Agile support with Kanban – some tips and tricks

By Tomas Björkholm

Foreword

A year ago I held an Open Space at Scrum Gathering in Stockholm about Agile Support. I have since received several requests to expand on the topic, so here it comes.

This is a mix of my experiences and ideas based on inspiring discussions with my colleagues Henrik Kniberg and Mattias Skarin and with David J. Anderson, one of the pioneers of Kanban. Please contact me if you are practicing what I describe here, and let me know the result. You can reach me at tomas.bjorkholm@crisp.se

I want to thank Henrik Kniberg for letting me use his picture for flow. I also want to thank Reza Farhang for giving me feedback on this paper and Yassal Sundman for correcting the English.

Content

Principles 1
Focus by limiting work in progress *2*
Balance demand to capacity *2*
Visualize and be transparent *2*
Let prioritization be blessed by management *2*
Visualize flow with a workflow board 2
Some alternative workflow boards 3
Board for a cross-functional support team *4*
How to handle urgent things together with daily duties and prevention work *5*
The reason to stick to limitations 5
Decide the limit 7
Metrics 7
Lead time *8*
Velocity *8*
Quality *8*
Flow *8*
How to decide what is next 8
FIFO *8*
SLA *8*
Cherry picking *9*
Round Robin *9*
Value *9*
Managed *10*
Levelling out work load *10*
Variation and queue size 10
Continuous improvement 12

Principles

Since support and maintenance looks different for every company, following values is more important than following a recipe. A recipe is what Tom Poppendieck, a pioneer

within Lean software development, describes as a solution to someone else's problem. Here you see the principles in short and later in this paper you can read more about them.

Most of the principles are based on "the Recipe for Success" by David J Anderson

Focus by limiting work in progress

Minimize task switching and limit the possibility to start new ones before previously started tasks are done. This is to avoid queues within the process. Collaborate and work together to finish started tasks.

Balance demand to capacity

Avoid stress because it will not help you improve efficiency. If you are stressed you are more likely to make mistakes, which will make you less productive. Don't commit to more work than you have capacity to fulfil. Promise behaviour and not a result.

Visualize and be transparent

Visualize what you are working with as well as your successes and problems. Let this information be seen by the stakeholders. The main reason for visualising the process is to be able to improve it, let problems and possibilities be seen.

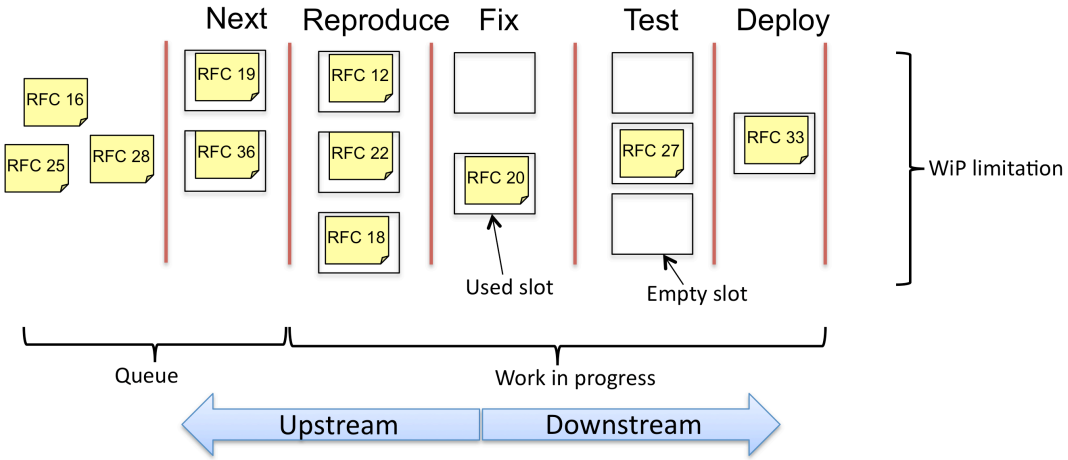
Let prioritization be blessed by management

Too often I hear about support teams who have a problem with customers (usually internal) who are upset because their case is not prioritized high enough. This takes focus from the team to get tickets done. My suggestion is to decide on a way (an algorithm) of how to prioritize and let it be blessed by management. If someone does not agree with your prioritization they should complain to the management team and not to you. This is to leave you to work with fixing support cases instead of arguing about what to fix first.

