NO LIMITS LEADERSHIP, INC.

Accelerating Projects Through Leadership

The Five Diseases of Project Management

Why your projects are late and what to do about it

Authored By:

Allan Elder

How can you finish more projects, faster, without sacrificing quality or content, when your resources are already over extended?

Do your projects suffer from these undesirable effects?

- late,
- resources overloaded,
- excessive changes (due to long project timelines),
- resources not available when needed (even when promised),
- changing priorities, rework

This paper defines the five human behavioral reasons your projects are late and if you don't address them, will continue to be late.

We have discovered through years of practice and research that projects suffer similar fates. Looking through our library of over one hundred books on project management and leadership we discovered that our oldest and newest book on project management identified the same complaints. For over 50 years projects have suffered from the same effects. Why? What could be causing such failure, universally, for so long? In case you believe where you work is different we also discovered that it does not matter what kind of projects you manage. Projects from the DOE, DOD, IT, Construction, and dozens of other fields suffer an identical fate.

Five reasons your projects struggle have been identified:

- Bad multi-tasking,
- Student syndrome,
- Parkinson's law,
- Task dependency, and
- PM math where 2+2=5.

You should be concerned about these five causes because they delay project benefits and results to your company and your clients. These causes also delay the cash flow of a finished project and allow your team and client to encounter a longer opportunity window to make changes that threaten the project itself. Imagine the reduction of change requests if your project were to complete twice as fast. Remember, if you keep doing what you have always done you will keep getting what you've always got, late projects.

REASON #1: Bad multitasking

Do you or your team constantly face shifting priorities that cause you to stop one task and work on another? Is someone waiting for the output of your task before they can do their work? This is the definition of bad multitasking. With this said, not all multitasking is bad. When nobody is waiting for your output there is nothing wrong with switching between various tasks.

Why do we multitask? For some of us it is the boredom of working on one thing at a time. Our mind demands higher stimulation and therefore we continually shift subjects. Often the culprit is poor prioritization. We are asked to start several tasks simultaneously and each of them has a "customer" waiting for the output. Each customer wants progress to be made on *their* task and constantly asks "is it done yet?" forcing us to repeatedly switch to their task to get something done and report progress. While working on this task other customers request status on their respective tasks. This cycle forces us to task switch repeatedly. Naturally, while working on one task you are not making progress on any others. If your customer is counting on quick delivery they will take their business elsewhere. For some customers progress alone is sufficient. It doesn't

have to be fast as long as it is being done. However, even if you are fortunate enough to have such customers what is the impact on you and your business?

Any amount of time not working on a task means the task is being delayed longer than if you dedicated yourself to its completion. Therefore, multitasking *always* makes a task take longer than it should. Other factors that add up include the thinking time it takes to "get in the groove" to become creative. For tasks such as engineering, programming, and writing, this time can be a significant part of the total task time when multitasking. For manual labor it may include machine setup time, getting the right tools and equipment ready and putting them back. There are some tasks where the setup time is negligible and not a factor but these are few in the world of knowledge work. Estimates indicate that setup, or think time, can equal or even exceed actual task time for highly cognitive tasks. An example includes writing this paper. When I walk away from it to do something else and then return I must read the entire segment over again to discover where I was and what I was thinking when I stopped. This takes extra time, time that could have been devoted to more writing.

Consider how multitasking would effect you at the grocery checkout line. Imagine if instead of doing one person at a time, the checkout process included scanning one product for each person in line and then repeat. If there was one person in line the time required to check out would be only the time required to scan your products, take your money, and bag your groceries. However, if after you get in line and one or more of your items were rung up, another person gets in line, instead of completing your order, the cashier takes one item from the new person, rings it up, and then takes one item from you, rings it up, and repeats. Now it would take twice as long for you to complete your purchase. While the cashier is ringing up your items and simultaneously the person behind you another person gets in line. Now the cashier takes one from you, then one from the next person, and then one from the new person, and repeats. What happens when another person gets in line? As you can see, the more people that get in line the longer it takes for you to complete your transaction. Would you shop at this store more than once? No. So, why do you do this to your team and your customers? The fastest way to complete a transaction is to start it and do it until done. You can then concentrate on the task and the customer. It's faster and provides better customer service. Nobody complains unless the line gets too long. When task switching, the risk of quality problems also escalates. We forget what was done and what was not done. We rush so we can return to another task. We overlook small details in our setup. The pressure from irate customers adds stress to the job making us less satisfied and prone to neglecting good service. Management involvement increases to deal with “important” or highly impatient customers and expediting begins to become a way to deal with them. As more and more customers begin demanding faster service management begins to focus on complicated prioritization methods to satisfy everyone and keep employees focused on “doing the right thing.”

