PharmaSUG 2022 - Paper LS-137

Empower Your Programmers: Get the Most from Your Programming Team

Carol Matthews, Advance Research Associates, Inc.

ABSTRACT

Experts agree that empowering your workforce can provide a path to increased productivity, higher quality, and greater employee retention. While the concept of empowerment seems like an obviously good idea, too often leaders behave in ways that do just the opposite. It is important for leaders to understand what empowerment really means, how to foster an environment that promotes employee engagement, and how that can be translated into a happier, more competent, and more productive workforce. Particularly in the highly regulated technical world in which pharmaceutical programmers work, it can be challenging for leaders to trust that their staff will make effective decisions on their own. We will discuss what it means to empower your programming staff and practical ways to do so while reducing risk and realizing the benefits to be gained.

INTRODUCTION

The concept of employee empowerment is not a new one – there are numerous books, articles and blogs that describe what it means and how leaders can behave in ways that allow employees to feel empowered. The abundance of information can be daunting to sift through – from books that take hours to read, to lists of "quick tips" that are so "quick" and general that they're difficult to see how to apply in specific situations. In this paper, we'll discuss what empowerment looks like in practice with a programming team in the pharmaceutical industry.

In addition to learning what an empowered programming workforce can look like, you'll learn various ways leaders can encourage an environment where programmers feel valued and able to actively engage in the research process beyond "just follow the specs." We'll discuss how to minimize the potential risks of implementing these techniques so managers can recognize the numerous benefits of an empowered workforce.

While for simplicity this paper will discuss empowerment from the perspective of managers and programmers, the concepts apply to all departments and all functions where there is a leader and those who follow that leader's direction (e.g., programming project teams, each with a lead programmer and several production and validation programmers). In addition, the concepts discussed here also have value to those who are *not* leaders – it will provide insight into what your managers are doing (or not doing) and why, as well as provide food for thought when trying to figure out what you are looking for from your own leadership and company work environment.

ELEMENTS OF AN EMPOWERED WORKFORCE

Defining "empowerment" in words is relatively simple – one source defines empowerment as giving employees *the ability to make certain decisions, rather than only following the standard procedures when working independently*.[1] Another way to describe empowerment is that *empowering your employees means giving your team members permission to take action and make decisions within your organization. It also means there is trust and understanding in place to ensure these actions are in line with company goals.*[2] While simple to define, empowerment is often much more challenging to understand what it means in day-to-day processes and behaviors, and to subsequently implement.

If you're a manager who wants to empower your programmers to share their ideas, grow their skills and work independently while in alignment with company goals, it's important to understand the specific behaviors you're trying to encourage. Table 1 below shows some examples of specific behaviors that managers may see when working with programmers.

Powerless Behaviors	Empowered Behaviors
Programmers just "program to the specifications" so… • Data issues are not found or communicated to others • Databases lock with issues or locks are delayed at the last minute and the project team looks for someone to blame	Programmers proactively check that data fits specifications so… • When data does not appear to fit expectations, programmers ask the data manager for clarification • New edit checks are added to the data cleaning process and data quality improves
Programmers don't volunteer for anything, so process improvements are left to a handful of leaders who have little time to develop them	Senior programmers notice a process that is repetitious and develop macros to make that process more efficient for everyone
Junior programmers lack mentorship, do not seek help, and fail to develop	Junior programmers actively ask questions of senior programmers and grow their skills
Managers *and* programmers feel overworked, frustrated, and unappreciated so leave after about 2 years	Managers *and* programmers feel like their work is important and appreciated so stay for 5 or more years

Table 1. Behavior Comparison

The specific behaviors above exemplify several characteristics that embody an empowered workforce. Let's look at these behaviors in more detail.

EVERYONE IS ENGAGED AND INVESTED

What does it mean to be engaged and invested in your work? Simply put, everyone is interested in what they're doing and cares about quality of their work. We want programmers to be committed to making sure the results of whatever is created by their programs, be it a dataset or a summary table, are correct. We want them to care about working efficiently so those results are delivered as quickly as possible while maintaining quality.

