

## It's Software, Not a Unicorn: Why Tools Won't Solve Organizational Dysfunction

Posted by Jim Schiel on June 3, 2008 at 9:27 a.m.

Jim Schiel is a CST with a strong background in enterprise level Scrum installations. Prior to joining Danube, Schiel worked at Siemens for 22 years, initially as a developer, then as a manager for 16 years, and eventually playing an instrumental role in creating one of the largest Scrum installations in the world. As a business process engineer at Siemens, he helped identify, document, and implement best practices for enterprise Scrum.

When I was much younger, it was explained to me that sticking my dad's screwdriver into the electrical outlet was not a good idea. As a good little boy, I obeyed and convinced my older brother to try it, instead. Fortunately, we used an outlet at the end of a long hallway, so my brother's resulting flight from the shock drew a clean and uninterrupted arc down the hallway.

For those of you now considering how heartless I was as a child, rest assured that Karma got me back a number of years later when I misused a multimeter while checking a 220-volt power line.

Whether I used the screwdriver or multimeter wrong, there was nothing that the tools themselves could do to keep me from using them incorrectly, nor would the tools do anything to fix what I had done. Similarly, the tools we use in software development cannot stop us from making poor decisions and they usually can't clean up the mess that ensues. Even though we would like them to be, tools aren't a cure for the poison of bad decisions.

Until the 17th century, it was widely believed that the unicorn horn cured almost anything. From being an antidote to poisons of all sorts, from the plague to epilepsy, the unicorn horn was priceless. The Church of St. Denis in Paris frequently used its unicorn horn by dipping it in water to create a restorative elixir drunk by the infirm. Similarly, there are many times when we would wish our software tools could miraculously fix whatever is poisoning our projects.

I've seen instances in which Sprint lengths were shorter some Sprints, but longer for others. When a software tool written to follow the concepts of Scrum can't properly represent the project burndown because of the varied Sprint lengths, it raises concern that the tool won't normalize the sprint lengths to provide a simpler view. In my opinion, however, the question shouldn't be why the tool doesn't support Sprints of varying lengths. The proper question is: Why does the organization use Sprints of varying lengths? The issue of Sprint length has been debated in various forums and, while we all have different opinions about how long a Sprint should be (a recent sampling suggested three weeks was the most popular), we mostly



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

agree that Sprint length, whatever we choose, should remain consistent. Rather than use a tool as a panacea to mask the problem, one course of action is to scrutinize the forces that drive Sprint length in the organization. Are large stories extending Sprint length? Is management forcing shorter Sprints? Why? Remember, the tool isn't supposed to fix organizational issues like this. In fact, when the tool doesn't support what the organization is doing, questions should be asked about organizational decisions.

Another situation I see frequently is teams that rarely reach their Sprint Review meetings with all of the committed goals completed. Strangely, many tools don't accurately reflect a team's real progress, presenting a relatively normal burn when a team's stories aren't actually done. So why doesn't the tool show what's obviously happening to the team? How good is the tool if it can't even show when a Scrum team is in trouble of missing their commitments?

So why isn't the team meeting its commitments? Usually, this phenomenon is caused by over-committing at Sprint Planning or by surprises during the Sprint that cause stories to become much larger than their original estimations. However, a third cause that would not appear on the Sprint Burndown is related to untracked changes in team load or capacity during the Sprint. When a new story is added to the team's load during the Sprint because "it just came up" or "it's really important" or "it won't take long," the team's ability to meet its original commitments is challenged. Likewise, when a team member is unexpectedly removed from the team (for many of the aforementioned reasons), the team will often not fulfill its Sprint Planning commitments. When either of these events occurs without properly following the Scrum framework (that is to say, cancelling the Sprint and reworking the team's commitment), the tool can neither stop it nor properly report it. A software tool can't force you to make good decisions or to track what you did.

I've also fielded many requests over the years for better time tracking in our software tools. These come in a variety of flavors: How many actual hours were spent on a task?; What's the difference between actual and estimated hours?; Who on the team has assigned themselves to too many tasks? Or too few? These questions always set off an alarm for me when I hear them because, almost every time, they signal a possible misunderstanding of Scrum that could be VERY detrimental to the team. Each of these is a potential poison. Accounting for these

**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

poisons in our software tools can't provide a cure for their effects – we have to look closer to the source for the proper cures.

Interestingly enough, in a marketing strategy to beat all strategies, the unicorn horns sold by Scandinavians through Viking and Arab middlemen ended up being nothing more than an incredibly well kept, centuries-long hoax. While the Scandinavians made millions, what they were selling to European royalty and the church were nothing more than the abnormally long teeth of the arctic Narwhal (<http://en.wikipedia.org/wiki/Narwhal>).

*There was no magic cure then and there's no magic cure today.*

In truth, we create the poisons that pollute our projects. We call them emergencies, business priorities, contractual requirements, customer satisfaction issues, must haves...you name it. Whatever we call them, we cannot rely on software tools to protect us from ourselves. We must rely on ourselves. The cure for what ails our projects is, quite simply, discipline. We must discipline ourselves to abide by the few guidelines that Scrum and Agile development put forth in order to be effective. Not that this is shocking news for software developers. After all, how often does the development process actually have a fairy tale ending?

**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