Dominica Degrandis

Making Work Visible: *How to Unmask Capacity Killing WIP*

DevOps Enterprise Summit London 2017

Making Work Visible

How to Unmask Capacity Killing WIP

DevOps Enterprise Summit London 2017

Dominica DeGrandis

IT REVOLUTION

25 NW 23rd Pl, Suite 6314
Portland, OR 97210

The contents of this eBook are a transcript of the complete presentation given by Dominica DeGrandis, "Making Work Visible: How to Unmask Capacity Killing WIP" at the DevOps Enterprise Summit London 2017. To view the original presentation, please visit https://www.youtube.com/watch?v=KR7Y8IUgyyA.

eBook published 2018 by IT Revolution Press.
For further information on this or any other books and materials produced by IT Revolution Press
please visit our website at itrevolution.com

Making Work Visible: How to Unmask Capacity Killing WIP

Thank you, everybody, for being here. It's really great to see you all here today. As Gene Kim said, I'm Dominica DeGrandis. My background [is as] a build engineer. That's what I did. So, I spent a lot of years doing configuration management and trying to make things visible for people to see what build was on what server, and what environment it was in, and where it was, and when it was going to production, and all of that—just making things visible. For the last ten years that's what I've been doing—helping people make things visible. I help teams see [their work] and [help them] measure the things that prevent them from getting work delivered. And then I help them design kanban systems so that they can optimize their workflow.

Today, I want to talk about something very near and dear to my heart, and that is that people take on more work than they have the capacity to do. It's a big problem because they overload themselves and they overload their teams, and that creates bottlenecks and constraints, and it delays the delivery of business value. So, that's what this talk is going to be about. I want to talk about how that happens and what to do about it.

Three things, really.

1. Why do we take on this overload?
2. How do we unmask that overload?
3. And then I'm going to show you some, hopefully, tactical examples, some kanban designs that you can take back home, as that can bring visibility to what's killing your team's capacity.

So first off, why are we so overloaded? Why do we think we can finish work faster than we actually can? I think we tend to underestimate the number of distractions that side track us from focusing on our work and finishing it on time, and we tend to say yes to requests when we should probably decline them or at least postpone them. And before you know it, it's 2:30 on a Sunday afternoon, and we're working on that thing that's due Monday morning at 9:00 a.m.

So, why do we overload ourselves to the point of making Sunday the new Monday? Because when somebody asks us to do something, we say yes. We say yes. We take on more work than we can complete in a timeframe. And just to clarify, there are sometimes burnout cultures that play a role here, and I want to acknowledge that. But often we do this to ourselves. Often, it's us that say yes all the time.

So, I want to argue that there's four reasons that we say yes. I think there's a few more based on my research, but I'm just going to talk about four today.

The first one is, nobody wants to be the person responsible for the team losing. Right? We are team players and we love to pull one for the team.

Number two is shiny-new-object syndrome. Starting something new and shiny is much more enjoyable than doing the grunt work it takes to finish something older and unglamorous. It's like, "Gee, do I want to implement new microservices using Docker containers or do I want to prepare for this security audit?" Right? Do we want to go have fine dining for dinner tonight with some prime rib and farm-fresh vegetables or do we want to stay home and eat three-day-old tuna casserole?

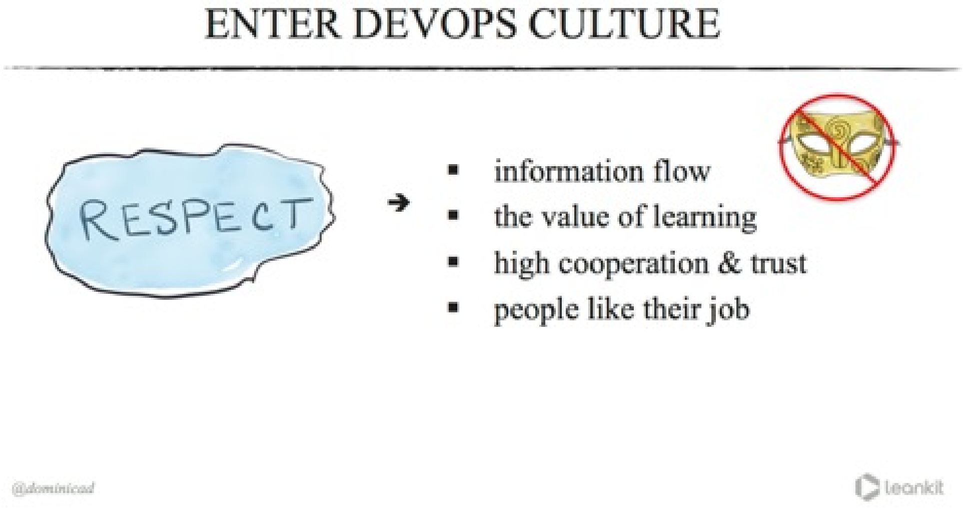
Number three. Everything always, always takes longer than we think it will, especially other people's work.

