SOFTWARE & ENGINEERING · GUIDE

The skills matrix *for software teams*

Engineering teams live and die by who can do what, who owns the database, who can be trusted with production, who is one bus away from taking critical knowledge with them. A skills matrix makes all of it visible: depth, breadth, and the single-expert risks hiding in your stack. This is how to build one for a software team.

Dr Alex J. Martin-Smith
CMGR · MBA · LLM · DBA

Reading time 12 min · **Method** Upleashed 0 to 5 capability framework · **Updated** May 2026

THE SHORT ANSWER

A software team skills matrix maps engineers against the skill areas the team depends on, frontend, backend, data, infrastructure, testing, architecture, with a proficiency level in each cell. Read across a row and you see one engineer's profile, their deep specialism and their breadth; read down a column and you see how many people can genuinely own each area. In short: **it shows who is deep, who is broad, and which parts of your stack rest on a single person.**

KEY TAKEAWAYS

- **Map skill areas, not job titles.** "Senior" means little; "owns the backend, can mentor on it" means everything.
- **Profiles are usually T-shaped.** Strong engineers are deep in one area and capably broad across others; the matrix shows both.
- **Watch the bus factor.** A skill area only one person can truly own is a risk, however brilliant that person is.
- **Level by scope, not tenure.** Define levels by autonomy and impact, so a rating reflects what someone can own, not years served.
- **Score on evidence.** Real projects and code, not "seniority vibes", and calibrate so a level means the same across squads.

START HERE

What a software team matrix *actually maps*

A skills matrix for a software team is the same grid used anywhere, people down the side, skills along the top, a level in each cell, but the columns are the engineering capabilities the team depends on, and the reading is shaped by how software work really splits into depth and breadth.

Skill areas, not job titles

The columns of a software matrix are **skill areas**: frontend, backend, data and databases, infrastructure and DevOps, testing and quality, architecture, plus the essential skills like code review and mentoring. Mapping these is far more useful than mapping titles, because "Senior Engineer" tells you little about what someone can actually own. Two engineers at the same level can have completely different shapes; the matrix captures the shape, which is what you staff and plan around.

Most engineers are T-shaped

The defining pattern in software teams is the **T-shape**: deep expertise in one area (the vertical stroke of the T) and capable, working breadth across several others (the horizontal stroke). A strong backend engineer who can also hold their own in data and infrastructure is T-shaped; a generalist sits flatter and broader; a junior is still shallow across the board. A skills matrix

makes these shapes visible at a glance, and the shape is exactly what tells you how to deploy and develop someone.

The column is where the risk lives

Reading a software matrix down its columns surfaces the risk every engineering manager worries about: the **bus factor**. If only one person can genuinely own the deployment pipeline or the core data model, that area is fragile, however talented that engineer is, because a holiday, a resignation or a bad week leaves a hole nobody else can fill. The matrix turns that quiet anxiety into a visible coverage count, so you can spread critical knowledge before it walks out of the door.

WHY IT MATTERS NOW

"Senior" is not a plan, and knowledge *walks*

Software teams carry two chronic risks: capability that is assumed rather than known, and critical knowledge concentrated in one head. A skills matrix addresses both, replacing title-based guesswork with a clear, evidenced picture of who can own what.

8%

GARTNER, 2024

of organisations have reliable workforce skills data, so most staff projects on titles and instinct.

39%

WEF, 2025

of workers' core skills are expected to change by 2030, and in software the stack shifts faster still.

63%

WEF, 2025

of employers call skills gaps the biggest barrier to transformation, acute where engineering capability is scarce.

The cost of getting this wrong is familiar to anyone who has run a team: a project staffed on assumed seniority that stalls when the real expertise turns out to sit elsewhere, or a critical system that becomes untouchable the moment its sole owner is unavailable. A skills matrix is the antidote to both. It replaces "they're senior, they'll manage" with a clear view of **who can genuinely own each area, and where the team is one person deep**, so you

can staff honestly, cross-train deliberately, and turn vague worry about the bus factor into a plan.