Visualize flow with a workflow board

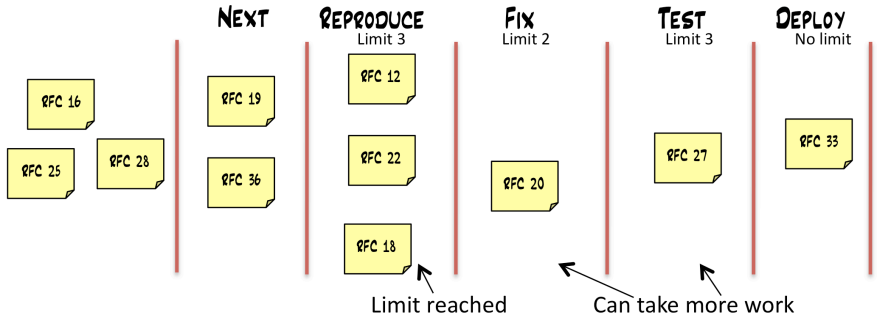
The first thing to do is to visualize the workflow on a board, both the work in progress and the work yet to be started. The board helps you see what you are working with and if there are problems with the flow. Since the board has a fixed amount of slots it also helps you limit the amount of work in progress (WiP). I have also seen improved

productivity just because it is easy to see what to do next.

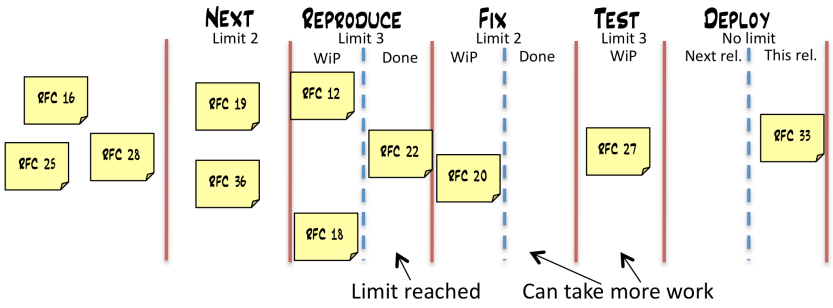


The basic rule of the board is that a team can only pull a support case if they have an empty slot. It might be that you have finished your work but have no empty slots. This usually happens when the team downstream has not been able to finish their work and are then not able to pull more work from you. This means you have a problem in the flow and instead of just starting new things it's better you help the next team to fix the clog. An alternative would be to extend your limit but that means you build a queue in the system, which harms the cycle time. Cycle time is the time it takes from when you start with a support case until it's fixed.

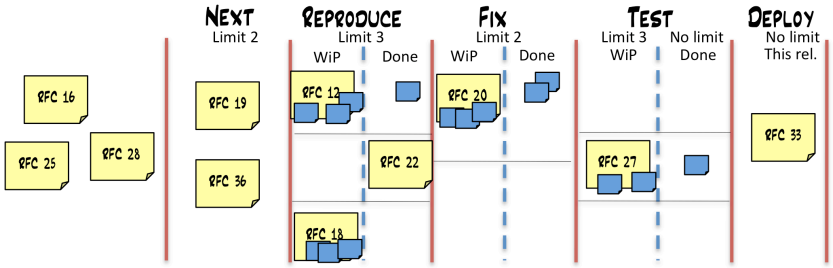
Some alternative workflow boards



Instead of having slots you just write the limit for each column.



To make it easy for the down stream team to see your progress you can split your column to one column for WiP (Work in Progress) and one column for things you have finished. Your limit is still valid for both columns.



