Engineering excellency

How to build a product development organization that excels in developing and nurturing its talent as part of becoming a better performing company

you only become a leader by leapfrogging! Russ Ackoff
https://youtu.be/OqEeIG8aPPk

Author: .... Hannes et al.,
Date: 2022-08-10ff
Version: ...... 0.95 (it's not rendered as being releasable, if this count is below 1.0)

Contributors

Currently I am contacting People in Software Development and HR and interview them to participate. They are:

Carola Lilienthal, WPS: invited

Christian Zweckerl: invited
Vincent Voss: invited

Felix Schad, dmTECH: invited

André Bahl, intersoft AG: invited

Andreas Stein: SumUp invited

Katrin Stamme: Freiheit.com invited

Robert Albrecht (Hathaway), equalexperts

Eberhard Wolff: INNOQ, invited

André Fleischer: Maiborn Wolff, invited

Jo Bager, invited

Matthias Ernst, invited

Michael Geers, invited

Niels von Stein, netlight, invited

Frerk Lättari, Tchibo, invited

Waldemar Spät, Tchibo, invited

Katharina Brunner, Tchibo HR, invite
Björn Kaiser: OTTO, invited

Anke Nehrenberg: invited

Iris Bögeholz

Table of contents

License - CC BY SA

About the genesis of this document

This started in August 2022, when Tchibo asked me (Hannes) to help them with reviewing their junior, professional, and senior software development career levels. I then felt a little reluctance to do that again, but then I did not find my notes from similar work in previous companies, and so I started by writing the "Why" chapter.

Then I caught fire on the subject, and soon I invited more people to join and give their views on the subject. Right now (September 2022) we have a regular meeting with Eberhard Wolff and Robert Albrecht on Fridays and many people left valuable comments.
Together with Robert and Eberhard, we came to the conclusion, to put this under an open-source license and make it available to the public and get additional feedback. I like the idea of pull requests and other additions from people, and would like to evolve this to something better.

Thankfully, Christian from Tchibo agreed to this. The Idea is that companies could fork and adapt this to their needs.

September 2022, Hannes

Why

Ensuring future success of company and individuals

"Ensuring future success of the company and individuals" is the fundamental reason **why management exists** (and to ensure communication in larger groups of people...). While companies are great at ensuring the company's success, the development of people is often not properly described and implemented. This paper aims to find a more formalized approach to the topic for software engineers and to help companies to implement a process at all.

Why in IT & digital product development?

The development of IT-Products differs from other work in the following areas:

- Knowledge work and intangibility:
  The things we work with are intangible in the sense that software development uses lines of code to move units of 100 electrons to other energy levels. This results in stored data, inputs, and outputs. And ultimately into systems that communicate information or let customers buy stuff online. They are in large parts intangible: you cannot touch, feel and experience lines of code, like you could touch, feel and experience a bicycle or a pair of headphones. Thus, large parts of the things we work with are hard to communicate to people, who are less involved in the process.
- Speed of ecosystem development:
  The tools with which we work evolve at very high speed. Just look at the change and development of frontend frameworks in the last 10 years. Thus, the stuff we produce tends to become legacy very fast.

This explains why "learning ability" and "communication skills" are vital to engineers. We will base large parts of this paper on the following theses:

1. Software development in larger organizations is done with many teams of people, who are

responsible for different aspects of the end product.
2. Successful software will pass though many engineers hands throughout its lifetime
3. successful companies will be like trainings camps for engineers

Formalized Software Engineering Development path Benefits

In many companies, the distinction between junior, professional, and senior engineers labels are arbitrary. This might lead to favoritism and might leave the managers in larger companies without orientation and common ground. It also leases the software engineers without orientation, on what the companies goals are towards their development. At last, the company itself might appear as being unconcerned with the positive development of their workforce and thus mediocre.

This leads to the following thesis:

High-performance companies
care about learning and improving itself
by serving excellent career paths for their employees.

Keep in mind, that only high-performance companies are attractive employers.

Manager versus Engineer

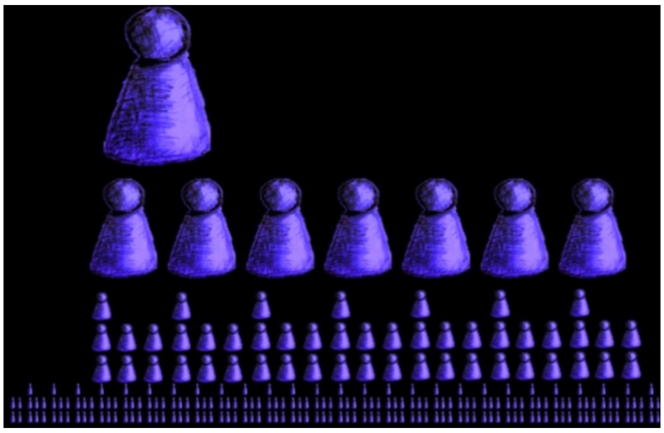
To create a long-term and sustainable career path for engineers, it is necessary to have a parallel path for managers and staff engineers (those without direct line management). People can progress through either and even switch between the two paths as a means to create further growth and learning over time.

To support these parallel paths and to provide sufficient long-term growth options for engineers, there must be an increase in what this paper refers to as Career levels. This expanded level range provides an indication to the level of experience both in terms of length of experience (years working) and also type/quality of experience/growth gained. These expanded career levels allow engineers to grow and be rewarded using the same scheme as managers. The difference between a management and non management role would simply be the amount of time a person spends in engineering vs. line management tasks.

