The Program-Level Value Breakdown Structure: How It Can Revolutionize Program Scheduling [1]

Stephen Devaux

Two nouns have been causing a great deal of confusion in project management circles: they are *project* and *program*! The PMBOK Guide defines them as:

- **Project**: a temporary endeavor undertaken to create a unique product, service, or result.
- **Program:** related projects, subsidiary programs, and program activities that are managed in a coordinated manner to obtain benefits not available from managing them individually.

That definition of *project* isn't terrible, apart from the slippery word "endeavor" which should be "investment in work". But the "word salad" definition of *program* tells you that the authors really don't know how to define it. "Coordinated manner" and "benefits not available" are fuzzy terms you throw around when you're not sure what to say.

The trouble is that the two entities *are* different, in ways that require that they be managed differently. Yet "coordinated manner" gives no hint of *how* the differences should drive the different management methods.

Evidence of this is that both individuals and organizations refer to "projects" when they really are performing programs. And we often hear terms like "megaproject". Which should be "megaprogram". And because the differing needs are not understood, those "benefits not available" remain unavailable because the techniques to manage them are lost in the fog of conflation. This is especially true of the scheduling process: scheduling a program is much more complex than a project.

This article will explore the key differences between projects and programs that should drive basic elements of each, like value generation, the interaction of different elements, scheduling and resourcing. And it will finish with a simple model program that illustrates the complexities of planning and scheduling programs. It will demonstrate that while each project *within* a complex program may be manageable, their interactions are fraught with inefficiencies that risk financial loss. This risk *may* be greatly alleviated through the use of program-specific techniques like a program-level value breakdown structure (PgVBS) and an appropriately-designed AI-based program management system.

[1] How to cite this paper: Devaux, S. (2024). The Program-Level Value Breakdown Structure: How It Can Revolutionize Program Scheduling; featured paper, *PM World Journal*, Vol. XIII, Issue VI, June.

The article will explore the following specific topics:

1. Why the differences between what are called projects and programs are significant for managing and scheduling work of each type.
2. How a plan for programs and projects that explores how value will be generated, including the value interactions between work items, should guide all decisions.
3. How value generation should guide a program's schedule—but within the discipline, scheduling on a program is much less well understood than scheduling for a project.
4. A critical path schedule is of great value on a project, but a program can have several different critical paths to various value generation points whose schedule date is important.
5. The program's scope, in the form of its projects and their value interactions, should be assembled by the program manager into a single document, developed collaboratively by the key stakeholders, called the program value breakdown structure (PgVBS),
6. The PgVBS identifies important value interactions among projects, and quantifies the value-added of *enabler* projects and *kindler* projects, whose interactions increase both their own value and that of the work they are enabling/kindling. They should be identified and their value-added estimated in order to target resources to individual projects for maximum value.
7. On a large program, the complexities of the value interactions would likely necessitate use of an AI system to maximize program ROI.

PART A

1. Nouns Can Cause Confusion!

When someone says the French word *projet*, do they mean exactly the same thing as when an English speaker says *project*? Indeed, when a UK or Australian English speaker says *programme*, do they mean the same thing as when an American says *program*?

While specific nouns may not be important, confusion about them often is. Important implications of differences that are not recognized often leads to them being managed inefficiently. Growing a "pomme de terre" in exactly the same way you'd grow a "pomme" doesn't work out well, even though some factors and techniques may be common to both projets!

In fact, what we usually call a "project" has this significant feature: it is a work effort that generates value *only* after it is finished. By contrast, "program" is often used for a work effort that has many projects within it, and each can start to generate value as soon as it's completed. This aspect of programs versus projects is crucial to planning, scheduling and managing such efforts if we want to generate maximum value. If we decide not to differentiate the terms according to this distinction, then we need to find other terms for these entities—because this distinction is crucial to managing programs and projects efficiently!

"Project" & "program(me)" are similar, but DIFFERENT. And because the words are not used carefully, those differences often go unrecognized. And that leads to inefficiencies.

This article stipulates that the *scope* of both projects and programs is not being adequately tied to the work's *raison d'être*, which is the *value* the scope is intended to produce! It's not being recognized and it's not being adequately quantified and monetized. And that value, along with *when* the scope that drives it is generated, can have a major impact on the value the program or project generates.

The value of good scheduling on projects and programs cannot be over-emphasized. But the differences in scheduling methods between programs and projects, and the data on which they should be based, is quite different:

- A project schedule should be driven by critical path analysis, modulated by considerations of resource availability, to maximize the project's "profit", or return on investment (ROI).
- A program may have many projects within it, and thus many critical paths. But the scheduling of projects within the program should go beyond the individual critical paths and be based on the value interactions of those projects to maximize the program's NPV.

All this starts with the estimation of the value to be generated for each scope item in a program-level value breakdown structure (PgVBS). This must extend the importance of program value, and of each project within the program, into the realm of professional scheduling!

2. The Fractals of Project Work

We need to be aware of the fractal relationship of levels of "project" work: from "activity" to "work package" to "project" to "program(me)". Each is a grouping of work within another grouping of work. They can "look" similar, differentiated by size. Yet while each is performed because it has value, increased size as we expand out to the program means increased complexity: personnel, skills, durations, budgets, and interactions.

The value of a little activity can be difficult to fully discern. Draining a swamp or putting a screen on a window in a Panamanian jungle may be perceived as pointless—unless one understands that it may reduce yellow fever deaths of those building that Path between the Oceans. Suddenly the value of activities that reduce yellow fever cases becomes huge, based on the lives saved and the work that the laborers contribute to the overall program (whose value has accumulated with the value of every cargo that has traversed the canal over the past 110 years!).

So activities, and work packages, and, yes, even projects must be judged in terms of how they "fit" within the larger effort! The worker operating a dredger and the carpenter putting screens in windows both were helping the program to accumulate value. Without the efforts to reduce mosquito-borne diseases, the canal may never have been completed.

But the use of varied terms in different organizations/industries—project, megaproject, program—leads to confusion and inefficiency. Each fractal's role within the whole is different, and therefore must be planned, scheduled, managed and judged differently.

3. Defining Entities, if not Terms

Activities and work packages, while they add value (otherwise, why include them?), don't *generate* value until they are combined with other activities and work packages into that thing we call a *project* (*projet*?)! If they did, they would themselves be projects, all by themselves. A car engine, or a ball bearing, or a user input field for a new software package can be sold and create revenue all by itself, and therefore their creation can be a project. *But they only can be used, and therefore only really generate value, after completion.* Therefore, the completion of a project—and the duration of its *critical path*—is of prime importance since no value is actually generated until completion! *That's* why the duration of the critical path is so important—because it determines when a project's value will start to flow: *after* the project is complete! And managing the schedule to reduce the time till the value starts to be generated is a hugely important part of good project management.

A program(me) is different. After completion, the output of a project *may* be combined with the outputs of other value-driven projects to create a more complex product of greater value. And that is what we usually call a *program*! But if that term is not acceptable, we can call it something else. An *effort*? An *endeavor* (endeavour)? A *mission* (well suited to the military!)? A *bucket*? How about a *Romeo,* or *a rose?* Whatever term seems apt.

But that work effort, to create a more complex product of greater value, whatever it's called, is increasingly how the working population of the world is employed. And it's what this article is about.

The specific name doesn't matter—but it's important to call a *program* something different from *project*! Or even *megaproject*! Because it *is* different from a project, in important ways, and therefore needs to be managed differently.

