Templates and Tools

From: The Fastest Way by Pieter Amersfoort

Printable templates and tools -- A4 format

Created: 2026-08-17

Introduction

Most problems remain hard not because they are impossible, but because people start too quickly, define them too vaguely, or head in the wrong direction. The Fastest Way gives you three tools: the Rules to think clearly, the Map to see where you are, and the Processes to choose the right approach.

These templates and tools are the practical companion to the book. Print the ones you need, fill them in as you work through a problem, and build a personal archive of problems solved and lessons learned. Start with the Problem-Solving Journal; it gives the rest of the toolkit a record to build on.

Make Explicit (Ch. 3)

Structured Writing Protocol

Use this protocol when a problem lives in your head but resists simple articulation. Follow the four steps below, spending about five minutes on each. Return three days later to review with fresh eyes.

The Four Steps

Step 1: Free Write (5 minutes). Write everything about the problem without organizing: facts, feelings, questions, assumptions. Do not edit or filter. The goal is volume, not quality.

Step 2: Organize (5 minutes). Review your free writing and group similar items. Identify the core problem amid the details. Separate observations from interpretations.

Step 3: Problem Description (5 minutes). Write your problem description, answering two questions: What is it? Why is it a problem? For the full framework for developing this into a formal problem definition, see Problem Definition (Ch. 8).

Step 4: Review (three days later). Return with fresh eyes. A problem that felt urgent three days ago may look different. Unclear writing reveals unclear thinking. Revise the problem description based on what you see now.

Format Tips by Problem Type

Problem Type	What to Write	Your Notes
Simple	Answer "What is the problem?" and "Why does it matter?"	
Complex	Add where, when, and desired outcome	
Ongoing	Maintain a dedicated document you update as you learn	
Team	Write in a shared document so others can add perspectives	

Storage: Keep written problem descriptions in a place where you will review them periodically. As you gather information, problems evolve; your written statement should evolve too.

Journaling Template

Use this template when a problem recurs and you suspect a hidden pattern. Record daily entries for a period proportional to the problem's cycle; two to four weeks is typical for problems that recur weekly, but adjust to your situation. Review weekly for patterns. Each occurrence that feels isolated may be connected by a systemic trigger.

Daily Entry (5 to 10 Minutes)

Field	Prompt	Your Entry
Date and Time	[YYYY-MM-DD HH:MM]	
What Happened	Brief facts only. What occurred?	
Emotional Response	How did it affect you?	
Trigger	What triggered it, if applicable?	
What You Tried	Did you attempt a solution? What was the result?	

Weekly Review (10 Minutes)

Look back across the week's entries and answer these four questions:

1. Do any patterns emerge? Same time, same activity, same people involved?
2. Do any solutions you tried actually work?
3. What triggered the best days? The worst days?
4. What do you still not understand?

Timeframe: Most patterns emerge after capturing several full cycles of the recurring problem; for weekly recurrences, that typically means two to four weeks. Avoid concluding after just a few entries. Variation is normal; patterns require time to reveal themselves.

Data Tracking Protocol

Use this protocol when a problem involves quantities, frequencies, or trends. Follow the five steps to connect your measurement to the problem and establish a consistent tracking cadence.

The Five Steps

Step 1: Identify what to measure. Connect your measurement to your "what" and "why." If you suspect response times are slow, measure response times, not something easier but less relevant. Ask specific questions that transform subjective experiences into measurable data: "Team morale seems low" becomes "What is our voluntary turnover rate this quarter compared to last?" "The project feels behind" becomes "How many of the 12 planned milestones have been completed on schedule?"

Step 2: Choose a simple tracking method. A spreadsheet, a notebook, or a dedicated app. The best tool is the one you will actually use consistently.

Step 3: Set a collection cadence. Daily, weekly, or per-event, depending on the problem. Be specific: "Every Monday at 9:00" is better than "regularly."

Step 4: Review after a defined period. Two weeks is a reasonable minimum for most patterns to emerge. Avoid concluding from a few data points.

Step 5: Visualize trends. A simple chart transforms raw data into patterns the whole team can see, shifting the discussion from feelings to facts.

Intuition Log

Use this log to capture signals you cannot yet name or measure. This is the capture phase of Make Explicit (Ch. 3) Step 1. Over time, patterns emerge that convert intuitive knowledge into repeatable triggers for action. For recurring known problems, use the Journaling Template instead.

Questions to Draw Out What Your Intuition Already Knows

Before you start writing in the log, or when a new hunch surfaces, ask yourself:

- What does this remind me of?
- When have I seen something similar before?
- What patterns am I recognizing, even if I cannot name them yet?
- What specific observations triggered this feeling?

Your answers become the first rows in the log.

Log Format

Date	Hunch or Feeling	Specific Observations	Outcome (Later)	Pattern Identified

Converting Intuition to Action

Over time, your intuition log reveals the implicit signals you naturally detect. Once you identify a reliable pattern (for example, "longer email response times predict client dissatisfaction"), create an explicit trigger: "When a client's response time increases by more than 48 hours, schedule a check-in call within the week." This converts intuitive knowledge into a repeatable process.

Prompts for weekly review:

- What problems did I sense before the data proved me right?
- Which hunches did not pan out, and what signal were they actually about?
- Which patterns have I now seen twice; can they be promoted to an explicit trigger?

Decompose (Ch. 4)

Integration Contract Template

Complete this contract before teams begin working on decomposed parts in parallel. This is Step 2 of Decompose (Ch. 4). For each interface between parts, fill in the table below and schedule a joint verification at the midpoint, not the end.

The Three Questions

For each interface between decomposed parts, answer three questions:

1. **What does this part promise to deliver?** Deliverable format, quality standard, completion criteria.
2. **What does the receiving part need?** Required inputs, handoff conditions.
3. **How will we verify the connection works?** Joint review, pilot test, dry run.

Schedule a joint verification at the midpoint, not the end. Discovering incompatibilities at integration time is the most expensive possible moment.

Contract Format

Inter-face	Promis-ing Part	Receiv-ing Part	Deliver-able	Required Input	Verification Method	Midpoint Date

One row per interface. If two parts exchange several distinct data streams (for example commands, telemetry, and firmware updates), that is one row each, not one combined row.

Worked Example: Hardware/Firmware Voltage Agreement

A hardware team plans a 3.3-volt interface while the firmware team begins from an assumed 5-volt input. Before either team builds, the Integration Contract places those requirements in adjacent cells and exposes the mismatch. The teams agree on 3.3-volt logic and record the resolved interface in the filled row above. The joint bench test then verifies the agreement on a combined prototype. The contract prevents the mismatch from surviving until integration week.

Create Focus (Ch. 5)

Priority Matrix for Multiple Problems

Use this matrix when you face multiple problems simultaneously. Plot each problem on urgency and importance, then address them in priority order.

The matrix categorizes problems on two dimensions: **Urgency** (how soon must this be addressed?) and **Importance** (how much business impact if not addressed?).

How to Use the Priority Matrix

Step 1: List all your current problems without prioritizing yet.