Number four, it's much easier to say yes to the boss or to people that we like than it is to say no. As an executive consultant, trainer, coach, I work with teams all the time for many years now, and people, when I ask them this question, "Why do you take on more work than you have the capacity to do?" they say, "Because the boss asked me, and it's really hard to say no to the boss." So, you bosses out there, keep that in mind.

I think this is a call for a change in behavior, right? We need to train ourselves not to say yes all the time. We need to give ourselves some no-cred.

I think it's helpful to look at some of the practices that we find in DevOps culture, so here's just a few attributes here.

You see these all the time. I'm not going to go into detail. We hear high cooperation and trust a lot, and that people should like their job, and that information is flowing across silos and teams, and people are aware and can anticipate what's headed their way, and that the climate for learning is considered an investment and not just an overhead or a cost. But all of these attributes at some level require respect for people, which is a main pillar of Lean, and many of the DevOps attributes stem from the Lean movement. We’ve got continuous improvement. We’ve got systems thinking, and value streams, and creating value fast for the customer, and flow. So, it’s these Lean attributes that embody the Lean house.


ENTER DEVOPS CULTURE

→

- information flow
- the value of learning
- high cooperation & trust
- people like their job

@dominicad

leankit

Figure 1: DevOps Culture

So, I just want to call out that respect for people is a major pillar of Lean. People use that term all the time, respect for people. I want to tell you my definition of that, because I don’t hear people defining that. I think respect for people is all about giving honest feedback, and it takes a lot of courage to do that. It takes a lot of courage to tell people why they may not have gotten that promotion or that merit review, or why there was a problem and how to deal with some of the problems. It’s respect for people that provides the leadership basis, the foundation for people to have that safety, to be honest. And that’s what enables continuous improvement, which in turn helps improve customer value. It’s just one big ecosystem.

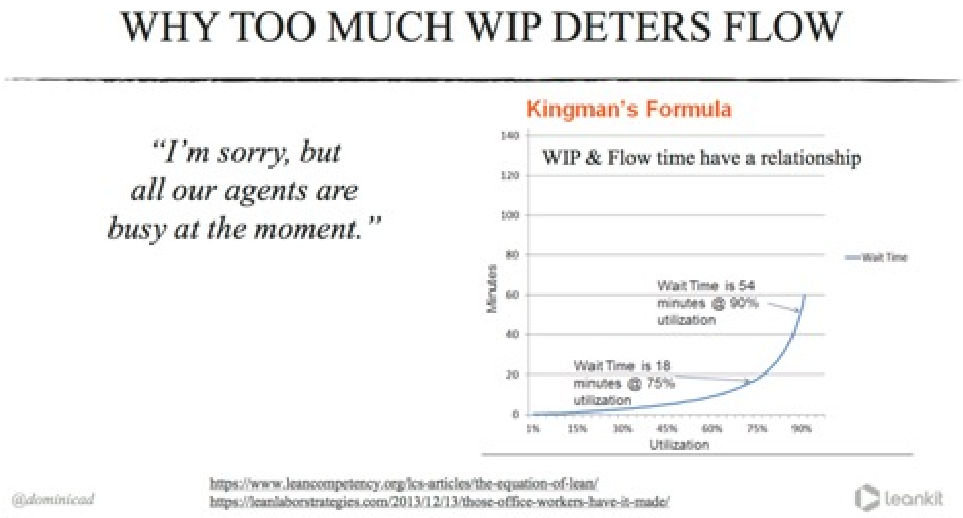
And then flow. flow is a major pillar. All right, so real quickly, what is FLOW? Just smooth and fast movement of all the activities that it takes, that the work goes through from the beginning all the way to the end, including delivery, maintenance, and production. It's not done when it's delivered. There's still some maintenance that needs to be done and some support. And that's the biggest deterrent to flow: too much work-in-progress, too much WIP.

So there's three things that come into play with improving FLOW. The first one is making work visible, and then we've got batch size—reducing the batch size of things—and then trying to build quality in. But in this talk I'm just focusing on that first bit, making work visible.