WHAT IT REVEALS

Four things a software matrix shows you

For an engineering team, a skills matrix surfaces four things that titles and org charts simply cannot. Each turns directly into a staffing, development or risk decision.

REVEALS 01

Depth: who can own what

The deep specialisms across the team, who can truly take responsibility for the backend, the data layer, the platform, so you staff critical work with people who can actually carry it.

REVEALS 02

Breadth: who is versatile

The T-shape: engineers with capable breadth beyond their specialism, the people who flex across the stack, review widely and adapt as priorities shift.

REVEALS 03

Bus factor: where you are thin

The skill areas only one person can own. The single most valuable thing the matrix shows, because it is the risk that hurts most and is easiest to miss.

REVEALS 04

Growth: the next step

For each engineer, the obvious development move, deepen the specialism, or broaden the T, made concrete and evidence-based rather than a vague review-time conversation.

Together these turn a software matrix into a planning instrument rather than a record. Staffing a project becomes a question of reading depth and breadth against what the work needs; managing risk becomes a matter of watching the coverage counts; and developing the team becomes a set of deliberate moves to deepen specialisms and broaden T-shapes where it matters most. The matrix does not replace an engineering manager's judgement, it **gives that judgement something solid to work from.**

THE SCALE BEHIND THE SCORES

The 0 to 5 capability framework

A software matrix needs levels defined by scope and autonomy, not years served, so a rating reflects what someone can own. This framework, developed by Dr Alex J. Martin-Smith, does exactly that: each level describes what an engineer can do and how independently.

0

Not required for the role EXCLUDED

The skill area is not part of this engineer's role within the next year, for example deep infrastructure for a dedicated frontend specialist. Excluded from their score, not counted as a gap.

1

In training / Learning WEIGHTING 25%

Learning the area, works on it with guidance and review. Up to 75% of the way to proficiency and does not yet fully grasp the quality bar. Should not own work here unsupervised.

2

Developing WEIGHTING 50%

Can complete standard tasks alone, but complex or unfamiliar work still needs review before it ships. Consistent quality is not yet evidenced.

3

Capable WEIGHTING 75% · THE OWNERSHIP LINE

Ships consistent, quality work in the area unsupervised, and can be trusted to own features and fixes. This is the line at which an engineer genuinely counts on a skill, the broad part of a healthy T.

4

Subject Matter Expert / Mentor WEIGHTING 100%

Deep expertise; owns complex work in the area, sets the quality bar, reviews others' work and mentors on it. The vertical stroke of the T. If unused for a period, reconfirm against current practice.

5

Strategic ownership / Architecture WEIGHTING 100%

Defines standards, architecture and direction in the area, with impact across teams. Staff and principal-level scope. The purple flag marks your technical leaders and architecture owners.

Levels by scope, and the T made measurable

The clearest anchor is Level 3, "ships quality work unsupervised, can own features". Below it, work still needs review; at Level 4, the engineer mentors and sets the bar; at Level 5, they shape architecture across teams. The weightings, Level 1 = 25%, 2 = 50%, 3 = 75%, 4 and 5 = 100%, with 0 excluded, turn a row into a capability figure, and the shape of the row, one or two high bars amid solid threes, is the T made visible.

A worked example. Read a T-shaped backend engineer's row across six skill areas:

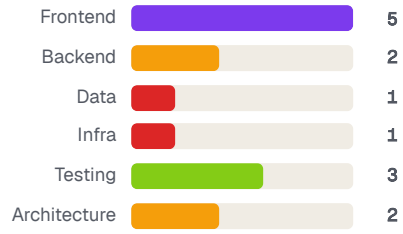
```
Raj's row  Frontend 2, Backend 5, Data 3, Infra 2, Testing 2,
Architecture 3
weightings  50, 100, 75, 50, 50, 75  =  400 ÷ 6  =  67%
a clear deep spike (Backend 5) on a broad, capable base — a
textbook T.
```