Manager versus Leader

The critical success factor for engineering excellence is a high density of great leadership, in both managers and engineers. Leadership is not a skill that is exclusive to managers, in fact, great leadership within senior engineers has a significant impact on the quality and excellence of the whole engineering team. Ensuring that the recruitment and development processes/practices within a company ensure the best leaders are hired and leadership is developed/encouraged as a key skill within all engineers should be one of the primary objectives of any modern company.

Career Levels



The career levels vary in different companies by naming and description. Generally, something like this is in operation:

- Junior Engineers
- Professional Engineers
- Senior Engineers
- Principal Engineers or Staff Engineers

There are many side paths to this linear approach with special roles towards software architecture, personnel management (engineering manager), quality assurance, technical product management. Furthermore, technically, there might be distinctions between frontend backend, full-stack, or DevOps, SRE and many more. But for now, let's try to keep it simple and start with the top four.

After discussing career levels in length in the group of authors, we noticed a shared discomfort with the terms “Junior” and “Senior”. These terms are one dimensional and in part misleading. A “Senior” sounds like someone with years of experience, whereas a “junior” sounds young. Yet, we experience that expertise does not always grow with the years in the job and often, the young people bring excellent knowledge of new technologies and frameworks into the community.

Career levels in companies vary. Often Companies have junior to senior levels. In some companies, they have salary bands, sometimes they have grades from A-G, like the German Post Management levels. And occasionally these levels are coexisting in the same company. Traditionally, the levels indicate how high in the pyramidal system a person is advanced. Usually, payment and benefits get better, the higher you rank. Career levels might be connected to skills and positions.

With the arrival of knowledge-work and digitalization, companies started to evolve the pyramidal line organization to more complex structures. In some companies, an expert will outrank a manager and is being better paid than the manager.

Generally, we believe it is a good idea for companies that rely on knowledge work to allow for expert careers in terms of recognition and payment. Otherwise, an expert who wants more recognition or better payment, will either have to move to management or leave the company. In any case, the company loses the expertise.

The following chapters describe how career levels might be implemented.

Example: 6 Level Scheme

Adding extra career levels and allowing engineers to progress without becoming managers provides a longer and more sustainable scheme. That allows employees to grow and companies to harness the key skills of their best engineers, ensuring that the critical skills and experience are retained within the engineering teams to ensure long-term growth.

Instead of labeling these levels it's recommended that they are simply numbers 1-6 or alternatively, as at Google, Facebook and others, 3-9 as this allows levels 1 and 2 within a company for roles outside of engineering across the wider organisation. For the sake of simplicity this paper uses 1-6.

Level 1

Expected to write codes for production and conduct tests under minimal supervision. Well-versed with software testing tools, source control, and code review technologies. Level I is typically designed for internship seekers. They work under close supervision of their superiors and have little to no decision-making authority. Usually college graduates or employees with less than two years of professional experience.

Level 2

Must possess basic knowledge of software application design. Responsible for making minor design decisions independently and work towards understanding and developing system applications without supervision. Established procedures typically drive decision-making at this level. Must possess a minimum of two to four years of experience.

Level 3

Expected to possess in-depth domain knowledge, problem-solving abilities and be well-versed in system design and architecture to function as a technical architect. Considerable decision-making responsibilities as well as involvement in hiring, training, and mentorship of lower-level Software Engineers. They also have budgetary responsibilities and operational planning roles and require people skills to lead junior personnel. Must have four or more years of engineering experience.

Engineering managers start at this level and would be responsible for training and mentoring junior engineers.

Level 4

Equivalent to staff/principal engineers at companies like Google, they lead significantly larger teams of Software Engineers while working autonomously. They facilitate coordination between different teams and are expected to mentor other engineers to foster growth by helping them build their skill-sets and realize their full potential. Must have 5 to 9 years of experience.

Level 5

Equivalent to Staff Software Engineer at Google. Must have extremely strong interpersonal skills and will be responsible for leading larger projects and teams. Must have 9+ years of experience

Level 6

Will act as visionaries and strategic engineering leaders and work towards large scale, competitive business growth. Part of the role will be focused on improving company culture, shaping policies, launching large-scale projects, and incorporating evolving technological practices. Would be considered the most skilled coders at the company.

Example: Expert career levels in the tech organization—junior to senior

This chapter still works with "Junior" and "Senior" and is deprecated. Throughout our discussions while writing this document, we came to the conclusion that the terms junior and senior are misleading in more than one way.

- Seniority is not something you earn by just getting older
- Juniors sometimes bring excellent know-how into an organization. Like Christopher, who came to XING at the age of 17 with spellboy.de as working and revenue bringing website, or Fynn, who works at SAP at the age of 19 and runs his small ERP-System for building sites. These very young people might lack experience in the organization, but start as juniors with some level 10 skills in their portfolio already.

Organizations might choose to still use a junior, professional, senior and principal nomenclature in their job offers and ranking of engineers, yet the authors would like to inspire companies to let go of them and replace them.