A project, even one within a program, *can* and often *does* start to generate value, even *revenue*, immediately upon its completion! In other words, upon completion of *its individual* critical path, even while other projects within the program continue, or perhaps haven't even started yet! That means that in a program, value generation is often sporadic, occurring at the completion of individual projects *even before completion of the entire program!* The full value for which the program was undertaken will likely not be achieved until after (perhaps, like the Panama Canal, more than 110 years after!) the last project has finished—but what we may call "value packets" can be "cashed in" with the completion of individual projects along the way.

4. Value Scheduling at the Program Level

For more than a quarter century, I've been insisting that project management is a specialty within the discipline of microeconomics. (As evidence, I'd point out that in 1975 Leonid Kantorovich and Tjalling Koopmans shared the Nobel Memorial Prize for Economics for their work on "techniques for the optimal allocation of resources". Surely that's something all project management should concern itself with?)

Each program and project is an investment. The job of an investment manager is to maximize the generated value above cost, or ROI.

What variables are vital to the ROI of a project or program?

1. The value of the product scope(s).
2. The moment (or period) when that value will be generated (i.e., the *schedule).*

For a project, critical path scheduling is a relatively straightforward process and, in some industries like energy generation and construction, is being performed fairly well. (Standardization of metrics like critical path drag, drag cost, and resource availability drag [RAD] could allow project scheduling to be done even better!)

But programs, and program scheduling, are NOT being performed efficiently! The main reason is that the value of the program often accumulates in a very complex manner, from different projects (and non-project work) that both interact with each other AND start value generation at different times. Competence in program management requires both an acquaintance with some of the fundamentals of microeconomics and an awareness that all programs are investments!

5. Scope Generates Investment Value, and Thus Should Drive All Decisions

For good investment decisions to be made regarding program value generation, the value interactions of the product scope MUST be planned! This mandates a collaboratively-developed program-level value breakdown structure (PgVBS) that clearly designates:

A. *Mandatory* vs. *Optional* projects/work. (On a program, most projects are usually optional, included only because they are judged to add value.)
B. *Enabler* projects, which must occur for other work either to be performed or to add value.
C. *Kindler* projects, which add value to that being generated by other projects.

Every project has a critical path, and each activity on the critical path has critical path drag (even if no one has bothered to plan or compute either of those data items). But without information regarding each project's value, how can a program manager (or program scheduler!) possibly know how much the acceleration or delay of a specific project within a program might be worth? In other words, what is this project's cost per unit of drag? If we could add $500K of resources to

activities on Project X's critical path and thereby compress its duration by 3 weeks, should we do it? Doesn't that often depend on Project X's interactions with the value (and schedule, and resources!) of the other work of the program? Value and work of which Project X's scheduler may have little or no knowledge?

If Project X's total value contribution to the program, through both its own value and through the value it will immediately kindle in other projects, is $250K per week, then finishing it three weeks earlier will increase the program's value by $750K at a cost of $500K.

But there is an important caveat: if *all* of Project X's value comes from being a *kindler* (i.e., increasing the value) of Project Y, X's value may not be generated till Y is completed. Therefore, accelerating Project X's completion would add *zero* value to the program *unless* Project B's completion is also accelerated by three weeks. The program scheduler should therefore coordinate the simultaneous completion of both projects, just as Napoleon would coordinate multiple marching armies to arrive at a prearranged (and secret!) assembly point right before battle, to mutually kindle their military potential.

As any professional scheduler reading this will know, program scheduling is an order of magnitude more complex than project scheduling. Yet it's *important*! And *valuable*! On very complex programs (scores or even hundreds of projects), it would almost certainly take an AI system to schedule the project and allot budgets and resources in the best way.

But a skilled program scheduler (or program manager!), if armed with a program-level value breakdown structure, can add significant value. And should be remunerated accordingly. (I believe that good project schedulers are paid far, far less than they should be considering the value they can add. Which, of course, results in there being a shortage of good project schedulers…)

PART B

6. Using the VBS to Guide Program Scheduling

This second part of the article is primarily about what I see as an extremely important difference between project scheduling and program scheduling. In exploring this, there are several somewhat controversial issues I'll have to discuss:

- The differences between *project* and *program*;
- The metrics that should inform all decisions (i.e., ROI/NPV/mission value/expected project profit!);
- Why planning and scheduling based on future projections of value are so important (vs. simply being "agile");
- Why the complexities of value interactions across a large program are so complex as to require a robust AI system.

Ultimately, we will show how much a good PgVBS can add to an efficient program schedule.

7. The Differences between *Project* and *Program*

That fractal relationship among *activity*, *work package*, *project*, and *program(me)* that was discussed in Part A of this article is central to program scheduling. From a practical standpoint (and no matter what the wide variety of differing definitions may say!), the most important distinctions are:

- *Activities* and *work packages* add value, but do not generate value until integrated with other activities/work, whereas:

- *Projects* can and often do generate value, but not *during* execution—only at completion and then *not* in the form of improvement/revenue/savings *until* integrated with other work. (An example is a research project, which may generate the value of knowledge but causes no improvement, revenue or savings *until* its conclusion(s) has been reached and combined, with publication or policy, into a *program*.)

- *Programs* generate value (including improvement. revenue and savings) through the *value-generating interactions* of projects and non-project work. Therefore value, of any sort, does *not* have to wait until program completion, but is often generated at and after the point of such interactions *during* the execution of the program.

But in terms of value generation (the *most* important characteristic!) among these types of project-oriented work, these distinctions are *real* and *important* in terms of how they should be planned, scheduled, resourced, and managed!

8. Value Generation in a Program

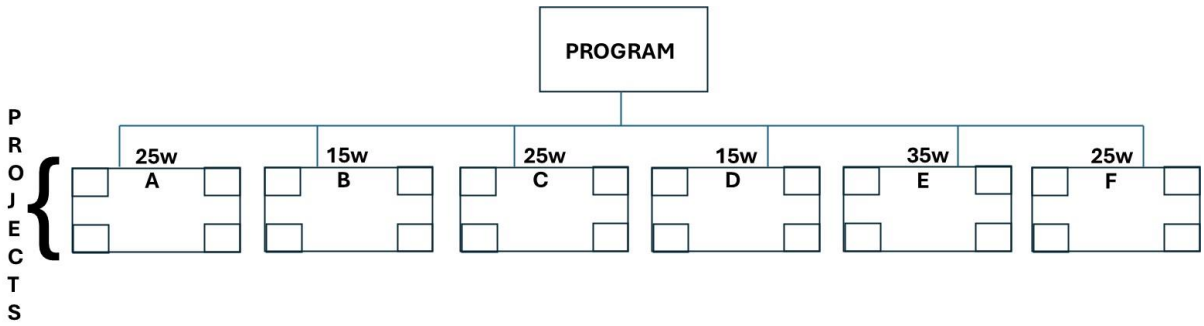

Figure 1: A program consisting of six projects

As we now dig into the techniques that these program value-generation characteristics require, I will try to keep the examples as simple as possible to avoid chasing rabbits down irrelevant holes:

a. I will deal only with project work, even though programs often include non-project work (manufacturing, marketing, sales, distribution, training, maintenance, manning strongpoints or phones, etc.). Those "operations" can sometimes require very complex planning and resourcing to generate value. But the focus of this article is on projects within a program, so I'm assuming that our sample program consists exclusively of project work.

b. Value can take many forms: lives saved due to a public health program; children taught to read in a literacy program; territory controlled in a military program. But for our example, we'll assume all the value is in the form of generated revenue, just because that is the easiest to quantify. But whatever form of value a program or project is expected to generate, that is its reason for being implemented and funded. And therefore, it should be quantified to ensure its value is understood and, always, greater than the cost to complete it. As with any investment, we don't want to pay out more than the expected return—yet for a variety of reasons, that happens all too frequently at the program, project and even activity level!