For programmers to truly engage, they need to understand that what they do has direct impact, both on the health of patients and on the health of the company. It can really energize a programming team to know how their work will be used to prove that a treatment works, and how that could impact patients' lives. For example, knowing that a cancer treatment they're working on could extend a patient's life by 4 months so that patient can see a child graduate or get married can really put the importance of meeting timelines into new perspective.

In addition, programmers need to understand that they'll be held accountable for their decisions and the quality of the work that they produce – both positive and negative. If there is no recognition for work done well, and no feedback or consequence for work done poorly, there is little reason for programmers to believe that what they do has any impact. If what they do doesn't matter, then what does it matter what they do?

INDEPENDENT DECISIONS ARE ALIGNED WITH COMPANY GOALS

Programmers make decisions every day, from how they'll execute the specifications they're given ("Do I use SQL or a data step to combine this data?") to how best to check what they've produced is correct ("Do I just look at the intermediate dataset or do I program a few specific checks?"). Beyond the everyday decisions made while writing code, programmers also make decisions on whether to actively participate in team meetings, whether to stay a little longer to finish a task, and whether to ask a clarifying question. It's important for programmers to understand the larger framework of the world they're working in, from overall company goals and values to specific output quality expectations, so they can make those decisions in a way that is aligned with those goals, values, and expectations. In addition, it is also important that programmers understand what risks are acceptable, what the consequences would be for success or failure, and that they can take pride in their successes and learn from failure.

IDEAS ARE SHARED

A key to a dynamic team is the sharing of ideas for new and improved ways of approaching challenges. This sharing of ideas usually starts with a question. How do I do this? Is this really the best way to approach this task? Is there a faster/easier/better way to solve this issue? When programmers are comfortable asking questions of their peers and leaders, discussions can occur where everyone benefits. From junior programmers learning new skills to senior programmers working together to develop new macros that benefit the entire programming team, having a safe environment for everyone to collaborate can yield many rewards. For ideas to be shared freely, it’s important that all programmers feel that they can offer ideas without fear of recrimination and feel that their voice is heard.

INDIVIDUALS STEP UP AND GROW… AND STAY

In an environment where programmers understand their part in the drug development process, understand how they add value to the company, and feel that their contributions are appreciated, where the sharing of ideas is actively encouraged and accomplishments are acknowledged, most will willingly “step up” to help resolve issues and meet challenges. Stepping up can include anything from contributing discretionary time without being told or asked, to offering to mentor others who need help. When programmers know that personal and professional growth is expected and rewarded, they are more likely to actively look to improve and grow their skills. Growth is not just learning new skills; in many cases, senior programmers get more comfortable speaking up and have renewed sense of purpose by mentoring others. Having a broader sense of purpose, contributing tangibly, and feeling appreciated often foster a greater sense of job satisfaction which then leads to programmers staying longer.

METHODS TO EMPOWER YOUR WORKFORCE

Now that we understand the kinds of positive behaviors exemplified by an empowered workforce, the next step is learning specific methods to provide an environment where programmers can effectively contribute, make good decisions independently and grow. For managers and programmers alike, knowledge is the basis for good decision making, and the fundamental way to impart knowledge is through open communication. Once everyone knows what needs to be done, why, and how, leaders then need to plan for and support growth. After managers and their programmers have agreed on a plan, managers can support their programmer’s independence and encourage safe failure. Let’s examine these concepts in more detail.

OPEN COMMUNICATION

The fundamental method that impacts all aspects of empowerment is communication. Knowledge is the power that fuels success in an environment where everyone is engaged and working toward common goals. Information needs to flow in all directions – to and from managers and programmers as well as between different members of various project teams. There are a number of ways to foster these levels of communication, both formal and informal.