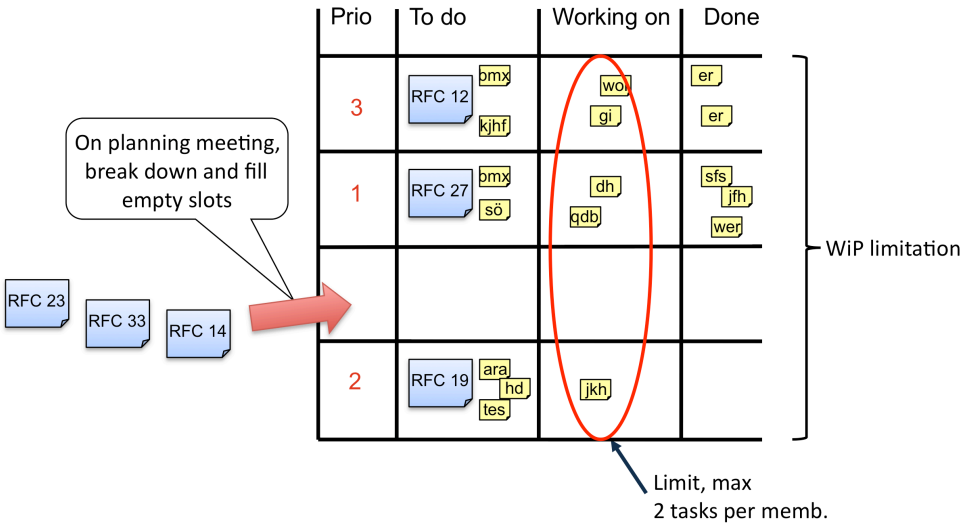
If it makes it easier for you to break down support cases into tasks (technical things needed to fix the support case), you should do that. To make it easy to see which tasks that are belonging to which support case, you can add so called swim lanes (borders).

Board for a cross-functional support team

The boards shown above are divided in different columns for different teams to show their work. If support cases are taken care of by one team, without distinct differentiation between who does what, they are called a cross-functional team. If so, they can use a board with only three columns, one for "Next" or "To Do", one for "Work in Progress" and one for "Done". This is very similar to a typical Scrum board.

The difference to Scrum is that the number of rows on a Scrum board is limited to the number of Backlog items the team commits to for current sprint. Here the limitation is a predefined number of support cases allowed to be processed at the same time. While Scrum works in iteration, here you work in a never-ending flow of support cases.

A board for a cross-functional team, with support cases that can be broken down into technical tasks, can look like this.



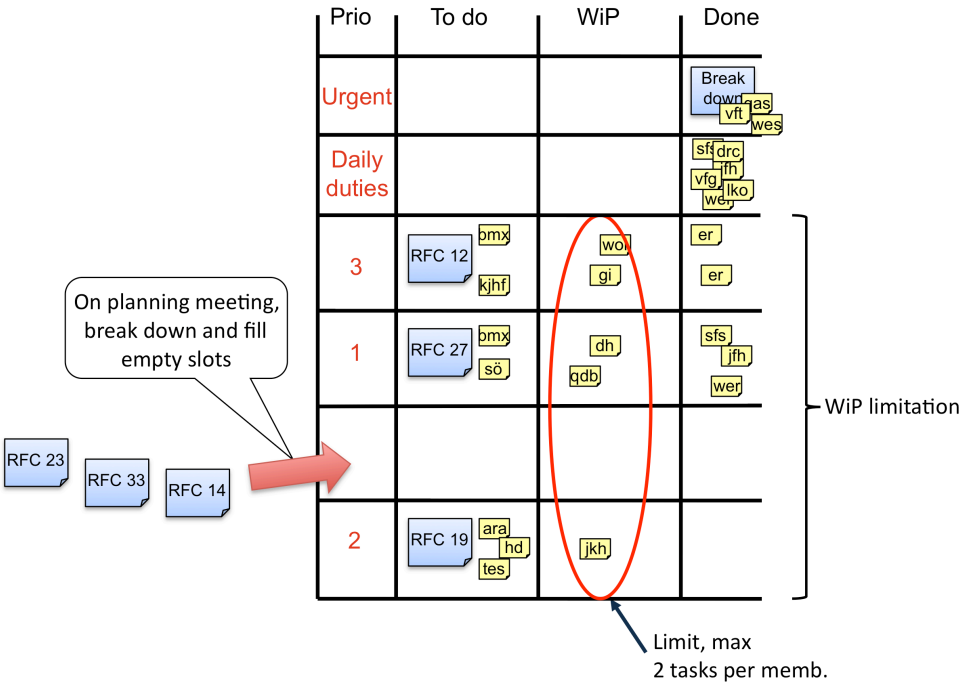
To minimize risk of overburden or lost focus there are two limits. One for the team, which is on case level, and one for every team member, that is on task level. In the example above maximum four cases, which are in process, and each member is limited

to work with at the most two tasks at a time. I have seen teams using two magnets each, with their name on, to mark their current work. This is a great idea to show who is doing what and to ensure that limits are kept.