One of the initial steps on any program should be to figure out exactly how each project (or non-project work) interacts with others. Whereas many of the activities on a project are usually *mandatory* (based on something akin to Newtonian physics: your building *has* to have a roof, and it can't be put on till you've built the walls; to have a drivable car, you must put on tires; you can't debug the software code till you've written it), the projects in a program are usually much more *optional,* discretionary, and included because they've been identified as likely to add more value than they are expected to cost.

The value of the projects in the program tend to interact in two ways:

1. They can be *enabler projects*, which either allow other work to occur (like building the walls that are necessary for the roof to be put on) or allow other work to add value (such as an advertising project that will allow a previously unknown product to sell). If Project W will generate zero value without its enabler, then the Enabler Project V is worth every dollar of value that Project W generates, because without V there would be no value from W. And if Project W's value generation can't occur until V is completed, then the drag cost (i.e., the cost of delay on a project's critical path) on all of V's critical path activities includes any decrease in the value of W due to Project V delaying its value generation.

2. They can be *kindler projects*, *not* essential for generating value from another project, but which add value (and sometimes a great deal!) to another project or projects. If we would sell $75M of our product without advertising, but we estimate that an ad campaign would increase that to $120M, then our advertising project is a kindler that's worth $45M. As a non-monetary example, if distributing a flu vaccine in urban drug stores will reduce deaths from 300 per month to 250 per month, such a project would kindle the value of the vaccination distribution project by 50 lives saved per month.

3. For simplicity's sake, in our example we will quantify the value in dollar units of revenue. But it may often be more useful to quantify the value a project generates as a percentage of the full expected value of the program. If our whole program is expected to generate $200 million of revenue and our advertising project's value-added is $45 million, we can say that its value is 22.5% of the program's value. In this way, if schedule delays or market changes reduce the program's expected value to $150 million, the value-added of the advertising project would be adjusted to 22.5% of $150M = $33.75M.

4. For a variety of reasons, some organizations might not want the program team to be aware of the expected monetary value of the program. In that case, an *encrypted currency* can be used instead of real dollars. If the whole program is expected to generate $500M, it can be assigned a value of $1,000 encrypted dollars. Then whatever percentage of the $500M a given project is expected to generate would be based on $1,000. A project expected to add $150M of revenue would be given a value of $300M in encrypted dollars. This would provide a way of prioritizing the projects within the program on the basis of their relative value in encrypted dollars.

5. Finally, it is important to remember that time to value generation is almost always a factor in any investment. Therefore, the impact of acceleration or delay for every project within the portfolio should be estimated. If a project is expected to add $20M per week if completed by July 1, but the project is delayed by three weeks, the project's value added would be reduced by the lost revenue of $60M.

On any program, the interactions of the projects in value-generation are of great importance for designing the scope, assigning resources, and budgeting—but also for planning and coordinating the schedules of those projects that interact with one or more of the other projects. The scheduling aspect is often overlooked—but there is usually little point in starting an ad campaign long before a product will be ready.

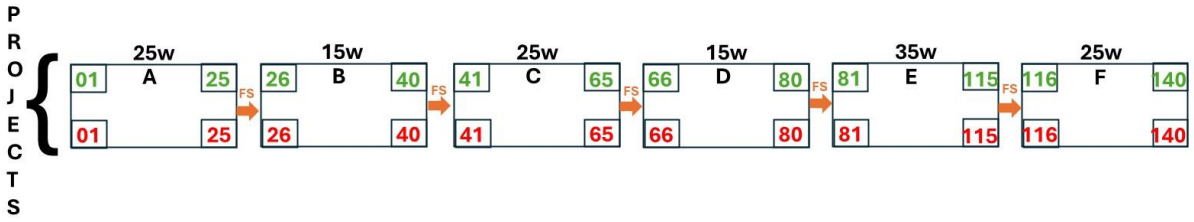

Figure 2: A program's projects scheduled serially, with all finish-to-start (FS) relationships

In Figure 2, we see that scheduling our six projects serially (or iteratively, if you prefer that term) results in a program scheduled to take 140 weeks. Tragically, scheduling a program in such a way (a default used far too often!) shows zero understanding that the benefits for which the program is being performed are being unnecessarily delayed. This occurs all too frequently due to seven possible reasons:

- A lack of imagination.
- A lack of available resources.
- A lack of knowledge about CPM scheduling.
- A phobia about the risks that parallel work can sometimes introduce.
- An unwillingness (of contractor or client!) to make future commitments.
- A funding source that relies on calendars rather than value of work for budgets.
- An "agile" (iterative?) methodology, where each project's deliverable is completed before then being enhanced/expanded by the next project.

The reader will notice that the last of the above is what is often referred to as *agile project management*. It's really not project management at all, but a program management methodology. And the other six aspects all provide reasons, perhaps sometimes even good ones, for reliance on a serial development process.

I'd point out that this confusion about project vs. program management is often at the root of the major criticisms of agile methodologies: the perceived unwillingness to estimate schedule and the associated lack of planning/scheduling. The great tool of project scheduling, critical path analysis, is rendered almost useless on a program of serial projects!

I revere critical path scheduling for projects, and bow my head to its originators. However, critical path scheduling on programs, while still of value, is *much* more complex than on projects! And the conflation of programs with projects has blinded us to how scheduling programs is very different from scheduling projects.

What does the critical path do on a project? Many things. But, first and foremost, it determines the amount of time till the end of the project, i.e., the project's duration. And if the critical path changes, the time of the end will change.

But why is the timing of the end so important?

Because that is when a project can start to generate its value!

Remember:

- An activity or work package generates no value until it is combined with something else.

- A project is undertaken as an effort that combines activities into something that *does* generate value (and, as an investment, should generate more value than it costs). But that value is only generated *at* or *after* the end of the project's critical path!

- A program can include many projects that generate value. It combines them with other projects (and non-project work) to generate even more value. But each point at which the value will start to be generated is, well, *critical* if we want (as we should!) to maximize the program's value. So a program almost always has multiple critical paths, driven not by the program's completion, but by those critical(!) moments when each project, by itself or in combination with others, starts to generate value.

Because if we can plan, schedule and manage that moment, we *may be able to* optimize the program's value! And shouldn't that be the goal of every investment?

9. Developing a Program-level Value Breakdown Structure (PgVBS)

And that brings us to the vital aspect of every investment and thus every program: what items will generate how much value? This should mandate, as part of the program initiation process, development of a type of document that I started recommending for projects years ago: a value breakdown structure (VBS). But I have recently been exploring all the benefits that a VBS at the program level could provide. It could:

- Allow multiple stakeholders to see the big picture;

- Stimulate collaboration and compromise on a single document that describes the value interactions among projects; and

- Once crystallized, it would serve to drive the program team to plan and execute the projects based on value to the program. Having a single documented listing of

projects and their value interactions will allow the team to use that as a guide, and not to be buffeted back and forth as conflicting needs may arise among stakeholders. (Of course, if during program execution the value generation profile changes, the PgVBS should also be changed for the remaining projects in the program.)