There are many settings for communication, both written and verbal. As leaders, regardless of the context, there are several principals to keep in mind:

- Provide purpose, context and direction so programmers can make decisions aligned with company, department, project and individual goals. It is important for programmers to understand not just *what* needs to be done, but *why* and *how* it needs to be done. From broad goals and strategies to individual programming tasks, understanding the big picture and where their work fits in is critical for engaging programmers and getting them invested in specific as well as overall outcomes. When making decisions, this provides a framework for them to work within to ensure they’re meeting company goals and expectations.

- Recognize contributions early and often, so programmers feel invested in outcomes and see that growth is rewarded.

- Ask for input where possible to encourage programmers to share ideas and demonstrate that their thoughts are valued.

- Provide positive feedback for communication to demonstrate that communication itself (asking questions, expressing concerns, offering solutions) is expected and encouraged while also showing that their participation in the communication process is appreciated.

Let's look at several ways that managers can provide venues where these principals can be put into practice and how those principals can encourage the empowered behaviors they're trying to support.

Department Meetings

Department meetings can be used to communicate broad goals, strategies, and what it means if these goals are (or are not) met. For programmers to make decisions in their day-to-day tasks that are aligned with the core values of the department, they need to understand what those core values are and why those values are important to follow. This is leadership's opportunity to provide programmers with an understanding of the general context for their work, along with a broader purpose and direction in which to couch their daily decisions. Providing this framework will help programmers align their decisions with company goals, as well as keep the entire team aligned and moving in the same direction.

Department meetings are also a great opportunity to recognize individual and team accomplishments and contributions, as well as to present challenges that were overcome such that the team can all learn from difficult situations others faced. By presenting both the good and the "bad" in a positive way, leaders can demonstrate a safe environment where independent decisions can be made and risks taken, but whatever the outcome nobody gets "beat up" for what they did.

Department meetings can also focus on soliciting feedback on broad-based initiatives or challenges that need to be overcome. For example, if the department is asked to find a faster way to generate tables so the company can be more competitive, a meeting can be partially devoted to brainstorming ideas for specific process improvements. On the other hand, if someone in the department worked out a new way to generate graphs more easily, a meeting can be devoted to that individual teaching the entire group how that's done so everyone can benefit.

Project Team Meetings

Beyond general department meetings, attending broader work team meetings can benefit programmers by showing how broader company and department goals apply to specific projects. These meetings can clearly define roles and responsibilities within the team, including expectations of cross-functional collaboration. Once initially defined, as the project progresses, the team can discuss challenges and solutions that may not have been reached if departments or individuals remain siloed. While these meetings take time, the benefits of including members from other departments (data management, statistics, medical writing) working on the same project promote the key aspects of an empowered workforce.

Project team meetings allow programmers to see and experience directly how their work fits into the process of completing a project with tangible examples that directly impact them and those around them. In addition, specific project information, challenges, and ideas for solutions can be discussed across the team so everyone understands where timelines and quality of work impacts those around them. When programmers understand the broader context of the project they're working on, they can better understand the impact of decisions that they make and align those decisions with project goals. Individuals who understand how the quality and timeliness of their work directly impacts the people they work with, will often volunteer extra time and effort without being asked; from learning new technical skills to working a few extra hours to meet a deadline.

One-on-one Meetings

The last communication tool for fostering an empowered workforce is one-on-one meetings. While these meetings are often discussed in the context of managers with direct reports, these meetings also occur between project leads from different departments, such as meetings between the lead programmer and project statistician or lead programmer and lead data manager.

When managers are meeting with their programmers, they can clarify expectations for individual performance, including the expectation that any issues or obstacles are communicated in a timely manner. It's one thing to say you have an open door, but you need to ensure that you make time to speak with people promptly when they come to you. While you need to hold individuals accountable by letting them know when they are and are *not* meeting expectations, you need to ensure that you're doing this in a timely fashion as well. If someone did a great job, tell them when you learn about it – not a month later! Similarly, if there is an issue with how a programmer is performing their job, discuss with them immediately using specific details rather than waiting for their next performance review.

