Sarah Zohn
C-Level Mentor and Author for B2B technology companies
April, 2020

VP R&D
Now more than Ever, Act to Improve Software R&D Productivity!

Does it really have to be so difficult? That's the question you should have on your mind continuously.

Let's start with an experiment. Forget your title for a moment and think about one of your engineering projects: a new feature, across the board quality improvement, a new architecture for the software product, replacing a fundamental component of your product, changing the underlying database, etc. How would you approach such a project? First, you'd understand the problems you're trying to solve, second, identify the main principles of the solution, third, identify qualifying tests, then you move to design and execution. And since a piece of software is a live entity, it needs ongoing maintenance throughout its lifetime.

You know from experience that in order to embark on this journey, you start with allocating quality time with yourself and then a bunch of meetings with your best engineers, until the subject is fully deliberated. Once it's well understood and designed, your best approach for the development phase is to employ Agile methods to execute the project with minimal setbacks. Once finished you move to ongoing maintenance with whatever methods you normally employ for this kind of work. That isn't so difficult, is it? You have definitely done this several times, or else you would not have had your title.

Now, can you realize that you are the best trained person to improve the productivity of your organization? The only difference from a software project, and it is a significant difference, I admit, is the human factor. But that too isn't a big secret, I can assure you because I have already mentored hundreds on excellence in execution. There are a few principles to study and practice on a smaller scale, and KABOOM, all of sudden you realize that you've become an emotionally intelligent VP of R&D, or to put it in plain language, you are now able to get out of your organization results that are a lot closer to your high standards of execution.

But first, let's understand why you feel that you're inside a pressure cooker most of the time. There are objective reasons for that. Responding appropriately to these noise creating factors, i.e., removing obstacles, is an *essential part of your job*.

- Firewall R&D is only a temporary solution.
  Some VPs of R&D with a tendency to centralize everything around them will react to the general disruptive ambient noise by putting a firewall around their unit. With time, this wall needs to get thicker and higher if it is to function at all. It's a defensive approach that doesn't consider the needs of the constituencies outside the wall, and therefore, it creates a workable environment inside the confines of the wall on the account of chaos outside it.

As a VP in the company, you should be thinking about the benefit of the entire organization - you are a company executive after all, and a member of the leadership team. If you like to serve your ego with your executive behavior, you'd take this approach, but, the only "benefit" is to you and not to the company. I hope I discouraged you from this behavior, and if you're already there, you should recognize it and start designing entryways into your organization that you manage and control as to minimize the disruptive effects.

What are the chronic reasons that disrupt R&D productivity? You will talk about productivity and velocity issues if there are delays in delivery, projects that overrun their original estimated hours, buglists that grow indefinitely, stops and restarts of work items. Do you recognize these symptoms? You're not alone. Let's review the typical reasons that cause so much disruptive context switching, and see what you can do about them.

- "Ah, if only we didn't have customers! How wonderful could life be!" is a common sentiment of development folk. Real life, however, puts the highest priority on customer issues that have to be addressed by engineering staff. If you analyze the time that's spent on customer issues, you'd realize that most of it is spent on looking for supportive information, analyzing the problem and finding the root-cause. This skill - debugging - is actually a hard skill that many engineers are not very good at, and therefore, if tasked to do it, they will spend even more time than necessary. Moreover, problems tend to come in clusters, i.e., if you resolve one problem at a time, you're missing the opportunity to prevent other problems that share the same or similar root cause.

  If you haven't done so yet, you should have a dedicated team to do the debugging of escalated problems. It will save you hugely in context switching - a common stealth productivity killer - and it will save you engineering hours - your scarcest resource. If you already have such a team, think of staffing it by periodical rotations; you will also contribute to improving over time the quality of your developers and their work.

- "The boss wants it now!" is the worst offense if you allow it to disrupt the work in your organization. If you were the TV producer of "Iron Chef Japan", would you allow the king of the country to interfere in the middle of a shoot? You have to put a stop to such disruptions and show publicly, albeit in a subtle way, that you did so. If you allow the boss to disrupt the work, there's no reason why others won't follow. This example of bad behavior serves as a lesson for everybody not to take plans seriously.