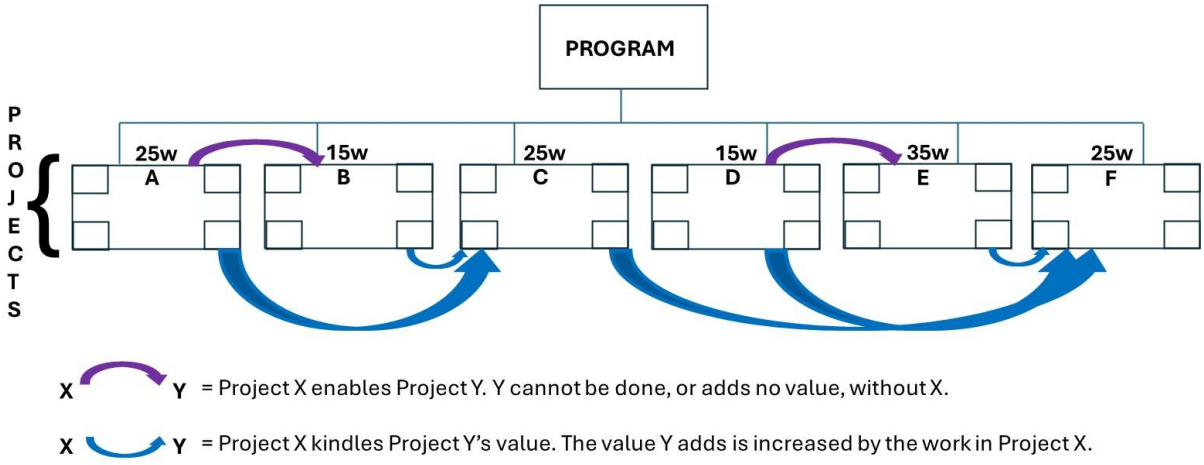

Figure 3: A program-level VBS, showing some examples of value interactions among projects in a program

The purple arrows in Figure 3 above show that two projects, A and D, are enabler projects of Project B and Project E, respectively. Additionally, the blue arrows show that Projects A and B are both kindler projects to Project C, and Projects C, D and E are all kindlers of Project F.

10. PgVBS Implications for Schedule Development and Optimization

Although it's not always the case (it can be much more complicated!), in terms of the scheduling process, *enabler projects* are usually finish-to-start (FS) predecessors to the work they enable. A roof has no value without walls because it needs the walls that hold it up. So wall-building enables roof-installing, both in terms of physical requirements *and* in terms of giving it value. In the fairly unusual event that the enabled project can start before the enabler is finished, *and if such an acceleration might increase the program's value,* then we should investigate changing the sequencing relationship to start-to-start (SS) (perhaps with lag). But for our example, as shown in Figure 3, we will assume enablers are FS predecessors of the enabled.

Kindler projects have a very special relationship with those projects whose value they kindle: projects typically don't generate value till after they finish. Therefore, for Project X to kindle the value of Project Y by 20%, Project Y must first be finished and ready to start generating value. If *all* of Project X's value to the program is as a kindler of Project Y, there is no value to finishing X until Y is also finished. Therefore, Y and X should have a mutual finish-to-finish (FF) relationship. However, if as sometimes happens Project X generates value for the program independently of its

kindler relationship with Project Y, there might be value in finishing it before Project Y finishes. This is exactly the sort of financial decision a good program scheduler (or an AI system with the ability to schedule based on program value) should understand and be able to implement.

What should now be obvious is that the program schedule should be driven by the PgVBS's layout of value generation. Each value generation point (often as with kindlers, the moment when kindler and kindled projects finish, perhaps simultaneously) should be the end of its own critical path. (As discussed earlier, this is the same as with a project, as a single project generates its value at its completion.) This makes program scheduling much, much more complex than project scheduling. If this suggests that project scheduling should be driven by economic considerations, that's absolutely the case! (Project scheduling should be too, of course—it's just that the rigid parameters of "deadline" and "budget" often hide the economic nature of a project.)

As mentioned earlier, physical requirements are rarely the drivers of the sequence of projects at the program level. Since programs are investments, the guiding factor should be to produce the greatest value for the least cost. And timing is usually a significant factor in generating "greatest value."

11. Quantifying the Value-added of Enabler and Kindler Projects within a Program

There are of course many different types of programs, and many different types of value to be generated. To keep our six-project example as simple as possible, we will stipulate that all the value on our sample project will come from weekly revenues.

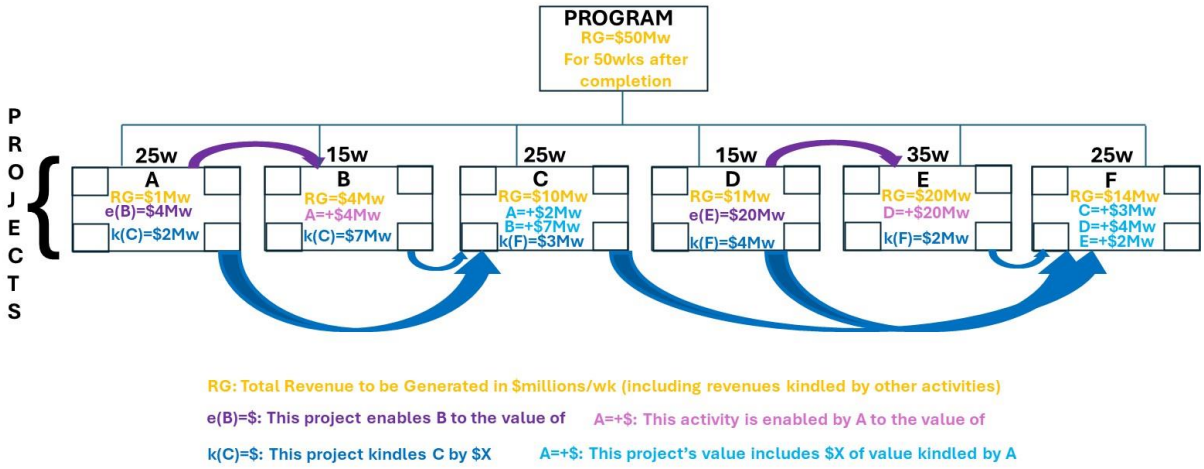

Figure 4: A program-level VBS, showing value of projects based on enabler & kindler interactions

There may be alternative ways of representing the value interactions of projects in a program—but Figure 3 shows the VBS expanded to show the value generation that should guide the scheduling of the program. Such a document (usually incorporating *many* more elements!) should be assembled at the start of every program, to be updated whenever conditions change or new projects (or tranches of projects) are added.

The total value-added of each individual project to the program is the sum of its:

(generated value + enabling value + kindling value)

As shown in Figure 5, if we calculate the value-added for each project, the one with the greatest value-added to the weekly revenue generation *once all the work is completed* will be Project D at $25M/week, *despite the fact that its own revenue generation is only $1M/week!* In contrast, Project F will generate the most revenue ($14M/week), but that is the total value that it's adding to the program, as it neither enables nor kindles anything else.

What should this tell us? *That enabler and kindler projects in a program can have a greatly increased value-added.* Therefore:

1. We need to identify such relationships.

2. We need to develop and use the VBS to quantify the total value-added of enablers and kindlers, which can often be much, much greater than might be suggested at first glance.

3. We need to develop our program schedule and make scheduling decisions such as where to add resources or prune scope to compress individual project schedules on the basis of the value-schedule interaction.

4. All this leads us back to the need to calculate the drag and drag cost of each of the critical path activities and critical resources on each of the individual projects. (For those unfamiliar with the terms, critical path drag is the amount of time that any item on the longest path of a project is adding to that path and thus to the project's total duration. And drag cost is the amount that delay of the project's completion due to an item's drag is reducing the project's value-above-cost.)

