HOW NOT TO OVERWHELM YOUR DEVELOPMENT TEAMS AND DISAPPOINT THE BUSINESS

A WHITEPAPER BY GARY GRUVER

Why Do Most Software Organizations Consistently Overwhelm Their Development Teams and Disappoint the Business?

Most software organizations have some common problems: the business is frustrated that development can't deliver what they need and expect; the development teams are overwhelmed with too much work leading to excess work-in-process (WIP) and burnout; everyone in the organization is frustrated and discouraged, while the problem keeps getting worse, and technical debt continues to pile up.

Why is this problem so prevalent in software organizations? Is it because it is hard to estimate? Because every time you create something new with software you are doing something for the business you have never done before? Because development teams are trying so hard to meet the needs of the business that they overcommit? Because the planning team wants to meet the business objectives so much so that they push the development teams to overcommit as well? Because the business assumes when they put out a request for a date and they don't get any pushback they falsely presume it's a commitment? Or, is it all of these reasons and more? The exact reasons will vary by organization, but this problem is significantly more prevalent in software than in other development teams.

The question, then, is how do we address these issues? We want the business to know what to expect. We want developers to feel good about their contributions. We want the organization to have some capacity to sharpen the saw by reducing technical debt.

While I don't think I have all the right answers, I recently got some helpful insights from a great project management office (PMO) leader. He leads the PMO for an organization that employs hundreds of engineers. He has grown his PMO reach from a smaller, more localized part of their business, into planning for several different parts of the business that are spread all around the world.

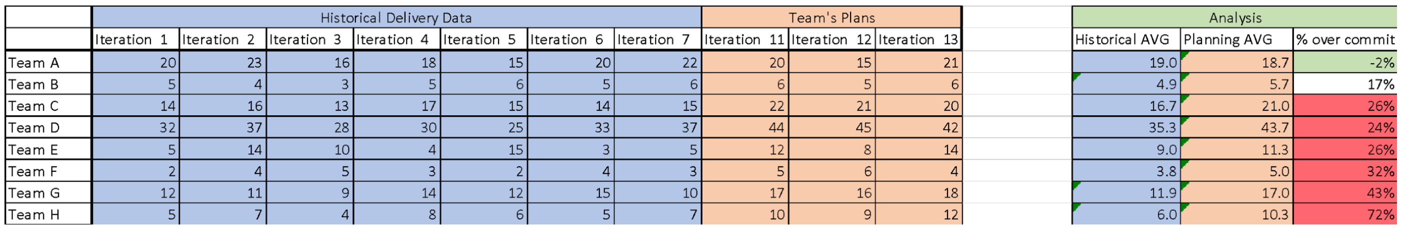
One of the most interesting things this leader said to me was that, "the most important parts of my job are understanding the capacity of the organization to deliver, ensuring there is room for sharpening the saw, and setting the expectations of the business." He felt accountable, not just for creating the plan, but for ensuring that the organization delivered what the business expected. It is a straightforward, common-sense approach for the job that is very different from what I've seen in many organizations.

In many organizations, I see a PMO working hard to create plans that can meet the needs of the business, getting the business to prioritize what they need, when they need it. Then, they work with the development teams to create a plan of record that will meet as many of the business's needs as possible. But many times, this results in the development teams overcommitting, because everyone wants to meet the needs of the business. It also starts the first phase of disappointment for the business, because they are being told they can't have everything they want when they want it. During development, when the teams realize they are overcommitted and are not meeting expectations, the business goes through their second phase of disappointment, and is frustrated yet again they can never get what they need and expect. The PMO blames the development teams because they are responsible for delivery. The development teams get discouraged and trades off sharpening the saw work just to deliver as much of the business commitments as possible. This only builds up more technical debt in the process, and the whole approach spirals down into frustrations and disappointments.

A Different Approach

The PMO leader I mentioned previously takes a very different approach. He makes it clear to the business that they shouldn't have any expectations, not until they can work through priorities and ensure the organization has the capacity to deliver. With this, he spends time collecting metrics to get a detailed understanding of the capacity of an organization. He works with the teams to ensure their plans include sharpening the saw work, because he understands if they don't, their capacity to deliver what the business needs will go down over time, and technical debt continues to pile up. He also helps teams make realistic commitments, so they can meet their expectations.