SEE IT ON A REAL TEAM

The team's *capability profiles*

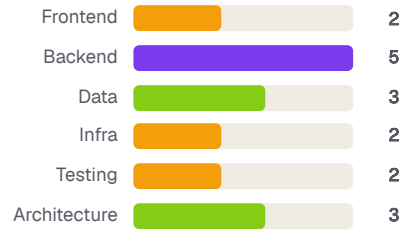
Here is a six-engineer team mapped across six skill areas, drawn as capability profiles. Each bar's length is the level on the 0 to 5 scale, so the shape of each engineer leaps out: the deep specialists, the broad generalist, and the junior still building. Read together, they show the team's depth, breadth and risks.

Maya — FRONTEND SPECIALIST



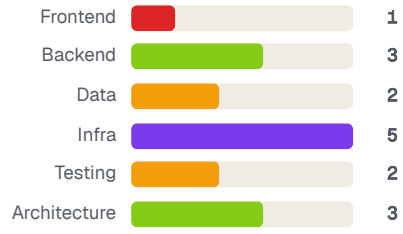
Deep T · frontend expert

Raj — BACKEND SPECIALIST



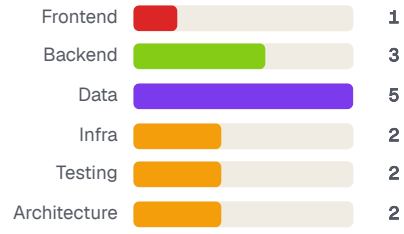
Deep T · backend expert

Chen — DEVOPS / INFRA



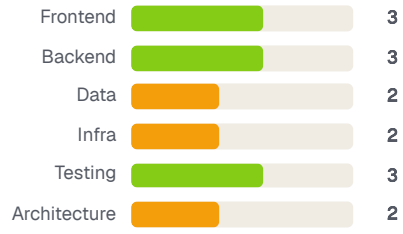
Deep T · sole infra owner

Sofia — DATA ENGINEER



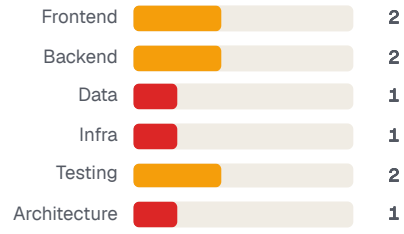
Deep T · data expert

Tom — FULL-STACK, MID



Broad generalist · no deep spike yet

Aisha — JUNIOR



Still building · shallow across

1 Learning · 2 Developing · 3 Capable · 4 Expert · 5 Strategic

1 **area owned by one person** (Infra, Chen), a bus-factor risk

4 **clear deep specialists**, plus one broad generalist and one junior

Illustrative team on the Upleashed 0 to 5 framework. Each bar's length is the level; the shape of each profile shows depth and breadth at a glance.

WHAT THE PROFILES TELL YOU

- **The T-shapes are obvious.** Maya, Raj, Chen and Sofia each show one tall bar on a solid base, deep in one area, capably broad across others. That shape is exactly what you want in a specialist.
- **Infrastructure is a bus-factor risk.** Only Chen reaches expert on Infra, and no one else is close. Brilliant as that is, the platform rests on one person, the top cross-training priority.
- **Tom is the flexible glue.** A broad, capable generalist with no deep spike yet. Ideal for covering gaps and connecting areas, and his growth move is to deepen one area into a T.
- **Aisha's path is clear.** Shallow but rising across the board, exactly right for a junior. The matrix shows her next step: build one area toward Level 3 to start forming a T.

READY-TO-USE EXAMPLES

Example skill areas to map for a software team

The columns of a software matrix should reflect how your team's work actually divides. Here are ready-to-adapt skill areas for the common engineering functions, a starting point to tailor rather than a blank grid.

