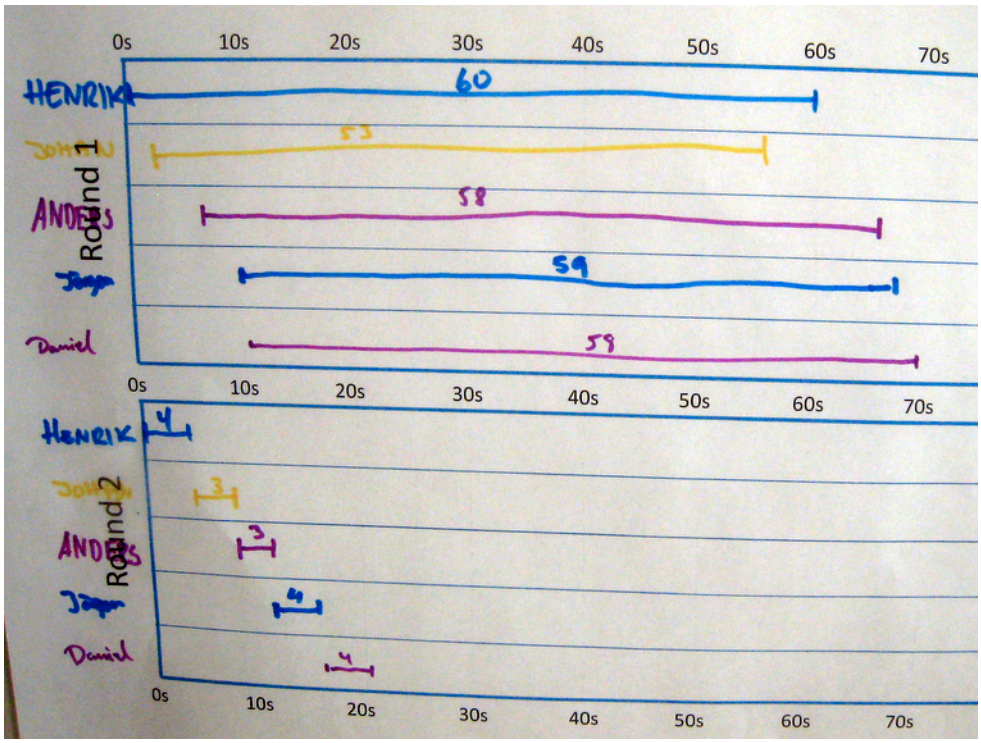
The Multitasking Name Game

or How Long Does it Take to Write a Name?

Henrik Kniberg
Version: 2011-12-07



About this article

Spoiler alert

This article is aimed at coaches, teachers, and anybody who is looking for a really cool way to illustrate the problem of multitasking.

If you think you might attend one my courses on lean & agile (or any other lean/agile trainer such as Mary Poppendieck), then I suggest you stop reading now. Reading the details of this simulation will reduce the fun in experiencing it yourself.

Really. Stop reading now if you think you might experience this simulation within the next few months.

Purpose of this article & the simulation

The purpose of this article is to describe a simulation that illustrates how bad multitasking is, and how easily we get drawn into it.

The article is primarily written for teachers and facilitators who want to know how this simulation works and how to facilitate it successfully. However, anybody else reading this article will probably gain an appreciation for the issue of multitasking and will hopefully see ways to reveal and solve the problem in their own workplace.

PS - if you don't know what the word *multitasking* means then don't worry. Just read on. Think of this article as an in-depth definition of the word :o)

About the simulation

Origin of the Multitasking Name Game

As coach, I'm continuously amazed by the amount of multitasking going on in most companies I've worked with, and how much unnecessary waste this causes. Over and over I find that one of the cheapest and quickest ways to significantly improve the productivity of any team or individual is to identify and reduce all sources of multitasking

Over the years I've been trying to find a simulation that illustrates the problem. If a picture is worth more than a thousand words, then a simulation is worth more than a thousand pictures. I found several simulations, some simple and some complex, but none that really nail the issue in a clear, simple, and repeatable way. So I stole the best ideas from some of the other simulations and created my own. To my surprise it worked really well the first time (as in, the participants were strongly affected by the exercise, several mentioned that it was the highlight of the course). Over the years I've continuously refined it, and it works even better now.

This simulation has now become very popular within the Lean & Agile training communities. One of the trainers that have adopted this exercise is Mary Poppendieck. She is normally very sceptical to simulations and doesn't use them. But after she saw this simulation during a lean workshop that we were co-training, she cheered loudly throughout the simulation and immediately adopted it as part of her own course material. Mary and other trainers frequently tell me stories of how strong impact this simulation has, especially on management teams.

Name of the simulation

The name of the simulation is "Multitasking Name Game", because that is exactly what it is. However, in class I often call it "How Long Does it Take to Write a Name", so that I don't ruin the surprise. Also, the second name is cool because I open the simulation by asking that simple question :o)

Licensing

Feel free to use this simulation as you like, but I appreciate if you mention or write somewhere that the simulation came from me.

The official URL is http://www.crisp.se/henrik.kniberg/multitasking-name-game

Time and materials needed

The core of the simulation takes about 20 minutes to run (regardless of groups size), plus another 10-20 minutes for debrief. The only tools needed are pen & paper & stopwatch. Ideally thick pens (whiteboard markers or sharpies), index cards, and a big stopwatch projected on a screen. Computer & projector is entirely optional. The simulation can be facilitated just fine using only a flipchart.

Executive summary

1. Write 5 names, all at the same time. Measure how long it takes per name, and in total.
2. Then write 5 names, one at a time. Measure again.
3. Marvel at the cost of multitasking

Detailed description

Step 1: Estimate how long it takes to write a name

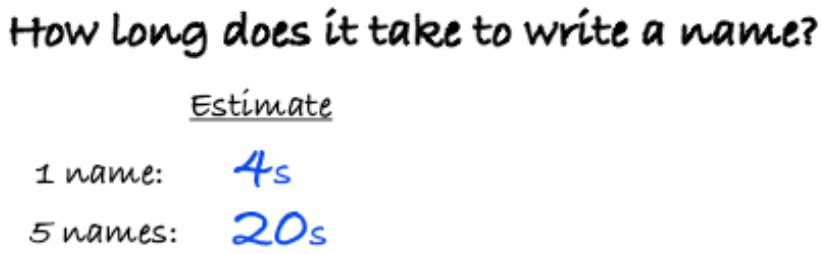
I skip all the preamble and start by asking the simple question "How long does it take to write a name?"

