



**Victor Szalvay**  
In 2000, Victor Szalvay co-founded Danube Technologies, Inc., a Bellevue, Washington-based software consulting company focused on software process training and coaching. A Certified Scrum Trainer with experience implementing Agile frameworks for software development teams in the public and private sector, he currently serves as Product Owner for Danube's flagship product, ScrumWorks™ Pro.

## Feedback-driven Release Planning

Posted by Victor Szalvay on April 18, 2007 at 10:20 a.m.

It's been a while since my last article because I've been busy as the Product Owner for Danube's ScrumWorks Pro product. That means I've learned a lot and I'd like to share one of the biggest lessons I've learned about the importance of feedback in release planning.

To provide some context, Danube first released ScrumWorks Pro on January 15th (Winter 2007 release). We planned for roughly quarterly releases thereafter and so our next target release date was April 17th (Spring 2007 release). I'm writing this after the 17th, so, clearly, we didn't make our intended release date. The adjusted release date is May 7th. But how did we end up missing our date? In Scrum, don't you just work on the highest priority items until your release date rolls around? And since quality is built-in (all phases of development happen each sprint), then there's no need for extensive stabilization/QA toward the end of the cycle, right?

That's generally true, but I fell into a different trap altogether: As Product Owner, I didn't wait for feedback on our January 15th release before proceeding to take on ambitious features in the current release cycle. Because I didn't wait for feedback, the priority and business valuation of my backlog was stale. That isn't to say that my backlog wasn't prioritized, though. In fact, I tried to out-smart the impact of feedback by prioritizing my Spring Release prior to releasing in January. In other words, the up-front release planning I did turned out to be inaccurate because the priorities of my customers changed given a chance to use the new features and enhancements in our Winter release.

But could it be that I just prioritized the Spring release incorrectly? The feature requests from our customer base prior to the Spring release were pretty clear. The problem is that our customer's changed their priorities after seeing and using the product of the Winter release.

But Scrum allows for flexible change of direction and each sprint the PO can make decisions to switch course, right? So why didn't I just re-prioritize and switch course after receiving user feedback? In an ideal world this would be the strategy, but I made the further mistake of taking on an ambitiously large feature right after the Winter release. This feature took us nearly three times the average time to build. In order for the feature to provide some value, it had to be developed to a certain point. Anything before that point made it nearly useless. So there was pressure to finish the feature to an acceptable level, and abandoning it didn't make financial sense.

**danube**

Danube Technologies, Inc.  
520 SW 6th Ave, Suite 940  
Portland, OR 97204 USA  
1.888.5.danube  
T: +1.503.248.0800  
F: +1.503.248.0801

Currently, we're working to incorporate some of the high priority requests from customers who have been using ScrumWorks Pro. To release without incorporating feedback might be interpreted as disregard for customers. The result is that we pushed the release date to accommodate the feedback. It's not a disastrous situation since the large feature that bogged us down is actually valuable. But had the situation been different, had we selected a feature of low value, or had our release date been firm, the result could have been dramatic.

In retrospect, I would have done things differently. I would have started the first couple sprints of the Spring Release working on smaller features that were recently identified as high priority. I would have then actively sought feedback from users and re-prioritized my entire backlog and release plan based on the feedback. This may sound crazy -- not having a firm release plan going into sprints -- but it's the best way to make sure our feature set is current and of high value to users.

As often as I try to think and plan in advance, our business requirements change out from under me. To gather some empirical evidence of this, I went back into our requirements wiki and counted up the instances where up-front feature planning paid off. I was shocked to discover that nearly 100 percent of the time our up-front requirements and plans were completely scrapped prior to the eventual implementation based on current customer feedback. Was all that work for not? Not necessarily, it's valuable to think through requirements details and scenarios, but it's not as valuable as I inherently thought it to be.

My advice based on this experience is to consider rejecting the traditional notion of a release plan and treat the release cycle as a constantly fluctuating set of goals that are continually influenced by customer feedback. Of course, this causes problems for marketing. But the fact is that more customers will be satisfied if their requests are actually turned around quickly. I balance feedback-driven release planning with the need to announce for marketing purposes by always delaying decisions until the last responsible moment.

The logo for Danube Technologies, featuring the word "danube" in a white, lowercase, sans-serif font on a blue rectangular background.

Danube Technologies, Inc.  
520 SW 6th Ave, Suite 940  
Portland, OR 97204 USA  
1.888.5.danube  
T: +1.503.248.0800  
F: +1.503.248.0801