The key to making this all work is the metrics. The PMO leader I spoke to uses story points provided by the development teams. An organization can also use T-shirt sizing to estimate demand. Although, the further you get away from high-level estimates to plans developed by teams that will be doing the work, the more accurate the data will be. The data is collected and summarized like the example below:



	Historical Delivery Data							Team's Plans				Analysis		
	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6	Iteration 7	Iteration 11	Iteration 12	Iteration 13		Historical AVG	Planning AVG	% over commit
Team A	20	23	16	18	15	20	22	20	15	21		19.0	18.7	-2%
Team B	5	4	3	5	6	5	6	6	5	6		4.9	5.7	17%
Team C	14	16	13	17	15	14	15	22	21	20		16.7	21.0	26%
Team D	32	37	28	30	25	33	37	44	45	42		35.3	43.7	24%
Team E	5	14	10	4	15	3	5	12	8	14		9.0	11.3	26%
Team F	2	4	5	3	2	4	3	5	6	4		3.8	5.0	32%
Team G	12	11	9	14	12	15	10	17	16	18		11.9	17.0	43%
Team H	5	7	4	8	6	5	7	10	9	12		6.0	10.3	72%

The PMO leader I spoke to has been rolling this approach out across the broader organization for years, and there have been some very interesting observations. **First and foremost, is that teams tend to overcommit and underdeliver**. Even when faced with historical data, they would still argue that, "I know I have only averaged 5 in the past, but this iteration we are going to do a 12!" This is a hard behavior to change and it takes time, persistence, and ongoing feedback. The leader would then go back to the team that committed to a 12, but only delivered a 4, and provide feedback for working on the next planning cycle. The team would sheepishly admit they didn't make it but felt, sure, they could do 10 the next iteration. This takes time, but as you can see in the graph above with teams, A and B, they eventually get better at matching capacity with plans. These teams are also who the leader worked with the longest.

After he had a better understanding of the real capacity of the organization, the next thing this leader did is he started setting more realistic expectations with the business. "I know the team is telling you what they are going to do, but this is what you should expect based on historical performance." He was very transparent with the business about the capacity of the organization and what they should expect. He also set boundaries for the teams: "I know you want to stretch and do more, but I am not going to let you put in plans that are more than 20% over your historical track record." They also needed to adjust their plans to include the work they hadn't finished in the last iteration. This work had to include the work that was for continuous improvement in order to ensure they were improving capacity over time. He tracked sharpening the saw commitments to completion in the same way as the business commitments, because they are usually the first things traded off in a crunch.

This leader also spends a lot of time working with both the teams and the business to set realistic expectations. For instance, he'll look for opportunities to get better at capacity estimations. When he sees Team E has historical data ranging from 15 to 3, he might suggest that breaking the stories down into smaller, more consistent sizes might help reduce some of the variability. Essentially, his method is all about providing visibility and analysis, then gently prodding the organization to get better over time.

There are also certain things the leader doesn't do that are equally important. He does not use the capacity numbers to measure productivity, because he knows as soon as he does, they will start gaming the metric to show productivity gains, and it will lose its value as a measure of capacity. Instead, if he wants to understand whether or not the teams are improving, he looks at the stories they are using to sharpen the saw, then he works to show the business the waste that they are removing will lead to greater productivity. He also doesn't use the capacity numbers to compare teams to each other, since teams usually have their own way of planning and estimating that works for them. The capacity numbers are not a measure of productivity that we can use to drive competition between teams, because they may start gaming the metric. And even if we could get planning metrics that are consistent across teams, the effort probably wouldn't be worth the benefit.

The PMO leader doesn't believe this approach is perfect so he is always modifying and improving it over time. It doesn't answer all of the reasons why software organizations continually overwhelm their development teams and disappoint their businesses. It does, however, provide some insights on some of the reasons for why organizations overcommit and how organizations can improve.

Typical Approaches

This PMO Leader's approach is very different from what I see in most organizations, in that he takes a very active role in setting realistic expectations with the business and ensuring the teams don't overcommit. This takes a lot of effort, and still, the teams have a natural tendency to overcommit. Now, imagine an organization where nobody is playing that role. Or, even an organization where the PMO sees their role as pushing the development teams to commit to as many business requests as possible. Or, where the business feels their role is to demand what they want, with no visibility into capacity constraints, and what the teams need to be doing to improve productivity.

This approach sets up the development teams to be consistently overwhelmed. They have so much WIP, they just rush from one emergency to another, and never feel as if they accomplish anything. Can you see how they would get frustrated and disengage? Can you also see why the business would be disappointed and feel like they don't have a very good technology partner? Can you see how this leads to spiraling frustrations that build up technical debt? I can see this, and I do see it in so many organizations. Hopefully, this leader's example can help people address these issues.

The Harder I Push the Less I Get??

