I Can Sing and Dance, But I Don’t Juggle

Date:	Nov 12, 2012	
Title:	I Can Sing and Dance, But I Don’t Juggle	
Code:	AR-JUG	
Abstract:	What is the true cost of too much multitasking? Is there even a cost or is your ability to multitask just plain expected as you advance through the software development career path? You’ll learn what steps to take so that you and your team can become more effective at focusing to “getting to done.” This is an excerpt from “I’m Not God, I’m Just a Project Manager” by the author.	
Version:	1.0 (2012-09-26)	Original, excerpt from “I’m Not God, I’m Just a Project Manager”
	1.0.1 (2012-11-12)	Includes Clarke’s column screen captures and biography reference

Interruptions Are Bad!

Let’s pretend that you have far more work assigned to you and your team than is reasonable. Like all of us these days, we’re expected to juggle a bunch of balls in the air.

Consider what Tom DeMarco and Timothy Lister state about the importance of maintaining focus in order to get things done. In their *Peopleware* book, they present a number of statistics regarding software developer productivity. Their results focus on software developer productivity and not the typical factory worker productivity statistics that you see in other publications!

It may be *your* fault that a developer isn’t performing!

If your working environment is noisy, offers no privacy, and is prone to interruptions throughout the day, then I’d suggest saving leasing costs by closing down your office and starting over with everybody working from home and communicating with each other via the Internet. It would be better than the environment you currently have. Phone interruptions doesn’t just take the five minutes of your time. DeMarco and Lister believe that the total time that call takes you away from your work is really 20 minutes because of an overall ***re-immersion time***. This refers to the process of getting back in gear where you left off — and it doesn’t even really count as work!

What about e-mails?

You may think that your team is responsive by instantly responding to incoming e-mails, but that is wrong. There's probably a large percentage of your team's day handling e-mails. That means that your team may be devoted to getting off track and out of those 100+ e-mails a day, how many of those e-mails are really that important?

Very few I'd assume.

The Brilliance of Attaining Flow

Great programmers are able to focus and almost enter a meditative form while programming. Not in need of much human contact, the modern-day software developer is at their best by writing code with the ease that an author can put thoughts into words.

You and your team are at their best when they get into flow.

Flow? Tom DeMarco in his insightful book *Slack* believes that this flow (or ***emotional inertia***) is what you want for your entire team. Few teams really operate in this nirvana, but when you have a team that is that productive, you'll *know* it. Some characteristics that I've seen with a team that is truly in the flow are:

- Intuitive radar-like (like the actor named Radar in the Mash TV show) where the team enjoys an unspoken ability to know exactly what to do (and what other team members are thinking).
- Almost no management is required on your part.
- Work consistently gets done on or ahead of schedule.
- Team meetings (for example, Daily Scrums) are effortless and productive.
- There's a noticeable positive, optimistic, and motivated energy.

You may have witnessed other characteristics, but the one thing is for sure: once you have a team operating like this, do not let management break the team apart! Mark Smith writes that according to Bruce Tuckman's mourning stage of team development, management has the tendency of thinking that breaking up a stellar team will naturally create other stellar teams.

Nothing could be further from the truth! You want other teams to rise to the occasion and become like your preeminent team. I've seen too many times that after a team completes a project that a "killer" (meaning "great") team is broken up to join less-productive teams. This unfortunately can result in all if the teams becoming now less functional. Now that we have a sense of a great development team's make up, there's a sure fire way that management can unknowingly hurt its chances of success.

The True Cost of Multitasking

For an individual contributor to advance through their career path they typically expected to handle concurrent tasks. In fact, the ability to multitask has been a key distinguishing characteristic separating a senior software engineer from a less experienced staff software engineer.

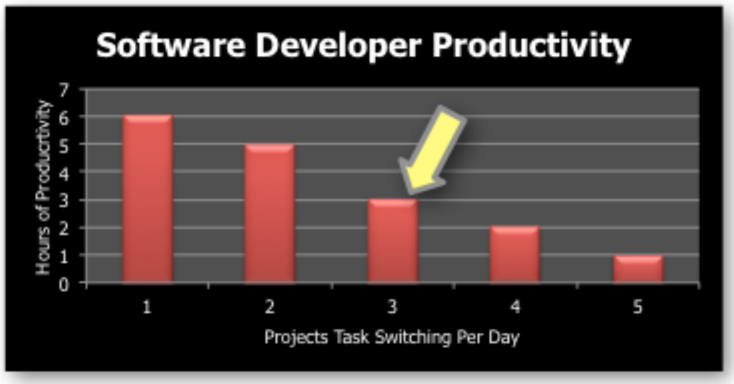
In addition to the cost of disruptions (in the previous section), we'll examine the cost of spreading developers over several projects at one time. Simply put, when your mind has to change gears, there's an efficiency impact and in the case of competitive software engineering, speed to task completion and speed to market means a great deal.