Skill level	**Expectations** **(next levels expectations add to the previous levels)**	**Comments**
junior (FDC: newbie, while in probation)	ability to work in a team Work independently and be able to do the work and raise the hand if any help/support is necessary.[1] basic communication skills with peers basic starter skill set of one programming language proven learning ability an idea, of what the next 2-3 learning topics to archive are curiosity coding ability	25% of all devs
professional	good communication skills with peers good sociability multi tech skills in at least two domains?	35% of all devs
senior	a profound track record of working with and developing juniors excellent communication skills with peers good communication with people from other teams collaboration skills in super-team-functional-units (like architecture meetings, etc.)	30% of all devs
principal	ability to solve and present complex technical and social problems in front of an audience in a way, that makes people want to listen	10% of all devs

The details of the skill level description and the naming of the roles will vary from organization to organization. It is however important that the roles are described within the organization and thus the organization's expectations towards engineers are manifested.

The distribution across the levels is something to be monitored and to be strategically agreed upon regularly, e.g., on an annual base among the engineering management. If the actual numbers are different from the agreed upon goals, strategic measures have to be taken.

E.g.

- Diagnosis: not enough seniors, measure: hire external seniors
- Diagnosis: not enough juniors, measure: take on an apprentice program
- ...

[1] From the editors: this is heavily discussed between the authors. Independence and teamwork seem to contradict each other.

Skill levels

In addition to the Career level a person attained, they also have a range of different skills they have gained over time, both technical (programming languages, engineering practices) and non-technical (communication, leadership, mentoring, project/program manager) skills. Each of the skills they have also have a level associated with it.

Let's collect some expectations toward the above-mentioned skill-levels. Organizations expect engineers to perform different tasks, to solve different problems or be responsible for specific things. These skills are grouped in skill-levels. Each skill is ranked with a skill-level.

Proposal: skill-levels ranking from 0-10

We propose to rank skills in skill-levels from 0-10 with the following nomenclature:

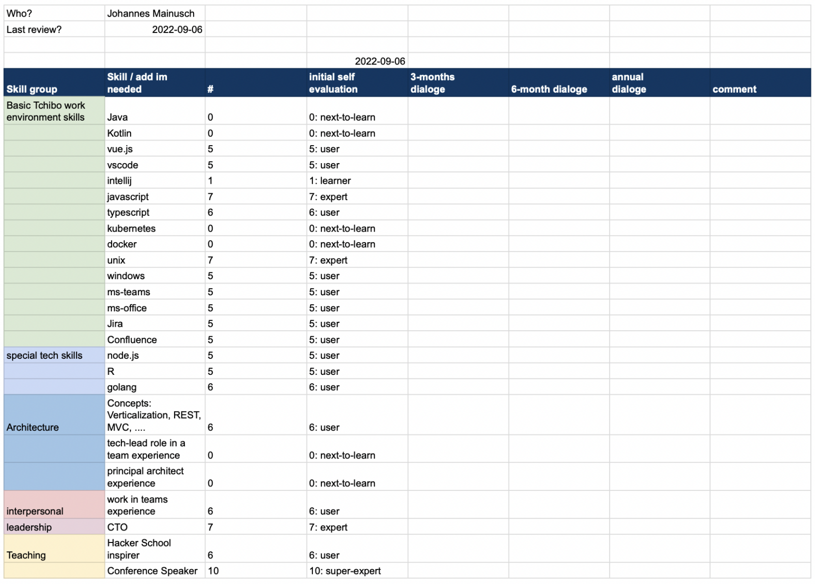
skill-level	expectations (next levels expectations add to the previous levels)	comments
0	this is new to me, I want to adopt this skill in the next 6 month to a significant higher level	this can be used to indicate, what's to be on the tab for learning
1-3	I am learning this skill, I do not yet feel proficient in it	this might correspond to junior
4-6	I am working with this skill naturally. People come to me for help, and I am able to contribute.	This might correspond to professional
7-9	I am teaching this skill to peers and contribute to decisions across our team boundaries.	This might correspond to senior
10	I am a proven expert in this area by external recognition, like participating on conference panels, publications, significant open-source contributions or similar merits, that are external to the company I work with,	This highest level can only be reached by company external recognition.

This can be applied to all roles within your organization. In this document, we trimmed it more towards the skills needed by software engineers. But it may be applied to scrum-masters, product-owners, ...

Skill-sets and skills by example:

skill-set	skill	comments
Technology	Dev Basics	please indicate the work set, with which your teams or organization works. Like: vs-code, git, Unix, vue.js, nuxt, Java, Kotlin, Windows, OS-X, Or if Agile Coach: Scrum, Kanban, Jira, Confluence,... Why: you might hire people, who come from a different working background. There are, e.g., still companies about there, who use subversion and not git. Or React and not Vue, Or scrum and not kanban.
Technology	vue.js	Indicate the versions you'd expect the person to know.
Technology	golang	
Technology	typescript	
Learning	self reflection	
Teaching		
interpersonal		
leadership		
team-building ability		
Autonomy		this is one of the most controversially discussed skills. It originated from "I'd like the new junior in my team to take on tasks by herself and ask for help if needed" to the controversial "the work is done as a team and not autonomously".
... beyond Tellerranding? Experience?		Take, what you need in your organization.