So why is flow the biggest deterrent? Let's cover that real quickly. Because it scatters your brain across multiple things, and you have interruptions and context switching, and you get out of the zone. When you're in flow, you're in the zone, you're focused. Nothing's interrupting you, and it's amazing the amount of work that you can get done. Too much WIP sabotages our ability to deliver work on time.

I worked with an Ops team once in LA: forty-one engineers—sys admins, network engineers, DBAs. They were building out six data centers all at the same time. They were supporting thirty-three revenue-generating projects on the business side, and they were still carrying the duty pager. They're trying to get all this work done, they're not getting a lot of sleep, and things are slipping through cracks. They're starting to lose respect from other teams. People are, "Oh, Ops. What a bunch of losers. They don't do what they say they're going to do." Their reputation really went down the drain. It was sad to watch. I have a whole talk on how we got out of that rat hole and came back. But the point is that when we take on new work faster than we can finish the older work, we're going to get more work in progress, and queueing theory comes into play.

For those of you interested in math, and if you want to learn more about queueing theory and what that is, here's Kingman's formula for you that basically shows the relationship that WIP and flow have. And the more WIP you have, the longer things take. The wait time increases dramatically as utilization approaches 100%. We don't let our servers get to 100% capacity to utilization. Let's stop doing that to ourselves too.



WHY TOO MUCH WIP DETERS FLOW

"I'm sorry, but all our agents are busy at the moment."

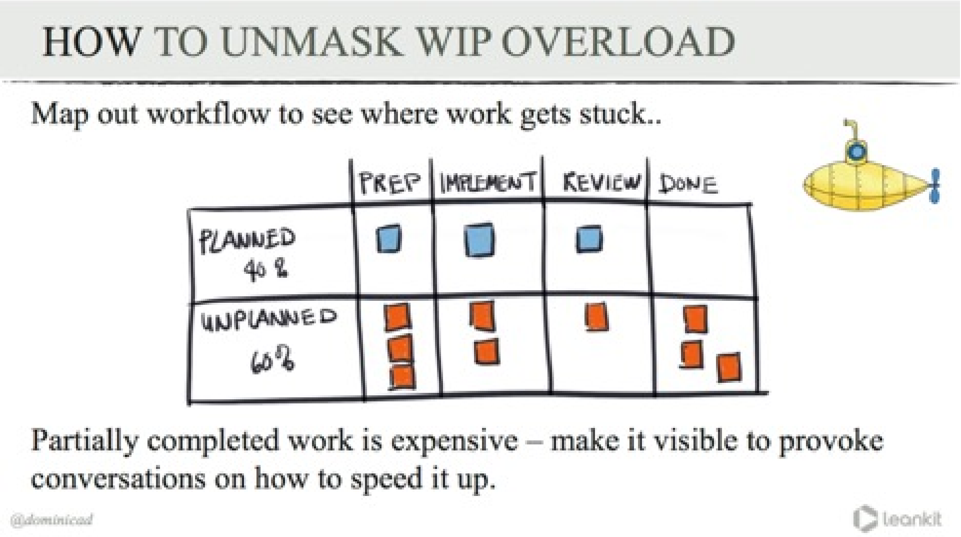
https://www.leancompetency.org/lcs-articles/the-equation-of-lean/
https://leanlaborstrategies.com/2013/12/13/those-office-workers-have-it-made/

Figure 2: Kingman's Formula

All right. So WIP, it's a primary factor inf low, the amount of work in progress we have. It's really what makes WIP a leading indicator. A lot of the metrics we have are trailing indicators. You look at cycle time, lead time, throughput, velocity. Those are all trailing indicators. You don't know how much you had or how long it took until after it was finished. If you're looking at work in progress as a metric, you know from the onset if something's going to take longer. If you're in an airplane and you're getting ready to take off, and there are ten airplanes queued up in front of you, you know it's going to take longer before your airplane takes off. Same thing with boats in a canal, going with the boat theme here.

Okay. So now that we know why people take on more work than they have capacity to do, and we know that [too much] WIP is the biggest deterrent to flow, let's talk about what to do about that.

Making work visible, just the art of putting your work up in a visual space immediately gives people something to view. The eye can take it in and they can see what's happening here. This is just a very elementary example to introduce this. There are many, many different ways to design kanban boards. I'm going to show you just a few here today. They won't all be the same. How you design your system depends on your context and what problems you're trying to shine a light on, and what you're trying to accomplish.