Bad multitasking forces people to give longer task duration estimates than necessary. If you know that you will *not* be permitted to start a task, work on it until done, and then go to the next task, you will be forced to give a much longer estimation of how long it takes to complete the task. If you know it would take you two days to complete a task but also know that you will be interrupted, you will include the interruption time in the estimate. Now, a two day task is estimated to take 10 days. Will your customer wait ten days? You decide. Would your customer be more motivated to do business with you if you promised two days instead of ten? Imagine the competitive advantage to the shorter duration promise. Imagine the improved work environment created for your team members when you eliminate the complicated prioritization methods, the constant expediting, the angry customers, and the constant management oversight.

Not all multitasking is bad. How do you know the difference? Remember, bad multitasking is when a task is being delayed and the person you owe the result is waiting for you to finish. In effect, bad multitasking can delay the completion of the entire project. However, if nobody is waiting for the result then it is not bad multitasking. It may not be efficient, but it may not matter. For example, you are asked to stuff 100 envelopes and put them in the mail for tomorrow. Since the mail only runs once a day and it's too late to put them in today it would not matter if you folded all the papers first and then stuffed each one (multitasking) or folded

one paper and stuffed it in an envelope and then moved on to the next one (no multitasking). However, if the mail carrier is due any minute and you have a critical item to send it would be unwise to multitask simply to be efficient. In this case the task requires folding, stuffing, stamping, and mailing without interruption.

Some jobs are multitasking driven. Not every job should eliminate multitasking. For example, a secretary, a chef, and many others are required to have many moving parts. Is this bad multitasking? Not necessarily. Due to the very short nature of the task, and time delays built into the task itself, the delay is usually tolerable. You may need to answer several phone calls and put people on hold. This delay is usually not a problem until it becomes excessive. Eventually, the customer puts a cap on how long you can permit multitasking delays before they hang up and take their business elsewhere. We would all like to have our call answered immediately, the problem addressed without going on hold, and then conclude the call. However, due to the short duration of the transaction we are willing to accept a certain amount of delay.

Task simulation demonstrates the effects of bad multitasking. In actual project simulations (using Tony Rizzo's three project simulation) performed in my seminars students perform three projects while multitasking. The effects created during this exercise are chaos, confusion, lots of commands, and many additional "management" activities. It is very stressful. When they are done I challenge them to do twice as many projects, six total, in less time than the original three. They never believe it is possible. They also don't want to face that much stress. However, once we remove the bad multitasking they always do twice as many projects in less time and without chaos, no yelling, and far less stress. The only difference between the two events is bad multitasking. When your team is multitasking it requires considerable management overhead. Someone must keep track of what is being worked on, the status, estimated completion time, and update the customer repeatedly. Every bit of this overhead can be eliminated. If this sounds like your experience you have the most to gain from removing this obstacle. If you desire to get twice as much done in the same amount of time while reducing stress, stop bad multitasking. The figure below provides a graphic example of the results of multitasking vs. not multitasking.