Example:

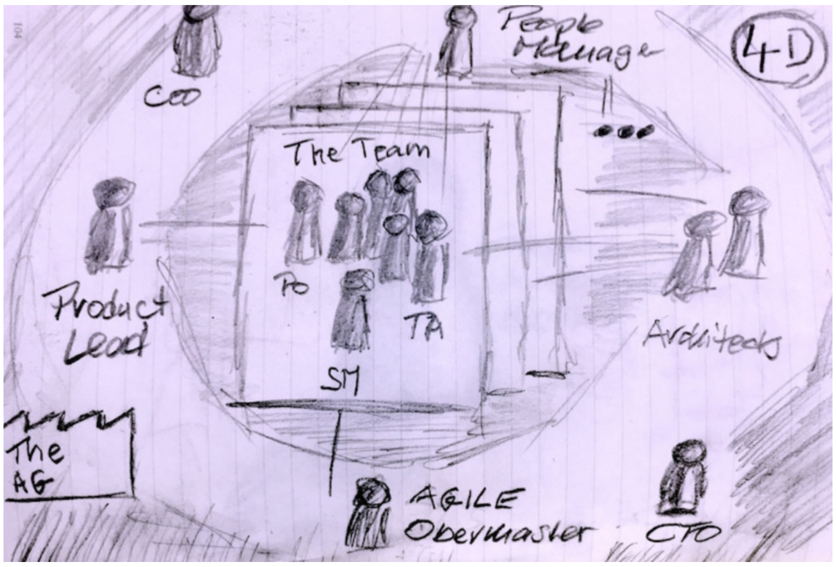


Who?	Johannes Mainusch						
Last review?	2022-09-06						
			2022-09-06				
Skill group	**Skill / add im needed**	**#**	**initial self evaluation**	**3-months dialoge**	**6-month dialoge**	**annual dialoge**	**comment**
Basic Tchibo work environment skills	Java	0	0: next-to-learn				
	Kotlin	0	0: next-to-learn				
	vue.js	5	5: user				
	vscode	5	5: user				
	intellij	1	1: learner				
	javascript	7	7: expert				
	typescript	6	6: user				
	kubernetes	0	0: next-to-learn				
	docker	0	0: next-to-learn				
	unix	7	7: expert				
	windows	5	5: user				
	ms-teams	5	5: user				
	ms-office	5	5: user				
	Jira	5	5: user				
	Confluence	5	5: user				
special tech skills	node.js	5	5: user				
	R	5	5: user				
	golang	6	6: user				
Architecture	Concepts: Verticalization, REST, MVC, ...	6	6: user				
	tech-lead role in a team experience	0	0: next-to-learn				
	principal architect experience	0	0: next-to-learn				
interpersonal	work in teams experience	6	6: user				
leadership	CTO	7	7: expert				
Teaching	Hacker School inspirer	6	6: user				
	Conference Speaker	10	10: super-expert				

An example of an assessment is shown here:

skills-evaluation

Functional careers in the tech organization



All the people around the teams serve functional roles. Also in this picture, the three roles in the Team "PO = Product Owner", "TA - Team Architect" and "SM = Scrum Master" serve in a functional role.

From: Scrum with user Stories

Functional roles in an organization serve a function beyond the execution of tasks. And in most organizations without execution of tasks. Mostly, functional roles serve a communicative purpose at the fringes or above team level.

Functional separation of organizational tasks aims at separating concerns and by thus gaining execution excellency and speed. Highly aligned and loosely coupled management is the aim. Specialization in management can be thus achieved. One of the main separation lines is the distinction between

- process organization,
  anything that has to do with building the product. In tech-organizations, this can be further divided into
  - product
  - architecture
  - methodology (agile, ...)
- line organization,
  anything that has to do with improving the structural fitness of the organization. Here are the classical managers of organizations.

Functional roles	**expectations**	**comments**
tech-lead	the tech-lead role in a team, architect of the team. Communicates the tech-needs from other entities and interdependencies to other teams. Part of the architecture steering committee. part of the process organization. [Andreas: take care about the delivery of the team. They need to organize plannings, should come up with estimates, helps the team to estimate, understands the technical challenges and environment, is able to communicate risk and challenges → to product or project management.] TODO: clarify the delivery role of the tech lead within the organization! TODO: Clarify, whether personnel management is in or out of scope of this role	1 for each team
product-lead	communicates product needs into a team backlog and the brains and hearts of engineers.	

	Part of the product and program committee. Part of the process organization	
methodologist	Agile Master, Scrum Master or similar. Part of the methodology convention part of the process organization	
engineering-lead	responsibility for personnel disciplinary responsible person part of the "structure organization"	1 for 20 people
Enterprise-architect	architect of the organization	1-3 per organization
department head	This is a placeholder for the line structure or the traditional organization. Usually, the nomenclature is: ● Team lead / group lead ● head of department ● executive ● Board But these will vary in name and number from company to company.	

It is, anyhow, important to describe these roles for your company individually.

When a person is assigned to a team or project, it is a combination of their career level and relevant skill levels that would be used to determine their role in the work. The higher their career level the more responsibility and leadership they would be expected to provide. However, if there are experts in specific skills relevant to a piece of work then this would allow that person to have more influence on the direction/decision making on relevant specific topics.

Committees

Committees are usually driving the process organization of a company. They are strictly spoken nor roles or career levels, but they usually carry a responsibility in the company. Committees in a process organization could be

- the regular meeting of all the IT-Architects from teams and enterprise. The might decide upon new Architecture decisions or Marco- and micro Architecture rules.
- The program committee: they might decide upon what's up next on the roadmap.
- The staffing committee: They might decide upon allocation of personnel budgets.

- Feedback committee: you might want to organize the regular feedback with a committee of line manager plus team rep (see chapter below)..