To minimize unnecessary extra work a new support case replaces the one that was just finished and cases are not moved around to reflect priority. A column is instead used to show prioritization.

How to handle urgent things together with daily duties and prevention work

Some support teams' workday is a mix between daily duties, prevention work and urgent fixes. One way you can manage to mix between those is to have a board like this.



Workflow board for team mixing urgent things with daily duties and prevention work

The day starts by moving the daily duty tasks to the "To do" column. The daily duty tasks are finished one by one until they are all done. After that, the team members can start working with the prevention work, in this example limited to four things. If an urgent issue comes, the other work is interrupted and the team's focus is moved to the Urgent row. When the urgent issue is fixed the work goes back to normal. A good rule for urgent matters that can wait just a little is to start working with them as soon as you are done with the task you started. If the issue is even less urgent, maybe it can wait until there is an empty row, that is, when you are done with the most important prevention item. You and your team can decide to change the prioritization of prevention work items at any time.

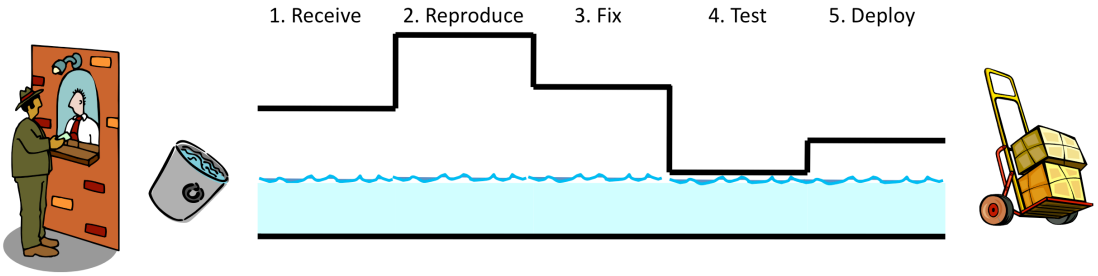
The reason to stick to limitations

When I talk about Kanban and limits I usually get asked why you should not break the limit. Wouldn't it be better to just keep working, as long there is something to do?

To answer the question I usually first show what happens if you exceed your limit and keep working.

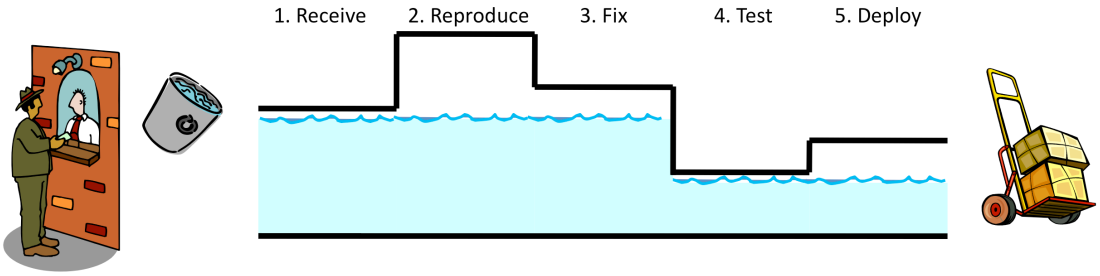
In the pictures you'll see a company with five teams needed to take a support case from a customer until a fix is deployed. The support cases taken in to the system are represented with water and the system is represented as a pipe. The width of the pipes represents the capacity of that team.

In the first picture the number of support cases taken in is not higher than the capacity that any of teams can handle. However team 4 (test) is working at full capacity while the other teams have extra capacity. The employees in these teams are idle part of the time. Since there are no queues in the system, throughput is quick.