Step 2: For each problem, evaluate two dimensions:

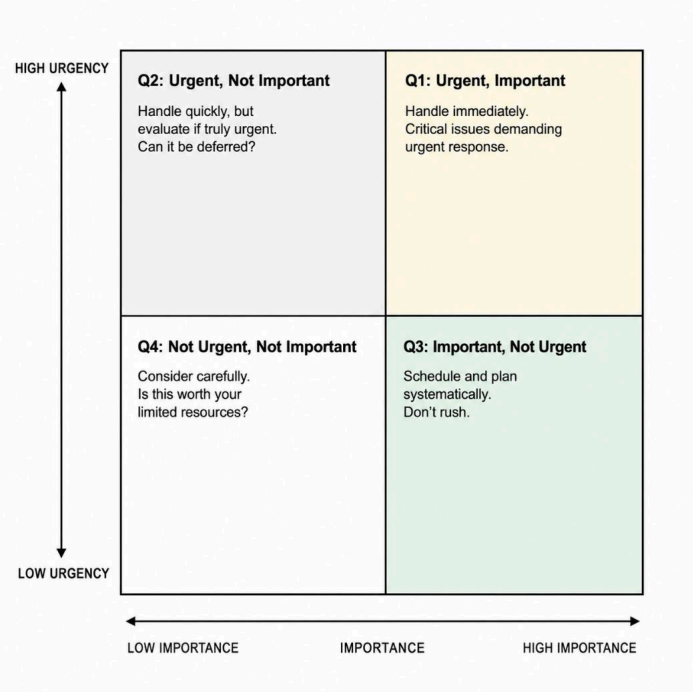
Urgency: How soon must this be addressed?

- High: Failing now or will fail within days (system down, disk full)
- Medium: Should be addressed within weeks (planned feature, upcoming deadline)
- Low: Can be addressed within months (documentation, nice-to-have improvements)

Importance: How much business impact if not addressed?

- High: Directly affects revenue, safety, customer experience, or critical operations
- Medium: Affects efficiency, productivity, or team capability
- Low: Nice-to-have improvements with minimal business impact

Step 3: Plot each problem on the matrix:



Priority Matrix

Step 4: Address problems in priority order:

- **Urgent-Important:** Handle immediately. These demand an urgent response.
- Example: System outage, security breach, product failure
- **Urgent-Unimportant:** Handle quickly, but evaluate if truly urgent. Can it be deferred or delegated?
- Example: Busy stakeholder's "urgent" request that is actually low impact
- **Non-Urgent-Important:** Schedule and execute systematically using the Primary Process (Ch. 13). Do not rush; plan properly.
- Example: Major feature with large business impact but not needed for three months
- **Non-Urgent-Unimportant:** Consider carefully. Is this worth your limited time and resources?
- Example: Nice-to-have improvements that do not affect operations

Example: Prioritizing Five Problems

Problem	Urgency	Impor-tance	Category	Action

Address Urgent-Important first, then Urgent-Unimportant, then Non-Urgent-Important, then Non-Urgent-Unimportant. This prevents "firefighting" on low-impact problems while important work waits.

Dependency Matrix for Problem Sequencing

Use after the Priority Matrix to determine execution order. Map dependencies between problems, identify independent work that can run in parallel, and find the critical path.

How to Use the Dependency Matrix

Step 1: List all your problems again.

Step 2: For each problem, ask: What other problems must be solved before this one can be started or completed?

Step 3: Map the dependency chain:

Evaluate (Ch. 6)

Assumptions Log

Start this log at the beginning of any problem-solving effort. List all assumptions, record evidence for and against each, and mark them verified only when tested against reality.

Assumptions Log Format

Assumption	Evidence Supporting It	Evidence Contradicting It	Verified? (Y/N)	Action

How to Use

1. List all assumptions at the start of the effort.
2. For each assumption, record the evidence supporting it and the evidence contradicting it.
3. Mark each assumption as verified or unverified. An assumption is verified only when you have tested it against reality.
4. Review the log before moving to the next phase of problem solving.

Q1: Problem Definition (Ch. 8)

Problem Definition Contract Template

Complete all seven components before committing resources. A good contract is specific enough that someone unfamiliar with the problem could read it and understand the gap, the target, and the boundaries.

The Seven Components

For each component, fill in the prompt.

1. Current State: "Right now, the measurable situation is: _____"

2. Desired State: "When solved, the measurable situation will be: _____"

3. Success Criteria: "We will know we are done when: _____"

4. Constraints: "The solution must respect these limitations: _____"

5. Stakeholders: "Who is affected or has decision authority: _____"

6. Problem Owner: "The single person who decides 'this is solved': _____"

7. Scope: "Explicitly in and out: _____"

Q2: Problem Analysis (Ch. 9)

Root Cause Analysis Quick Reference

Use this reference to select the right root cause analysis tool. Start with the selection guide, then follow the pocket reference for your chosen method.

Tool Selection Guide

Question	If Yes	If No
Is the cause-effect chain likely linear (A causes B causes C)?	**Five Whys**	Continue
Are there multiple potential cause categories (people, process, equipment, environment)?	**Fishbone Diagram**	Continue
Does the "fix" seem to make things worse, or does the problem keep returning after being solved?	**Systems Thinking**	Continue
Does the problem occur somewhere in a clear sequence of steps or components?	**Binary Search**	Start with Five Whys

When uncertain, start with the Five Whys. It is the fastest tool and will reveal whether the problem is linear (continue with Five Whys) or requires a different approach (switch tools).

Five Whys: Pocket Reference

Best for: Linear cause-effect chains where one thing leads to another.

How: Start with the symptom. Ask "Why?" Repeat until you reach a cause you can act on. "Five" is a guideline; stop at three or continue to seven as needed.

Watch for: If the chain branches ("it could be A or B"), switch to Fishbone.

Fishbone Diagram: Pocket Reference

Best for: Problems with multiple potential causes across different categories.

How: Place the problem at the head. Draw bones for major categories (People, Process, Equipment, Environment). Brainstorm causes within each category. Identify which causes explain the observed symptoms.

Watch for: If one bone contains a feedback loop ("fixing A makes B worse"), switch to Systems Thinking.

Systems Thinking: Pocket Reference

Best for: Problems where causes loop back on themselves, or where the obvious fix makes things worse.

How: Map the causal loop. Identify where the output of one action feeds back as input. Find the weakest link in the loop where an intervention breaks the cycle.

Watch for: Delays between cause and effect. A fix today may create a worse problem months later.

Binary Search: Pocket Reference

Best for: Problems in a clear sequence (pipeline, process stages, code execution) where you can test at midpoints.

How: Test the midpoint. Is the problem before or after? Eliminate half. Repeat. A 1,000-item search takes 10 tests instead of 1,000.

Watch for: Multiple independent faults. Binary search finds the first one. Fix it, re-test, and search again.

Q3: Solution Design (Ch. 10)

Solution Evaluation Scorecard

Use after generating multiple solution options and before selecting one. List three to five genuinely different approaches, score them on weighted criteria, and select the highest scorer.

Step 1: List Your Options

Write down three to five genuinely different approaches. Minor variations of the same idea do not count. If you have only one option, you have not finished generating.

Step 2: Eliminate Against Hard Constraints

Before scoring, test each option against the hard constraints from your Q1 Problem Definition Contract. Any option that violates a constraint is eliminated, regardless of how promising it looks.