Whatever the committees in a company are, they should be described because the functions and roles described above will have to run them or participate in them.

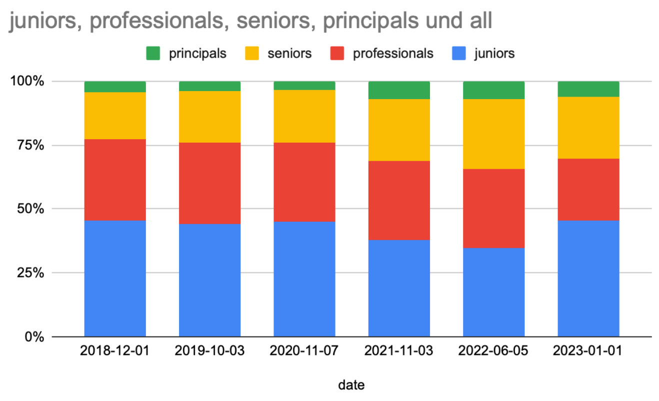
KPIs - ensuring continuity

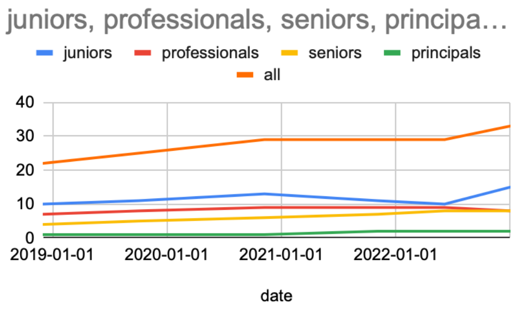
One of the products of a solid engineering management is experts!

How does the personal skill set (the numbers of people with individual skill levels) develops over the years? This should be monitored by engineering management.

https://docs.google.com/spreadsheets/d/1F319ydwBKFksq-_zdWq6a7TZ4nlW-PscvKCy4fHhbag/edit#gid=0

This is an example of a fictional company:





Another way of monitoring the progress of the whole organization could be to count the number of skill levels above 8 or the "stars" in the organization.

Development path from junior to principal

An organization should consider (and communicate) what expectations they have of their employees' paths. Do you expect them to reach a certain level in a certain time frame, at which level is it ok to have reached your potential and plateau?

Not everyone gets hired as "junior", not everyone will ever make "principal". Are you ok with a "professional" being "professional" for the next 10 years? Will they feel relaxed or pressured to advance?

The responsibility
to develop people and to maintain the overall skill set of the company
lies within the engineering management.

Thus, if a company has no management of engineering excellency, it

- might fail at product development in digitalization,
- it might fail to be an attractive place to work at.

Training people and providing learning opportunities within the company is the way to develop people. Here are some of the ingredients that will help engineering managers to archive and maintain an exceptional workforce. The following subchapters are methods an organization should consider implementing, if they are not yet in place.

Learn from Fail culture

Fails are great opportunities to learn from. Unless an organization punishes fails or unless fails are thrown under the carpet. Neither is good practice. Fails should be addressed promptly blameless, always with the possible "learn" in sight. A good KPI could be to count and publish the "learnings" of the organization.

These learnings will train people and align teams.

Inspiration:
google SRE: https://sre.google/sre-book/postmortem-culture/

Technical mentoring

Eberhard Wolff:

"I think technical mentoring might be a good idea. Actually, I would think it might be the most effective and important activity to enable personal growth of people."

One way of technical mentoring – pre-merge code reviews

Matthias Ernst:
“I consider the practice of mandatory, pre-merge code reviews – if taken seriously – to have a profound impact on learning within the team. It's beneficial in so many ways:

- it creates a constant opportunity for knowledge transfer in either direction, on any level from coding technique to product
- it forces engineers to communicate and summarize intent and feedback in constructive, high signal-to-noise language. (And all that on permanent record, an invaluable knowledge base.)
- natural limits on the size of PR reviews enforce iteration in logical, self-contained steps, which in turn has positive learning effects for how to design for such changes (loose coupling, yadda yadda)
- it enforces a negotiation in the team between short- and long-term benefits like the strive for high quality and the pragmatism needed to make progress where both author and reviewer need to juggle multiple interests on a personal and organizational level ("I will have to deal with this code months from now").
- It naturally promotes standards within the code base and enables shared ownership
- it's an equalizer in the power structure and creates incentives for senior engineers to educate more junior team members
- and, as a nice side effect, :) the organization profits from a higher-quality code base

I don't know how well all of this applies to organizations that don't share a long-lived artifact like a product code base, but where this exists, I consider it factors more important than allocating budget for seminars.”

Feedback cycles at regular short intervals

Learning and improving is only possible with feedback cycles. Feedback in systems needs to be fast and connecting, in human terms direct and with empathy. People managers should stay closely connected to the people they are responsible for. At least, they should have regular meetings with their people at regular intervals, e.g., every two weeks for 45 minutes (block 60). Feedback should be going in both directions because also people managers have to get the chance to learn.

KPI: measure the time for personal meetings per month. If a people manager has too many people, there will be little time for exchange. Organizational spans vary, the classical Dunbar number is 1:7 but with factorization of management functions some organizations reach 1:40 and beyond. Excellent people managers can handle this, if they are allowed enough time to relate to people.

Inspiration:

Radical empathy: [Sam Richards: A radical experiment in empathy | TED Talkhttps://www.ted.com › talks › sam ...]