Efficiency and focus can take many forms. Once again in *Peopleware*, the authors define something called an efficiency factor, or ***e-factor***, that illustrates algorithmically this phenomenon:

```
E-factor = Uninterrupted hours / Body-present hours
```

Your e-factor is 50% if you can focus for four hours of uninterrupted time per day (e-factor = 4 / 8—or 50%). If another developer on the same team has a lower e-factor, say 25%, then you could surmise that this worker has to work twice as hard as the first (with 50% e-factor) to complete work.

In DeMarco's *Slack*, survey results show that there are significant switching penalties when expecting developers to work on more than one project at a time.



In this study, there appears to be a dramatic drop in productivity after two concurrent projects.

Similarly, Mike Cohn's *Agile Estimating and Planning* offers similar results due to multitasking Although a little different than DeMarco's results, Cohn indicates that Clark and Wheelwright in 1993 studied the effects of multitasking on individual productivity and found that productivity dips significantly after three unrelated tasks (projects). At that point, the time spent switching (getting re-immersed to the task at hand) becomes a more tangible cost, burden, and increasing risk of completing any task!

(Interesting that in Cohn's study results, productivity marginally improves at two concurrent projects. Sometimes a change of pace throughout the day can be a positive, I guess.)

How Many Teams Can You Lead At Once?

Okay. We've discussed developer loss of productivity. What about your productivity? As a management or leader type, you may not have to be as focused into the details as a developer on a project team. I would imagine that you are expected to handle more than one project at one time.

I haven't seen any empirical information on the subject, but I have over the years found this to be true:

I can handle leading on up to two tactical projects destined for release. (It doesn't matter whether the projects are Agile either—there's lots of preparation and planning required to ensure your Agile project runs smoothly.) When I'm handling three concurrently, I start losing grasp of the details and find myself shifting from active mode to passive, "taking notes" mode.

In addition, I like leading on at least one strategic project. This type of project may be in the early planning or research stage. Finally, having the day-to-day responsibility for already released products should be doable. This implies that you're making sure that maintenance and support issues are under control (which might require borrowing core resources from your other projects for emergencies).

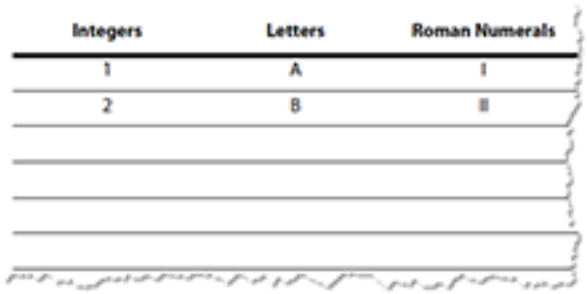
Jim Lewis, in his superb *Project Planning Scheduling & Control* book, strongly suggests to never overload your project management duties with too many concurrent projects. He believes that three to five projects is doable at a time, but there are many considerations in arriving at that number.

Keeping historical notes are a great way to track what works and what doesn't work for you. I've just recently asked the same question on about twenty LinkedIn discussion groups, and the early indication is that folks in project leadership positions can be successful leading one or two projects at a time. There was one respondent who said that he is constantly handed over the reins to at least 12 projects (and he just hangs on for dear life).

Whew!