A key part of one-on-one meetings between leaders and those they lead is active listening. These meetings are not just for leaders to communicate knowledge and feedback, but also for leaders to learn what goals each programmer aspires to achieve. While one programmer may be excited to lead a project team, that same goal may terrify another programmer and suggesting that as a goal may add undue stress for that individual.

By exemplifying empowering behaviors, leaders demonstrate how to work more collaboratively with others. When individual programmers meet with peers or team members from other disciplines regarding project-related topics, they are more likely to demonstrate empowering behaviors themselves if they are shown direct examples of those behaviors. They will be more likely to ask questions and share ideas, discuss challenges and potential solutions, and support one another in a way that enables the entire research process to become more efficient.

SUPPORT AND PLAN FOR GROWTH

Once managers have communicated goals and expectations, and learned what individuals are comfortable with as well as what they're interested in, the next step is to discuss how those interests align with company goals. Leaders can then work to provide opportunities for their programmers to grow and pursue their interests in a way that benefits both them and the company. Helping programmers to grow could require access to specific tools (such as software or online learning) or time with other individuals in the company (e.g., training time with a data manager to better understand a data collection system). The key is to have a plan that is mutually agreed upon between the manager and the individual, and regularly discussing progress toward executing the plan. It's important to acknowledge and reward progress toward growth – even a simple "great job" can go a long way toward encouraging programmers to continue improving their skills.

SUPPORT INDEPENDENCE, ENCOURAGE SAFE FAILURE

Personal and professional growth can occur in many ways, but in the end all programmers need to function independently while writing programs. Managers cannot stand behind a programmer's chair and tell them where to add logic checks in each program they write, nor can they tell each programmer when to research a new programming technique that could be more efficient than the technique they already know. That said, there are varying degrees of independence that each person is comfortable with, and ready for based on where they are in their respective career path. While a senior programmer with many years of experience in the pharmaceutical industry may welcome the opportunity to lead a programming team on a complex pivotal study, that same task is more likely to overstress a programmer with significantly less experience. The crucial task as a manager is to understand the needs of each individual and to delegate tasks and responsibilities appropriately, then step back.

It is important to keep in mind that independence comes with guidelines. Programmers need to understand company, project and individual goals and expectations, and practice independence within those parameters. While allowing individual independence, it's important for leaders to check in with programmers regularly and set milestone checkpoints. For example, if a programmer is preparing a set of outputs for delivery to a client for the first time, they should check in with their manager or team lead so more experienced individuals can review the package before it gets delivered. Similarly, if a programmer makes a decision that leads to an issue, leaders need to turn that issue into a learning opportunity rather than berate the individual. Those learning opportunities are more than just "this didn't work, you need to do better next time." Feedback needs to be constructive and specific enough that lessons can truly be learned. If programmers are "beaten up" for mistakes, they're very unlikely to offer ideas or think independently beyond explicit instruction.

RISKS AND BENEFITS TO EMPOWERMENT

Ideally, employee empowerment is part of the company culture, endorsed and supported from the top down. Even if your manager claims to endorse programmer empowerment, they may not support the methods required to foster positive empowered behaviors. It takes time to educate programmers about industry and company goals and values. It takes time to train people and earn their trust. Empowering programmers does introduce some risk, but the rewards can far outweigh them if you take steps to mitigate those risks. Table 2 below shows some examples of the potential risks and benefits of empowering programmers.

Method	Risks	Benefits
Open communication	Leadership time spent in meetings and group management rather than "billable" work	Entire team works collaboratively, everyone contributes to a higher quality end product
Support growth	Training time takes away from tight resources needed to produce outputs within the timeline	Technical and personal growth leads to a broader talent pool, which results in higher productivity and more capacity from the same resources
Support Independence	Programmers make poor decisions that cost time and money	Programmers develop more efficient processes that get more done, faster
	Programmers feel stressed about new expectations and leave	Programmers feel valued and stay longer

Table 2. Risks and Benefits of Empowerment

While the risks listed above are legitimate concerns, the key is to be aware of them so leaders can take reasonable steps to reduce them. Let's examine these risks and ways to mitigate them.