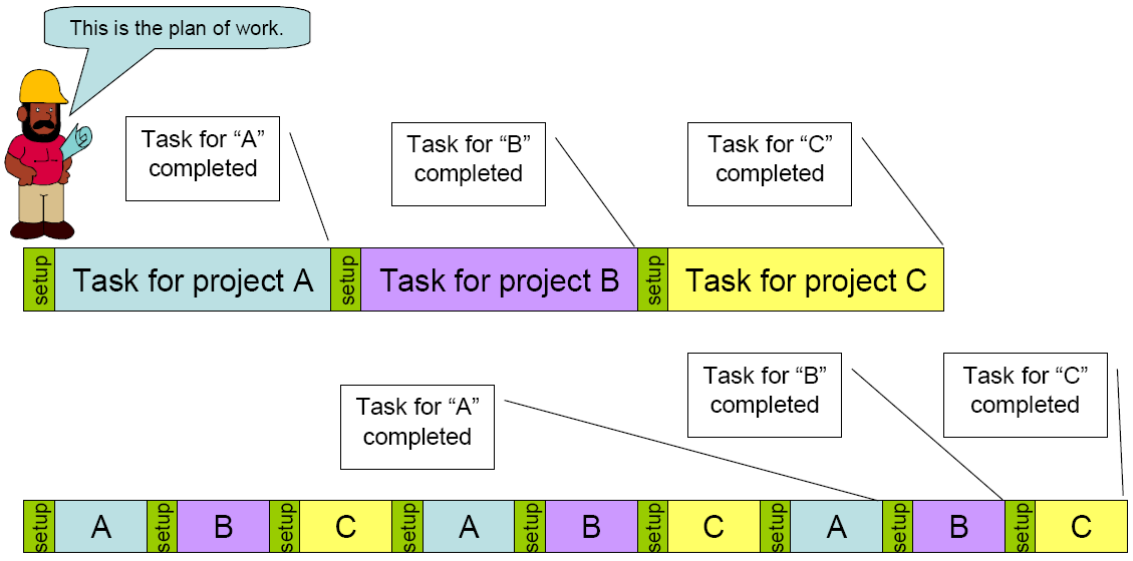


Figure 1

In Figure 1, notice the early completion time due to removal of setup (think) time. Also notice the delivery time of each task. Even if the two scenarios took the same overall amount of time (zero setup/think time), the advantage to not multitasking is significant. If someone is waiting for the results of Task-A before they can do their task it is easy to see that without multitasking the next task can begin considerably sooner. In addition,

notice that when multitasking the three tasks complete in rapid succession. If the results of all three tasks are going to the same resource the recipient now has inherited the burden of multitasking. This creates a pile of finished work that moves down stream as well as excess work in progress. In addition, time may have been wasted while the next resource was waiting for the output. They may have been engaging in "busy" work to make sure they were not caught with nothing to do (and risk being "right-sized"). The time spent on busy work did not move the project forward and may have contributed to delays in the project.

I am often asked about "dead time" during tasks. "Am I supposed to sit around doing nothing when I reach a point on a task where I am waiting on others?" No. Remember, it's only bad multitasking if someone is waiting for the results. That is, someone else is waiting for your output to do their job. For example, if you are cooking and you put the roast in the oven you are waiting for the oven to complete its task. You are free to move on to other tasks while you wait. However, you must be ready to immediately resume the previous task when the oven is done. Otherwise, you will burn the roast.

REASON #2: Parkinson's Law

Bad multitasking causes team members to embed greater safety in task estimates. Let's further examine the topic of task estimates and evaluate the validity and damage done by inflating the amount of safety in duration estimates. When a person is assigned a task, one of the first questions asked is, "How long will it take you?" Would you agree that people have a tendency to include "protection" in the duration estimate to accommodate outside factors such as "Murphy" and multitasking? Consider a recent task that you estimated. What level of certainty did you offer? Was it 100%? Not likely, something could occur that prevents you from ever finishing the task, such as death, and therefore 100% certainty is not attainable. Was it 10%? How about 50%? Even at this level of certainty you would deliver late 5 out of 10 times. Perhaps your offer was closer to 90%. This would mean that 9 times out of 10, you're on time. Does that seem reasonable? What level of certainty do you demand from your team? Do you demand their estimates to be accurate every time? Or, are you okay with them being late 5 out of 10 times? Most managers want people to provide "accurate" estimates which makes no sense because an estimate, by definition, is only an approximation.

Intellectually we understand that estimates will not be exact. Yet, we demand it. Why? There is an underlying belief that if we can accurately determine the time for each task, and make sure each task is finished "on time" then the project will finish on time. However, we also know that the goal is not to finish each task on time but to complete the project on time. The reality of project work is that uncertainty exists and therefore task time cannot be determined, only estimated. The result of demanding accurate estimates is that duration estimates are converted into commitments. Therefore, to provide a realistic estimate we must account for all of the things that could impact task duration by embedding safety.

When a "small" safety is added to estimations they are not considered as unreasonable because mentally we add the safety considering a normal distribution of time. In a normal distribution 50% of the time is to one side of the average and 50% of the time is to the other side of the average. Therefore, moving from a 50% accuracy to 80% does not appear to be significant (see Figure 2). However, task times are not "normal." In fact, there is no such thing as "normal." The normal distribution occurs only in

