What’s Holding You Back?

by Mike Cohn • 0 Comments • originally published in Better Software on 2004-07-01

What does agile software development and running have in common? I like to run, but I’m really bad at it.

Not as bad as I used to be, though. I used to get bored or tired after three miles of a planned six-mile run and stop. I broke that habit by allowing myself to stop running only if I could name the specific reason why I was stopping. Do my shins hurt? Are my legs tired? Is my heart rate too high? Am I just bored? In other words, what is the constraint that is stopping me from running any farther today?

Software development teams—like my running—also are limited by some constraint. One of our challenges as project participants is to pinpoint that constraint. Eli Goldratt was the first to write about looking for improvements in a process by finding the one current constraint on the process (the Theory of Constraints).

We all have an implicit understanding of how this works. Suppose we have a performance problem with a client-server application. We first try to determine whether the problem (that is, the constraint) is the database, the SQL, the network, the program design, or the program code. We’ve learned that if we identify the constraint and then correct it, we will have the most dramatic impact on the application.

At any point in time, each project has but a single constraint. Suppose you’re on a project with eight programmers and one part-time tester. Most likely, this project is constrained by the tester. Adding eight more programmers will not increase the overall throughput of this team (unless the programmers act as testers, which effectively means we’ve added testers). If this team wants to produce more software, team members need to address the constraint—having only a part-time tester.

The recent shift toward Agile software development has helped point us toward new places to look for these constraints. In the past, we would look for project constraints in our requirements-gathering techniques, in our team size and tools, in our models, and in our detailed written
communication and process descriptions. Like Bill Curtis’s drunk looking for his misplaced keys under the lamppost only because the light is better there, we were looking for our constraints in the wrong places. Rather than looking for constraints in the tools we use, Agile development encourages us to look for constraints in individuals and their interactions. Rather than looking for constraints in the details of our plans, we are looking instead for constraints in how we respond when the inevitable changes occur.

Agile development also supplies better ways to overcome these constraints. We’ve finally learned that we cannot write all requirements upfront; Extreme Programming introduced the notion of writing user stories on note cards and using those for requirements. We’ve stopped asking developers to write more documentation; Crystal shifts the focus to high bandwidth, face-to-face conversations. Rather than spend time planning beyond the line of sight at the start of the project, Scrum has teams spend a little time planning at the beginning of each iteration.

As you read this special issue on Agile software development, I encourage you to think about what is constraining your current project. Don’t let yourself off with a generality; identify the one constraint. Software projects are complex webs of interrelated activities, and there are many things you can improve that will speed things up slightly. But that’s like tuning the SQL when the network is congested. Maybe it results in a 1 percent improvement. That’s nothing. Finding your project’s constraint should be a ticket to at least 100 percent improvement. Productivity gains like this are far from uncommon for Agile teams. Now, if only I could get that type of gain we use, Agile software development encourages us in my running speed… .

Posted: June 30, 2004

Tagged: user stories, teams, programming, agile software development
processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the Better User Stories video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.