OPEN COMMUNICATION

Communication takes time, from department meetings to project meetings to one-on-one discussions. Many would argue that technical leaders should spend more time producing outputs that can be billed to projects than organizing meetings, educating, and managing their programmers. However, if a programming team can't make appropriate decisions independently, then more responsibility ends up resting on a handful of leaders. As a result, there may be six programmers working at 80% efficiency while their leader ends up working 110% to make up the difference. In contrast, one highly effective programming leader may enable an entire team of programmers to work more efficiently. Even if this only results in 5% higher capacity from those six people, it is much more substantial than 110% from one person.

Regardless of how the math works, it can still be a challenge for leaders to make time for meetings, education and management. However, at least some portion of a leader's time should be reserved for communicating with others and education. A side benefit of this, is that the leader can become the extra resource in times of crisis. For example, all programming resources are busy with various projects, but then one of the programmers has a health emergency and will be out of the office for several days unexpectedly. Rather than impact a critical project deadline, leaders can step in to fill the gap at short notice by delaying meetings or mentoring sessions.

The other benefit of open communication is engaging everyone to work together to solve problems. From technical problems to timeline challenges to communication gaps, the old saying that "two heads are better than one" still holds true. Allowing everyone to see the challenges being faced, and participating in discussions on ways to solve them, enables everyone to collaborate on solutions that may be innovative and discovered much faster than would otherwise occur.

SUPPORT GROWTH

Like the time concerns regarding adequate communication between leaders and their programmers, time spent training programmers to grow their technical and leadership skills is often overlooked in favor of focusing on producing project output to meet timelines. Unfortunately, when nobody has time to train and coach those with less experience, once the current generation of programmers leaves or retires, there aren't any programmers to replace them. In addition, if senior technical people aren't taught how to properly lead a team or manage a project, if the existing leader leaves then the team is left leaderless. It's critical to ensure both technical and professional development within each programming team, not only to help reduce turnover but also to mitigate the impact of the inevitable turnover that will occur.

There are many ways to allow for training and growth while minimizing impact. One example is online training courses, which often have more flexibility on timing (programmers can attend during non-work hours) and don't require expensive travel. Asking senior programmers if they'd like to mentor junior programmers can be a great way for both to grow; junior programmers learn new technical skills while senior programmers increase their communication and leadership skills. Leaders need to set expectations around these mentorship programs, but overall team growth can rapidly yield impactful results. The team's technical base gets stronger (more programmers can do more complex tasks) and the leadership base grows as well (more senior programmers have enough skill and confidence and to be technical leads on projects).

Giving programmers the opportunity to work on new tasks while being supported by more experienced individuals means more programmers have direct experience with a wider variety of situations. This direct experience, along with coaching and learning from successes and failures, leads to better decision-making on future challenges.

SUPPORT INDEPENDENCE

One of the main concerns with allowing programmers to have more independence is that they'll make poor decisions that are not aligned with company or department values and objectives, leading to loss of time and money. For example, if programmers don't adequately validate their work, it could result in significant delays and costs for rework. There are a number of ways for leaders to reduce the risk that programmers will make poor decisions. Making goals (e.g., no serious errors in tables delivered for statistician review) and expectations (e.g., each programmer will validate their own output prior to sending for independent validation) clear from the beginning, as well as ensuring that programmers understand that they will be held accountable for their actions is the first step in supporting good decision-making.

One common issue with programming is the balance between speed and quality. Particularly in the pharmaceutical industry, time is always a concern – patients may be suffering while waiting for a treatment, so of course researchers want to demonstrate the safety and efficacy of a treatment as fast as possible. However, this needs to be balanced with the absolute necessity for quality – it doesn't matter that the summary tables were delivered quickly if the content is incorrect or misleading. Many decisions are made around this key issue and it's important for leaders to coach programmers how to balance speed versus quality. For example, discussing how to determine the best time to start programming against a draft database to meet final project deadlines.

