A Open Framework for the Career Progression of Software Engineers

Intended Audience

This framework is aimed at teams of all sizes where conversations around career progression are unclear, for example:

- The company has no consistent way of articulating what it means to be junior, senior, etc, which leaves individuals uncertain of their own progression, and unclear what to expect of others who have more experience
- Where teams have promotion cycles that seem arbitrary or unjustifiable
- Where more experienced developers are focused entirely on coding and not upleveling the team
- Where there is attrition because of poorly managed career expectations
- For individuals to be able to get a feel for the kind of behaviours that the values-first companies expect of all employees
- For founders to instil the right culture from the start as they begin to scale
- For companies that are ushering developers into managerial positions as the only path forward

Backstory

The first version of this framework was created by the JIRA Service Desk (JSD) team at Atlassian in 2017. At the time, Atlassian was experiencing exponential growth, and while this success was an incredible journey in an environment of moving quickly, breaking things, progressing, creating, innovating, some of the supporting processes didn't grow as quickly as the rest of the organisation. It increasingly became a vital need to empower team leads with the tools to be able to steer career growth, while also ensuring the company values were a fundamental, accountable part of how we worked and how we succeeded as we rapidly scaled.

The difference between this framework and others is that it was written by developers, not by managers. Input, over a period of weeks and months was given by engineers of all levels from senior through to chief architects (the latter isn't listed here, as the differences between architects and principal developers varies wildly from org to org). Once a V1 was ready, trials were led with the development teams to measure its usefulness, while team leads, development managers and heads of engineering iterated over the language used to make sure it wasn't vague, repetitive, or that the delta between levels wasn't too wide. Atlassian have since moved to a different framework (which has this as its DNA of sorts), which enables me to open source it to the community. Since then, this has been adopted by organisations including Domain.com.au and has helped provide a basis for positive conversations to hundreds of eager engineers.

How To Use This Framework

The left-hand column of the table below lists the core behaviours that become instrumental when an engineer reaches that point in their career, so for example, junior developers will initially be focused on collaboration and improving their craft, whereas seniors as they progress will focus on not only their own technical brilliance, but on levelling up those around them. Principal developers are expected to have an even greater reach, as their focus increasingly moves into one of strategy and company enablement. However, it's the expectation that these core behaviours continue to grow, which is why the focus is one of continual growth as an engineer progresses, rather than switching to something else. Where there were opportunities for growth at other levels these have been added.

There is no right way to use this document – all I ask is that adoption isn't in a silo, that you adapt it to fit the values of the company you work for, and, most importantly, don't use it as a promotion gate. It's intended to be the basis of conversation around behaviours, and one in which developers can refer to over time, rather than have a series of checkboxes.

I welcome iterations, specialisations and other improvements. If you've found it useful, I'd love to hear about it.

Managing work	Able to complete tasks that are given to them in a reasonable amount of time.	Properly splits work into small individually deliverable items / PRs. Able to properly handle workload and expectations, which includes realistic estimates. Should be able to plan their week, understand sprint goal and help sprint execution. Is able to strike the right balance between home and work.	Can split up a feature to work on with 1-2 other developers to work on together. Can break down the feature phases and individual stories. Ensures that work gets done. Keeps delivery time promises and challenges unrealistic delivery expectations. Understand the priority of planned work vs side-projects; pragmatic when to execute on it. Can Lead a Feature from start to finish including investigation phase.	Looks actively for longer term goal items, manages several initiatives in parallel. Coaches others in striking the right balance between quality and getting things done (effective resource management). No job too big and no job too small.
Quality Orientation	Properly acts on PR feedback (seeks to understand and learns to not repeat the same mistakes twice) Has pride in their own code. Attention to detail.	Thinks about code maintainability, testability and follows best practices.	Work assigned to a Senior Dev gets done without requiring supervision/double checking. Balanced approach to tech debt, for themselves and the team. Negotiates with the product owner as necessary. Leaves code in a better state. Doesn't add to code debt because "it is too hard" (the boy scout rule)	Instils quality orientation in others, goes beyond 1:1 teaching. Pushes for solutions at the platform instead of individual team/product level. Actively reduces the risk of projects.
Core competencies for Developers and above				