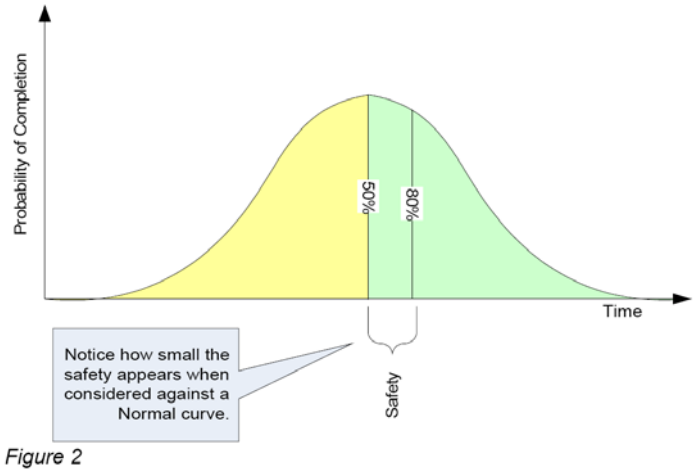
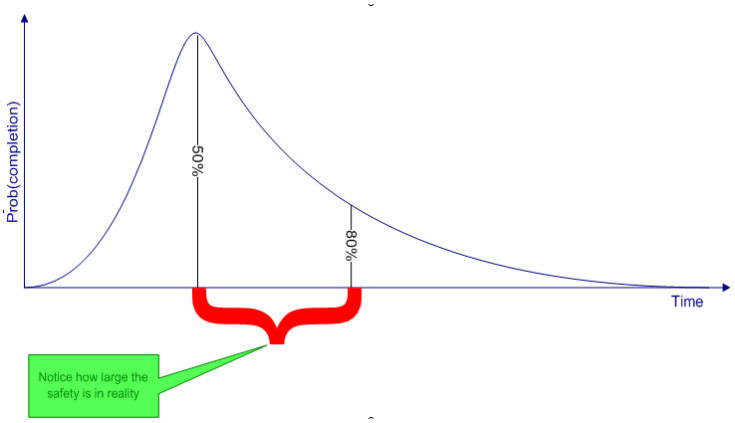


Figure 2



mathematics, not real life. Due to the Central Limit Theorem, if we include enough samples, eventually, it will provide the "normal" distribution. A silly example is that if I put one foot in a bucket of boiling water, and one foot in a bucket of ice water, on average, I should be comfortable. Task times do not follow a normal curve. Instead they start somewhere beyond zero (every task must take some amount of time) and then the probability of completing as promised ramps up quickly only to drop off with a very long tail (see Figure 3). When you compare the two graphs you see that the "small" safety embedded is in reality quite large. The higher the uncertainty of the task the longer the tail grows. **The result is a task with approximately half of its duration estimate being safety.**

Is the individual team member the only one to add safety? Never. The manager then takes the safety estimated tasks and adds his own safety. In addition, his manager adds her own safety. For every level of management more safety is added (see Figure 4). This additional safety unnecessarily extends the project completion date and does not, in fact, protect against uncertainty. Sometimes another disease occurs, that is everyone inflates their estimates because they know that the layer above them will cut it. Since the resource does not know how much the arbitrary cut will be they guess how much to inflate the duration. In addition, since the next level up has no idea how much safety was added they cut it by guessing how much safety may have been added. This is entire process is ludicrous.

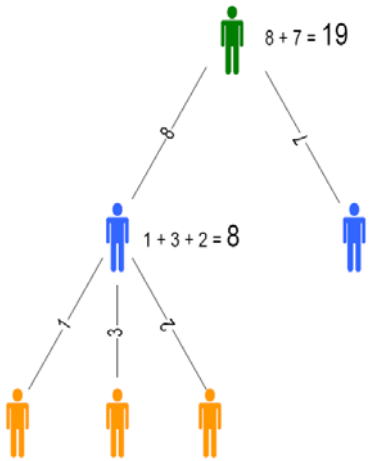


Figure 4

If there is so much safety embedded in each task, why do tasks continue to deliver late? Shouldn't they all deliver on time or early? If you agree that safety is embedded to account for the unknown (Murphy) and Murphy does not strike *every* task as predicted then most tasks should not finish on time. They should finish early. Only an occasional task that had everything imagined during estimation, and then even more bad luck, should finish late. If tasks are not finishing early most of the time then the safety is being wasted. Even a task that delivers on time is unacceptable since nearly half the estimated time was reserved for events that never occurred (Murphy).