There is usually some confusion as people ponder what I mean with that question. I hold up a card "For example like this."

First people are reluctant at guessing, trying to hide behind "well it depends". I coax an answer out of them by saying things like "You really can't estimate? So you mean it might take over a year? Or less than a second?". Finally they start calling out estimates, usually around 4 seconds.

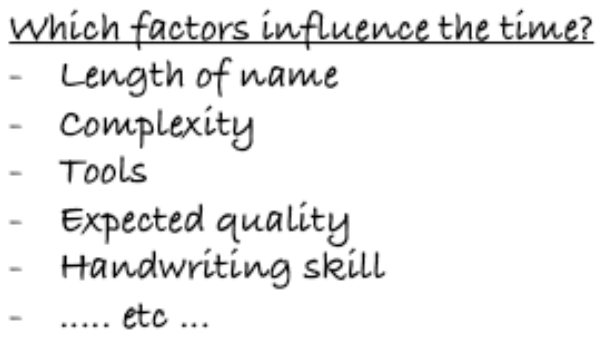
I then ask how long it takes to write 5 names. They usually guess around 20 seconds (the first number x 5)

I write these numbers on the flip chart.



Step 2: Discuss influencing factors

Next I ask "which factors influence this time? When you say 'it depends', what does it depend on?". The class starts calling out things and I write them down. Usually things like this:

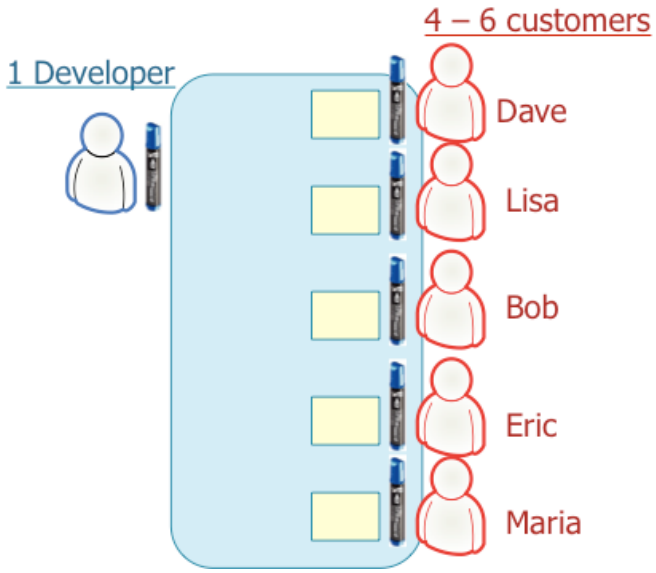


This is a trap. In the vast majority of cases, nobody will mention "multitasking" as one of the factors that influence the time. That's why I usually don't mention the name of the simulation, I want to illustrate that we tend to forget about the influence of multitasking.

Next I say "OK, let's find out the truth".

Step 3: Create groups and describe the Customer role

I ask people to divide into groups of 5 - 7 people, where one person in each group is Developer and the rest are Customers.



The only skill needed to be Developer is the ability to write letters on a piece of paper. The tool for that is a thick pen.

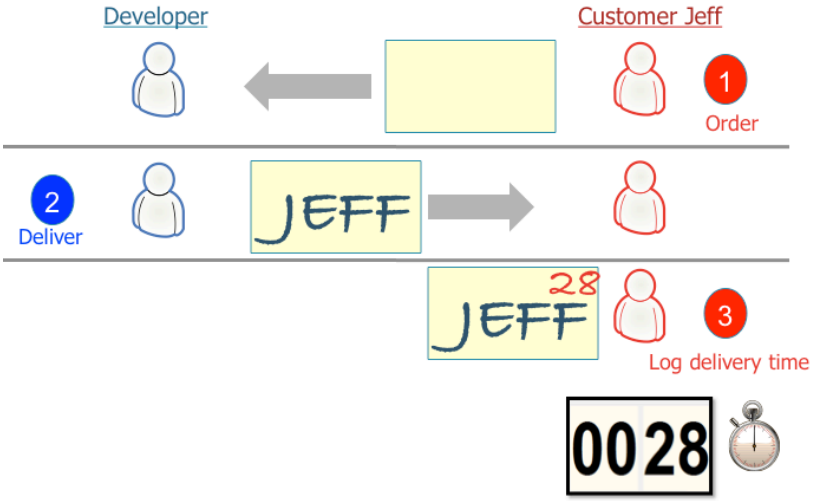
Here's what I tell the Customers:

Each of you have a blank index card. All you want is to have your name written on that card. The problem is, you don't know how to write letters - that's why you need the Developer!

Furthermore you, the Customer, want your name delivered as quickly as possible. It is your job to track this, so when your name is delivered you will write down how long it took (yes, you know how to write digits). To aid you with that, I (the exercise facilitator) will display a big stopwatch on the projector.

So the ordering process is:

1. Send your card to the Developer, tell him your name.
2. Wait for the Developer to write your name and deliver it.
3. Check the timer and write down the delivery time on the card.



If there is a bug (such as a mispelling) then don't log the time yet, instead send the card back for correction. The delivery is not considered done until it is correct.

You can talk to the Developer and answer questions. You just can't write letters.

Name tags and business cards and such should of course be put away for the duration of the exercise.