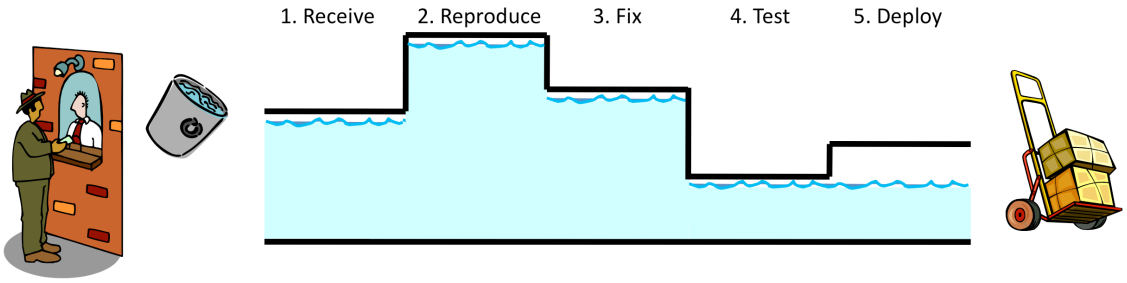
Picture 1: Intake is not higher than the bottleneck.

If intake is increased to the maximum capacity of the first team (receive and prioritize) then the demand on team 4, the bottleneck, is over the capacity and work is queing up. Output is still the same since output is decided by the bottleneck.



Picture 2: Intake is higher than the capacity of the bottleneck

If you continue to take support cases at the capacity of the receive team you will build queues all over the system. The output will still be the same but the throughput will be slower.



Picture 3: Support cases are continuously taken when there is capacity in the first team without considering if the next team is ready to start working with the support case.

Conclusion: If you keep working at your own capacity without considering the capacity of the other teams you will probably build queues for all the teams upstream of the bottleneck. All teams with a queue before them will think they have a capacity problem so you are actually hiding where the real problem is. The main point of this example is that your customer would have been happier if all teams respected their limit. Even if it means that some people are doing nothing for a significant part of their working day. Output would have been the same but throughput would have been higher.

The problem with low throughput is that customers might leave because you are slow to respond. Maybe support cases are out of date when they are deployed. It's better if those support cases had never entered the system because when they do, it means you have invested in them but without possibility for return.

If you respect your limits you easily see where there is a bottleneck in the system. When the problem is seen it's easier to solve it and also verify that the fix helped.

Decide the limit

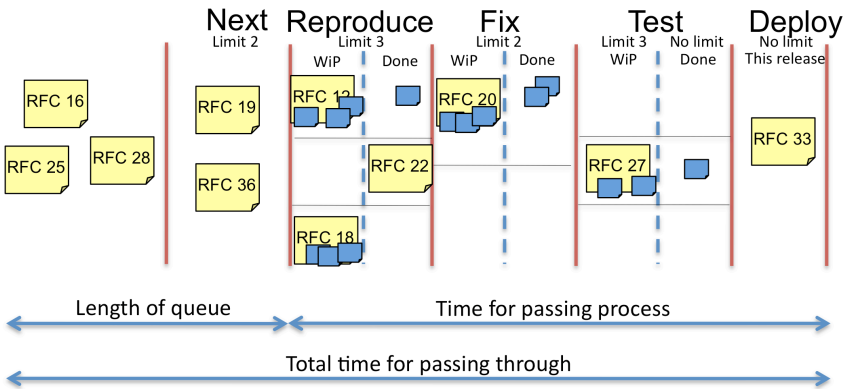
The size of the limit for each station is best found by trying and then measuring the result. If you want to improve cycle time you can try to lower the limit. It's obvious that it takes a longer time to complete the process if you have ten cases in the system than if there are only five. Too low a limit might result in higher costs since people are either doing nothing while waiting for the previous station or the next station. It can also be that people are helping other stations with work they are not experts in and therefore not quick at. Low limits mean you have a high throughput but you are not as protected against variation in size of the support cases. Clogs in the flow will occur more often.

Metrics

To make the right decisions and follow them up to see if they give the right result you need to measure some important performance indicators. Some indicators I can recommend are lead time, velocity, quality and flow. Read more about decisions made from these metrics in the section "Variation and queue size".

Lead time

There are two important ways to measure lead-time. One is the total time it takes on average for a support case to go through the whole system, from the time it enters until it's completed. The second method is the time someone has been working on it. The second one is good for calculating return of investment (ROI) since you start investing when you start working on the support case.



Velocity

How much are you producing/fixing during in, let's say, one week? You can count support cases or if you like you can count big cases as 3, medium as 2 and small as 1. The total sum is your velocity.