Option	Constraint Violated?	Status
Option A: _____	Yes / No. If yes, which: _____	Eliminated / Survives
Option B: _____	Yes / No. If yes, which: _____	Eliminated / Survives
Option C: _____	Yes / No. If yes, which: _____	Eliminated / Survives

Step 3: Score Surviving Options

Rate each surviving option on four criteria. Use a 1-5 scale (1 = poor, 5 = excellent).

Criterion	Weight	Option A	Option B	Option C
Risk: How likely is failure, and how costly?	_____	_____	_____	_____
Cost: Direct costs (licenses, hardware, labor) + indirect costs (training, maintenance)	_____	_____	_____	_____
Time: Can it be delivered within the deadline?	_____	_____	_____	_____
Resources: Do you	_____	_____	_____	_____

have the people, skills, and infra-structure?				
Weighted Total		_____	_____	_____

Setting weights: Assign a weight of 1-3 to each criterion based on what matters most for this problem. If failure would be catastrophic, weight Risk at 3. If the deadline is immovable, weight Time at 3. If all criteria matter equally, use weight 1 for all.

Calculating totals: For each option, multiply each score by its weight and sum the results.

Step 4: Check Effectiveness

A high feasibility score does not guarantee the solution works. After scoring, verify: **Does the highest-scoring option actually solve the problem?** Will it meet the Q1 success criteria?

If the top scorer falls short on effectiveness, consider:

- Combining compatible approaches (approaches that address different aspects of the same root cause)
- Selecting a lower-scoring but more effective option and mitigating its weaknesses
- Prototyping to validate uncertain effectiveness estimates

Step 5: Decide

Decision	When
Select and proceed	One option clearly leads on score and effectiveness
Prototype first	Top options are within 2 points of each other, or the leading option has unvalidated assumptions
Return to Step 1	No surviving option meets the success criteria

Solution Design Tools

Four structured techniques for generating solution options. Use in Q3 after identifying the root cause (Q2) and before evaluating options with the Solution Evaluation Scorecard.

Choosing the Right Tool

Each tool fits a different starting position. Use this table to choose:

Starting Position	Tool	Why
You have an existing process or product that needs improvement	SCAMPER	Systematically generates variations on what already exists
You face a contradiction: improving one thing worsens another	TRIZ	Resolves trade-offs using principles proven across industries
Existing solutions are inadequate, or the problem is genuinely novel	First Principles	Discards inherited assumptions and rebuilds from fundamental truths
You need broad creative input from a team with diverse perspectives	Six Thinking Hats	Structures group discussion so every angle gets heard

If you are unsure, start with SCAMPER. It is the fastest and produces the most options. If SCAMPER generates only incremental variations and you need a fundamentally different approach, move to First Principles. If the core challenge is a trade-off that seems impossible to resolve, use TRIZ. These tools can also be combined: use SCAMPER to generate a broad set of options, then apply TRIZ to resolve contradictions in the most promising candidates.

SCAMPER

What it is: A checklist of seven idea-spurring questions that systematically modify an existing idea or process. Formalized by Bob Eberle based on Alex Osborn's original brainstorming questions.

The seven questions:

Letter	Question	What it does
S	Substitute	Replace a component, material, person, or process step
C	Combine	Merge two functions, steps, or resources into one
A	Adapt	Borrow an approach from another domain or context
M	Modify	Change size, shape, frequency, intensity, or sequence
P	Put to another use	Repurpose an existing resource for a different function
E	Eliminate	Remove a step, component, or requirement entirely
R	Reverse	Flip the order, swap roles, or invert the process

When to use: When you have an existing process or product and need to generate variations. SCAMPER works best as a starting point: it produces many ideas quickly, which you then filter.

Six Thinking Hats

What it is: A parallel thinking technique developed by Edward de Bono. Instead of each person arguing from their own position, the entire group puts on the same "hat" at the same time and examines the problem from that single perspective before switching to the next.

The six hats:

Hat	Perspective	Question it answers
White	Facts and data	What do we know? What data do we have? What is missing?
Red	Intuition and feelings	What does my gut say? What feels right or wrong?
Black	Risks and caution	What could go wrong? What are the dangers?
Yellow	Benefits and optimism	What is the best case? What value does this create?
Green	Creativity and alternatives	What else could we try? What if we changed the approach?
Blue	Process and control	Where are we in the discussion? What should we do next?

When to use: When a group discussion is stuck in debate, when strong personalities dominate, or when the team keeps circling the same arguments. The hats force everyone to think in the same direction at the same time, which prevents adversarial debate and surfaces perspectives that quiet team members might not voice.

TRIZ

What it is: A systematic innovation method developed by Genrich Altshuller from a review of roughly 200,000 patents, of which he judged about 40,000 genuinely inventive. TRIZ identifies 40 inventive principles that recur across industries and problem types. Instead of relying on random creativity, TRIZ asks: "How have similar contradictions been resolved before?"

Core idea: Most problems contain a contradiction: improving one parameter worsens another. TRIZ resolves contradictions by applying principles that have historically worked for similar trade-offs.

When to use: Engineering, process design, and technical problems where the challenge is a physical or organizational contradiction (faster but heavier, stronger but more expensive, more thorough but slower). TRIZ is less effective for purely social or political

problems.

Selected principles (from the 40):

Principle	Idea	Example application
Segmentation	Divide an object or process into independent parts	Split a monolithic approval process into parallel tracks
Prior Action	Perform a required action in advance	Pre-stage equipment before a maintenance window
Feedback	Introduce or improve feedback to adjust behavior in real time	Add real-time quality sensors to a production line
Intermediary	Use an intermediate object to transfer or carry out an action	Route customer complaints through a triage bot before a human agent
Continuity of Useful Action	Eliminate idle time; make all parts work at full capacity continuously	Stagger shift changes so production never stops completely

First Principles Thinking

What it is: A reasoning method that deconstructs a problem to its fundamental truths, discards all inherited assumptions, and rebuilds the solution from the ground up. Where SCAMPER modifies what exists and TRIZ resolves contradictions, First Principles starts from scratch.

The process:

1. **Identify the goal** at the most fundamental level. Strip away jargon and inherited methods.
2. **List every assumption** that currently shapes how the goal is pursued.
3. **Question each assumption.** Is it a physical law, a regulation, or just "how we have always done it"?
4. **Discard assumptions that are not fundamental.** What remains is the foundation.
5. **Rebuild the solution** from only the fundamental truths.

When to use: When existing solutions are inadequate, when the problem is genuinely novel, or when incremental improvements have plateaued. First Principles is expensive in thinking time. Use it when the payoff justifies the effort.

Q4: Solution Implementation (Ch. 11)

Implementation Planning Checklist

Complete before beginning Q4 execution. Work through all five parts: setup, task translation, weekly drift checks, obstacle handling, and final verification.

Part 1: Pre-Implementation Setup

- [ ] Q3 design document is complete and reviewed
- [ ] Q1 success criteria are written at the top of the implementation plan (visible, not buried)
- [ ] Rollback plan is documented: how to undo each major change if it fails
- [ ] Rollback plan has been tested or reviewed for feasibility
- [ ] Stakeholders who signed off on the Q1 problem definition are informed that implementation begins

Part 2: Task Translation (Preventing Translation Loss)

For each significant task in the implementation plan, verify:

Task	What to Do	Why This Task Exists (Design Intent)	Which Q1 Criterion It Serves
1			
2			
3			

The "Why" column is the Translation Loss prevention mechanism. When implementers understand the design logic, they make better decisions when reality forces deviation from the plan. If you cannot fill in the "Why" for a task, the design intent has already been lost; return to the Q3 design owner and recover it before proceeding.