HOW TO UNMASK WIP OVERLOAD

Map out workflow to see where work gets stuck..

	PREP	IMPLEMENT	REVIEW	DONE
PLANNED 40 %	🟦	🟦	🟦	
UNPLANNED 60 %	🟧 🟧 🟧	🟧 🟧	🟧	🟧 🟧 🟧

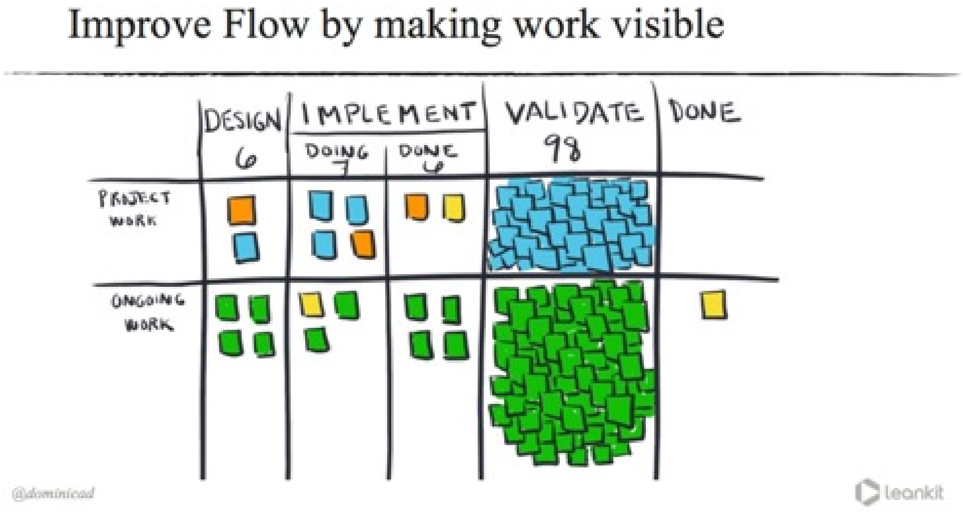
Partially completed work is expensive – make it visible to provoke conversations on how to speed it up.

@dominicad

leankit

Figure 3: Kanban Example 1

So this design [see Figure 3], we've got the states that the work moves through. Prep work, implement work, review, done. That's a fairly simple, generic kind of workflow. Often, we just see planned work on the board, but more often than not a lot of teams have unplanned work. So this is a view that's bringing visibility to unplanned work to show that partially completed work can be very expensive. So make it visible and hopefully it'll provoke some conversations about how to speed it up.



Improve Flow by making work visible

Figure 4: Kanban Example 2

Here's a flow that we're trying to bring visibility to where things get stuck [Figure 4]. Where are things stuck here? Validate, right? When you make work visible, you can start to see the problems. This one's a little bit obvious, but this is a real example here. This is the team of forty-one engineers who had a hundred things stuck in their validate queue. When you see things pile up like this, it should start a conversation about what to do about it. How can we avoid work getting stuck in validate? (By the way, if you don't have this capability yet, then just start out by querying your ticketing system or your workflow system, whatever you use, and just query everything that hasn't moved or been touched in over thirty days.

It’s probably a pretty interesting list. Some of you may need to go sixty or ninety days.)

Then you put your work up on the board, and then you can stand back and ask this question: “Why is this expensive piece of business value stuck?” And then you can consider what’s the cost of the delay of that value being stuck. That should help get some things moving.

I’m going to argue that your expensive piece of business value is stuck, or isn’t moving, because it’s competing with one or more of the following obstacles:

1. Unplanned work
2. Conflicting priorities
3. Dependencies

I see these as some of the biggest impediments to getting work flowing again, so let’s look at them.

Unplanned work. I’m defining that as the interruptions of the day. Many of them are emergencies or incidents. Production’s down. What happened to the NFS mounts? Where did they go? People interrupting you in your queue. These are things that never go in the backlog. These are born in doing. They just appear in doing. And it’s why you can’t get to your to-do list until 6:00 p.m. sometimes. The interruptions and the context switching cut into the business value, or the customer value, or whatever kind of value that it is you’re trying to deliver that’s going to generate revenue or protect revenue in some way.

I also want to be clear that I’m not saying that we can bring unplanned work down to zero. That would be delusional. We’re always going to have unplanned work, so let's just plan for unplanned work because of all the uncertainty. Plan for it so you can put yourself in a position so that it doesn't kill you. One way you can do that…I'm going to talk about maintenance. Keeping a regular cadence of maintenance is super helpful for reducing unplanned work. Pull a maintenance card or two. If you have a work card type of maintenance, pull some of those on your board, so you always have one or two that you can be working on.