Core Competency	Grad/Junior	Dev	Senior	Principal
Competencies for all developers (this should be read as items from the right encompass items to their left)				
Collaborating	Able to work with others		Actively fosters/drives collaboration, be it pairing with developers or encouraging others to do so. Include developers in technical discussions giving them a chance to chip in. Can actively engage with other people outside the team to solve specific problems.	Knows how to nurture and foster relationships with key individuals across the company.
Communication	Able to articulate problems and raise concerns. Shares information that concerns others with blogs. Actively listens to others.		Keeps emotions in check when communicating. Able to explain complex concepts and decisions to others. Can be through internal workshops and brownbags to help foster a good development culture. Awareness of external (i.e. non tech) stakeholders, knows how to adapt language to audience. Manages information effectively, guides others to change media / approach when communication issues arise (Slack vs email vs page vs meeting)	Can manage information effectively and widely with teams across the company.
Continuous learning	Has had experience in a few different languages and is able to self-learn new languages. Quick learner, curiousity to learn what they don't understand. Learns unguided when pointed in the right direction.	Knows what is happening in the field / technology advances. Starts to challenge themselves, get out of their comfort zone; learns what is necessary to get the job done.	Has working knowledge about emerging frameworks, technologies, libraries etc. Is balanced in discussions about pros/cons. Is not defensive of failures; is able to accept it and move on.	Gains deep knowledge of technologies, understands what is considered *magic*. Is able to learn whatever necessary to solve any technical problem.
Initiating Action	Doesn't sit idle when running out of work. Seeks help when stuck, but with proper understanding of the problem they are stuck with.	Figures out themselves what needs to be done, i.e. can handle non-fleshed out stories. Understands the responsibilities of the product team (PM, Design, Engineering)	Understands scope and feasibility of initiatives. Manages scope creep and avoids over-engineering. Solution orientated, i.e. doesn't just bring up problems, investigates possible solutions upfront. They unblock themselves - often engages with other teams or just does what's needed directly.	Takes the initiative to unblock others. Tackles the challenges the rest of the team consider "too hard" Can influence roadmap decisions based on technical expertise.

Work Standards		Code in PR is already in a good state in terms of best practices and agreed ways. Doesn't cut corners. Receptive to feedback in PR. Has shadowed recruitment interviews.	Confidently reviews PRs and guides others in improving architecture, code maintainability, readability and testability. Is constructive and empathetic when giving feedback. Adapts feedback for the recipient, and remembers to be solution orientated in feedback. Maintains the quality bar for the team. Is actively involved in recruitment (resumes, phone calls, exams, face to face)	Sets the quality bar for the wider team. Consistently and efficiently works at a high standard.
Stress tolerance		Keeps a cool head when approaching problems	Knows when to step back from a conflict or potential conflict. Reacts calmly during an incident. Manages own emotions.	Actively resolves conflict, for example decision impasses and disagreements, and passionate technology discussions. Steers others through high pressure situations, for example incidents and tough deadlines.
Monitoring information		Keeps up with information concerning their ongoing work. Updates outdated information.	Is aware of what is going on in other parts of company in terms of technology.	Presents relevant information where and when it might have an impact.
Decision making		Should be able to make decisions on most stories, for example minor design queries, bug fixes, filling in gaps when they're obvious. Is aware of trade-offs and know that they don't know everything. Balanced approach to escalation.	Should be able to make some wide-ranging decisions, such as feature scope, relative priority of tasks, technical direction, shipping decisions (such as timing, releasing early) and so on. Is able to make technical trade offs that last years and is able to escalate / bring agreement where necessary. Guides decisions on the best technical approach and lead the team to adopt those technical approaches. Involves developers in the decision-making process so they can learn. Draws on facts (e.g. analytics) to drive their decisions. Where those facts aren't readily available, will seek them out and create tooling if necessary. Understand that own opinion/gut feel is incomplete. Makes a call on technical architecture and decisions, involving stakeholders where necessary (principal engineers, product management, founders). Has the self confidence to make a call on their own (i.e. if no input is readily available) in order to unblock the team. Seeks forgiveness rather than permission.	Should be able to make organisation-wide decisions, such as framework adoption, feature priority, complicated release mgt, technical approach etc. Is not afraid to make hard calls, ensures decisions don't drag on, balanced approach to effort/return.
Adaptability		Should be able to react to change positively. Able to see and analyse decisions from different points of views.	Should be able to react positively, and support others, through change. Doesn't react emotionally when things beyond their control change. Runs with a decision once it is made without continuing to argue. Doesn't hold grudges about decisions.	
Continuous improvement	Learns and follows processes.	Shares knowledge with others. Actively looks to improve their own skills. Has interest and curiosity in improving their skills, processes and patterns adopted by the team.	Helps others to learn from own past experience. Should be looking at new ways to approach work and make the team more efficient (e.g. suggest new patterns and process improvements, training) Preemptively learns the skills and technologies that are on the horizon (e.g. React, Redux, Flow vs Typescript).	Helps the wider team master new technologies, the internals of the product, etc. Drives the team towards better patterns and processes (e.g. moving towards *You build it you run it* and what that looks like for the team). This could be via brown bags, workshops, blogs, bootcamps etc.
Competencies expected for Senior and above				
Customer orientation		Shows empathy for customers. Can wear the customer's hat.	Encourages developers to strive for customer value, breadth of understanding not just technical purity. Balanced approach to customer value. Guides Knows when and how to steer other developers to hit the product and platform right level, knows when to decisions. stop/ship. Progress over perfection. Should always be thinking and can articulate the vision in terms of customers and of the company in terms that not just code. Can argue against technical perfection when ne cessary. Have enough understanding of the product, and empathy for how customers use it, to be able to steer product decisions where necessary. Knows the business well enough to be able to influence better decision making in the team.	Have enough context and breadth of understanding across both products and platform. Knows when and how to steer product and platform decisions. Fundamentally knows all aspects of the company's business, and can articulate the vision of the company in terms that can be translated into technical direction. Can perfection when identify cross-cutting risks.