Part 3: Drift Detection (Weekly Self-Check)

Answer these three questions at the end of each work period (daily for short projects, weekly for longer ones):

1. **Am I still solving the original problem?** Compare current activities to Q1

success criteria. If the answer is no, you are drifting.

2. **Have I added anything that was not in the Q3 design?** If yes, is it necessary for meeting Q1 criteria, or is it scope creep?

3. **Can I still explain how today's work connects to the Q1 problem definition?** If the connection requires more than two sentences, the link is weakening.

Any "no" or uncertain answer is a signal to pause and realign before continuing.

Part 4: Obstacle Decision Tree

When implementation stalls, work through these four questions in order. The first "no" tells you where to return.

Question	If No	Action
1. Do I know what "done" means?	Requirements are ambiguous	Return to Q1
2. Am I solving the right cause?	Root cause analysis was wrong	Return to Q2
3. Does the design work?	Approach is flawed (verify it is not Translation Loss first)	Return to Q3
4. Is there another way to execute?	No execution path remains	Return to Q3 to revise the design

Part 5: Verification

- [ ] Solution tested against normal use cases, edge cases, and error scenarios
- [ ] Q1 success criteria checked with objective data (not "it seems to work")
- [ ] Stakeholders have confirmed the problem is resolved
- [ ] Results documented: before and after measurements
- [ ] Lessons learned captured while fresh

The Processes (Ch. 12)

Process Selection Decision Guide

Answer the four decision tree questions in order at the start of any problem-solving effort. Stop when an answer names a process. Use the switch signals to detect mismatches during execution.

The Decision Tree

Answer each question in order. Stop when an answer names a process.

1. Is failure catastrophic or irreversible?

- *Examples:* Patient safety, regulatory compliance, one-shot deployment, structural engineering
- Yes: **First Time Right Process.** Stop here.
- No: Continue.

2. Is the problem familiar, in a context you have navigated before?

- Not just "have you seen something like this?" but "have you solved this exact type of problem, in the same environment, with the same constraints?"
- Yes: **Primary Process.** Stop here.
- No: Continue.

3. Can you describe what a working solution looks like?

- Can you name the shape a working solution would take, even in broad terms?
- No: **Exploratory Process.** Stop here.
- Yes: Continue.

4. Can you afford repeated attempts?

- You are here because the problem is unfamiliar, so the work left is learning. Judge affordability in time as well as money, against this project's timeline.
- Yes: **Iterative Process.**
- No: **First Time Right Process.** Use bounded simulations, prototypes, consultation, or other safe tests to reduce uncertainty before the full-cost commitment. Switch to **Primary Process** only if that learning makes the remaining work familiar and recoverable.

When answers conflict across questions, the higher-risk answer wins.

Signs You Chose Wrong (Switch Signals)

Signal	What It Means	Switch To
Reviews catch nothing; every gate passes unchanged	Rigor without risk. Safeguards are overhead, not protection.	Primary or Iterative
Each phase reveals problems that should have been anticipated	Moving through familiar motions in unfamiliar territory	First Time Right or Exploratory
Improvements have flattened but the team keeps iterating	Iteration without convergence	Primary (for execution) or First Time Right (for verification)
The team locked onto a solution early without testing alternatives	Premature convergence; commitment without exploration	Exploratory
Experiments produce data but no actionable direction	Exploration has become drift	Iterative (if a direction exists) or stop and redefine the problem

Process Selection Matrix

The Selection Matrix is a profile view of the four processes. Read by column to see what each process assumes about the problem; read by row to see how a single dimension separates the four options. Use it as a cross-check on the Decision Tree, not as a replacement for it.

Dimen-sion	Primary	First Time Right	Iterative	Explora-tory
Familiarity	High: solved before, same context	Any: overridden by risk	Moderate: general direction known	Low: unknown territory
Reversi-bility	Reversible: mistakes can be undone	Irrever-sible: no second chance	Reversible: each cycle is low-cost	Reversible: experiments are disposable
Failure tolerance	Moderate: errors are costly but recoverable	Zero: the full-cost commitment cannot fail	High: imperfect versions are acceptable	High: experiments are designed to produce learning, though some are inconclusive
Solution space	Known: proven solutions exist	Any: uncertainty must be reduced safely before commitment	Partially known: details emerge through feedback	Unknown: the path must be discovered

Read the matrix by column to see the profile of each process. Read it by row to see how a single dimension separates the four options. When the matrix points to different processes on different rows, use the Decision Tree to break the tie.

The Primary Process (Ch. 13)

Primary Process Execution Template

A structured walkthrough from Q1 through Q4 for the Primary Process. Fill in each quadrant's elements, apply the four rules, and pass each Quality Gate before proceeding.

Q1: Define the Problem

Element	Your Entry
Problem statement (one to two sentences: what is happening, what should be happening, why it matters)	
Success criteria (measurable, with threshold and duration)	
Scope (what is in, what is explicitly out)	
Problem owner (single person who decides "this is solved")	

Rules applied: Make Explicit (state the problem in writing, not just in your head). Decompose (is this one problem or several? If several, which one are you solving first?). Create Focus (which aspect matters most right now? If multiple problems compete, use the Priority Matrix from Create Focus (Appendix D)). Evaluate (does the problem owner agree with this definition?).

Quality Gate 1 self-check: Can you state the problem in one to two sentences with numbers? Do affected people agree this is the problem? Do you know what success looks like? If any answer is no, stay in Q1.

Q2: Analyze the Root Cause

Element	Your Entry
Initial hypothesis (what do you think causes the problem?)	
Evidence supporting it	
Evidence contradicting it	
Alternative explanations considered and ruled out	
Root cause (the condition whose removal prevents recurrence)	

Rules applied: Make Explicit (write down your hypothesis before testing it).

Decompose (break the problem into components; where does the failure occur?). Create Focus (which cause explains the largest share of the problem?). Evaluate (does the evidence support your root cause, or are you believing your theory over the data?).

Quality Gate 2 self-check: Can you explain the root cause with evidence? Have you ruled out alternatives? Would a colleague reviewing your analysis reach the same conclusion? If any answer is no, stay in Q2.

Q3: Design the Solution

Element	Your Entry
Options considered (at least two genuinely different approaches)	
Selection rationale (why this option over the others)	
Hard constraints respected (budget, time, resources, regulations)	
How you will verify it worked (measurement plan)	

Rules applied: Make Explicit (write down the options and the selection criteria). Decompose (break the solution into implementable parts). Create Focus (resist adding features that do not address the root cause). Evaluate (does the chosen solution address the root cause identified in Q2?).

Quality Gate 3 self-check: Does the solution address the Q2 root cause? Can you implement it within constraints? Do you know how you will measure success? If any answer is no, stay in Q3.

Q4: Implement and Verify

Element	Your Entry
Implementation steps (actionable tasks with owners and deadlines)	
Progress tracking method	
Verification results (before and after data against Q1 success criteria)	
Stakeholder sign-off (does the problem owner agree the problem is solved?)	