But this approach is so different and counter-intuitive for most organizations that it is worth exploring in more detail. Most leaders feel like the way they deliver more is to get the organization to make stretch commitments and hold them accountable for delivery. This way, everyone is working hard and delivering the most value. **But this overlooks the fact that the most inefficient way to run anything is by being booked to full capacity**. Manufacturing learned a long time ago that when you try to keep a factory running at full capacity, you just end up building a lot of WIP. This excess WIP results in wasting lots of energy expediting, task switching, and managing the inventory to get anything done. This is well-supported[1] by queuing research, and the same principles also apply to software. If we try to make plans that ensure everyone is busy all of the time, as soon as there is any unknown that causes a change to the plan, we have to switch tasks and expedite work in order to complete anything.

How then, as a leader, do I make sure my organization is productive and getting as much done as possible? Just like with manufacturing, Goldratt teaches us it starts with understanding and optimizing the bottleneck. We need to understand where the bottleneck is, and make sure it is never starved for work, because this will define the overall rate of the system. Then, we need to make sure we are doing everything we can to optimize the throughput of the bottleneck.

The bottleneck could be in the requirements, development, or release processes. Usually, it is in the development or release process. If it is in the release process, there

[1] "The principles of Product Development Flow Second Generation Lean Product Development" By Donald G. Reinertsen, Celeritas Publishing, Redondo Beach, California, 2009, Chapter 3

is lots of automation that can and should be leveraged from the DevOps community. If this is your bottleneck, you should work to address these issues and move the bottleneck to the developers. You might think you never want your developers to be a bottleneck for the business. That might be a good idea, but not very practical since it is so much easier to come up with ideas you can do with software than it is to deliver them. Therefore, we want the bottleneck to be with the developers, since they are the ones developing new value for the business. Then, we want to do everything we can to optimize the throughput of the developers.

As a leader, you might think, yes, this is what I am doing by getting the organization to make stretch commitments and challenging them to deliver, — working to optimize the throughput of the developers, which are the bottleneck. Although this overlooks all of the well-documented, queuing research that shows that booking to full-capacity is the most inefficient way to run any process. Because as soon as something doesn't happen according to plan, you have to start expediting and moving everything around to get the most important stuff done. Any coordination across teams must get renegotiated, or developers get blocked. Everyone is so busy rushing to tend to the latest emergency that nobody is very productive, and the improvement efforts may get ignored.

But Then How To Optimize the Value the Organization is Delivering?

We need to start with realistic plans that are based on historical performances, like the PMO leader in this example. We need a plan that organization can deliver, even when everyday uncertainties arise. The expectations for the business should be based on this realistic plan, and until their request is part of this plan, they shouldn't have any expectations. It is not that we don't always want the developers working at full capacity, it is just that we don't want them planned at an overcapacity that requires replanning and adjustments when any new discoveries show up. Next, as Goldratt teaches us, we want to make sure there is a buffer of work waiting for developers to start working on, especially if uncertainty results in them finishing early. This should be a prioritized list that includes business requirements and technical debt that is defined and ready, so they are never left waiting.

Next, we want to do everything we can to optimize the bottleneck. When I led the large-scale transformation at HP, we did everything we could to help the developers be more productive. We bought the developers large and multiple monitors because we knew they would be more effective if they could see more of the code all at once. We also bought everyone noise-cancellation headsets, because we knew that while software development requires collaboration, it also requires periods of intense concentration, which is hard to do in today's noisy work environments. We looked at the repetitive tasks the developers were doing and tried to automate as much as possible. We did everything we could think of to help the developers become more productive, because we understood that was *our* bottleneck to delivering value to the business.

If you want to understand if the organization is optimizing the delivery of value, look at sharpening the saw work, and quantify the waste that is being removed so the business sees and appreciates everything being accomplished to be more efficient. This makes the business feel better about their partners. Then, you'll want to work to make the processes across teams more visible so it's easier to identify if there is broader waste in the organization that is slowing things down, that was previously beyond the purview of the individual teams. This type of focus — trying to understand and optimize the bottleneck — is what truly optimizes the flow of value.

Summary

There are many possibilities to why software organizations consistently overwhelm their development teams and disappoint their businesses. Based on this example, it would appear that everyone is so motivated to meet the needs of the business that there is a tendency to be overly optimistic. If organizations are going to create realistic plans and credible expectations with the business, they are going to have to bound their commitments with metrics that represent historical performance. This will help them avoid the waste associated with having to rework plans and expedite deliveries. Once they have these realistic plans, they can ensure they are optimizing delivery by having a buffer in front of the bottleneck, and then work together to optimize the flow through the bottleneck.

Gary Gruver is the CEO of Gruver Consulting, an acclaimed author, and in-demand speaker. Gary brings a proven track record of transforming software development and delivery processes in large organizations.

For more information, visit GaryGruver.com