Step 4: Describe the Developer Policy for round 1

Here's what I tell the Developers:

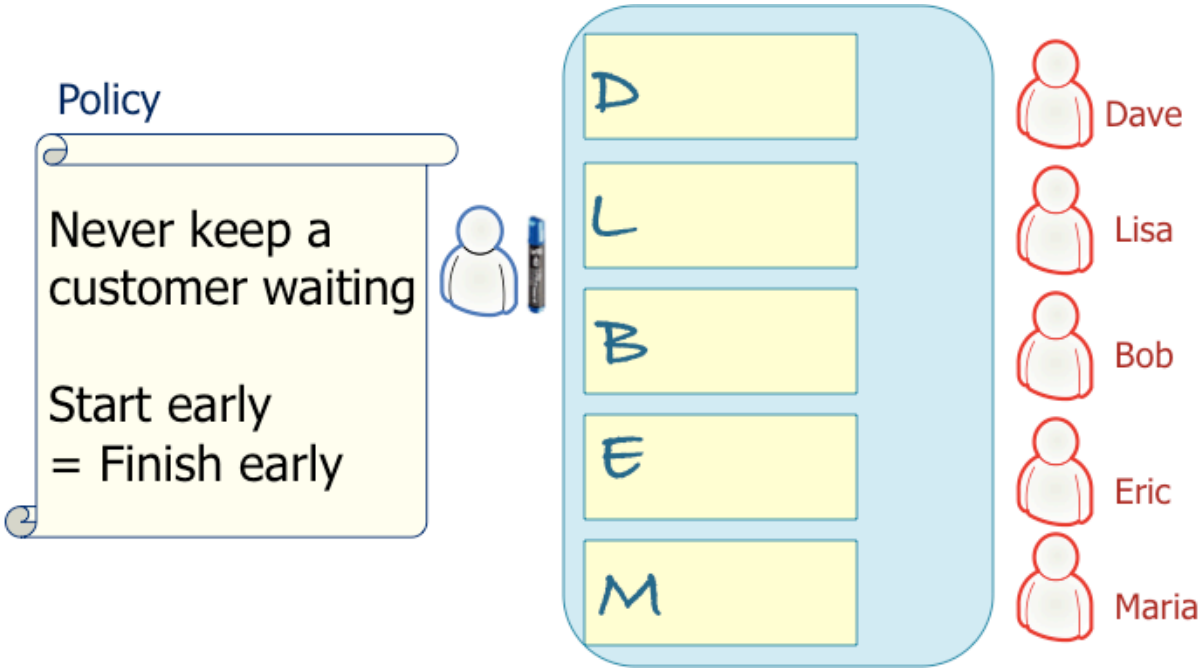
Your job as Developer is simply to write the customer names. However, you have to follow Corporate Policy!

Our Corporate Policy is *Never keep a customer waiting*, because that's bad business. We might lose a customer who has to wait. Furthermore, we believe that *the earlier you start something, the earlier you finish*. RIght?

> *At this point I paus and check carefully to see if anyone seems to object to that statement. Usually nobody does, because it sounds obvious. One of the purposes of the exercise is to illustrate how false that statement is.*

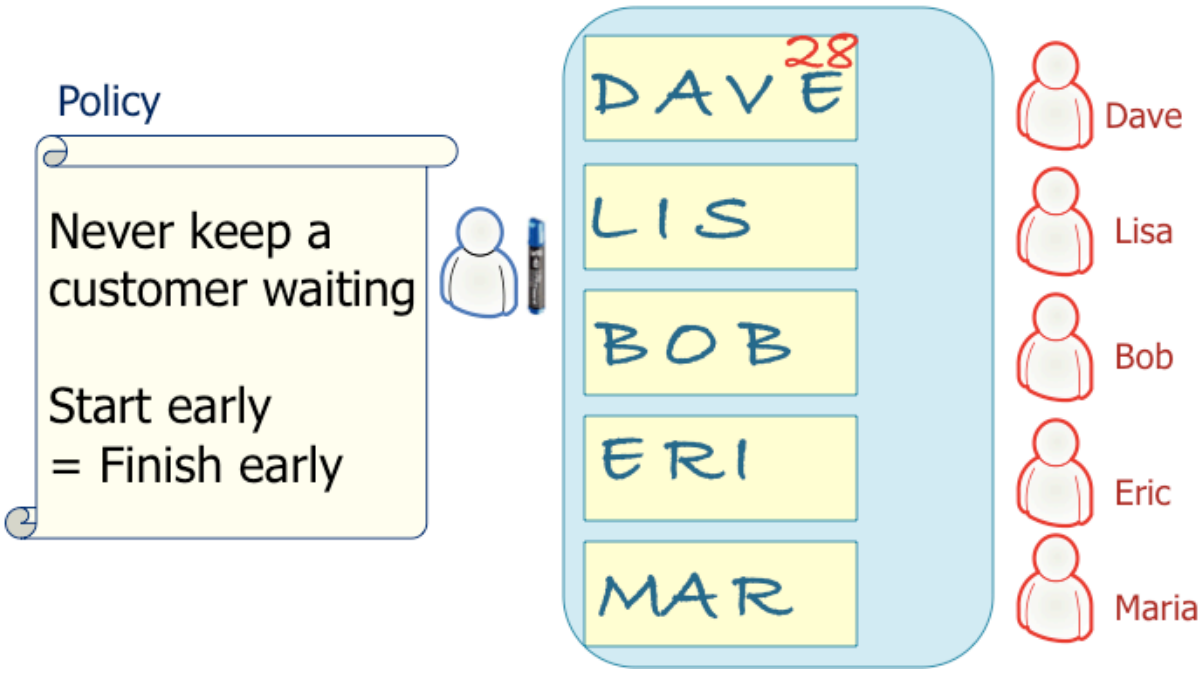
So, to fulfill this Corporate Policy, you as Developer must execute all projects at the same time!

When I start the stopwatch, all Customers will simultaneously hand you their blank card and tell you their name. You will write the first letter of the first customer, then the first letter of the second customer, etc.



When the first letter of each customer is written, go back and start writing the second letter of each customer. Etc.

Whenever a name is finished, deliver that card to the Customer so that he/she can write down the delivery time on the card.



There, that's the first round. I check carefully that everyone has understood the instructions, that everyone has a pen, and that each customer has a blank index card.