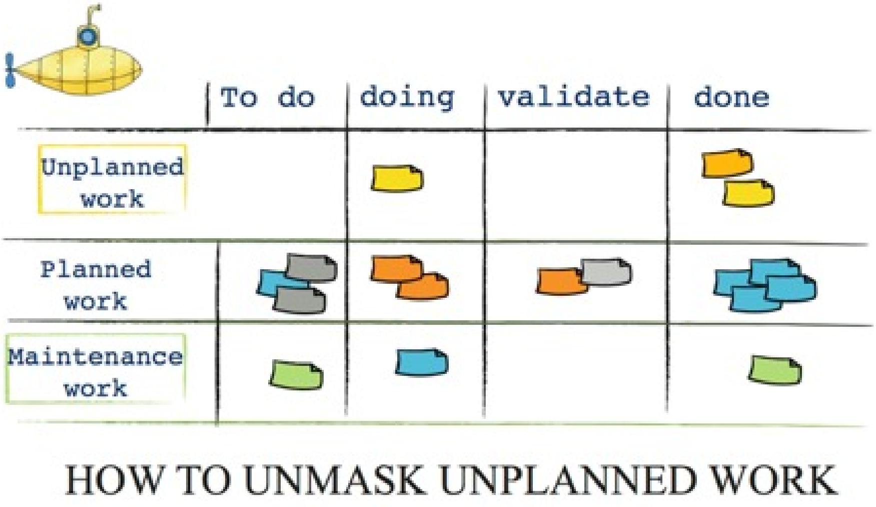


Figure 5: Kanban Example 3

Here's a board: How to Unmask Unplanned Work. We've got a swim lane for maintenance work so we can always have some maintenance work in play. It's also got an unplanned work row up on top there that's bringing visibility to unplanned work. It never goes to to-do, it just always shows up in doing, and these are flowing through the system and interrupting the planned work.

Usually there's some resistance to this, and it's usually in the form of somebody like platform Ops manager Eric, who will say, "I don't have time to create a card or a ticket every time I get interrupted. Not doing that." And then after weeks of interruptions, the CIO wants to know why Azure isn't up and running yet in production. And what does Eric say? Eric says, "I've been busy." But there's no evidence. It's like a perfect crime. I'm not suggesting that you do this forever, but if in your context unplanned work is a problem and it's invisible then maybe bring some visibility to it. Experiment with it for a couple of weeks and see how much unplanned work you have. If you can bring visibility to unplanned work, then you can show it to people and explain, "This is why I wasn't getting that other stuff that you wanted done on time, because all these emergencies popped up."

Next up, conflicting priorities. Competing requests is a major culprit of too much work in progress or too much overload. When you hear, "My project is more important than that project," you know there's a problem there or that priorities are unclear. Thirty-three projects all happening at the same time. That can be a problem because it competes for the same people's time and the same resources. You're competing with all those other projects and resources. It blocks flow. It slows flow down, and it increases partially completed work.

Everything *cannot* be a top priority. We can have a lot of priorities, but there can only be one top priority. That's where prioritization policies come into play, so I need to be really clear. Let me just show you one example of that.

#2 CONFLICTING PRIORITIES

Figure 6: Conflicting Priorities

This is an accounting team. The finance teams—I hope that some of you can go befriend the finance team, because they are probably one of the last remaining hold-outs of being Lean and trying to get on board with flow and DevOps—but they have regular cadence work. They have end-of-the-month invoicing that needs to occur so sales teams can meet their monthly quotas. They have to get expense reports out so people can get reimbursed. And that competes with other work that they have to do, like customer upgrade requests. So they're juggling a bunch of work too, just like we are in Ops. For cadence-driven work, whether it's end-of-the-month invoicing or quarterly business reviews or annual security issues, recognize that those have to be prioritized over other work, and so some of that other work is going to be delayed. And to have that conversation, you can have a board like this that's showing all the different kinds of work that you have, right?