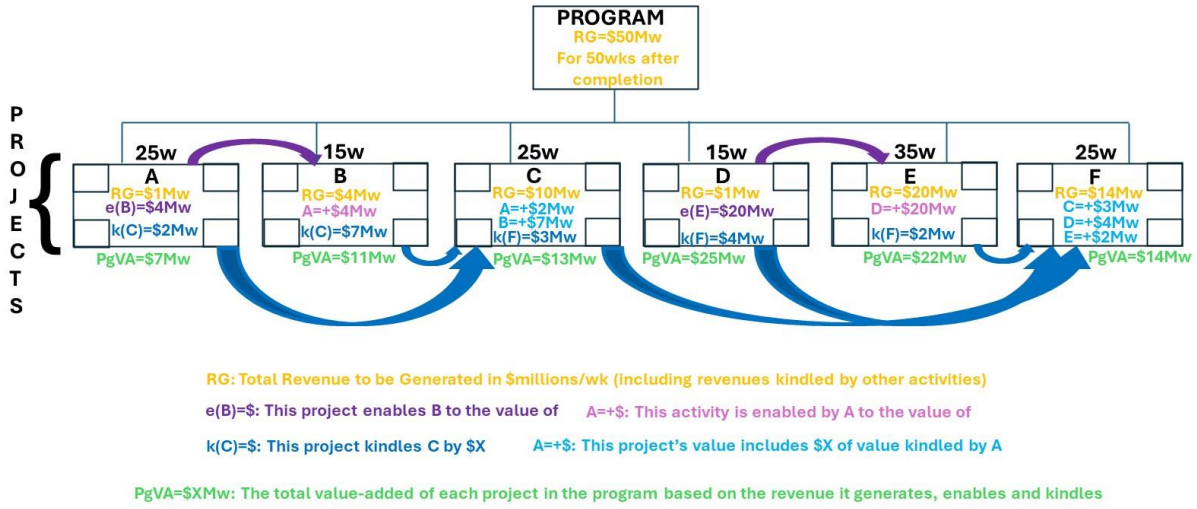

Figure 5: A program-level VBS, showing value-added of each project based on enabler & kindler interactions

12. Some Consequences of NOT Looking Ahead to Plan and Schedule

Now let us once again look at what may happen if we *don't* do this, *don't* plan ahead, and simply arrange all our projects serially (i.e., with finish-to-start relationships) as shown in Figure 6:

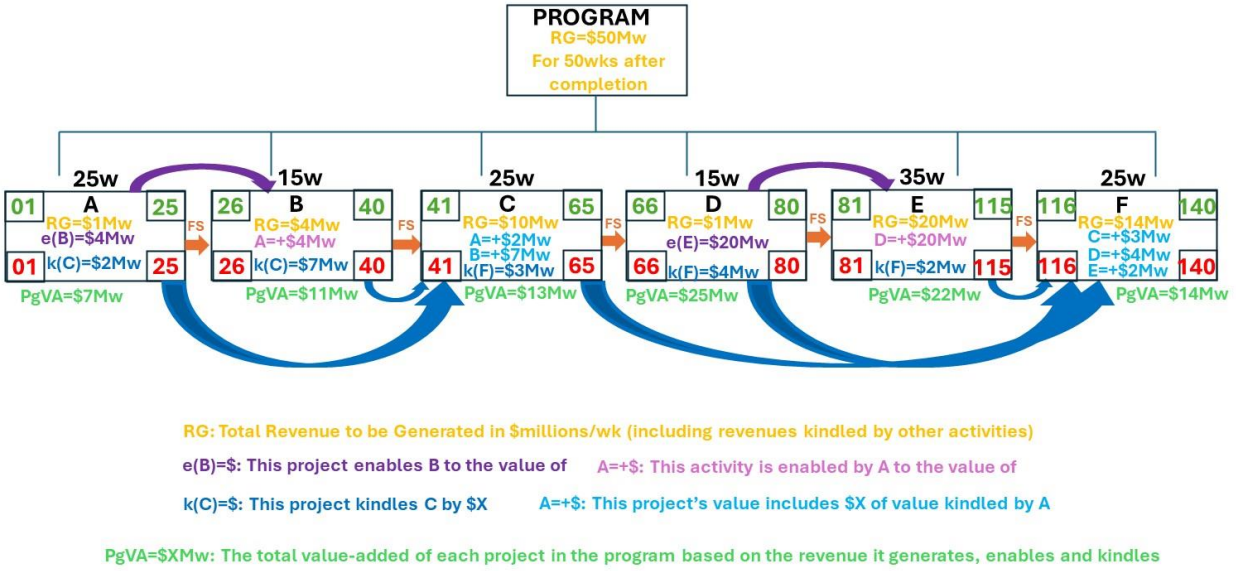

Figure 6: The program scheduled serially, showing when revenue will be generated

Those who are dismissive of planning ahead in schedule development sometimes defend their position as allowing the program to avoid unwise commitments, unattainable goals, and the risks of parallel work to which inaccurate estimates might lead. However, the potential for one of the most likely causes of schedule problems, resource unavailability, is increased if there is no planned schedule to determine when specific resources will be needed for specific work.

Blind to the benefits of earlier revenue generation (or whatever benefits the program is being performed for), with little or no forward vision or planning, the project will take much longer than it needs to. And what is the result of that? Figure 7 shows us:

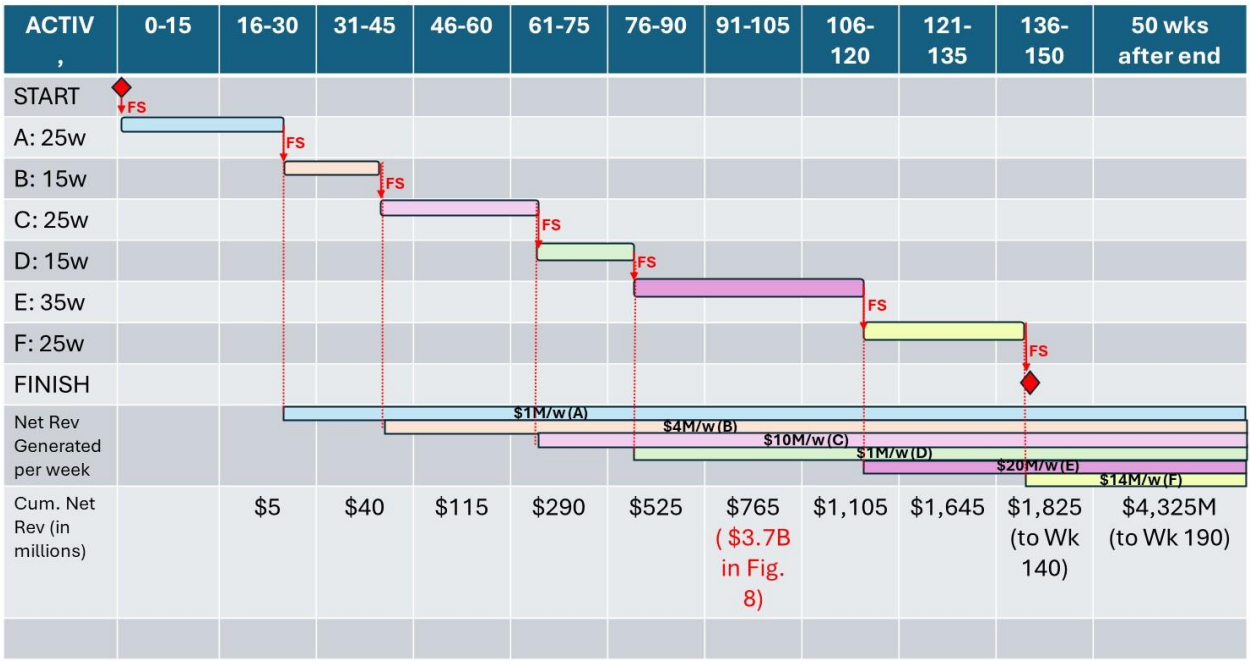

Figure 7: Gantt chart of serial program schedule, showing revenue generation schedule beneath.

