.
D3-T15	C	Deploy and operate machine learning models.
D4-T01	A	Explain and maintain a CI/CD pipeline.
D4-T02	A	Handle secrets safely.
D4-T03	A	Prevent common application vulnerabilities.
D4-T04	A/C	Manage dependencies and the software supply chain.
D4-T05	C	Apply least privilege and zero-trust networking.
D4-T06	C	Threat-model a system.
D4-T07	A/C	Make systems observable.
D4-T08	C	Run and scale containerised workloads on an orchestrator.
D4-T09	C	Manage infrastructure as code.
D4-T10	C	Respond to incidents and learn from them.
D5-T01	**A**	**Communicate technical information clearly in writing.**
D5-T02	**A**	**Collaborate within a delivery team.**
D5-T03	**A**	**Give and receive code review feedback constructively.**
D5-T04	**A**	**Act ethically as a practitioner.**
D5-T05	**C**	**Lead a squad toward clear outcomes.**
D5-T06	**C**	**Mentor and develop other engineers.**
D5-T07	**C**	**Align engineering careers with expectations.**
D5-T08	**C**	**Build psychological safety and a learning culture.**
D5-T09	**C**	**Shape team boundaries to manage cognitive load.**
D5-T10	**C**	**Work with stakeholders on scope and trade-offs.**

People Leadership in Engineering

Writing, collaboration, code review, ethics, leading and mentoring, careers, psychological safety, team design and stakeholders.

Part of the IICTP Library. Every book is free to download at iictp.org.

- Domain Guide 1: Architecture & Systems Design
- Domain Guide 2: Engineering Governance & Agile Delivery
- Domain Guide 3: Applied AI & Emerging Tech Integration
- Domain Guide 4: Infrastructure & Security
- Illustrative Questions
- Glossary of Technical Practice
- Casebook
- Code of Ethics and Professional Conduct

IICTP · Version 0.2 · 22 October 2026