MUST BOTTLENECKS IN THE SOFTWARE PROCESS BE FEARED?

Paradigm Shifts Are Uncomfortable

Lt Col Joe G. Baker, USAF

The books *The Goal* and *The Race* stem from the Theory of Constraints concept. *The Goal* is written as fiction with the main characters using this concept to solve a series of manufacturing problems. *The Race* is a follow-on book, but is more "how to" in presenting the Theory of Constraints concept.

Memories surfaced after I read these books. Just pictures from the past, but these flashbacks took on new and disappointing meanings. Disappointment came as I began to realize that the Configuration Control Board (CCB) meetings I chaired and the software development division I directed were not maximizing the resources available anywhere close to the level they could achieve.

I did not understand "bottlenecks," constraints or, according to *The Race*, "capacity constraint resources." I know the definition of a bottleneck, but I did not know what the real bottleneck was in the software development process I sought to control as the Chief of Software Development and the CCB chairperson. This was a small software activity of about 10 government and 20 contract programmers who maintained the application code on an IBM mainframe. The atmosphere in this environment was casual. Individuals were trusted to do the job, and this was good. Management opinion of workload was subjective, and this was bad.

Before the CCB, the functional users would rank, in order, the Problem Reports and new requirements. Our programmers reviewed this list, met with the users, and determined what could be done in the next release. Most applications were stovepiped with one or two programmers per application. I moved our organization into this process so the CCB meetings would not encompass one whole day and accomplish only the official approval of the contents of a release. During the CCB, I asked each lead programmer what he or she could accomplish. The functional users then openly agreed and the meeting ended. Though the meeting was short, decisions were made and output was maximized. Most CCB attendees appreciated this process change.

After following this process so far, do you perceive what others and I considered the constraint in our software process? The programmers were viewed as the constraint or bottleneck. Some of you may be curious about the testing resources. When we developed our release schedule, we blocked off time for testing. From experience, we automatically set aside a certain number of weeks per release. Our software process assumed the test time set aside for each software release was adequate. I should have invested a little time into the activities conducted during our test time and, thus, possibly increased the number of changes we included in each release. It was now too late. What I once viewed with some pride and satisfaction took on new emotions. Paradigm shifts are not painless.