Even more remarkable is that people strive to comply with the unreasonable request to provide accurate estimates. Why is this? Would you agree that most people want to be considered reliable? If so, doesn't this mean that we try to meet our commitments? If this is true the result is we add safety and we struggle to prevent that safety from being removed by others. If we have extra safety built in, that was not needed, then we use that extra time to do a better job rather than report an early completion. After all, if we demand 6 weeks and deliver in 4 weeks what will be "reward" for delivering early the next time we ask for 6 weeks on a task? Our safety will be cut. This may cause us to miss our commitment, thereby causing us to be late and therefore unreliable. This phenomena is so prevalent that it has its own name: Parkinson's Law. This Law states that "Work expands to fill the time available." By now you should agree that people do add safety but that safety is not being used appropriately. Your proof is that most tasks do not deliver early as would be expected.

REASON #3: Student Syndrome

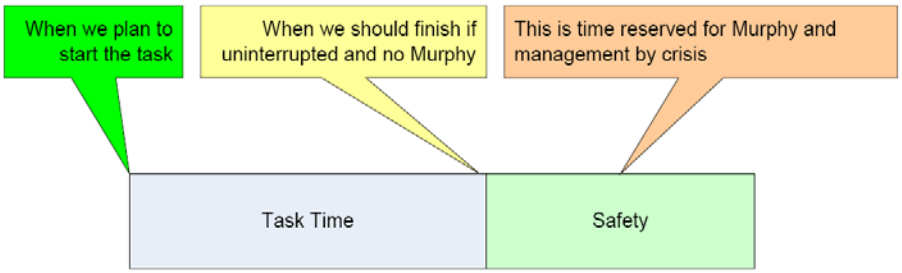
Student Syndrome is also known as procrastination. The big difference is the reason for putting off the work. Procrastination is being lazy or irresponsible. Student syndrome is a natural defense mechanism. It means to put off the work until the last possible moment not because we are lazy, in fact we are working very hard. We all fall prey to student syndrome on occasion. Student syndrome got its name from how students handle homework. Imagine your professor tells you that you have a final exam in 19 weeks. He gives you all the material, the book, the objectives you will be tested on, and the date. When do you begin studying? The night before the test. Why? You have time, that's why. Other tasks are more pressing and therefore you delay starting a task until the last moment to allow yourself time to complete other work, most likely also being done at the last moment.

Often people say they cannot estimate with any accuracy how long a task will take. I disagree. The evidence for this statement is most people do know the last possible minute they must start a task or risk being late. When was the last time you put something off until the last minute and was able to choose the actual last minute? You work all night to complete the task and print it out right before the big meeting. The paper is still warm upon delivery, but you made it. This problem becomes more serious when we consider the implications on quality. Yes, you did choose the last possible moment to complete a task but what was the potential consequence on quality? If anything does go wrong there is no time to fix it. How often do you miss choosing the last minute on important tasks? I suggest it is rare. I know this from personal experience. Therefore we can conclude that most people do know how long a task will take (given some probability) when they can dedicate themselves to it. If you want more accurate (but never accurate) estimates identify the delivery date and ask yourself, “If I started this task at the very last minute, when would it be?” You now have the task duration estimate.

The difficulty of task estimation is not knowing how long the task takes but guessing how many other factors must be accounted for since you know you will not be permitted to work on the task without interruption. If people are still not sure how long a task will take it is most likely due to not knowing when they will get all of the required inputs to really start doing the task without having to stop and gather more information. This is not a problem. Remind the resource providing the estimate that you are only interested in how long it will take them to complete the task without interruption and with all necessary inputs available. Then document the inputs they require to ensure they are not asked to start the task until all such inputs are in their possession. Your job as the project manager is to ensure they have what is needed to do the work.

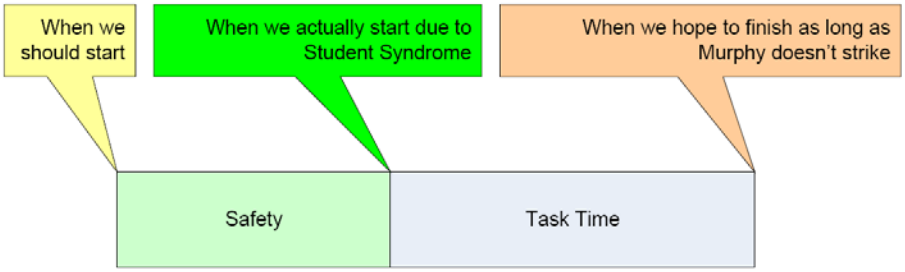
As long as you overburden people with tasks and allow them to inflate the time line to accommodate other tasks you perpetuate student syndrome. This problem is magnified by multitasking. We have our team working on multiple tasks with inflated durations and expect them to prioritize those tasks. Urgent tasks will take precedence over important tasks. This encourages embedding safety in tasks, which provides more time than actually needed, so we delay the start of tasks until we absolutely must start due to competing demands. Now what happens when the task does encounter a problem? Where is the safety (a better question would be, "when is the safety?"). It is in the past. We can no longer use it (see Figures 5 and 6). Notice in the graphic that when we embed the safety we mentally place the protection at the end of the task. However, due to student syndrome we don't begin the task immediately and therefore start the task late. When trouble strikes the safety we counted on is no longer available.