Earning trust		Doesn't omit testing and silently cut corners. Follows up on what they promise. Can estimate their own work and set realistic expectations.	Go-to person for technical questions. Has confidence, and a track record, to make solid technical calls. Leads can rely on Snr Devs to take over and own a problem/task. Sets realistic expectations with stakeholders. Always takes ownership of own mistakes, never blames others.	The default go-to when there's no-one else to ask. Able to communicate and back a contentious decision, and take full ownership of the results of that approach. Doubts are eased because of implicit trust. Has a track record of making successful calls. Actively encourages developers to share knowledge.
Influencing		Provides constructive input to technical discussions. Provides constructive opinion in design sparring, workshops and ongoing feature development.	Could work closely with PO and Design as part of the triad to come up with solutions that are well balanced in terms of technical complexity/feature value. Takes ownership / responsibility of the technical implementation of a feature.	Raises areas of improvement. Encourages and guides other developers with implementation thereof (i.e. doesn't just do it themselves) Helps drive technical decision making, keeps everyone on track (outcome focused). Influences developers to implement technical solutions in a consistent approach across the company. Influences organisation-wide decisions where appropriate, e.g. protocol and tech stack decisions, testing/monitoring approaches (e.g. contract testing)
Planning and organising		Can deliver a well-defined feature from start to the end, works with principal engs, PO and Design to deliver a quality outcome. Helps break down into epic/stories.	Leads other developers in bigger features. Notices when things are going pear-shaped. Can identify risks. Can manage/plan sprints. Should be able to see opportunities where duplicated work is being performed and prevent it. Should be able to know when duplicated work is actually desirable to decouple timelines. Should be able to track future work to fix up workarounds that helps us ship today but fix it in the future.	Needs to have awareness of what multiple teams are working on, Sees and prevents cross team inefficiencies. Takes a practical and realistic approach to estimation, manages tradeoffs and scope creep. Mitigates risk in a complex environment (e.g. external/platform teams). Plans and orchestrates complex (and potentially risky) rollouts, involving many dependencies and teams.
Coaching			Mentors more junior developers through PRs and in-person. Shares past architectural mistakes to avoid future ones (this goes beyond simple code/design mistakes). Encourages others to learn new things/techniques. Should be encouraging empathy to developers to take a customer's point of view when coding, e.g. how would people use this. Tailors the coaching to the person receiving it - empathy.	Proactively ensures that new starters are brought up to speed at scale and are effective. Empowers those around them to educate the team. Tangible success in teaching the team. Always eager to share experience with positive attitude. Actively encourages others to share knowledge. Actively looks for knowledge gaps in the teams and fills them. Provides growth support for Senior Devs in bringing them to the next level of their career.

Innovation			Raises the technical excellence of the team by proposing and is the driver of technical/architectural changes, including education and roll-out. Evaluates suitable technologies not used inside the company today. Curiosity about using new things (i.e. pet projects)	Improves productivity/technical excellence across multiple teams. Role model for other developers to strive for in terms of knowledge and effectiveness. Promotes a culture of innovation, encourages, and backs others in implementing ideas. Delegates/shares ownership of implementation to/with others in order to grow them.
Competencies expected for Principal				
Inspiring others			People want to work with you.	Inspires by technical excellence. Leads by example, hands on. People are motivated by your approval.
Facilitating change			Brings the local team along when there are significant changes happening across the wider team (e.g. feature flag adoption, monitoring, React, platformisation). Sees change through to the end.	Takes ownership of leading change, e.g. pioneering the adoption of new technologies and company initiatives.

Positive approach			Manages own emotions; awareness of effects of one's emotions to others on the team. Is a role model for all of the company values. Upholds the Tech Principals in a balanced way.	Always keeps a positive attitude, even if the decision isn't personally preferred. Understands the importance and impact of voiced opinions; negative outbursts have a magnified negative impact on those who hear or read it. Is empathetic during contentious technical decisions.

Credits

Simon Keefe – Snr Eng Mgr @ SiteMinder, Brad Baker – Snr Architect @ Atlassian, Scott Harwood – Co-Founder @ Code Barrel, Michael Ruflin – Principal Eng @ Atlassian, Marty Winsen – Snr Team Lead @ Atlassian, Clem Capeaux – Team Lead @ Atlassian, Eddie Kaiser – Team Lead @ Atlassian.