Function	Example skill areas to map (the columns)	Watch out for
Product engineering	Frontend, backend, API design, feature delivery, code review	One vague "coding" column instead of the areas people specialise in
Data & platform	Data modelling, pipelines, databases, analytics, ML where relevant	Lumping all "data" together when modelling and pipelines differ
Infrastructure / DevOps	CI/CD, cloud infrastructure, observability, incident response, security	Treating infra as one skill when it hides several deep specialisms
Quality	Automated testing, test strategy, performance, release management	Assuming everyone who codes can also own testing to standard
Essential / behavioural	Architecture, mentoring, technical communication, incident leadership	Mapping only hard skills and missing the ones that scale a team

Take the functions your team covers, trim each to the skill areas that genuinely matter, and add anything specific to your stack, a particular cloud, a critical service, a regulated domain. Resist the urge to list every language and tool; map the *areas* people specialise and grow in, since that is what depth, breadth and the bus factor are really about. As always, few enough columns that the matrix stays maintained.

AVOID THESE

Six mistakes on a software team matrix

MISTAKE 01

Mapping titles, not skill areas

"Senior" tells you nothing about what someone can own. Map the areas; the title is not the capability.

MISTAKE 02

Levelling by tenure

Years served is not ownership. Define levels by scope and autonomy, so a 4 means "mentors and sets the bar", not "has been here a while".

MISTAKE 03

Ignoring the bus factor

A team of strong individuals can still have areas only one person owns. Always read the coverage down each column.

MISTAKE 04

One giant "coding" column

Collapsing the stack into one skill hides every specialism and gap. Map the areas people actually divide along.

MISTAKE 05

Scoring on vibes

"Feels senior" is not evidence. Anchor ratings to real projects and code, and calibrate so a level means the same across squads.

MISTAKE 06

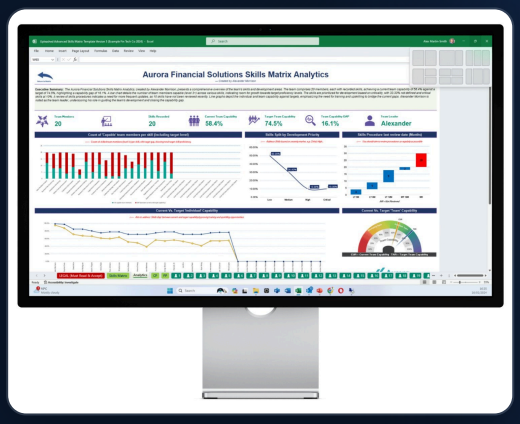
Mapping only hard skills

Architecture, mentoring and incident leadership scale a team. Leaving them off misses the capabilities that matter most at senior levels.