In addition to clearly stating goals and expectations, and coaching how to make good decisions, leaders should set pre-specified checkpoints with programmers to ensure everyone stays aligned and on target with objectives. This could include things like weekly team meetings to review current work status to ensure tasks are on track to complete on time and reviewing quality control documentation before output is delivered to ensure it is compliant with internal requirements. Programmers working independently does not mean that programmers are just assigned a list of tasks and then left to fend for themselves. The key is for leaders to work with their programmers to determine the best balance of autonomy and guidance that allows programmers to leverage their expertise and expand their skills while working efficiently toward business objectives.

OVERALL EMPOWERMENT

As discussed, the key is to know how much independence and responsibility each programmer is comfortable with and what kind of support they need. There are many aspects to empowerment and providing an environment where programmers are empowered doesn't mean everyone will take advantage of all aspects equally. Empowerment doesn't look the same for everyone and can actually be detrimental when forced on programmers in ways that they're not comfortable with. For example, many programmers have no interest in speaking out in groups or leading teams. Forcing those individuals to take leadership roles will only increase their stress and will likely lead to less productivity. Instead, it might be a better option to give them the opportunity to play a supporting role, where they have increased responsibility but are not required to direct others or speak in a group setting. For example, they may take responsibility for archiving project files and preparing required documentation, which the team leader then reviews and communicates outward.

The goal for leaders is to support their programmers in ways that allow them to effectively contribute as much as they're able to the group in ways that are comfortable for them while being aligned with business needs. In this environment, programmers feel valued and fulfilled and are more likely to stay longer.

CONCLUSION

Empowering your programmers isn't easy – if it was, everyone would be doing it! It takes deliberate planning, effort, and reinforcement to create an environment where individuals communicate freely, make independent decisions aligned with company goals, and grow professionally. It can be a challenge to share goals and expectations in a clear framework, and to consistently reinforce that framework, so individuals can have the power to make decisions and to have a voice in a way that does not become a chaotic free-for-all.

Even if you provide a culture and opportunities that empower your programmers, not everyone will take full advantage of them – everyone has different personal and professional goals. Ongoing support and guidance are required to allow everyone to realize their maximum potential while minimizing business risk. Despite the effort and risks involved, the potential benefits are significant. Empowerment fosters collaboration, innovation, and dedication. This combination can result in a highly effective and efficient team of programmers, who regularly outperform expectations, and are happy to stay where they can contribute and feel appreciated.

REFERENCES

[1] Leonard, Kimberlee. "What Are the Benefits of Employee Empowerment?" July 2, 2020. https://smallbusiness.chron.com/benefits-employee-empowerment-1177.html

[2] Bosworth, Patrick. "How to Empower Employees in the Workplace – 8 Tips." Accessed April 2, 2022. https://leadershipchoice.com/empower-employees-in-the-workplace/

Cornerstone OnDemand. "The Empowered Workforce: Crucial to Success in the New Economy." 2010. http://www.dresserassociates.com/pdf/whitepapers/The-Empowered-Workforce-Crucial-to-Success-in-the-New-Economy.pdf

Encyclopedia of Business, 2nd ed. "Empowerment." Accessed April 2, 2022. https://www.referenceforbusiness.com/encyclopedia/Eco-Ent/Empowerment.html

Lee, Allan; Willis, Sara and Tian, Amy Wei. "When Empowering Employees Works, and When It Doesn't." March 2, 2018. https://hbr.org/2018/03/when-empowering-employees-works-and-when-it-doesnt

The Saylor Foundation. "The Benefits of Empowering Employees." Accessed April 2, 2022. https://resources.saylor.org/wwwresources/archived/site/wp-content/uploads/2013/02/BUS208-5.2-The-Benefits-of-Empowering-Employees-FINAL.pdf

Zetlin, Minda. "5 Things Smart Leaders Do to Empower Employees." August 4, 2014. https://www.inc.com/minda-zetlin/5-things-smart-leaders-do-to-empower-employees.html

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Carol Matthews
Advance Research Associates, Inc.
cmatthews@advanceresearch.com
www.advanceresearch.com