Training Budget

No Premier League without training

A yearly training Budget per employee ensures, that there will be time and money for formal education and training. The budget in various companies varies from:

- k: €2500 per person and year, large retail company: €300 per person and year
- MW: 1,5 the amount of the monthly salary per person and year, may be used for training, conferences, travel, books. And also allowed to "pay with your own time", e.g., if you want to learn for a certificate or do an online-course, you will need more time than attending an on-site training. Your budget will be reimbursed by your hourly cost rate.

Inspiration time

Freedom and boundaries are the two ingredients for innovation. Many companies are good on boundaries but lack the trust that people will work to their benefit, when set free. But luckily this is changing, and companies work more often with hackathons, free hacking time, design and innovation sprints, solution labs and so on.

Instigating High-performance teams

High performance in my experience always starts at the top. Just proclaiming to be high performance, or to challenge teams to be high-performance teams, might lead to cynicism or frustration. As management team, ask yourself

- What would a high-performance management team look like?
- Can we become a role-model for our organization?
- Do we understand the needs of our teams?
- Do we understand the abilities of our teams?
- Do we serve and challenge them beneficially?

KPI: there are several team performance metrics out in the field, with self assessment and foreign assessment. Apply one of those and use it regularly (i.e., once per year).

Annual career path review

In addition to feedback at regular short intervals, there should be regular more formalized "career path reviews" At regular intervals, e.g., every 6-12 months, people manager and employee evaluate the career path.

- Assess status-quo together

- set learning goals

André Fleischer:
"The annual review should be based:

- economic impact/contribution (based on project/product)
- personal growth (what did you learn, new skills, mastery shown in daily work, where do you need to invest time to learn)
- contribution to cultural growth, contribution to team aspects, did you worked on extra topics where the team/department/.. benefits"

An example of an assessment is shown here:
skills-evaluation

The Feedback committee—why not just the boss?

Matthias Ernst: "Your manager is unlikely to understand whether you're actually a 7 in "Kubernetes" and how that even relates to what actual value you bring to the team ("I heard this is important"). And this is where things get interesting - how do you solicit this feedback and from who? What visibility do employees have into peer assessments and how does that shape the peers' incentives? How do you calibrate this across teams ("Team A all think they're JS experts, and the manager nods along because it makes them look good"). And how does cross-team calibration influence the need for self-promotion across the company ("visibility" is a trigger-word for many people in the corporate...)?"

We propose to form a committee of people from team and line management to give the feedback. E.g., one person from the team together with the line manager.

The form of the feedback could be given in the form of "try this, it might help" and "keep doing this, it's great".

Keep and Try Feedback → Eberhard has good experience with that

Salary negotiation

Salary negotiations are part of the job of a line manager together with HR. Usually, there is not much to negotiate, as companies often give narrow bands of possible raises to their managers. 2-3% annual rise in budget for a department was something usual some years ago in bigger companies.

As manager, this is one of the very unpleasant responsibilities. Typically, the wishes of employees are beyond the possible scope, and it is hard, to find fair solutions.

Some smaller companies have come up with new forms, like empowering a committee to become responsible fair and understandable salaries.

Feedback for mangers—how to become a level 5 leader

Jim Collins:

Level 5 leadership is a concept developed in the book Good to Great. Level 5 leaders display a powerful mixture of personal humility and indomitable will. They're incredibly ambitious, but their ambition is first and foremost for the cause, for the organization and its purpose, not themselves. While Level 5 leaders can come in many personality packages, they are often self-effacing, quiet, reserved, and even shy. Every good-to-great transition in our research began with a Level 5 leader who motivated the enterprise more with inspired standards than inspiring personality.

People frequently become managers without training. Moreover, organizations typically lack feedback for managers. So, how could we expect them to learn and become better at their job? Here are things, companies should consider implementing to become better in their management. We deliberately included some more uncommon methods to inspire:

- Formal management training
  Most of the older companies have their individual management trainings. Yet startups often lack this
- coaching for managers
- feedback for managers
  The most commonly known feedback method is the 360˚ feedback, which goes back to the German Reichswehr and Wehrmacht.
- Pair doing / peer responsibility in management
- regular votes on delegees to management-circles
  This is quite uncommon in economic companies. Yet very common in political systems of in NGOs.

Inspiration:
https://www.jimcollins.com/concepts/level-five-leadership.html

Personnel Marketing - tell the story

How can people know that you have a great place to work at? Personnel marketing is the answer. A KPI here could be the number of Talks on conferences/meetups per year.

Inspiration:
google "thoughtworks" and click on videos.

Rob: "What about amazing people that are too humble/modest to do this?"