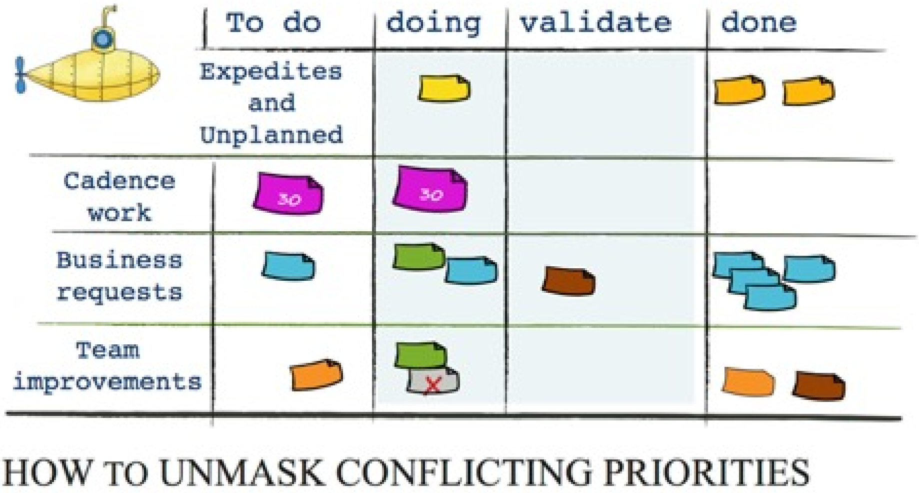


Figure 7: Kanban Example 4

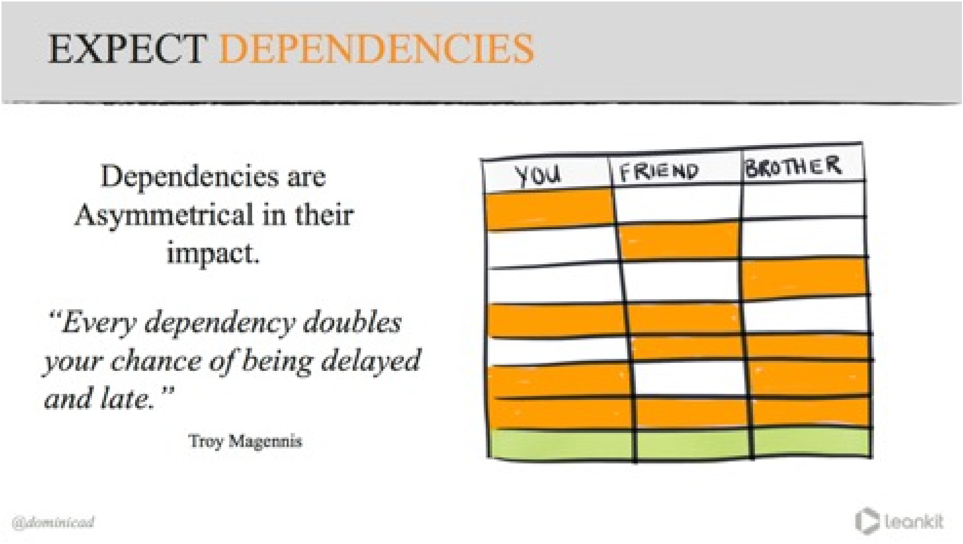
And the cadence work there, that's that second row. We've got these things, and they have a hard due date. This work has to be done on the 30th or payroll won't happen. So, make them big, make them bold, put some dates on them, make them *very* visible, and show them there along with all the other work that's going on. We still have expedites or unplanned work that's happening. There's still business requests, feature requests, revenue-generating work that's got to get done that we promised customers we'd do, so we need to get that done.

And what about all the internal team improvements we're trying to make? And the automation we're trying to do? If you look at everything in the doing column there, now instead of people arguing face to face about what to do, you can look at some data and see what's competing with each other, and then have a conversation and say, "Look, that red X on that gray ticket on the bottom there isn't getting done because we have this cadence work we're doing and we have an emergency."

When you can see the work like this, when you unmask it, you can start to have the conversations that need to be had about how are we going to prioritize. The main point here is, be really clear on what your prioritization policy is. Is it really just whoever gets paid the most, or talks the loudest, or cost of delay, or whatever that is—help people understand.

Number three, dependencies, which affect just about everybody. I'm going to give it a marketing example here, because I was working with a marketing team a while ago. The marketing team editorial director waits on an author to write a blog so they can do the copyediting. And then the social media director waits on the editorial director in order to promote the post. And then the data analyst waits on the post to go out so she can do the social media part about it. The reality is that we work in webs of dependencies, if not on people, then on architecture, and it's just hard to do. This is one of the hardest things we do, I think, is dependencies.