Some people point out that all of this cannot be true. As a matter of fact, when my team gives me an estimate they almost always hit that estimate. This may be true. However, it is true because it is a self-fulfilling prophesy. The task falls victim to the before mentioned issues causing it to complete as predicted because people want to be seen as reliable. As you can see, if the task is started immediately and there are no serious problems the embedded safety is used up by Parkinson's Law. If we don't start the task immediately and there is a problem the task will be late. If we start the task late (student syndrome) and nothing goes wrong



How it looks in our head when we add safety.

Figure 5



How it actually happens

Figure 6

we complete the task "on time." Either way, the safety time was wasted. This is time that caused the duration of your project to be estimated for much longer than necessary. This also makes your project less competitive.

REASON #4: Task Dependency

On projects all tasks are dependent on other tasks. In his book, "The Mythical Man Month," Fred Brooks answered the question of how projects become late, "One day at a time." Have you noticed that if your project end date slips it is nearly impossible to catch up? Have you also noticed how easy it is to get behind schedule but how difficult it is to get ahead of schedule? If you do, then you understand the issues created from task dependency.

One negative effect caused by task dependency is explained in the following example. If you have a task that was estimated to take 5 days including safety, you started immediately, and completed the task "early," is the person that receives your output ready to use it *immediately*? Not usually. Therefore, if you deliver the results in 3 days the next person will not touch it for 2 additional days because they are not scheduled to start their task until that time. Now, the embedded safety is wasted even though the task was delivered early. To overcome this problem **you must have a project system that ensures all tasks begin, not when they are scheduled to begin, but when the required inputs are available.** This is especially vital with tasks on the critical path (or critical chain).

Another negative effect caused by task dependency is well known from probability theory called the "probability of dependent events" (also known by other names). This theory states that the total time required for dependent events, in terms of probability, is the product of the probability of all dependent events. Here is how this impacts you (see Figure 7). If you have three tasks that are dependent on each other and each has a 90% chance of being done on time what is the probability of all three completing on time? About 73%! We must calculate the probability of finishing Task-1 (90%) and then calculate the probability of finishing Task-2 given its dependency on Task-1 (90% x 90% = 81%). As you can see the probability of finishing Task-1 and Task-2 on time is now only 81%. We can then calculate the probability of completing Task-3 given its dependency on Task-1 and Task-2 completing on time (90% x 90% x 90% = ~73%). With only three tasks, each with a 90% chance of finishing as promised, there is only a 73% chance we will, in fact, finish all three as promised. It doesn't take many tasks to reach a zero probability of finishing the project on time.

You may be thinking, "I don't have this problem because I perform the tasks in parallel rather than in series." Let's examine this solution (Figure 8). If each task has a 90% chance of being done as planned what is the chance the entire project will be done on time? Your first reaction may be 90%. However, since the completion of the project depends on the completion of all of the tasks we must use the product of each

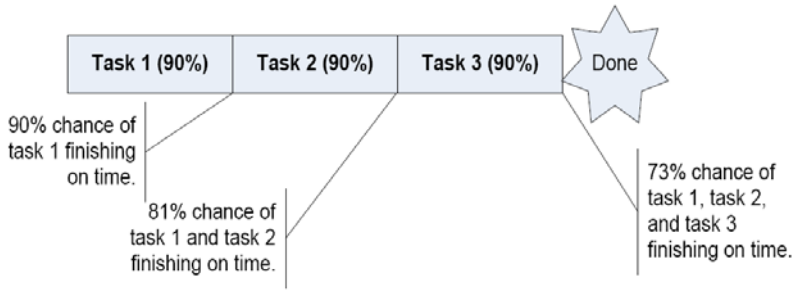


Figure 7

dependent event to determine the probable completion time. In this case we would multiply 90% for each parallel task revealing the same 73% chance of being done on time. Now you can see why trying to get every task done on time does not mean the project will finish on time. Every dependent task will be required to finish as planned in order for the project to finish on time. This effect has caused project managers to conclude that the only way to finish a project on time is to ensure every task, does indeed, finish on time. Such a solution would require that every task be estimated with 100% accuracy. Even if this were possible it would lead to task times of unimaginable duration.