Hannes: "good question. Maybe they could contribute in a more quiet way, like maintain stuff on the company's GitHub open-source page…"

Personnel balance—the annual review

This chapter raises controversy. In creating this document, we receive comments from experts like:

this is a very outdated concept

and often abused by companies

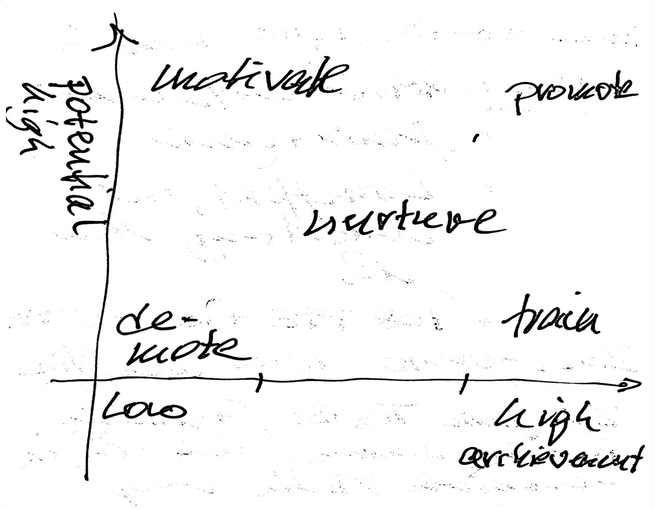
to reduce the number of people in the company

So treat this with awareness and care.

Once a year, engineering managers should meet, preferably together with HR and a works Council representative, to share insights about all the personnel. The classical way is, to go through each development level and discuss the dimensions "potential" and "achievements" and place each person on a 2D map in the fields "below expectation", "as expected", "above expectation".

People above expectations should be promoted, i.e., be confronted with new challenges. People with low potential, yet high achievement might need extra training. High potentials with low achievement might need motivation. low%low people should be demoted and everyone needs nurturing.

Generally, there should be an up and up movement. This could be an KPI, like number of promotions per year.



Sources/inspiration:
https://www.controllingportal.de/Fachinfo/Funktional/Personalcontrolling-Personalbilanz-Mitarbeiterbefragung.html

Money, Salary, compensation, € and $

There is a great interest in this chapter by many parties. Eberhard Wolff had one comment from Sonia in his podcast (linked below) who said: “Seniors are better at hiring negotiations”. Senior positions in companies are better paid than junior positions. We in this area, we traditionally have a distinction between juniors and seniors.

Or course this immediately leads to senior labelling of people to gain more salary of a higher daily compensation for freelancers. It might also lead to a very static image of being senior, not catering for the fact, that when learning new topics, we always start as junior. And this in turn is counterproductive for building a learning organization with learning people in it.

But how much?

This article is not about money. Yet, to satisfy the questions, here are some numbers I heard recently (2022, if not stated differently). The variation is very high and this is highly circumstantial.

Freelance:
India per diem, sub-company of a German Industry: €200 (I would like to know, how little the engineer gets, probably something like 5€/h as senior).

Ukraine per diem (in wartime, not negotiated): €80/h = 640€

German consultancy internal full cost(2019): €50, customer pays €80-120€/h

German freelance top expert of top consulting company: €1500

Usual fee of German consulting companies: €900-€1200

Employee:

DevOps Individual on XING: 130.000 p.a. / 200 days = 750€/ day (I doubt, he'll get it)

Boston senior Java Engineer (2019): $150.000 - $180.000

Senior Java engineer, Hamburg: €70.000 - €85.000

Junior engineer: as low as €35.000

So, whatever of these numbers might be correct, the salary span is worldwide very high, ranging from probably well below < €10.000/year in the far east up to 180.000 in the US. This of course is an estimation. But a factor of almost 20 in salary span is large.

A random search in the internet reveals similar stuff:
https://codesubmit.io/blog/software-engineer-salary-by-country/

https://qubit-labs.com/average-software-engineer-salaries-salary-comparison-country/

https://www.levels.fyi/comp.html?track=Software%20Engineer&showAll=true&ref=home_page_notification&ref=homepage

Other reasons, why experts like to work at a company

Freedom to pursue things they render important, Open Source.

tbd.

Heterogeneous people in heterogeneous teams

engineer are of different types:

- the jumper
- the legacy chainsaw
- the qa-liker
- the frontend-tiger
- the backend gruffy
- the generalist
- the expert

Disorder, Chaos and innovation - small improvement steps or leapfrogging innovation?

This whole document tries to describe the Human resources process and responsibility of engineers in the company from day 1 to them leaving the company. Order and responsibility are crucial to the success of people and company here. Yet there is something to be said for not sticking to order, to the fringes of the process. Excellency and innovation happen at the fringes of order. They need order as a pillar but at the same time freedom and areas of chaos to develop. Therefore companies that are effective in change and in innovation, i.e. excellent companies, will have the ability to manage disorder, to embed some chaos and to develop people and teams in ways that are not known to them today.

As a management group you should be able to ask yourselves, whether perceived disorder has to be forced back to order, or if there might be a chance to learn something new.

High performance teams will to my experience always look out for diversity and some level of argument or embrace queerness.

Leapfrogging or continuous improvement, what to choose. I think both are needed in excellent organizations. Leapfrogging will produce a lot of failure for companies, before you hit gold. So you might want to have a stable, continuous improving reliable growth culture aside from your leapfrogging organization's parts to be able to learn from all the fails, the leapfrogging produces before eventually you strike gold.