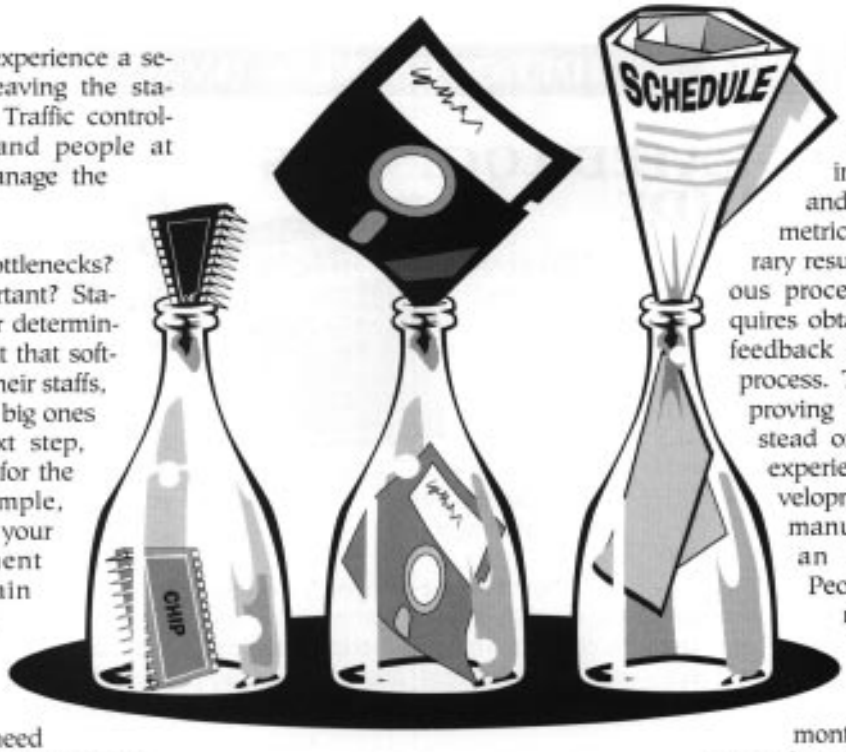
Problems Provide Opportunities

My point is that for every problem there is an opportunity. For example, the opportunity to improve the capacity and speed of software delivery exists. This may sound like I'm writing about manufacturing instead of software development or software maintenance; but, in some ways "software manufacturing" may be a more appropriate paradigm.

A bottleneck is an opportunity to know how many software changes you can insert per release, and the opportunity to control and maximize throughput through each component of your process. For example, when you attend a crowded music or sport-

Lt Col Baker is Chief, Communications-Computer Systems Services Branch, Communications-Computer Systems Directorate, Aeronautical Systems Center, Wright-Patterson AFB, Ohio.

ing event, don't you experience a series of bottlenecks leaving the stadium or parking lot? Traffic controllers erect barriers, and people at strategic locations manage the crowd.



Where are your bottlenecks? Which is most important? Statistical means exist for determining these, but I submit that software managers, with their staffs, probably can pick the big ones out subjectively. Next step, determine the reason for the bottleneck. For example, suppose you selected your software development system as the main bottleneck. Is the solution to buy another? Stop and think. Why is it your constraint? Does it need additional memory or an upgraded operating system? Do you need to juggle schedules to add another shift during peak times. Before you spend heavily on new equipment, ask yourself how you are utilizing your time on different components of this system. You may need to break off a part or buy a subset of the system on which to do your prototyping or unit testing.

I was fortunate to be associated with a project where individuals sought to maximize throughput as each bottleneck was encountered. Briefly, they costed each change after determining the requirements by discipline such as programming, building, integration, testing and documentation. Since building each release through the build shop was time-consuming, they grouped changes together that used the same software modules. This reduced compilation time by the build shop, and shortened the time the integrators used on the system to check out the new code.

When the testers did their formal testing, their standard tests encompassed several changes with one test, reducing the time spent using the critical system. As you might guess, as soon as one bottleneck was eliminated another surfaced. However, they worked these also. Ultimately, they could identify potential new bottlenecks before they eliminated an old one. This began to influence their solutions to the original constraint. *The Goal* and *The Race* go into more detail about constraints.

Identifying and manipulating your bottlenecks should help you sleep at night. Then again, you may lie awake thinking about them.

Conclusions

To improve your throughput, you can hold meetings, announce goals, and produce impressive metrics to achieve temporary results. Serious, continuous process improvement requires obtaining good, ongoing feedback and monitoring the process. The difference is improving throughput once instead of continually. In my experience, a software development, maintenance or manufacturing project is an evolving process. People, technology and requirements change yearly. In some organizations, changes occur monthly. Unless you can see and understand your bottlenecks, how can you adjust schedules and resources effectively? You are bound to the person who for that month seems to have the right answers. In the next month, the knowledgeable person is wrong because his or her subjective opinion is not based upon fact.

For the sake of credibility, I will avoid postulating a maxim that controlling bottlenecks will solve all or most problems. From my experience, the source of anxiety will be removed as you begin to get a firm grip on your process. Identifying and manipulating your bottlenecks should help you sleep at night. Then again, you may lie awake thinking about them.

References

1. Goldratt, Eliyahu M., and Cox, Jeff. *The Goal: A Process of Ongoing Improvement*, North River Press, Inc., Croton-on-Hudson, New York, 1986.

2. Goldratt, Eliyahu M., Robert E. Fox, *The Race*, North River Press, Inc., Croton-on-Hudson, New York, 1986.