Rules applied: Make Explicit (capture the implementation plan in writing). Decompose (divide the plan into tasks small enough to complete in a day or two). Create Focus (execute one step at a time; track progress against Q1 criteria, not just the task list). Evaluate (verify with data that the problem is solved).

Quality Gate 4 self-check: Do the metrics confirm the problem is resolved? Do stakeholders agree it is solved? What did you learn? If either of the first two answers is no, use the Obstacle Decision Tree from Q4: Solution Implementation (Appendix D) to determine which quadrant to revisit.

Documentation Checkpoint

After Q4, record these three items (two minutes is enough for routine problems):

1. **What was the problem and root cause?** (One sentence each.)
2. **What solution worked?** (One sentence.)
3. **What would you do differently?** (One sentence.)

For significant problems, use the full Problem-Solving Journal template below.

Problem-Solving Journal

Complete one entry after each significant problem-solving effort. Review monthly for recurring root causes, common wrong assumptions, and effective approaches.

Problem-Solving Journal Entry Template

Date: [YYYY-MM-DD]

Problem: [One-sentence summary]

Q1: Problem Definition

- Current state: [What is happening?]
- Desired state: [What should be happening?]
- Impact: [Why does this matter?]
- Success criteria: [How will we know it is solved?]

Q2: Problem Analysis

- Initial hypothesis: [What did we think caused it?]
- Root cause found: [What actually caused it?]
- Evidence: [How did we verify the cause?]
- Wrong assumptions: [What did we assume that was incorrect?]

Q3: Solution Design

- Chosen solution: [What approach did we select?]
- Why this solution: [Why did we pick it?]
- Time invested: [Hours spent on Q1-Q3]

Q4: Solution Implementation

- Outcome: [Did the solution work?]
- Success metrics: [Before/after numbers]
- Time invested: [Hours spent on Q4]

Reflection and Learning

- What worked well: [What approaches proved effective?]
- What would we do differently: [How would we approach this again?]
- Lessons learned: [What should we remember?]
- Similar problems: [Have we solved something like this before?]

The First Time Right Process (Ch. 14)

Gate Evaluation Checklist

Answer these five questions for each Quality Gate in the project. Redesign the gate on any "no"; two or more "no" answers signal a systemic gate problem.

For each Quality Gate, answer these five questions:

1. **Does this gate actually stop problems?** Review historical gate outcomes. How many times has this gate caught a genuine issue? If the answer is zero across 20 to 30 projects, the gate may be overhead rather than protection.

2. **Is the gate independent?** Can the reviewing authority genuinely block progression, or do they rubber-stamp approvals? If the reviewer has no real authority to stop the work, the gate is theater.

3. **Is the gate proportional to risk?** A two-day review for a low-risk transition is overhead. A two-day review for a safety-critical transition is insufficient. Match gate rigor to actual consequence of failure.

4. **Does the gate address real stakeholder concerns?** If stakeholders do not care about what the gate checks, it is bureaucracy. If stakeholders consistently ask "Why are we not checking X?", the gate misses what matters.

5. **Can the gate be streamlined without losing rigor?** A three-day expert review can sometimes be condensed to a two-hour focused review with better-targeted questions. Look for efficiency, not shortcuts.

Scoring: If any answer is no, redesign the gate before proceeding. Two or more "no" answers indicate a systemic gate design problem; review the entire gate structure, not just the individual gate.

Reviewer Authority Matrix

Use this matrix to match reviewer authority to the consequence of failure. Pick the lightest gate type that fits the risk; escalate the gate type when uncertainty grows or earlier gates reveal unexpected issues.

Gate Type	Who Reviews	Authority Level	Use When
Self-check	The person doing the work	Can proceed if satisfied	Low-risk transitions within a familiar domain
Peer-check	A colleague with relevant expertise	Can flag concerns; escalates if unresolved	Moderate-risk transitions or unfamiliar territory
Formal	An independent reviewer or review board	Can block progression until criteria are met	High-risk transitions where failure is irreversible

Choose the lightest gate type that matches the consequence of failure at that transition. Escalate gate type when uncertainty increases or when earlier gates have revealed unexpected issues.

Escalation Decision Rule

Apply this rule whenever stakes change mid-project. Escalate to First Time Right when an error becomes catastrophic or irreversible, introduces safety, legal, or regulatory exposure, affects substantially more people, or otherwise becomes too costly to absorb. A tenfold increase in cost is one warning signal, not a required threshold. Revalidate completed work against First Time Right gate criteria before proceeding.

Use this decision rule when a project started with the Primary Process but new information suggests higher stakes:

1. During Q1 or Q2, ask: "If we get this wrong, can we fix it?"
2. Stop the current process if the answer changes from "yes" to "no," or if new safety, legal, regulatory, stakeholder, or cost consequences make an error too costly to absorb.
3. Escalate to First Time Right. Revalidate all completed work against First Time Right gate criteria before proceeding.

It is cheaper to escalate early than to discover too late that light enforcement was insufficient.

The Iterative Process (Ch. 15)

Iterative Process Review Templates

Complete after each iteration cycle. The Results Review evaluates what you learned; the Process Retrospective evaluates how you worked. Use the stopping criteria to decide whether to continue.

Results Review Template

Conduct the Results Review to evaluate outcomes and what you learned about the problem and solution.

1. **Iteration Goal Achievement:** Did we achieve the goal set at the start of this iteration? If not, why not?
2. **Outcome Measurement:** What were we trying to measure or improve? What do the actual measurements show?
3. **Problem Understanding:** What did we learn about the problem in this iteration? Did our understanding change?
4. **Root Cause Insights:** What did this iteration teach us about the underlying cause? Are we closer to the true root cause?
5. **Solution Effectiveness:** What did we learn about the solution's effectiveness? What worked? What did not?
6. **Customer or User Feedback:** What feedback did we get from actual users or stakeholders? What surprised us?
7. **Next Iteration Direction:** Based on results, what should we focus on in the next iteration?

Process Retrospective Template

Conduct the Process Retrospective to evaluate how effectively the team worked and identify process improvements for the next iteration.

1. **Planning Effectiveness:** Was our iteration plan realistic? What planning assumptions were wrong?
2. **Team Coordination:** How well did team members collaborate? Were handoffs smooth?

3. **Decision Speed:** How quickly did we make decisions? What slowed us down?
4. **Scope Adherence:** Did we stay within the planned scope, or did scope creep occur?
5. **Tool/Process Effectiveness:** Which tools or processes helped? Which hindered?
6. **One Improvement:** What is the single most important process change for the next iteration?

Know When to Stop Iterating: Decision Framework

After completing both reviews, use the diagnostic questions below to decide whether to continue iterating, pivot, or stop and deploy.

Quick Diagnostic

Answer these five questions first. Any "No" points you to the relevant stopping criterion below.

1. **Progress?** Have we made meaningful progress toward solving the problem? If no, see Criterion 2 (Diminishing Returns) or Criterion 5 (New Information).
2. **Learning?** Did we document what we learned, and can we apply it next iteration? If no, conduct deeper reflection before continuing.
3. **Direction?** Is our next direction clear based on results? If no, conduct brief exploratory work to clarify before selecting a direction.
4. **Resources?** Do we have time, budget, and team for another iteration? If no, see Criterion 4 (Budget or Time Constraints).
5. **Value?** Is there still significant value in continuing, or have we reached diminishing returns? If diminishing, see Criterion 2.