To better understand how delays are passed forward and early is not, consider Figure 9. This project is intended to take 17 days. How much earlier would this project finish if the first task were five days early? 17 days. The reason is we are dependent upon the completion of all five tasks. Therefore, even if one task is early the project will not finish any earlier. What if the first four tasks completed 5 days early? The project duration is still 17 days. Now consider the project duration if even one task is 5 days late. The project will take 22 days. As can be seen, early is not passed forward, late always is.

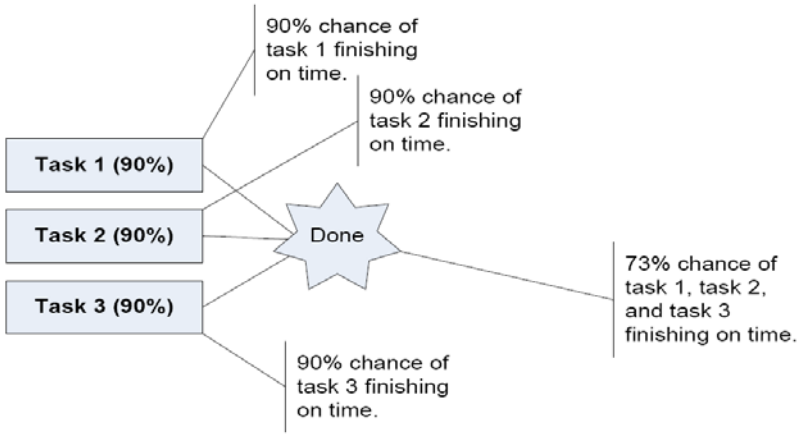


Figure 8

An additional dependency not discussed, and usually forbidden, in critical path methodology is resource dependency. In the following graphic (Figure 10), we have a project that includes task dependencies and resource dependencies. Note that tasks A and D both require the use of the "programmer" resource. To complete tasks C and D require the completion of tasks A and B. To start task D also requires the completion of task A due to the resource dependency. Given that each task has a probability of completing on time at 90%, what is the probability this project will complete on time? Only 73%! Although there is no arrow attaching task A to task D the dependency remains. With that dependency we encounter all the problems previously mentioned. However, resource dependencies are not calculated in traditional project networks.

Let's now examine the vicious cycle of logic that occurs due to turning task duration estimates into commitments. Recall that people want to be seen as reliable and therefore try to complete tasks as close to the commitment date as possible. This prevents having protection removed on future estimates and aids being seen as reliable.

When a project is late what is the typical response? Perform a lessons learned to identify the tasks that caused the project to be late. What behavior does such a discovery create? The next time we create a project

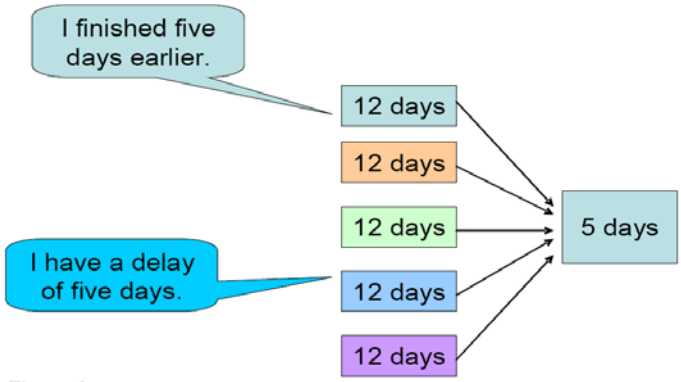


Figure 9

plan we add more safety and detail to those types of tasks. However, if we add more safety, and we don't change how we measure people to allow early completion and late completions to cancel each other out, then the project will not finish "on time" the next time. Also, if the tasks blamed for making the project late add more safety, that project will be scheduled to take even longer. If gains and early completions don't cancel each other out and the project still does not finish on time we then do a lessons learned to find the tasks that caused the project to be late. As you can see, this becomes a negative loop. We are late, blame tasks, add more safety, we are late again, blame tasks, add safety, and so on. Eventually, your customers will move on to your competitor because your project durations are too long. **Your customers determine how long you are permitted to grow the schedule to account for poor management.**