Just to demonstrate how dependencies create a WIP overload, let's imagine you go out to dinner to a fine-dining restaurant. You must think I like fine-dining restaurants. Really, I do enjoy just eating at home. But it's the fine-dining where they don't seat you until you all arrive on time. So say three people are going: you, your friend, and your brother. And you all work at different places, so you're all getting there independently. You're not arriving together; you're meeting at the restaurant. There are eight possible outcomes of arriving at a certain time. Right? Either you're on time, and your friend and your brother are late; or you're late, and the friend is on time, and the brother is late; or you and your friend are late, and the brother is on time. You get the idea.



EXPECT DEPENDENCIES

Dependencies are Asymmetrical in their impact.

"Every dependency doubles your chance of being delayed and late."

Troy Magennis

@dominicad

Figure 8: Dependencies Example

There's eight possible outcomes, and only one of those is going to guarantee that you're going to be seated on time for dinner because dependencies are asymmetrical in their impact. It's not a 33% probability that you're going to arrive on time. It's an 87% that you're *not* going to arrive on time, because seven out of the eight times…it's much more likely that one of those seven is going to happen than the eighth, so it's much more likely that you're going to be late. And every dependency doubles the chance of being late.

So what to do about it? This is one example. [See Figure 8] We have many, but this is between two teams because often the dependencies between two teams is what gets really hard, and three team, and four teams, and 100 teams, and portfolio level across boards. Here, just showing a card on Team Dev board and two child cards on Team Ops board. We're trying to bring visibility to these dependencies, and we're using whatever method we can. Orange cards here mean dependencies. We've got headers in the top, so this is task A or project A or feature A, and it's got two kids, and they live on another board. So, we've got one of two and two of two on the other board. And if you open those cards up, there's links that take you to the other cards, and communication and comments, and you can see all the dependencies that are in there. And you can do this at many different levels. But it's very expensive when teams are mutually unaware of critical information, so trying to bring some sense of visibility to the dependencies is helpful, and just to give downstream teams a heads-up.

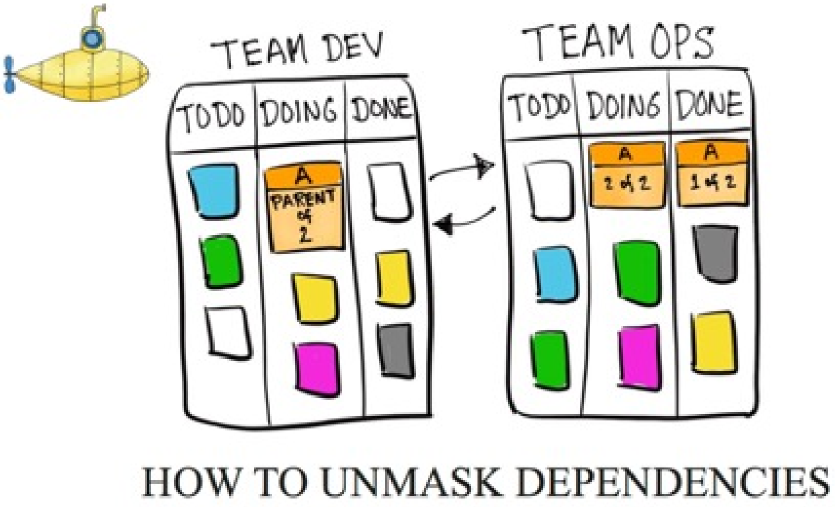


Figure 9: Team Dependencies Example

All right. Next step. In order to get more out of a kanban flow there's two practices that are essential. One is bringing visibility to the work, and two is limiting work in progress. Limiting WIP happens by setting the amount of work that you think the team can handle at one time. The team comes up with that decision, actually. WIP limits intentionally insert tension into the system. They are what put the constraints in the system, because it's what gives people permission to say no. If your WIP limit is ten, and you're at ten, and somebody asks you to do something, this is what gives the honesty, the truth, the trust that says, "We're full right now. Sorry. That's going to have to wait," unless your prioritization policy kicks in, which says if production's down, then drop everything else and go work on this production issue.

But think of a WIP limit as a friendly constraint. It's like if you went to…I got a food thing going on…if you went to a food buffet. You can't eat everything at the food buffet. They give you a plate, a certain size. That's a constraint. You can only fit so much food on the plate. My youngest son is really creative with this, and he finds a way to…he puts the French fries out off the plate edges so all the food piles on. He can get creative. The WIP limits, it's not that you wouldn't ever break them, but if you're not using WIP limits you're at a disadvantage, because it prevents blocking flow. So start limiting your WIP.