Then I start the clock.

Step 5: Execute the first round

At this point I observe the developers to make sure they are following Corporate Policy (usually no problem, as long as I described the policy well).

People usually have a lot of fun doing this, because everyone can see how bad it is and most people recognize the situation from their own projects.

There is a lot of communication and confusion going on, as the Developer has to repeatedly ask the Customers for their names and (in some cases) spellings.

Sometimes the Customers micromanage the Developer by verbally feeding him the individual letters. That's fine, I allow that since the results will be more or less the same.

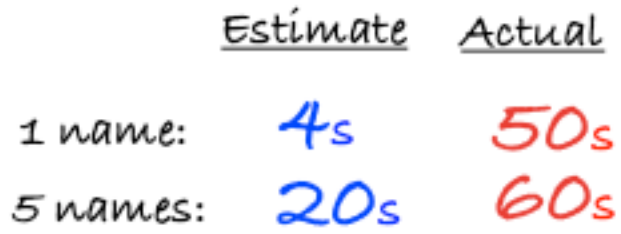
I check and (if necessary) remind the Customers to write down the delivery time as soon as they receive their name card.

When everybody has stopped writing I stop the timer.

Step 6: Capture metrics from first round

I ask each table to report the median of the delivery time ("if you order the cards by delivery time, what does it say on the middle card?"). I write down the median of the each table's response (i.e. the median of the medians...). Typically it is 50 seconds or so.

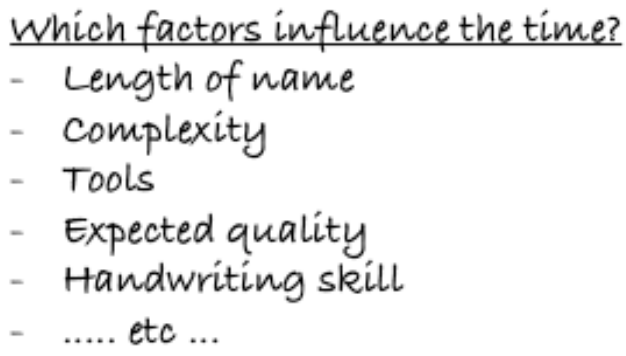
I also ask each table to report the total delivery time, i.e. how long it took to get all 5 names delivered, and I write down the median of all those numbers. Typically 60 seconds or so.



Time for a fun little rant. "You estimated that each name would take about 4 seconds to deliver. But it took more than *12 times longer*! I mean, some project managers secretely multiply all estimates by PI, but in this case even multiplying by PI *twice* wouldn't get us close to the real delivery time! "

And then the killer question:

"So, please enlighten me. Which of these influencing factors caused this incredible delay?" (pointing to the flip chart)



"Did you all have incredibly long names?"

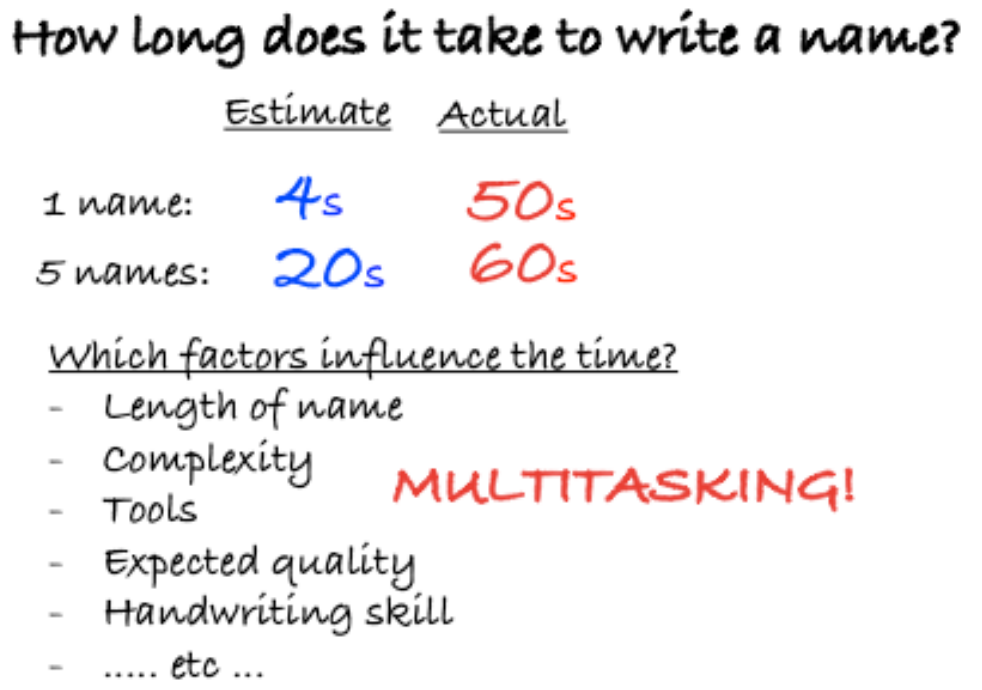
"Were the spellings very complex?"

"Did your pens not work properly?"

"Did you write beautiful caligraphic letters, or hack them into a stone tablet?"

"Did we just have really crappy developers?"

After some laughs and discussion the obvious is stated - the problem had nothing to do with those influencing factors. The list is missing the most important influencing factor of all - multitasking! I add it to the list, with emphasis!



The other influencing factors pale in comparison.