If all five answers are "Yes," proceed with the next iteration. Otherwise, apply the criteria below.

The Exploratory Process (Ch. 16)

Experiment Documentation Template

Complete before and after every experiment. The pre-experiment checklist ensures you use the cheapest valid approach. The experiment record captures hypothesis, setup, results, and learning.

Pre-Experiment Verification

Before running each experiment, verify you are using the cheapest and fastest approach that still produces trustworthy results.

Proof of Concept

- [ ] Identified the single uncertainty being tested
- [ ] Scope limited to testing only that uncertainty
- [ ] Target duration: a small fraction of total exploration time (e.g., four to eight hours in a month-long effort)
- [ ] No production-quality work included

Prototyping and Small-Scale Testing

- [ ] Prototype works end-to-end but cuts corners on quality
- [ ] Target duration: 5 to 15 percent of total exploration timeline
- [ ] Small-scale test uses a representative sample (not full rollout)
- [ ] Success and failure criteria defined before testing

Time-Boxing

- [ ] Strict deadline set before the experiment starts
- [ ] Evaluation criteria defined for the deadline
- [ ] Team agrees to stop and evaluate at the deadline regardless of status

Resource Constraints

- [ ] Using fewest people needed
- [ ] Using simplest tools that produce trustworthy results
- [ ] Large investments reserved for candidates that survived early screening

Experiment Record

Experiment number: [Sequential number]

Date: [YYYY-MM-DD]

Hypothesis: [What do you expect to happen, and why?]

Setup

- What are you testing: [The specific variable or approach]
- Method: [How you will run the experiment]
- Scale: [Sample size, duration, scope]
- Cost: [Time, money, resources required]

Measurements

- Metric: [What you will measure]
- Measurement method: [How you will measure it]
- Baseline: [Current value before the experiment]
- Success range: [What result indicates the hypothesis is supported?]
- Failure range: [What result indicates the hypothesis is refuted?]

Results

- Actual result: [What happened]
- Hypothesis supported: Yes / No / Partially
- Unexpected findings: [Anything you did not anticipate]

Learning

- What did this experiment teach us: [Key insight]
- What should we test next: [Next hypothesis or direction]
- What should we stop pursuing: [Approaches this result eliminates]

Bringing It All Together (Ch. 18)

Prevention Log Template

Log every prevented problem with evidence and estimated value preserved. Review monthly with your team; present the quarterly summary to leadership.

Prevention Log Entry Template

Date: [YYYY-MM-DD]

Potential problem: [What would have happened without intervention?]

Early warning: [What signal alerted you?]

Action taken: [What did you do to prevent it?]

Result: [What did not happen?]

Value preserved: [Estimated cost avoided, using conservative numbers]

Evidence: [How do you know this was a real risk, not speculation?]

Prevention Log Table

Maintain a running table for quick reference and quarterly reporting:

Date	Potential Problem	Early Warning	Action Taken	Result (What Did Not Happen)	Value Preserved

Quarterly Review Template

Use this template when presenting prevention results to leadership:

Period: [Quarter and year]

Total incidents prevented: [Count]

Total value preserved: [Sum of conservative estimates]

Top three preventions:

1. [Highest-value prevention with brief description]
2. [Second highest]
3. [Third highest]

Patterns identified: [Recurring risk categories that suggest systemic investment]

Recommended investments: [Based on patterns, where should the organization invest to reduce recurring near-misses?]

Quantification guidelines:

- Use conservative estimates; round down, not up
- Base estimates on comparable past incidents, not worst-case projections
- When exact numbers are unavailable, use ranges ("USD 20,000 to 40,000")
- Only log preventions with evidence of real risk, not speculative scenarios

Common Pitfalls Diagnostic Checklist

Check this list during or after any problem-solving effort. Each pitfall maps to a specific Rule or quadrant violation, so the diagnosis tells you exactly where to look.

Pitfall	What Happens	Which Rule or Quadrant Is Violated	How to Avoid It
Insufficient Decomposition	You try to solve an overwhelming problem as a single unit; no progress is made	Rule 2 (Decompose)	If a problem feels overwhelming, decompose again until each piece is addressable
No Evaluation	You implement a solution but never measure whether it worked; mistakes repeat	Rule 4 (Evaluate); skipping Q4	When designing a solution, simulta-neously design its measurement; define success criteria in Q1
Choosing the Wrong Process	You use the First Time Right Process for experimen-tation or the Exploratory Process for safety-critical work	Process-context mismatch	Match the process to the risk profile: Primary for routine, First Time Right for high stakes, Iterative for complexity, Exploratory for uncertainty
Working Alone on Collaborative Problems	You design a solution without involving the people who must implement or live with it; they reject it	Skipping stake-holder identifi-cation in Q1	In Q1, identify who is affected; involve them in Q2 analysis and Q3 design so they own the solution
Solving Symptoms Instead of Root Causes	You replace a failed component without understanding why it failed; it fails again	Rushing through Q2	Use Q2 to identify root causes; ask "why" repeatedly; verify your solution addresses the cause, not the symptom
Over-complicating Solutions	Your solution is hard to implement and harder to maintain; complexity signals incomplete understanding	Failing to apply Rule 3 (Create Focus)	Ask: "What is the simplest approach that addresses the core issue?" Prefer proven solutions over clever ones

Now You Know, So Use It (Ch. 19)

Know Your Style: Self-Assessment

Fill in the questionnaire below before reading the profiles. Score each statement from 0 (never) to 5 (always), sum each quadrant, then divide each quadrant total by the grand total to get your percentage distribution across the four quadrants of the Map: Problem Definition (Q1), Problem Analysis (Q2), Solution Design (Q3), and Solution Implementation (Q4). The profile descriptions tell you what your distribution means. To understand what the quadrants are and how to strengthen your weaker ones, read The Fastest Way.

Every person has a natural style. Some excel at defining and understanding problems (Q1 and Q2). Others shine at getting things done (Q3 and Q4). Neither is better than the other; they are different. Recognizing your style lets you play to your strengths and shore up your weak spots, whether by deliberate practice or by teaming up with people who complement you.

Self-Assessment Questionnaire

Read each statement below carefully. For each one, assign a score from 0 to 5, where 0 means "Never" and 5 means "Always," based on how accurately the statement describes you in your problem-solving work.