This team that had over one hundred in validate, first of all, there's many, many different ways to set WIP limits. This is one. You see this a lot with the WIP limits at the top of the column, and people ask, "Well, how are we supposed to do that? What should our WIP limit be?" I'm just going to tell you my favorite way, which is probably very different from many of the other Lean coaches and what they say for how to limit WIP. My favorite way is to accurately reflect how much work is there and put that number on the board. You got one hundred things in validate? Let's put one hundred there, and then stand back, and look at it, and ask the question, "Does this make sense? Does it make sense to have one hundred things on the board in validate?" No, that doesn't make sense. Okay. So, let's lower it. How much lower should it be? At this point, it doesn't really matter. Eighty is better than ninety-eight. And then you work with that for a little bit and ask the question again. Your volume comes down to eighty. Does that make sense? Probably not. Lower it again. Keep lowering your WIP gradually over time until you can find some smooth flow through your system, and then you know that's about right. But maybe new people join the team, or maybe there's some attrition. WIP limits are meant to be adjusted. They're not permanent.

Okay, there's no rule that says you have to have WIP at the top. You can have WIP limits on rows like this. Here [see Figure 9] we've got a WIP of one Silver Bullet. I was going to explain that, but I'm running low on time, so come afterward to ask the speaker if you want to hear about Silver Bullets. We've got three items in Team Improvements, and we've got a WIP limit of five in Business Requests, and we've gone over that. There's eight things. So now you have this conversation. That's money flowing through there on its way to being delivered, so which one is the most valuable, and what one do we need to drop right now?

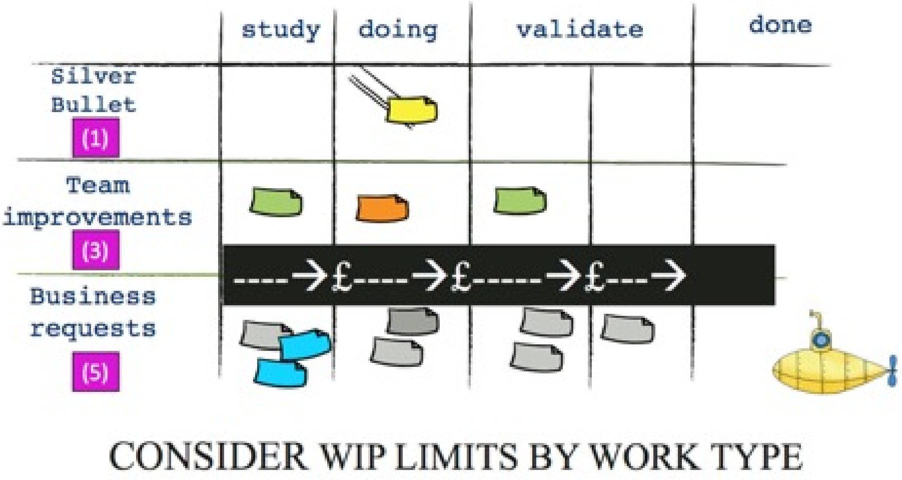


Figure 10: WIP Limit Example

Keep in mind that partially completed work results in unmet business goals, and one way to be more predictable to meet those business goals is to limit your WIP. Maybe 80% of capacity is a nice general rule of thumb. And then to sum up, just in order to reduce the overload, think about using a pull system, a real pull system like kanban that has WIP limits, has a constraint in there, gives the team permission to say, “Sorry, we’re full. Can’t take that on right now. That would overload us,” so that you can do something about it.

So, I hope I’ve provided you with some tactical ways to limit your work and make things visible, basically lose the mask. There’s still a problem here, though. People are taking on more work than they have capacity to do. So, the leaders in the audience, please consider your positional power that you have over your teams when you ask people to do things, because they want to say yes, and more likely, they will say yes, even if it overloads them and overloads their teams and prevents you from your ultimate goal of delivering business value.

About the Author

Dominica DeGrandis is the foremost expert in Kanban Flow within the IT industry today. Her work has shown working IT teams how to effectively improve workflow and optimize throughput to produce the best result throughout the value stream. Her passion involves the use of visual cues and transparency across teams and organizations to reveal mutually critical information. As Director of Digital Transformation at Tasktop, Dominica combines experience, practice, and theory to help teams level up their capability.

She blogs at ddegrandis.com and tweets at @dominicad.

Making Work Visible: Exposing Time Theft to Optimize Work & Flow

By Dominica DeGrandis