Performed to this schedule, the entire program will take 140 weeks, just under three years. At the end of 105 weeks, the program will have generated revenues of $650M:

Project A total revenues = $1Mw for 80 weeks (Weeks 26-105) = $80Mw.

Project B total revenues = $4Mw for 65 weeks (Weeks 41-105) = $260Mw.

Project C total revenues = $10Mw for 40 weeks (Weeks 66-105) = $400Mw.

Project D total revenues = $1Mw for 25 weeks (Weeks 81-105) = $25Mw.

That totals $765M, with two projects (E and F) that are expected to generate $34M/week (of the $50M/week expected when all the projects are finished) yet to contribute a single dollar. And, of course, 35 weeks of project costs, including overhead, is still left to pay out.

One of the nice things about this serial schedule is that it's pretty easy to compute the drag cost of each of the six projects. Because of the rigid finish-to-start relationships, every week by which Project A's drag (which in this particular case is its duration) is compressed will mean that *every* project will start generating revenue one week earlier. That means its drag cost is $50M for every week of its duration. (With a planned duration of 25 weeks, Project A will start with an expected drag of $25 weeks and a total drag cost $50M x 25 weeks, = $1,250 million.

For Project B, its drag cost is only $49M/week, as shortening it will impact all the descendant activities *but not* Project A and its $1M/week of revenue generation. Project C will have drag cost of $45M/week, again based on the $10M/week of revenue that it will generate when it is finished PLUS the delay in revenue generation of the succeeding projects D ($1Mw), E (20Mw) and F (14Mw) that each is delayed by waiting for C to finish as shown in Figure 8. This is the information that should guide increased spending on the critical paths of each of the projects to reduce their drag costs.

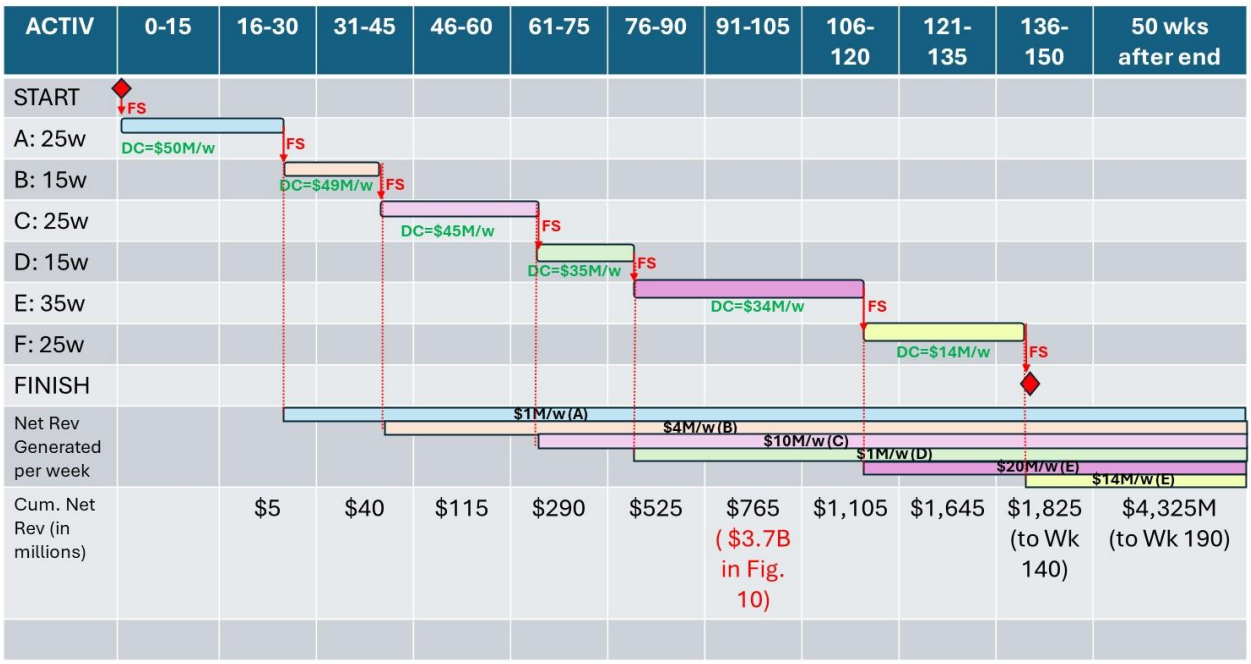

Figure 8: Gantt chart of serial program schedule, showing drag cost per week for each project.

It's surely worth investing an extra $10M on Project A to reduce its duration, and thus the drag and drag cost on its internal project critical path, at a rate of $50M/week. Despite the fact that that project, by itself, will only generate $1M/week when it's finished! But the whole program would

thereby generate each of its "value packets" a week earlier, including reaching the $50M upper limit at the end of Week 139 instead of the end of Week 140.

That is one of major benefits of developing a PgVBS and thus understanding the value interaction of the program's projects.

Figure 9 illustrates graphically how the program revenues would accrue over a 200-week period:

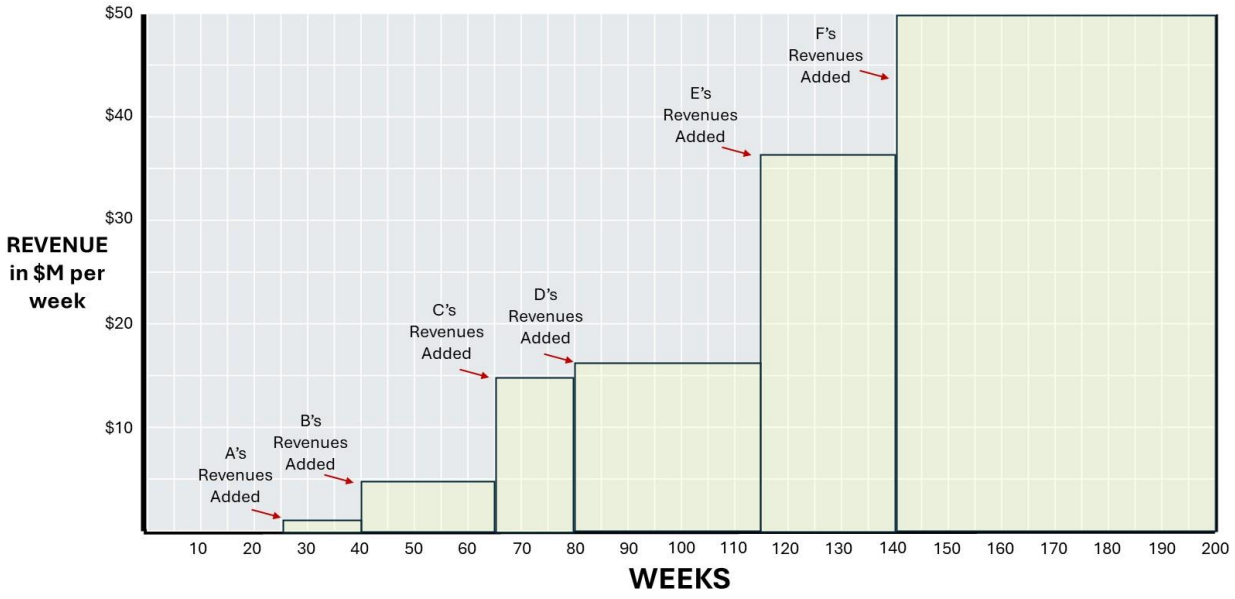

Figure 9: Graphical illustration of how program revenues will accrue with the serial schedule.