Quadrant	Statement	Score (0-5)
Q1: Problem Definition	I clarify precisely what the problem is before trying to solve it.	_____
	I confirm that I am solving the right problem, not just addressing a symptom.	_____
	I write down the problem statement to ensure clarity.	_____
	I validate my understanding of the problem with others.	_____
	I define what "success" looks like before starting work.	_____
Q1 Total		_____

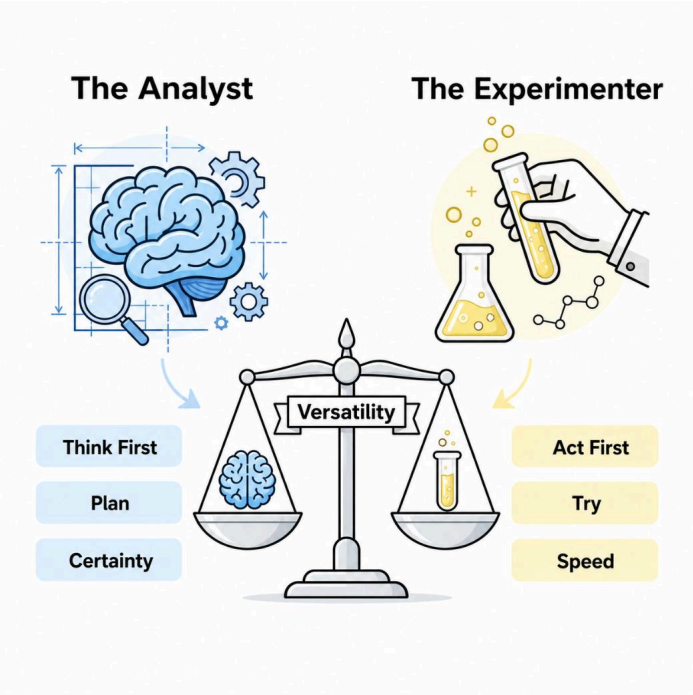
Q2: Problem Analysis	I take time to understand *why* the problem is happening.	_____
	I use tools like the "Five Whys" and data analysis to identify root causes.	_____
	I break complex problems down into smaller, manageable parts.	_____
	I challenge my initial assumptions about the cause.	_____
	I look for patterns and evidence rather than guessing.	_____
Q2 Total		_____
Q3: Solution Design	I generate multiple options before choosing a solution.	_____
	I evaluate trade-offs (costs, risks) for each potential solution.	_____
	I aim for the simplest solution that works (Create Focus).	_____
	I create a clear plan for implementing the solution.	_____
	I check if the solution actually addresses the root cause.	_____
Q3 Total		_____
Q4: Solution Implemen-tation	I take action rather than analyzing forever.	_____
	I test my solution in small steps (fail fast) rather than all at once.	_____
	I measure the results to see if the problem is actually solved.	_____
	I finish what I start and ensure the solution is sustained.	_____
	I learn from the outcome, even if it failed.	_____
Q4 Total		_____
GRAND TOTAL	Sum of Q1 + Q2 + Q3 + Q4	_____

For each quadrant, divide its total by the Grand Total and multiply by 100. The four percentages form your style distribution.

Example: Q1 = 20, Q2 = 22, Q3 = 15, Q4 = 18. Grand Total = 75. Distribution: Q1 = 27 percent, Q2 = 29 percent, Q3 = 20 percent, Q4 = 24 percent. This reveals a slight Analyst tendency (Q1+Q2 = 56 percent) versus Builder tendency (Q3+Q4 = 44 percent).

In Problem Solving When You Are Not Alone (Ch. 17), you saw five role preferences for team composition: Analysts, Creatives, Experimenters, Implementers, and Evaluators. Those preferences describe contributions to team work; this personal profile is a broader lens on where your energy concentrates across all four quadrants when you work alone. Analysts often lean toward the Analyst profile. Creatives, Experimenters, and Implementers may lean toward the Builder profile, depending on their full distribution. Evaluators work across all quadrants and do not map automatically to either profile. A Balanced Expert shows even energy across all quadrants.

Three Problem-Solving Profiles



Analyst versus Experimenter

The Analyst (High Q1+Q2, Low Q3+Q4)

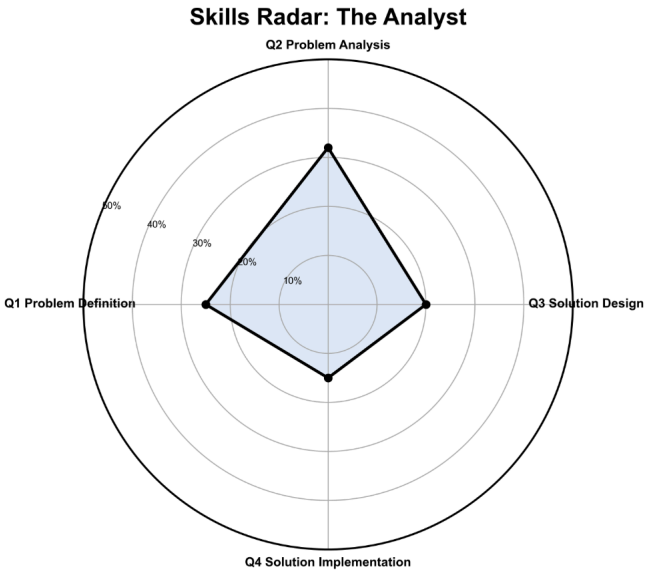
You are in the third meeting about the same problem. Everyone else wants to start building, but you can see two unresolved assumptions that will surface as bugs in week three. You are the colleague whose slide deck has forty pages and no recommendation. You excel at analysis and planning but may struggle to move into concrete action.

Strengths: Defining and understanding complex issues is where you excel. Good questions come naturally, and you challenge assumptions rather than accept them. Solutions wait until understanding is complete.

Challenge: The risk is Analysis Paralysis: spending weeks analyzing when days would

suffice. Decision-making can stall because there is always more information to gather. See Problem Analysis (Ch. 9) for the full treatment.

Development: Set explicit analysis deadlines and time-box your Q2 work. Partner with Builders who can push you toward action. Perfect information is impossible, and waiting for it guarantees failure.



Skills Radar: The Analyst

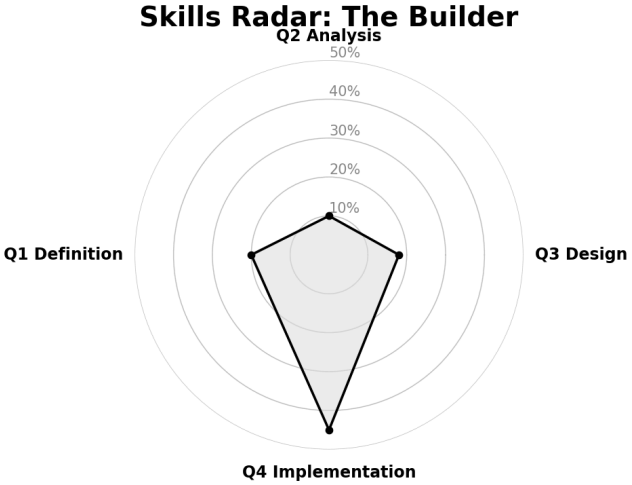
The Builder (High Q3+Q4, Low Q1+Q2)

The meeting is at minute forty-five and still defining requirements. Your laptop is open. You have already sketched three possible solutions and you are itching to test the first one. You are the colleague who has already opened a terminal while everyone else is still explaining the problem. You observe problems and jump to implementation, often skipping analysis and design.

Strengths: Getting things done is where you shine. Execution is quick; tests launch early; results feed the next round. Teams appreciate the bias toward action and delivery.

Challenge: The risk is Solving the Wrong Problem: rushing to action before the root cause is clear. Significant resources may go toward fixing symptoms instead of causes.

Development: Force yourself to pause before acting: write down your problem statement (Q1) and ask "why" at least three times (Q2) before designing. Partner with Analysts who can help you understand problems deeply. Speed built on misunderstanding is not progress.



Skills Radar: The Builder

The Balanced Expert (Even Q1+Q2+Q3+Q4)

Strengths: You can navigate the entire problem-solving lifecycle independently. You know when to spend time analyzing (and when to stop), and when to push forward with action. Your balanced approach enables you to work effectively alone and serve as a bridge between Analysts and Builders in teams.