  Work-in-progress in development should be protected by a freeze of two or three weeks, typically. In small companies it's a symptomatic behavior of the CEO - the boss - to make special requests and interrupt the work. You should treat such a request as another one that goes into the queue, like customer requests or Sales requests with its appropriate priority. It's your job to *push back on high-level interrupts*.

- "Why are we missing estimates so badly?" You probably are so used to estimation overruns that you have tired of asking this question. The proportion of development tasks that run on overtime is large, very large indeed. There are two main reasons for these underestimations. The first, we've already discussed, and that is the context switching that comes from outside pressures. The second, which is related to the first reason, is that the wrong people do the estimations. If a product manager or a

development team-lead does the estimation, there's no doubt that they would have missed architectural or interfacing complications.

That's why the Agile team approach to Sprint Design and Planning gives the best estimates. Moreover, once developers are under pressure to deliver a task on time, they will look for shortcuts. I am sure you participated in such "conspiracies" and you certainly felt the conundrum of "pay me now or pay me later". Overall you'd pay less if you pay now - you know that, so insist more, as much as possible. Another reason for underestimations is that the definition of development requests, typically produced by Product Management, are too broad or too vague. Here the approach is to educate everyone on the principles of MVP (minimally viable product) and insist on practicing them.

- What about quality of work? You keep paying for it dearly, and you've become complacent. "I don't have enough QA resources" is your justified excuse, but you learned to live symbiotically with this illness. QA is so behind development that you test tips of icebergs and you miss underwater cracks and opportunities to prevent calvings.

  First, complacency isn't part of your job description, so quit accepting this as an "act of god". Second, find someone in your organization for whom quality of software development is a passion, or hire one - they do exist. Empower this person to do Continuous Improvement initiatives throughout the organization in the forms of: Coding Manifestos, Code Reviews, Educational seminars, Study of the Principles of Test-First, etc. All aiming to raise awareness of developers and attack the quality problem at its root.

  Then, over the other side of the fence, focus on QA automation, and increasing the coverage of the test suites. Unfortunately, throughout my career I have come across only one place where there were 3 times more QA engineers than development engineers. This was the one and only place where I experienced that velocity and productivity issues were kept at a minimum, to the extent that I could have declared them as non issues.

- A weak structure is a formula for disasters. It's true for all types of engineering: constructing buildings, bridges, or designing the architecture of a software product. Once the product is released and is in use by customers, the normal pressures for more features that are coming either from Sales, Customer Success, Marketing or from customers, usually push investment in the architecture to a lower priority. You know it's a mistake that eventually you will be paying for, dearly, and your key engineers know it too. You've heard it called "Technical Debt", already signifying that it is something you will have to pay for, eventually, with interest.

  Your job is to put some sense behind the allocation of the - always in shortage - engineering resources. Here's a baseline for you to consider: an allocation of 50% to satisfy requests for sellable features, 20% for addressing bugs, 30% of your organization's capacity dedicated to underlying technologies and lower layers - the architecture. Under no circumstances should you allow this allocation to go below 20% continually, i.e., in every Sprint, or Freeze. If you are having difficulties justifying this allocation to your management, think "Performance at Scale" and there you will find

how to justify it. Consider it a *hard principle* of your function, and with it you would build your reputation as a VP R&D of high quality.

So far I have covered those aspects of your job as VP R&D that concern the technical sides; these are always important, and in times of crisis management, they are also urgent and critical to the success of the company. But, above and beyond the technical sides, the absolutely most important aspect of your job is to optimize the nurturing of the R&D employees and relate to their unique needs, i.e., the *human sides* of your job. R&D work has only three types of raw material: People, Time and Tools. Knowledge is the collective brain power of your staff, and high motivation is the main productivity driver.

Here are a few points you should consider. I have seen time and again that even simple and small progressive steps in following these principles will make noticeable improvements in the contributions of individuals and teams, i.e., improved productivity.

- Organizational stability has many faces, i.e., the structure itself and the vulnerabilities inside the structure.