What we have now is all the information the program should need to greatly accelerate and increase its revenue generation! What follows in this article is an illustration of how to do this in, again, a simplified example. We will explore only two examples of possible schedules. There would be a vastly larger number in a complex program. But these two examples will demonstrate the opportunities for schedule optimization based on a value generation schedule aided by a PgVBS.

13. Scheduling the Model Program Based on the PgVBS's Value Packets

It's *not* simple to schedule a program, even one with just six projects and with all value just as revenue. We have chosen this simple example (ignoring complicating issues like resource availability, budgets, and items whose value is *not* monetary returns per week!) to demonstrate that, even when greatly simplified, program scheduling based on a PgVBS is (a) complex, and (b) hugely profitable if appropriate time and effort are invested in creating a program schedule driven toward maximized value based on the scheduling of what we are calling the *value packets*. Even a haphazard attempt at developing a schedule based on value generation would likely do better than

the serial schedule we've examined so far (even though such schedules are far more common than most organizations would care to admit!).

A really good project scheduler, armed with a detailed VBS, should be able to do what is shown below. On a large and complex program, with dozens or hundreds of projects generating value at different times, approaching anything like a value-optimized program would probably require an AI system, guided by a program profit metric such as the DIPP (which is: [Expected value ± acceleration/delay value or cost] ÷ Cost estimate-to-complete), and operating like the chess engine Stockfish: modeling vast numbers of possible decisions, analyzing the program value impacts of different resource profiles across projects, and reporting them to the program manager in descending order of potential profits. The program manager may then (as chess grandmasters often do) choose the option that, for pragmatic reasons, is a little less than the "objective" best for pragmatic reasons: less risk, perhaps, or less "churn".

Or they may select the option that the AI system is suggesting will generate the most value.

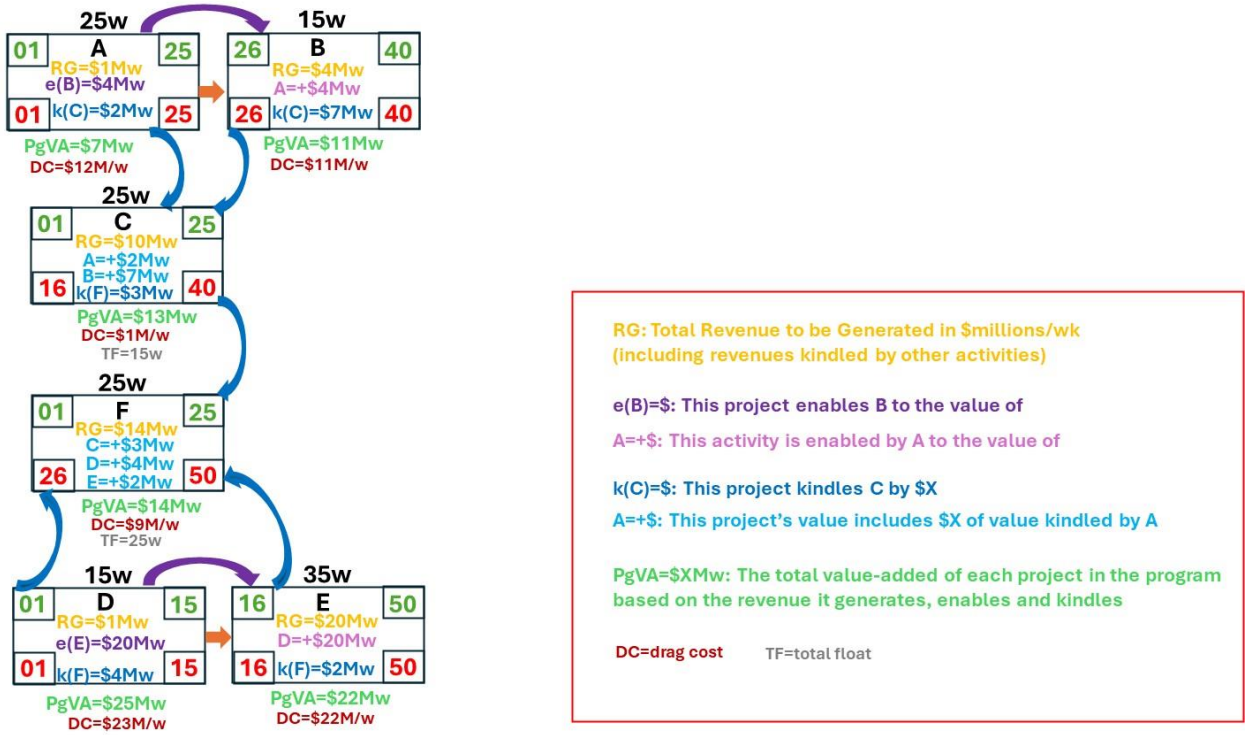

Figure 10: Network schedule of projects with substantial parallelism to accelerate total revenues.

1. In the example in Figure 10, we are assuming that both A enabling B, and D enabling E, call for a finish-to-start (FS) scheduling sequence. There may be occasional exceptions, where an enabler may not have to be finished before the enabled activity can start. Sometimes an enabler may also be a revenue generator on its own, in which case starting it and finishing it as early as possible may increase the program's value. Whatever the precise circumstance, this is the info that the scheduler should use to maximize program value.

2. Notice that, based on the VBS, Project D has neither an enabler nor a kindler. That means that it should be possible to start Project D as soon as the program starts, without waiting for any predecessors. This is shown in Figure 10.

3. A and B are both kindlers of C: A will add $2 million per week to C's revenue and B will add $7 million per week. But those revenues will not start to accrue until C is finished and starts generating value! That means that while we'd like to start generating the $4M/week of revenue from B as early as possible, the $7M/week that B adds to C can't start to accrue until C also finishes. That suggests a mutual finish-to-finish relationship between B and C, where neither is scheduled to finish until the other finishes. (More on this to follow.)

This reasoning allows us to create a program schedule that will let revenue accrue much earlier and more rapidly, and thus more profitably, as shown in the compressed schedule in Figure 10.

Figure 10 shows important differences between program scheduling and project scheduling. Since project sequencing in program scheduling is seldom based on physical requirements, much more parallelism of work is possible. But the consideration that *should* take primacy is the program's value-above-cost, or profit. And that value is based in the value-generating interactions or value packets of the projects.

In Figure 10, we are denoting enabler projects A and D as finish-to-start predecessors of B and E respectively, because that is usually the case with enablers. But Projects C and F have no (what would normally be considered) "logical predecessors." C and F should be scheduled based on revenue generation implications. A and B are both kindlers of C's revenues, and C, D and E are kindlers of F's revenues.