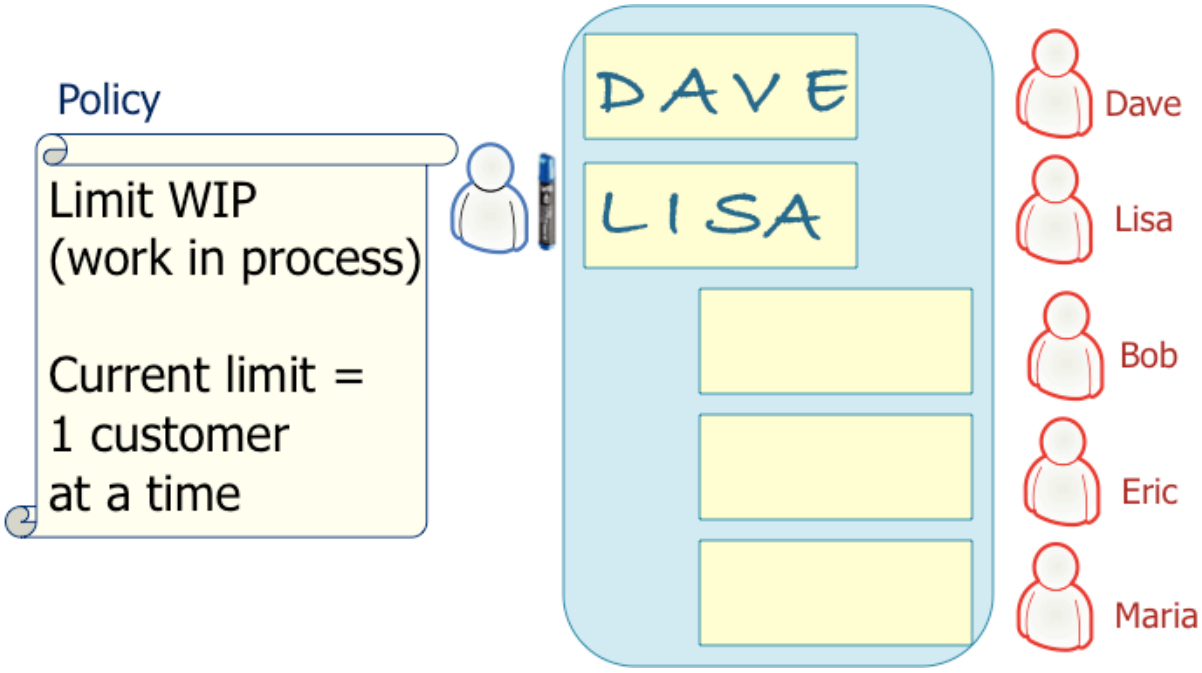
This was the simplest possible project - just write a name on a piece of paper! Yet, multitasking caused us to blow the delivery time by 1250%. Not only that, it also reduced our *productivity* to one third of what we expected. We thought it would take 20 seconds to write 5 names, but it took 60 seconds!

Time for next round.

Step 7: Rotate developers and introduce the second round

I ask each developer to rotate to the next table, so that each table has a new, fresh developer. I tell them that they have started working at a new company, with a completely different Corporate Policy.

The policy of this company is *Limit WIP* (Work In Process). And the current limit happens to be 1.

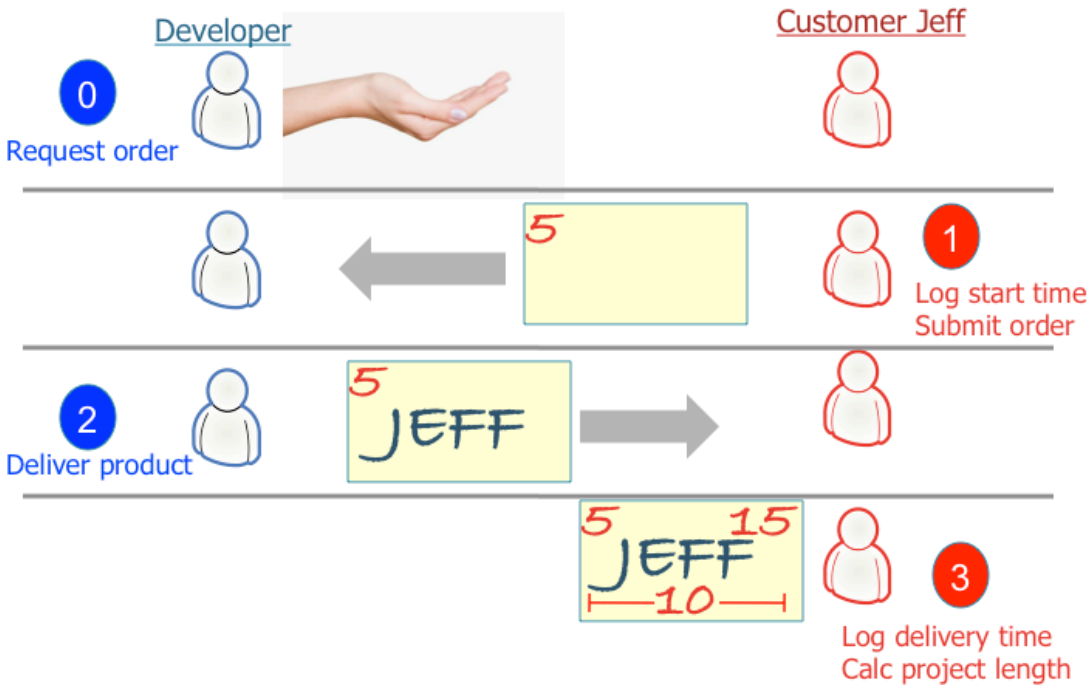


That means the Developer is only allowed to work on *one customer at a time*. So he won't start writing Lisa until Dave is done. And poor Maria will have to wait the longest, her project won't be started until all other customers are finished.

This has a very important implication. The Developer is now in control of his input stream! The Developer decides when to start a project (he "pulls" projects in), instead of the customer (who previously "pushed" projects in).

So now we have a *pull system*, rather than a *push system*.

This adds a new step to the beginning of the ordering process:



The Customer is no longer allowed to push his card at the developer. Instead, the Developer decides when he is ready for the next project, and holds out his hand towards a Customer and asks for his card and order.

Now, since all projects no longer start at the same time, the Customer needs to note both the start time and the end time of this project, and calculate the project length by subtracting these two numbers. So he logs "when did I get my delivery" as well as "how long did it take, from start to finish".

OK. Ready to go. Once again, I make sure everyone understood the new instructions, and make sure the customers have new cards (and that the old cards are put away temporarily).

Then I start the clock.