Quality

How much of your work adds value? Good things are making the customer happy or using your time on preventive work. Bad things are urgent things or helping an annoyed customer again because the last fix didn't help. To get a good quality metrics you can divide the amount of value added (good) work with the total work (good + bad work).

Flow

To know if your capacity is at the right level, you can measure how often your flow is blocked. This happens when there are no completed items upstream or the team downstream is blocking.

How to decide what is next

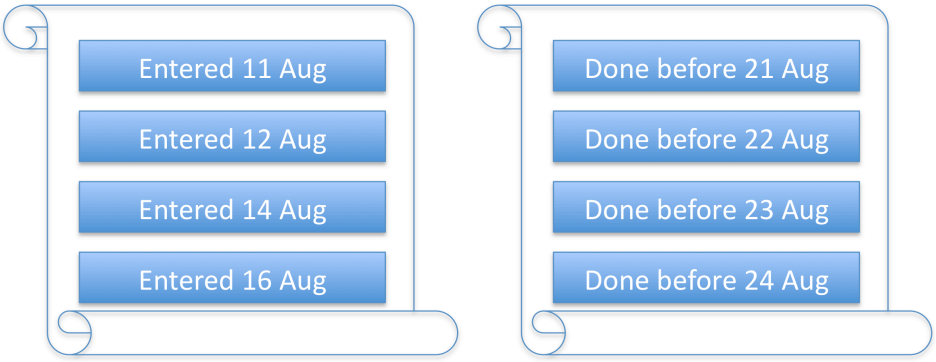
There are many ways to decide which support case to take into the system when there are empty slots in the first column. Here are some suggestions. Pick or mix the ones that are most suitable for your situation.

FIFO

The easiest way to prioritize is to take the oldest case first by using a first-in-first-out algorithm.

SLA

If you have a service level agreement (SLA) with guaranteed response times, you should prioritize cases based on their deadline.



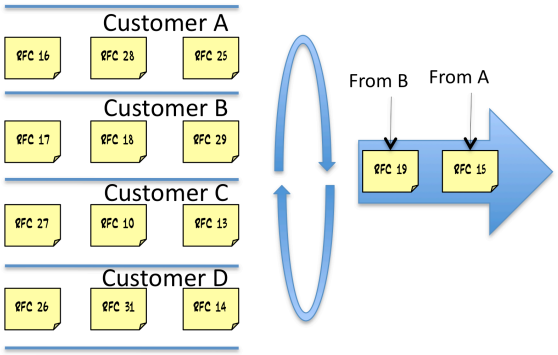
Example of two queues, the one to the left prioritized according to FIFO and the one to the right prioritized according to SLA response time.

Cherry picking

To quickly reduce the number of cases in the queue you can pick the easiest first. This means the output will be high but there will be some hard cases that will be hanging around for a long time. Those stoppers will lower the average lead-time.

Round Robin

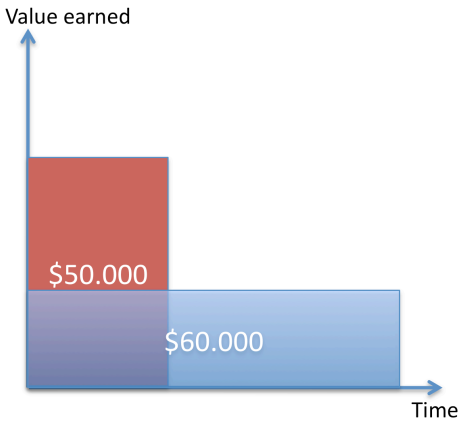
Do you have a limited number of customers, e.g. if you are supporting a number of sales offices divided into different countries or regions? Then they can have their own queue and the authority to prioritize it. You can then alter between the queues, one by one, and pick the one with highest priority. This is known as the Round Robin algorithm.



A queue prioritized by each customer and pick according to the round robin algorithm

Value

A really good algorithm to use is to optimize on value. High value first. But it's also interesting to know how value is spread over time. Some tickets might have a high value for a short time and should be prioritized higher than a ticket with an even higher total value but spread over a longer time. It is also sometimes best to prioritize tickets with continuously increasing value over a ticket with an end date.