  As most VP R&Ds you will find yourself often thinking about the structure with the hope that a better structure will solve most of the problems. Unfortunately, the endemic problems often remain unsolved, because the structure isn't at fault. We've already reviewed above most of the failure mechanisms of R&D organizations and none of them pointed to the weaknesses of the structure. My personal advice to you is to refrain from reorganizations as much as possible. There are only a few reasons that justify a reorg, and those are when your scope of responsibility is abruptly changing, or when you adopt a new overall process, like Agile. Especially refrain from reorganizing if you are a new VP R&D to this organization; doing so will hurt your reputation as someone who is trigger happy without knowing the situation in depth.

- Inside the structure you should look for all kinds of bottlenecks and address them proactively.

  - When you have knowledge areas that are covered by just one person, it's a knowledge bottleneck. Team up two people to cover two areas, as in: Rob and Emily cover two areas together instead of each of them covering one area independently. This pairing of responsibility will create the minimal redundancy and alleviate the bottleneck.

  - Accept that it takes three to six months to ramp up new developers to their full potential. Yes, these learning curves can be shortened with proper mentoring and by creating buddy relationships, however, they can't be ignored. Even a star programmer needs to learn all the intricacies of the software he's working on. And by the way, TBHs can't do any development work. I have seen plans that calculated working hours of employees who are yet to be hired as if they were already available and in full capacity. That's a form of lying to yourself - try not to add insult to injury.

  - Find those developers who have a multiplier effect, i.e., they are not only excellent at what they do, they're also the ones making other people's work better, either by creating tools for everybody, or by informally advising others on better approaches. Find these people and find ways to use their skill to the

fullest. You could even rate your employees on a scale of 1 - 10 based on their ability to improve the work of others. Assign them as internal coaches, put them in influencing positions, team them up with others, or use your own creativity to make best use of this talent. These multipliers hold the key to resolving all bottlenecks. And don't forget to treat them differently, *recognize this special talent and contribution*, even if it costs you managerial cycles.

- Enhance the collective knowledge by periodical rotation of people among teams. Because expertise and teamwork are so critical for software development, I would not recommend commoditizing your developers, i.e, don't use all developers as generalists that can be assigned to any task. I will give you a rule of thumb for rotating engineers: give engineers the right to request a transfer to another group, once a year, and ensure that no more than 20% of them rotate in one year. This way you will enhance both knowledge and motivation in your organization.

- If you employ Agile as a key process, take it up a notch and follow the principles more tightly, or expand its coverage to include regularly interfacing groups such as Product Management or Customer Success. If you're not, then at least study the method and its principles, and focus on the problems it's addressing. There are many lessons that can be learned from Agile that are applicable without adhering to its rules.

- Last, but not least, the team leads of your organization are your favorite children and that's how you should treat them. Productivity and Quality fall directly on their shoulders; they are the last line of defence. No matter how many layers of management are between you and them, make an effort to show them recognition, and personal presence. You want them to see, each and every one of them, that you know what they're working on, and that you appreciate them.

Management and leadership are about organizing work, overseeing quality of delivery and leading people. In R&D the distinction is clear between the needs of work and those of people. You should see this as two separate sets of skills that you should have, or, two different toolboxes at your disposal. The more you see it like that, the clearer your role becomes, and with it the rainbow that will appear in your clouds.

Sought after C-Level Mentor, consultant and speaker, Sarah Zohn is a highly respected expert on Operational Strategies for Growth and Go-To Market for hi-tech companies. She brings broad experience from her positions as VP in mega multinational corporations as well as from coaching dozens of B2B technology companies and guiding them with game-changing strategies. With hundreds of consulting interactions with startups, Sarah also knows well the nature of startups and their ecosystem.
Sarah is a graduate of B.Sc.E.E and MBA from the Technion Institute of Technology in Israel. Her executive level positions at EMC Corporation over 15 years covered hardware and software Engineering, Services and Customer Success, Product Management and Sales. She also served as Deputy Executive Vice President, where she participated at the top of the decision making

processes of EMC’s leadership. Sarah’s experience spans over continents, cultures, several technology-based industries, and hundreds of mentored managers and leaders.