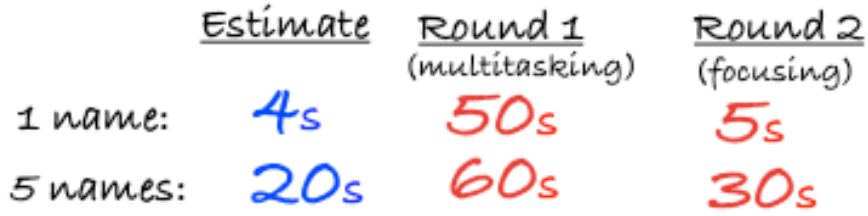
Step 8: Execute second round

Watch and enjoy.

Step 9: Capture metrics from second round

After everyone has finished writing, I stop the clock and gather statistics in the same way as in round 1. That is, the median time to get one project done (from start to finish), and the median time to get all 5 projects done.

The difference is dramatic. Typically something like this:



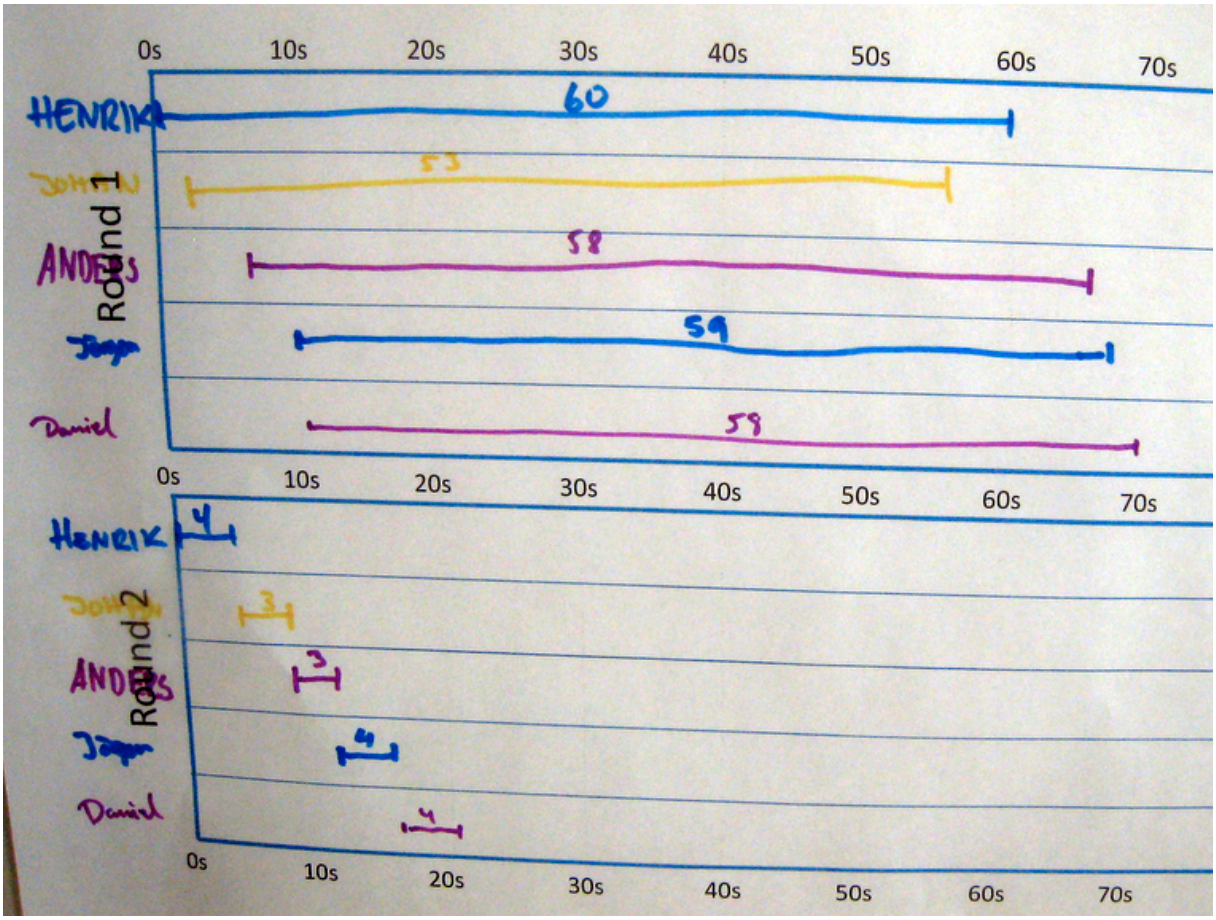
I point to these numbers, and ask about the difference between the two rounds.

In this case we executed each project 10 times faster (From 50 to 5 seconds). and since the total project time was cut in half (from 60 to 30 seconds) we also doubled producitivy. With 30 seconds to spare after Round 2, we could have delivered another 5 projects - for example 5 new customers, or another iteration on the first 5 customers (perhaps writing the last name as well).

That's how bad multitasking is, even for such a simple project as writing a name on a card. Now imagine doing something complex, like writing a software system, where context switching is *much* more difficult. This causes even further delay, quality problems, and stress. In a real project the scale might be weeks instead of seconds, but the proportions are often similar.

Looking at the original estimate, we can see that in Round 2 the numbers are fairly close to what we originally thought.

I also show this graph, showing a very typical result. If time permits I have each table draw their own graph.



The graph provides a nice visual representation of the cost of multitasking. I ask people if the shape of this graph looks similar to their own numbers, and it usually does.

At this point I facilitate a discussion. Here are some examples of typical questions that I ask, and typical answers that I get (or offer myself):

How did these two rounds *feel* to you as Developer or Customer?

- Round 1 felt stressful and confusing for both parties, Round 2 felt calm and focused.
- As customer in round 2, it felt nice having a focused discussion with the developer while he was working, and then be done. In round 1 the developer kept coming back and repeating questions like "what was your name again? Was that with K or CK?"

What was it like being the last customer in round 2, compared to in round 1?

- Felt fine having to wait, because I could see the other projects getting done very fast and I could see my turn coming up. In round 1 my project got started quickly but I had no sense of when it would be done.