References

https://youtu.be/YnfWJg-0Zr4, Eberhard Wolff on junior and senior engineers

the OTTO story, Johannes Mainusch, from Scrum mit user stories chapter 12

Scrum mit User Stories, Ralf Wirdemann, Johannes Mainusch, Astrid Ritscher

https://www.jimcollins.com/concepts/level-five-leadership.html#:~:text=Level%205%20leadership%20is%20a,and%20its%20purpose%2C%20not%20themselves, Jim Collins on leadership

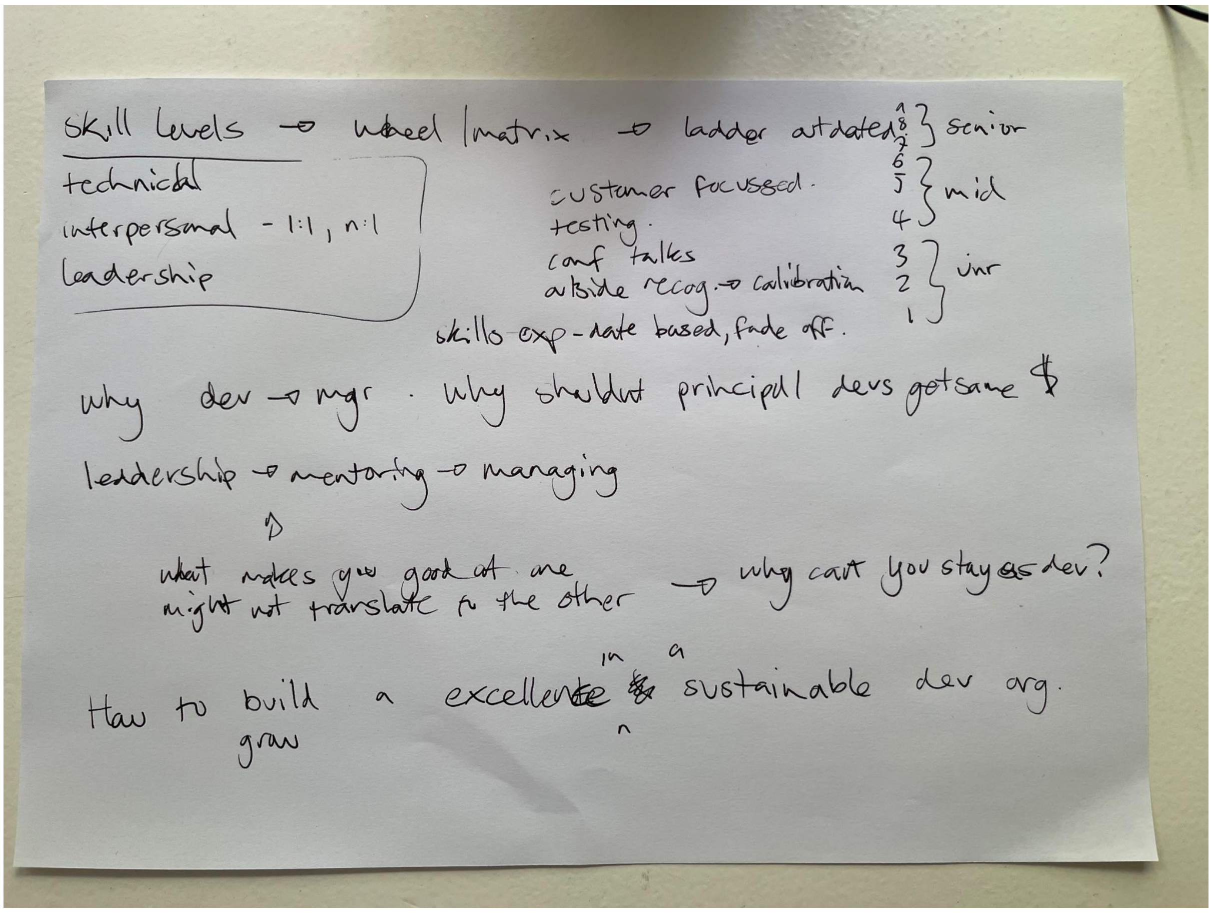
EOD

In the hope, that this will serve to improve companies to become top class in digital product development.

EOD

Additions

Robs Braindump



skill levels → wheel | matrix → ladder outdated

technical
interpersonal - 1:1, n:1
leadership

customer focussed.
testing.
conf talks
outside recog. → calibration
skills exp - date based, fade off.

9
8
7 } senior
6
5
4 } mid
3
2
1 } jnr

why dev → mgr. why shouldnt principall devs get same $

leadership → mentoring → managing

what makes you good at one might not translate to the other → why cant you stay as dev?

How to build / grow a excellent in & a sustainable dev org.

I did a 3 minute google search for software development competency model and found this page which gives some ideas that you could turn into a wheel (although this list format also works)

https://sfia-online.org/en/sfia-8/sfia-views/software-engineering-competencies?path=/glance

I'm not suggesting that this would be directly used or that I agree with all of it but just to give a little idea. This list misses things related to leadership and communication/collaboration skills so misses key areas of a individuals development

Google Level System

Here is some more info on the levelling system at Google. Basically the levels apply to everyone in the company and have salary and stock grants bands associated with them...this means that you can get paid more as a software engineer than a manager if you're more experienced. The only difference between software engineer and software engineering manager is the amount of time spent coding vs managing people.

@johannes: I think gives you a possible walk to answer Tchibo's request about Junior vs Senior engineering with something that has levels but is essentially more fine grained...as a reframe for them this is the same scheme they have but with more levels and something that can be applied company wide to ensure fairness AND a better promotion model for software engineers that don't want to do too much management...more sustainable so Tchibo can retain its best engineers:

https://candor.co/articles/tech-careers/google-promotions-the-real-scoop-on-leveling-up

This webpage shows the different levels at top tech companies

https://www.levels.fyi/?compare=Google,Facebook,Microsoft&track=Software%20Engineer