Title: Taming the customer support queue: A Kanban Experience Report

Presenter: Karen Greaves

Stage: Insights  Session Type: Experience Report  Duration: 60 minutes  Level: Practising

Summary

In Aug 2010 we reduced the size of our development team. We needed to handle customer support issues and new development with the 3 remaining teams. We had over 600 open defects and the list was growing daily. On top of that our first clients were going live. Clearly some radical change was needed. We implemented Kanban.

The result? In Dec 2010 we had no open severity 1 issues. Our queue was down to 54 issues. SLAs are met in most cases, and our resolution rate exceeds our incoming rate.

I’d like to share our story, and inspire you to tame your support queue too!

Learning Outcomes

- Practical tips you can use to actively manage your customer defect queue
- What metrics to track with Kanban to help make informed decisions
- How to go about designing an evolving a Kanban board and setting WIP limits
- Getting stakeholder buyin to a new process especially with looming deadlines

Process/Mechanics

I’ll use a presentation with a few slides, mostly showing pictures of our board or graphs and metrics we collected. I’ll talk through our story, and allow lots of time for discussion and questions about some of the decisions we made.

Detailed Description

Below are some of the points I will cover for each of the learning Outcomes

Active Defect Management

We applied some blanket rules to clean up issues i.e. we closed the following: issues on older versions, all internal issues, external issues older than 3 months, lowest priority issues.

We changed from having 5 different categories for priority/severity to only 3. We separated bugs raised by product development from other bugs, and changed our dev process to say any issue found needed to be fixed before we worked on new functionality.

Once the queue was cleaned up, we make sure we still actively manage it, so we regularly clean it up by doing the following: Resolving issues if we have been waiting for a response from the reporter for more than 3 days. Closing issues older than 3 months
Kanban board design

Designing a Kanban board can be overwhelming. We just started with the basics: To do, In Progress, Test, Done. We had no WIP limits. But we recorded our CFD daily from day 1.

Today our board looks like this:

We set our WIP limits by looking at the number of people doing that work, and agreeing how many items they could work on at once. If there was a bottleneck we changed the limits.

An important insight was that without buffers (Ready for Dev & Ready for Test) we forgot it was all about pull, and instead were pushing work to the next column. Once these buffers were in place the team could more easily self organize to optimize the throughput of the whole system.

Engaging Stakeholders

We did some analysis on the size of the problem. Incoming rates, closure rates etc, and had a conversation with our stakeholders (Professional Services arm of our company) that we would not be able to solve all issues. We got executive guidance to focus on Sev 1 issues for our highest priority client first.

We explained that although minor issues were still issues in the product, given our capacity constraints, logging them wouldn’t really achieve anything, and got understanding from our stakeholders about the importance of focusing on the right stuff.
We made sure we kept other stakeholders in loop even when we were not working on their issues, and gave a prediction of when we could start. Our predictions (which were based on actual data from our CFD) turned out to be pretty accurate and this bought us a lot of respect. People trusted that if we told them we’d look at their issue in 3 weeks, that we really would look at their issue in 3 weeks.

**Metrics and Quantitative Decision Making**

We track how many items were in each column daily from the start in a CFD. We also kept comments on our CFD about things that happened. We also kept stats on the number of issues in our full queue by priority and type (defect or feature). Although we didn’t know what the data would tell us at the start, having the data proved useful to spot problems or see what impacted our cycle time.

Data showed that we slowed down every time we released a patch, we slowed down whenever we had to do merges, and that most of our urgent issues (expedite) items were resolved without code fixes.

Although these might be intuitive, we can put numbers to them. So we know that if we do 9 patches in a month rather than 4, we will do about 50% less issues. We can use this data to calculate the sweet spot between business need and support cost. This helps us make good decisions.

**Contact info:**

Email: Karen.greaves@gmail.com

Skype: Karen.greaves

Twitter: @karen_greaves

Mobile: +27 78 457 7203