What do think about the statement 'if we start early, we'll finish early'?

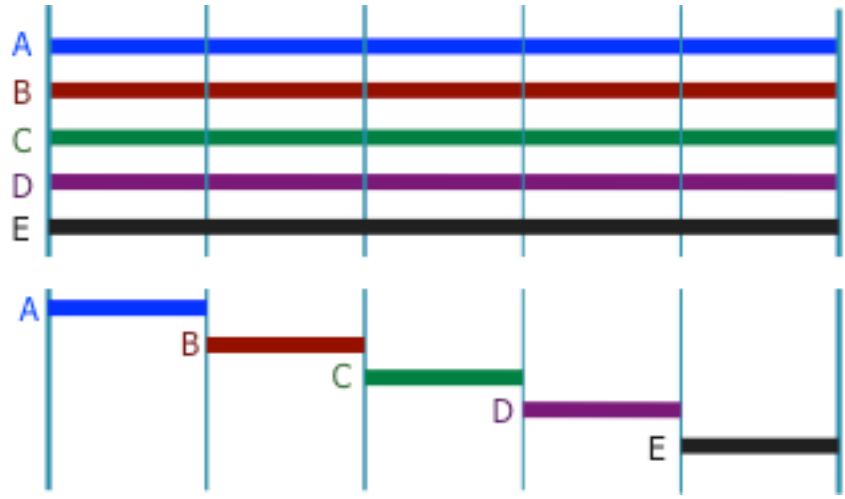
- Busted. This simulation shows an example of the opposite. In Round 2 every single project (except the first) started later, and yet every single project finished earlier. In fact, in Round 2 the projects finished earlier *because* they were started later. Weird but true.

How does this influence release planning? For example, what do we know after 10 seconds in round 1 vs round 2?

- In round 1 after 10 seconds we have no idea when anything will be done. In round 2 after 10 seconds we have already completed and delivered 2 projects, so we can reliably estimate when the next projects will be done.
- In round 1 estimation is basically a guessing game, even if all the other influencing factors (length of name, etc) are known in advance, because we never know when yet another project might be added to the fray.
- In round 2 we could estimate fairly reliably (compared to round 1) even if one of the subsequent names is long and complex.

What would have happened if the developer was perfect at task switching?

The 30 seconds of productivity loss was because of task switching (time "lost" switching from project to another). But what if task switching was free, i.e. the developer can jump back and forth between projects without losing any time? In that case the difference between the two rounds would look something like this:



The total time to complete all projects is the same - so no productivity is lost in this scenario. But *each individual project still takes 5 times longer* when task switching! That's because of Little's Law, which basically says that if you do X things simultaneously, then each thing will take X times longer.

So don't focus on hiring good multitaskers. Instead, hire people who hate multitasking and focus instead on creating an environment that minimizes the need for multitasking - through WIP limit policies etc.

How does this influence product quality?

- If the developer misunderstood the customer needs (for example mispelled his name), it would take 50 seconds to find that out in round 1, and just 5 seconds in round 2. Furthermore, the increased productivity in round 2 means we have time to iteratively improve the name if necessary. So product quality is dramatically affected by multitasking.

Does this multitasking problem feel familiar? Who is experiencing this right now? Who has experienced this in the past?

Just about every person raises their hand here...

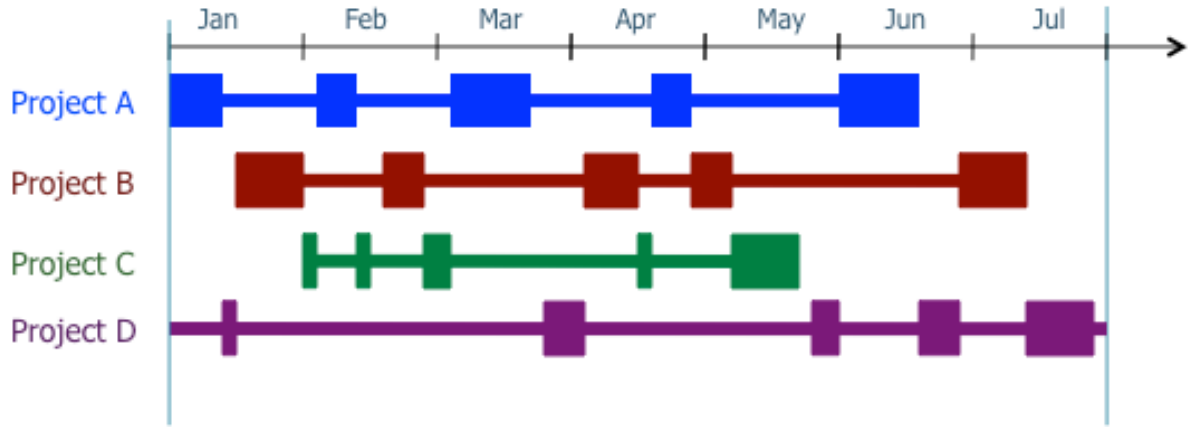
What causes us to do this? Why is the problem so common?

- We are eager to please, and it is easier to say "yes" to a new customer than to ask him to wait.
- In Round 2, while we are working on customer #1, when customer #3 arrives at our door and asks if we can start his project, it feels weird to tell him "We will start your project later, so that we can finish it earlier". Even though it is true.
- We focus too much on starting things rather than finishing things. For example by assigning sales bonuses for signing new clients, rather than for delivering.
- We sometimes try to "lock in" customers by starting their project early, to reduce the risk of losing the customer. But this is a lose-lose proposition. It might work in the short term, but if there is intelligent competition in the market this strategy will backfire in the long run. The market will sooner or later figure out that company A does projects 10 times slower and twice as expensive as company B.