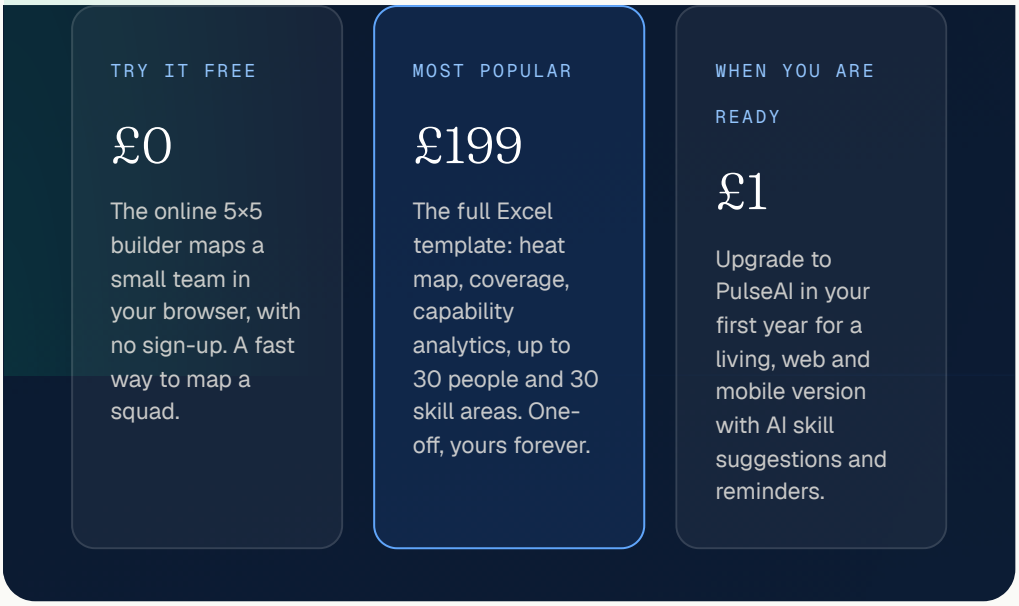
FROM ORG CHART TO CAPABILITY

The method is free. A ready-made matrix just makes depth, breadth and *risk obvious.*

Everything here works in a blank spreadsheet, and that is a fine place to start. A purpose-built template just makes the engineering view effortless: score each area on the 0 to 5 scale, and the profiles, coverage counts and capability figures calculate themselves, so the T-shapes, the broad generalists and the single-owner risks in your stack are visible at a glance rather than buried in people's heads.



The Advanced Excel Skills Matrix shows capability and coverage across skill areas at a glance, so depth, breadth and bus-factor risks stand out, all on the same 0 to 5 framework used throughout this guide.



COMMON QUESTIONS

Quick *answers*

Q What is a skills matrix for a software team?

It is a grid mapping engineers against the skill areas the team depends on, frontend, backend, data, infrastructure, testing, architecture, with a proficiency level in each cell. Reading a row shows one engineer's depth and breadth; reading a column shows how many people can genuinely own each area.

Q What does T-shaped mean for engineers?

A T-shaped engineer has deep expertise in one area (the vertical stroke) and capable, working breadth across several others (the horizontal stroke). It is the healthy shape for most specialists, and a skills matrix makes it visible, alongside broad generalists and juniors still building depth.

Q What skill areas should I map for engineers?

Map the areas your team's work actually divides along: frontend, backend, data and databases, infrastructure and DevOps, testing and quality, and architecture, plus essential skills like code review and mentoring. Map areas people specialise and grow in, not every individual language or tool.

Q How should I level software engineers?

By scope and autonomy, not tenure. On the 0 to 5 framework, Level 3 means "ships quality work unsupervised, can own features", Level 4 adds mentoring and setting the bar, and Level 5 means shaping architecture and standards across teams. Score against real projects, and calibrate so a level means the same everywhere.

Q What is the bus factor, and how does the matrix help?

The bus factor is how many people would have to be unavailable before a critical area has no owner. A matrix surfaces it by counting, per skill area, how many engineers are genuinely capable (Level 3 or above). An area with a count of one is a risk, the clearest signal of where to cross-train next.

Q Do I need special software for an engineering skills matrix?

No. A well-built spreadsheet maps depth, breadth and coverage perfectly well, and most teams should start there. Dedicated software helps when you want it live across many squads, kept current as the stack and people change, with coverage and capability views that update automatically.

ABOUT THE AUTHOR

Dr Alex J. Martin-Smith

CMGR · MBA · LLM · DBA

Alex is the creator of the Upleashed capability framework that powers Skills Matrix Template, the award-winning Excel skills matrix. A Chartered Manager with an MBA, an LLM and a doctorate in business administration, he has spent more than two decades helping operations, HR and quality teams turn capability from a gut feel into something they can measure, manage and prove.

Connect on LinkedIn: linkedin.com/in/alexmartinsmith

Dr Alex J. Martin-Smith

SOURCES

Gartner. (2024). *Talent management research: Workforce skills data.* Gartner.

Martin-Smith, A. J. (n.d.). *The 0 to 5 capability framework.* Upleashed Limited. https://upleashed.com/capability-framework/

World Economic Forum. (2025). *The future of jobs report 2025.* World Economic Forum.

Map the stack, *not the titles.*

You now have the engineering method. The quickest way to start is to list your skill areas this week, score each engineer on the 0 to 5 scale, and look at the shapes. The deep spikes, the broad generalists and the single-owner risks will tell you how to staff, develop and de-risk your team.