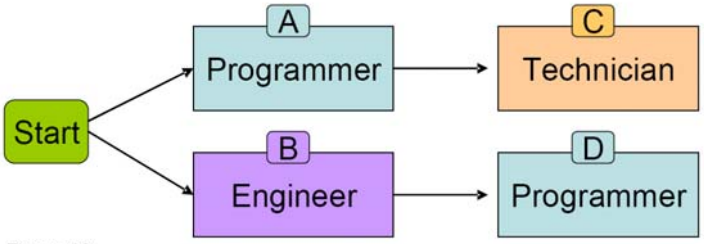


Figure 10

One last issue relates to tasks that integrate with one another. Anywhere there are integration tasks there is added risks. Things do not always plug together as we had hoped. What happens when integrative tasks have problems? The project is delayed. Yes, you could build in more safety at these points but it would be wasted as already described. This problem can only be overcome by a scheduling method that accounts for variability and uncertainty in task completion times. It cannot be overcome by simply stating that people "must" deliver on time. The scheduling method that manages these issues is called Critical Chain. Readers are encouraged to read "Critical Chain" by Eliyahu Goldratt for more information on this subject.

REASON #5: 2 + 2 = 5

In project management things are not always what they seem. This negative effect is quite simple yet often neglected. If you had a project with two tasks, each taking 2 days, how long would it take? Without consideration of the above issues the simple answer is 4 days. However, you now know that the probability is much lower than you thought. Here is the scenario: you are working on Task-1 and your output goes to the resource to perform Task-2. Let's say you are a saint and you started your task on time, worked on it until done with no multitasking and completed it at the end of day 2 as promised. Is it normal to complete your work, immediately take the results to the next person (as if you really knew who it was) and deliver the results for that person to begin? No. However, you conclude that you are done on time so you report the completion time to get credit for on time completion and call it a day. The next morning, you get your coffee, read your email, and do any other quick catch up work. You then take your task results from yesterday and deliver them to the next person in line. We now have lost a half day.

Part II of the scenario: Does the person you deliver your results to immediately stop what they are doing, clear their desk, and begin working on the next step for this deliverable? No. They thank you for the delivery, appreciate you being mostly on time, and exchange gossip with you. They realize there is no hurry to start immediately since there a safety net built into the duration estimate. Now it's lunch time. They return from work, catch up on other work, and consider starting their task but, it's the end of the day, nothing like a fresh start tomorrow. They go home. Now another half day is lost. When they begin the task the next morning they

are already a day late, causing them to be late, and now the project is late. This is how a 2 day task plus a 2 day task equals 5 days. Some may argue that the second person will most likely just speed up and get it done in one day and the project won't be delayed. That could happen. It probably does happen often. But, this admits the task wasn't a two day task at all but a one day task with one day safety, that was wasted, delaying the project. This can easily be overcome by ensuring that each task name includes the hand-off. For example instead of "Complete report" as a task it might be "Marketing is provided with completed report." Now, the person doing the task does not get credit for its completion until it is in the hands of the next resource. In addition, the person doing a task cannot mark their own task complete. Only the resource that needs your deliverable can validate that you are "done." This prevents late delivery as well as provides the next resource in line the opportunity to reject poor quality results that impact their work. This method has been termed the “roadrunner” effect by Dr. Goldratt. The effect is similar to a relay race where we must make sure the next person in line is always ready to start work on a project task when delivered. Many project managers go so far as to contact the next resource in line and notify them when their predecessor task will be completed so they have an opportunity to clear their desk in preparation for the incoming work.

CONCLUSION

Now you know the five major diseases that cause your projects to be late. To remove these obstacles you will need to stop bad multitasking, develop a system that allows early and late tasks to cancel each other out, account for the probability of dependent events, stop the effects of Parkinson's Law, ensure that when one task completes the results appear almost instantaneously to the next task in line, and stop the practice of adding safety to each task.

About the author

Allan Elder is the president of No Limits Leadership, Inc., a consulting firm dedicated to helping organizations deliver more projects, faster, through effective leadership. Allan has worked as the director of MIS for the second largest corporate insurance firm and the largest private security company in California. Allan has been certified as a PMP, is a Theory of Constraints “Jonah,” holds a B.S. in Telecommunications, a Masters in Project Management, and a PhD in Organization and Management. In addition to his consulting work, Allan is a lead project management instructor for the University of California, Irvine, has consulted and taught for the UCI Graduate School of Business, and was a Senior Examiner for the California Award for Performance Excellence (CAPE) for three years. You may contact Allan at aelder@nolimitsleadership.com.