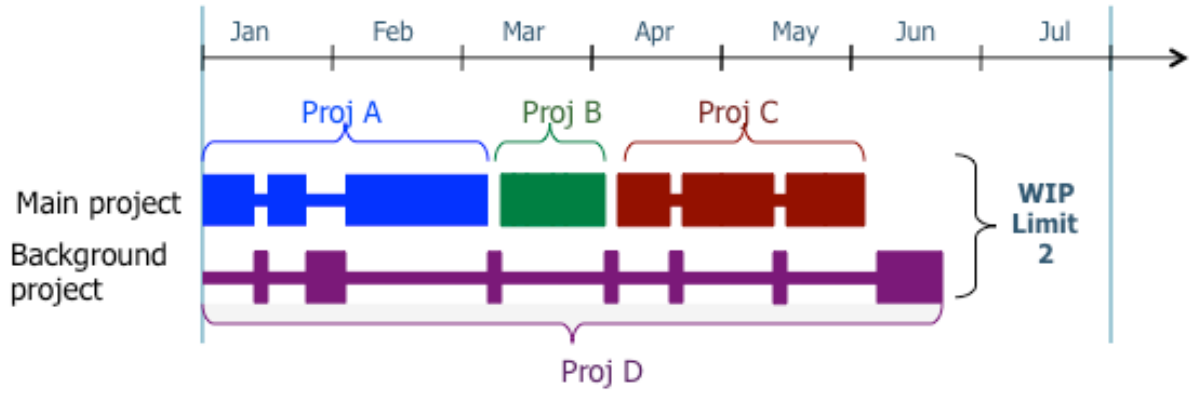
When is multitasking a good idea? Who benefits in round 1?

- Multitasking is almost always a bad idea. Nobody benefits in round 1.
- Limited multitasking can be useful in situations where Project A is blocked because we are waiting for something, so we go work on Project B in the meantime. We need to limit this though, for example to 2 simultaneous projects. That way we create pressure to remove impediments rather than keep starting new projects.
- An important learning point is that although WIP limits (work in process limits) are important, the limit doesn't necessarily have to be 1. Any limit is better than no limit.

Here is an example of a team working on 4 projects in parallell. The thick areas show when actual work was being done on each project.



Here's what happens if we limit WIP to 2 projects - one "main project" that we focus on, and one "background project" that we work on only when the main project is blocked. In the main project we have release plans and commitments, in the background project we don't .



What can you do to combat this problem in your environment?

You can do a number of things, regardless of your role in the organiation.

- Visualize the problem. Collect metrics from existing projects, draw the above chart.
- Measure multitasking. Ask people to list how many projects they are involved in. Or have a stickynote on your desk where you jot down a mark every time you are forced to context-switch. Aggregate this for the whole team and discuss how you can reduce it.
- Run the multitasking name game with your colleagues, managers, customers, etc.

Keep in mind, though: the first and most important step is getting people to agree that there is a problem. Only then can you effectively solve it. I hope this article will help!

Scaling this simulation

The simulation works best with a small class (10-20 people or so) grouped around small tables, since that allows for proper discussion and debriefing. However, it works with groups of hundreds people too. If people can't create small groups around tables (because you lack tables, or because it is too crowded), then you can do the simulation on stage instead.

For example, invite two groups of 6 people to come up to stage. Place each group in front of a flip chart. Instead of using index cards to write names, the developer writes on the flipchart instead, and the customer writes the delivery time next to his name as it gets done. Other than that, the simulation is run in pretty much the same way.

It works to invite just one group too, if you only have one flip chart. The only disadvantage is that you won't be able to swap the developer as easily.

Variants

This simulation can of course be extended or adjusted to illustrate other things. For example what happens if the customer changes his mind in the middle of the project, or decides (after delivery) that he actually wants it all in capital letters, or green letters? What if we add cash flow aspects such as payment on delivery, or return on investment?

I haven't experimented much with these things though, so far I've focused strongly on the multitasking aspect of this simulation.

Facilitator FAQ

During the initial estimation, isn't it better to write an estimation range (4-6 seconds) instead of just one number?

Technically, yes. Estimation ranges are better because they convey the uncertainty level. But all those extra number complicates the exercise by making the flip chart harder to read. So this is an intentional simplification.

When collecting the results, why do you just write the median?

I used to write the full range of results on the board, but that became misleading. Often 10% or so of the participants get a really weird result, for example because they have an incredibly short name, or because someone dropped a pen, or because the developer misspelled the name and had to rewrite it, or because the customer wrote down the wrong time. The median gives me the most relevant number.

I also found that every piece of additional detail on the flipchart dilutes the message, as people focus more on understanding the numbers instead of focusing on the lesson learned. So I keep it simple.

I use median instead of average because it is faster to figure out, and no calculator is needed. The median of [12,20,24,25,50] is 24. The average is... uh... can't be bothered.

Why names? Why not fruits, or colors, or something else?

Names are useful because every Customer has one (I don't have to give it to him), and there are sometimes non-obvious requirements (special spellings, etc), which is a nice source of customer-developer conversation, and possible bugs. It also feels more "real" with names, people can relate to "I need someone to print my name for me", more than "I need someone to write the name of this fruit".

And, as a positive side-effect, people learn each others' names :o)

Has the simulation ever backfired?

Not that I can remember. Once in a while I have a course participant who insists that the first round represents a better sales strategy, and that locking in the client is a good idea. But that usually leads to strong protests from the other participants, and some very interesting discussions about short term vs long term thinking, so I wouldn't call that a backfire.

Once in a while one of the groups will misunderstand the instructions and not follow the Corporate Policy in round 1, which messes up their statistics. But then I just focus on the numbers from the other groups. Or redo the round.

Sometimes there will be a developer who is exceptionally good at multitasking, that can dilute the results a bit. But it hasn't been a major problem so far.

Sometimes a customer will get overly enthusiastic and start role-playing as "difficult customer", changing the requirements and being overly picky about quality. If I see that happening I will politely ask him to Play Nice.

Sometimes somebody will complain that the simulation is overly simplistic, and that this multitasking effect doesn't apply to "real" projects. The rest of the class will usually provide strong counterarguments to that, using examples from their own experience. Quite fun to watch.

In the end, I remind the participants that this is just a simulation, it is by definition artificial. It is up to each individual to decide what to learn from it and how this applies to his/her own context.