Prioritize a case with high value during short time before a case with higher value but spread over longer time

Managed

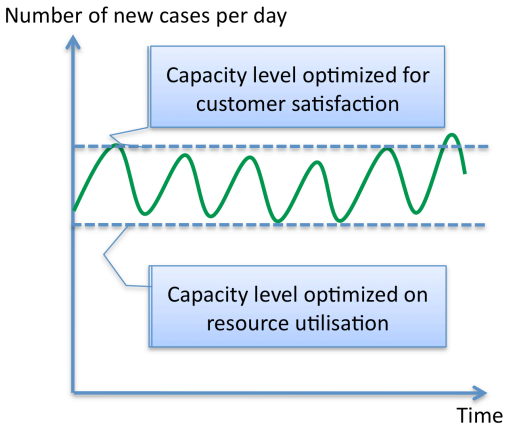
If a manager is helping you to decide which case to take next it is nice to have a next-queue so you know which one to take the next time a slot is empty. This is of course not needed if the queue of cases is already in prioritized order.

Levelling out work load

If one support case needs a lot of work from, let's say, the reproduce team it will stop the flow and leave the teams downstream without work. If the limit for "Reproduce" is higher than one case, parallel cases that are easy and quickly finished can be taken on to insure that there is work for the downstream teams.

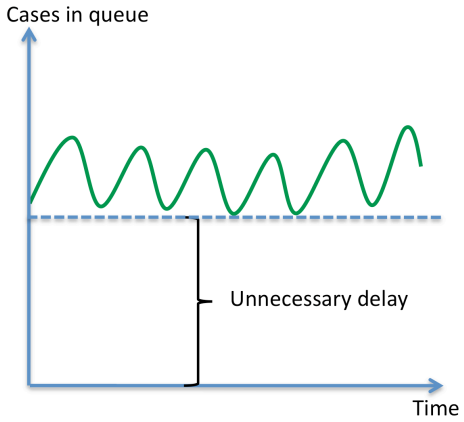
Variation and queue size

Unless you are willing to pay for unused capacity there will be a queue as input to your support process. This is to even out the unevenness that is natural when dealing with input generated by support need.

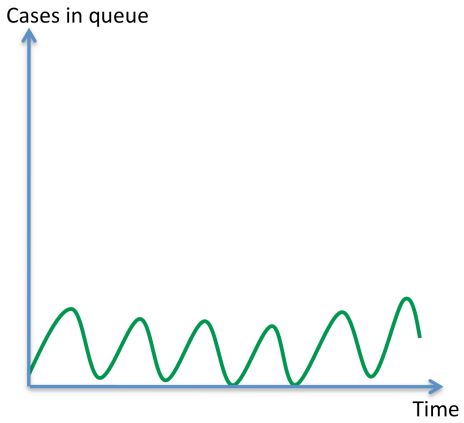


Number of incoming support cases varies over time

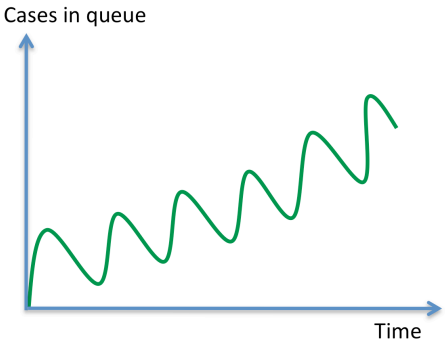
If you want to optimize on resource utilisation then you should staff so that the queue of support cases never reaches 0 for more than a few minutes. At the same time there is no point in having a queue that never goes down to 1. The distance between the minimum and 0 does not add value. It does add irritation.



If you want to optimize on customer satisfaction, then you should staff so there is always capacity to immediately start working with incoming support cases. That is, the queue is always 0.



The right capacity level for you may lie somewhere in between. You can staff so that your VIP customers get an immediate response in 50% of the cases, while the other customers get an immediate response in 5% of the cases.



By keeping track of the number of cases in the queue you can determine if you are staffed correctly. In the case above it is obvious that the capacity is too low.

Continuous improvement

A good feedback loop is even more important than a good start. My tips and tricks do not describe the perfect support process. You need to customize the process to meet your organization's needs. Since circumstances change no process stays perfect over time so you need to continuously improve it. My recommendation is to have a meeting, at least monthly, to talk about problems and improvements.

Good Luck!

/ Tomas Björkholm