A Final Word

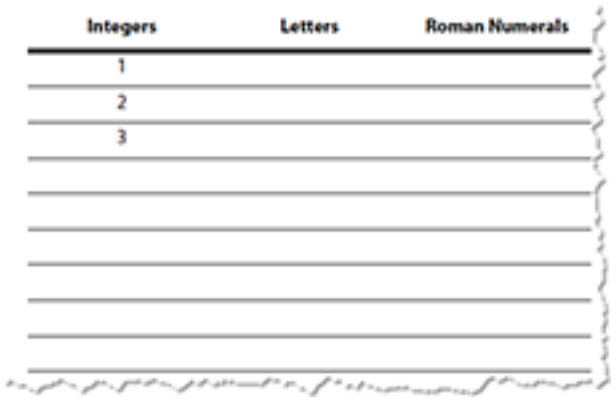
Chances are all of this “limiting the number of concurrent tasks” mumbo-jumbo doesn’t resonate with your management. If so, I’d recommend running the executive team of your company through a ten-minute exercise in the prior “The True Cost of Multitasking” section. You have 30 seconds to enter an ordered sequence of numbers, letters, and roman numerals across each row (I’ve started the first two rows):



Integers	Letters	Roman Numerals
1	A	I
2	B	II

Ok? Start your 30-second counter and then go! When done, calculate the total number of cells completed.

Next, do the same thing, but this time create a similar ordered sequence of numbers (to 26), then letters (to Z), and then roman numerals down each column:



Integers	Letters	Roman Numerals
1		
2		
3		

Start your 30-second counter and go! When done, calculate the total number of cells completed and compare this count versus the first. Surprised? Originally published in the May 14, 2008 issue of “Iterations,” Clarke Ching believed that Agile software development is far easier to have each person working on one task at a time with minimal task switching. When your brain focuses on one train of thought (the second exercise), you’ll get much more done than if your brain is switching gears (the first exercise). I’ve seen, on average, more than 30% more cells completed among participants in workshops I’ve lead. In a few cases, I’ve even seen 100% more cells completed!

This innovative 10-minute exercise will convince your managers to change their habit of overloading you and your team with too many concurrent tasks to juggle.

If you’re still expected to drive lots of concurrent project and practical expectations and realistic workloads doesn’t change management’s behavior, it might be time to hunt for a new company to work for!

I’m anxious to hear from you all regarding this question. You can contact me through the http://www.leadingswmaniacs.com Web site.

Bibliography

Ching, Clarke. “The Agile Experience: Multitasking Is Evil.” *Sticky Minds*. 14 May, 2008 (http://www.stickyminds.com/sitewide.asp?Function=edetail&ObjectType=ART&ObjectId=14287).

Cohn, Mike. *Agile Estimating and Planning*. Upper Saddle River, NJ: Pearson Education, 2006.

DeMarco, Tom and Timothy Lister. *Peopleware: Productive Projects and Teams, 2nd Edition*. New York: Dorsett House, 1999.

Lewis, James P. *Project Planning, Scheduling & Control: A Hands-On Guide to Bringing Projects In On Time and On Budget*. Fifth Edition. New York: McGraw-Hill, 2010.

DeMarco, Tom. *Slack: Getting Past Burnout, Busywork, and the Myth of Total Efficiency*. New York: Broadway Books, 2002.

Smith, Mark K. “Bruce W. Tuckman - Forming, Storming, Norming and Performing in Groups.” The Encyclopedia of Informal Education. 2005 (http://www.infed.org/thinkers/tuckman.htm).

Whitaker, Ken. *I’m Not God, I’m Just a Project Manager!*. Seattle: Leading Software Maniacs, 2011.

Bio

Ken Whitaker of Leading Software Maniacs® (LSM) has more than twenty-five years of software development executive leadership and training experience in a variety of technology roles and industries. He has led commercial software teams at Software Publishing (remember Harvard Graphics?), Data General, embedded systems software companies, and enterprise software suppliers. Ken is an active PMI® member, Project Management Professional (PMP)® certified, and a Certified ScrumMaster (CSM). Sources for LSM’s material come from case studies, personal leadership experience, the PMI *Project Management Book of Knowledge* (*PMBOK® Guide*), and Ken’s leadership books: *Managing Software Maniacs*, *Principles of Software Development Leadership*, and *I’m Not God, I’m Just a Project Manager*. PM University, http://www.pmuniversity.com, is a new addition to Leading Software Maniacs’ online, eLearning curriculum focused on pragmatic project management and software leadership courses. Ken is also the creator of PM Chalkboard, http://www.pmchalkboard.com—the fastest way to learn basic project management principles with entertaining, no cost tutorial videos.

Leading Software Maniacs is proudly associated with:

PMI Registered Education Provider Project Management Institute