Development: Focus on maintaining balance. Be aware that in high-pressure situations, you may default to your stronger quadrant. Continue practicing all four quadrants to prevent skill atrophy in weaker areas.

For principles of effective team problem solving, including how to compose teams with complementary styles, see Problem Solving When You Are Not Alone (Ch. 17).

Personal Playbook Template

Build this playbook over four weeks of deliberate practice, then update it quarterly. Each section records one dimension of your problem-solving style: your profile, the processes you default to, the techniques that have worked, the patterns you have noticed. Every field names a concept from the Fastest Way (Rules, quadrants of the Map, Processes). The book explains what each concept is and how to use it; this page is where you record how you apply them.

Your playbook is the document you consult when you face your next significant problem. Build it from your journal entries, self-assessment results, and four weeks of practice.

My Problem-Solving Profile

- Style: [Analyst / Builder / Balanced Expert, from Self-Assessment]
- Strongest quadrant: [Q1 / Q2 / Q3 / Q4]

- Weakest quadrant: [Q1 / Q2 / Q3 / Q4]
- Known blind spots: [What do I tend to skip or rush?]

My Default Process Selection

- High stakes, well-understood problem: [Primary Process / First Time Right]
- Familiar problem needing refinement: [Iterative Process]
- Novel problem with high uncertainty: [Exploratory Process]
- Notes on my tendencies: [Do I default to one process when another would be better?]

My Go-To Approaches

- Preferred clarification method: [Socratic questioning, rubber duck, expert consultation, Five Whys]
- Preferred decomposition approach: [Top-down, bottom-up, horizontal, vertical, MECE]
- Preferred focus tool: [80/20, bottleneck, thought experiment]
- What works for me in Q2: [Specific analysis techniques that have produced results]

Lessons from Practice

- Most effective approach I discovered: [What worked better than expected?]
- Biggest mistake I made and what I learned: [What went wrong and why?]
- Pattern I noticed in my journal: [Recurring root cause, repeated assumption, etc.]

Under-Pressure Checklist

When stakes are high and time is short, follow this sequence:

1. [ ] State the problem in one sentence (Q1)
2. [ ] Ask "why" at least three times (Q2)
3. [ ] Generate at least two solution options (Q3)
4. [ ] Test the smallest version first (Q4)
5. [ ] Verify against original success criteria (Evaluate)

Quarterly Review

- Date of last review: [YYYY-MM-DD]
- What changed since last review: [New lessons, updated defaults]
- Next review date: [YYYY-MM-DD]

The Fastest Way on One Page (Reference Card)

Print this page and keep it where you work. It carries the entire method on one sheet: the four Rules that apply at every step, the four quadrants of the Map, the four Processes that match your pace to the problem, and the artifacts you produce along the way. This card names every piece. The Fastest Way is where you learn how to use each one.

This is the entire method.

The Rules (apply at every step)

1. **Make Explicit.** Define the problem before solving it. Write it down.
2. **Decompose.** Break the problem into parts small enough to handle.
3. **Create Focus.** Find the vital few causes driving most of the impact. Set the rest aside for now.
4. **Evaluate.** Check your work against explicit success criteria. What would prove you wrong?

The Map (know where you are)

Quadrant	Question	Output
Q1: Problem Definition	What exactly is the problem?	Problem definition with success criteria
Q2: Problem Analysis	Why is it happening?	Root cause identification
Q3: Solution Design	What is the solution?	Solution design with implementation plan
Q4: Solution Implementation	Does it work?	Verified result against Q1 criteria

The Processes (match your pace to the problem)

Ask in Order	If Yes	If No
Is failure catastrophic or irreversible?	**First Time Right Process**	Continue
Have you solved this exact problem before, in the same context?	**Primary Process**	Continue
Can you describe what a working solution looks like?	Continue	**Exploratory Process**
Can you afford repeated attempts?	**Iterative Process**	**First Time Right Process**; reduce uncertainty first with bounded, safe tests

The Artifacts (what you produce)

- **Problem Definition Contract:** written in Q1, validated with stakeholders
- **Process Selection Matrix:** chosen in The Processes (Ch. 12), documented with rationale
- **Prevention Log:** maintained in Bringing It All Together (Ch. 18), reviewed monthly
- **Personal Playbook:** built in Week 4, revised quarterly

Continuous Practice Tools

Question Library

Maintain this library as you build domain expertise. Capture questions that unlock problems, organize by domain and quadrant, and review quarterly to refine.

Generic questions get you started. Domain-specific questions make you an expert.

The Loop:

1. **Capture:** When a question unlocks a problem, write it down.
2. **Organize:** Tag it by Domain (e.g., Software) and Quadrant (e.g., Q2).
3. **Refine:** Review quarterly. Remove what does not work.

Question Log Template:

Domain	Quadrant	Question	First Use	Effective-ness	Notes

To seed your library, take the Q2 core questions from Questions and Quality Gates

(Appendix C) and rephrase each for your domain. For example, in healthcare, "Can I decompose this further?" becomes "Can I isolate which organ system is involved?" This domain-specific rephrasing is the first step toward building expertise.

Maintaining Your Library:

Monthly, review the last 10 questions you asked and mark effectiveness (High/Medium/Low). Quarterly, delete questions marked Low more than twice, keep consistently High questions, and add two to three new domain-specific questions from recent successes. If all your questions cluster in one quadrant, rebalance: heavy Q2 may mean over-analyzing without defining problems clearly.

Summary

You just used a collection of tools. Here is what you received, grouped by the three pillars of the method.

The Rules (clear your thinking):

- Rule 1 Make Explicit: Structured Writing Protocol, Journaling Template, Data Tracking Protocol, Intuition Log.
- Rule 2 Decompose: Integration Contract Template.
- Rule 3 Create Focus: Priority Matrix, Dependency Matrix.
- Rule 4 Evaluate: Assumptions Log.

The Map (know where you are):

- Q1 Problem Definition: Problem Definition Contract Template.
- Q2 Problem Analysis: Root Cause Analysis Quick Reference.
- Q3 Solution Design: Solution Evaluation Scorecard, Solution Design Tools (SCAMPER, Six Thinking Hats, TRIZ, First Principles).
- Q4 Solution Implementation: Implementation Planning Checklist.

The Processes (match your pace to the problem):

- Overview: Process Selection Decision Guide.
- Primary Process: Primary Process Execution Template, Problem-Solving Journal.
- First Time Right Process: Gate Evaluation Checklist, Reviewer Authority Matrix, Escalation Decision Rule.
- Iterative Process: Iterative Process Review Templates.
- Exploratory Process: Experiment Documentation Template.

Practice and synthesis:

- Consolidation: Prevention Log Template, Common Pitfalls Diagnostic Checklist.
- Personal reference: Know Your Style Self-Assessment, Personal Playbook Template, The Fastest Way on One Page Reference Card.
- Deepening: Question Library.

Three pillars, one method. The Rules clear your thinking. The Map shows you where you are in problem-solving space. The Processes match your pace to the problem. Combined, they are the Fastest Way. Each tool in this pack is a practical handle on one of those pillars; the book is where you learn how they fit together, why each piece works,

and how to pick the right tool for the situation you are in. If you found these tools useful, the book behind them is *The Fastest Way* by Pieter Amersfoort.