14. Computing Drag Cost for the Projects Based on the Value Packets of their Interactions

When one project kindles the value of another, it typically requires that *both* be finished in order for the full amount of value that's kindled to be realized. Of C's $10M per week of revenue generation, only $1M/w is dependent solely on C being finished. $2M/w is dependent on A's kindling, and a further $7M/w is dependent on B's kindling. That means that as soon as C is

finished, it can generate $1M/w. But to generate another $2M/w, both A and C must be finished, and the earliest A can currently finish is the end of Week 25.

1. A's drag cost for each week of drag = $1Mw (lost revenues for each week of A's duration) + $4Mw (lost revenues from the enabled B because every added week in A delays B by a week) + $7Mw (lost revenues from the completed B kindling $7Mw of revenues in C, which can only generate $3Mw till B is completed in Week 40) = $12Mw for every week into the future following A's completion. (Note that since C can finish Week 25, it's duration is currently having no impact (i.e., no drag) on F's finish and thus the $3M/w in kindled value for F is not drag cost. Essentially, C is not on the critical path to a value packet. Instead, it really has TF = 15w.

2. B's drag cost is $1Mw less than A's because it is the same as A's EXCEPT for the $1M/w that A can generate unrelated to B.

3. C's drag cost is just the $1Mw of its revenue generation which ISN'T being driven/delayed by the kindling from A and B. And neither A nor B is scheduled to finish before C's current earliest possible finish of Week 25. Thus, shortening C to finish earlier will, by itself, not increase revenues *unless* we can also shorten A.

4. D and E are the two projects adding the most value to our program, and therefore it's not surprising that they have the biggest drag costs. (This means that these two projects are good opportunities for schedule compression, if necessary allocating more budget/resources to reduce their drags and drag costs.) Every week that D slips will do more than simply delay its own revenue generation of $1Mw. As D is an enabler of E, E's revenue generation of $20Mw would also be delayed, giving D a drag cost of $21Mw. If D were one week shorter, E could start and finish one week earlier and thus begin generating $20M a week earlier. Additionally, if either D or E were one week shorter, the $2M of value that E will kindle in F could start a week earlier! Thus D's drag cost = $1Mw + $20Mw + $2Mw = $23Mw!

5. Any week that E slips will not only impact E, but will delay both E's revenue generation of $20Mw AND the $2Mw that a completed E will kindle in F. In other words, the value packet of $2Mw that starts once both E and F are finished will be triggered once both E and F are completed – and with the current schedule, it's E's finish that is delaying that value. So E's drag cost is $22Mw.

6. F is even more complex. It generates $5Mw on its own, so that's the least its drag cost can be. And if kindler D finishes Week 15 as scheduled and F finishes Week 25, that's an additional $4Mw of kindled value, taking its drag cost to $9M/w.

Those drag costs should trigger the respective projects to consider adding resources to their critical paths to reduce their drags (and drag costs) and thus increase the program's value.

15. Shared Drag Costs Due to Value Packets in Kindler Projects

The kindler relationships between A and C and between C and F remains complicated. Some value packets can be tied to more than one project, where a change to the schedule of either one will have no impact, but accelerating both could be very beneficial.

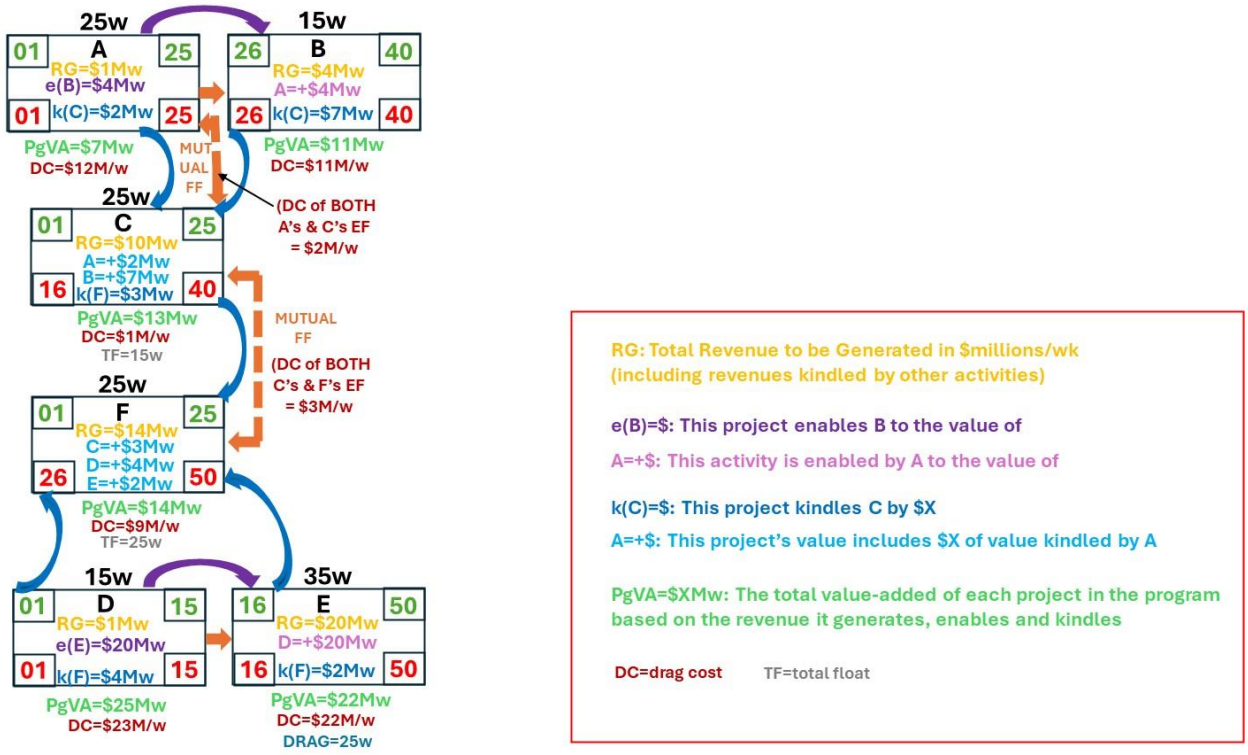

Figure 11: Network schedule of projects with substantial parallelism and MUTUAL FF for kindled relationships.

In Figure 11, we have denoted a couple of *mutual finish-to-finish (FF)* schedule relationships to our program schedule, between A and C and between C and F. This means that, due to the value interaction of a kindler/kindled relationship, we want *both* to finish at the same time.

1. IF both A and C finish at the end of Week 25, then an additional $2M/w would immediately start to be generated due to A kindling that amount in C. And if *both* projects were to be shortened by the same amount, thus still finishing at the same time, every week of that further shortening would add $2Mw to the program. Thus we denote A and C's mutual FF relationship as having a drag cost of $2Mw.

2. Similarly, the $3Mw that C is kindling in F's revenues is dependent on both being finished. If they both finish Week 25, that revenue will start to be generated at the beginning of Week 26. But every week by which we can shorten *both*, in parallel, will generate additional revenues of $3Mw. Once again, the drag cost (that shows how much is to be gained by perhaps increasing the budgets of both C and F) is attributable to both C and F.

With the current early schedule, shorten ONLY one and it will not impact that $3Mw of kindled revenue generation. To get the additional $2Mw of kindled revenues from E, F would have to wait till Week 50.

16. A Comparison of Value Generation Using PgVBS Data for Schedule Optimization

How much has our new program schedule increased the revenue generation? Here is a graphical illustration of the revenue generation with our value-optimized schedule:

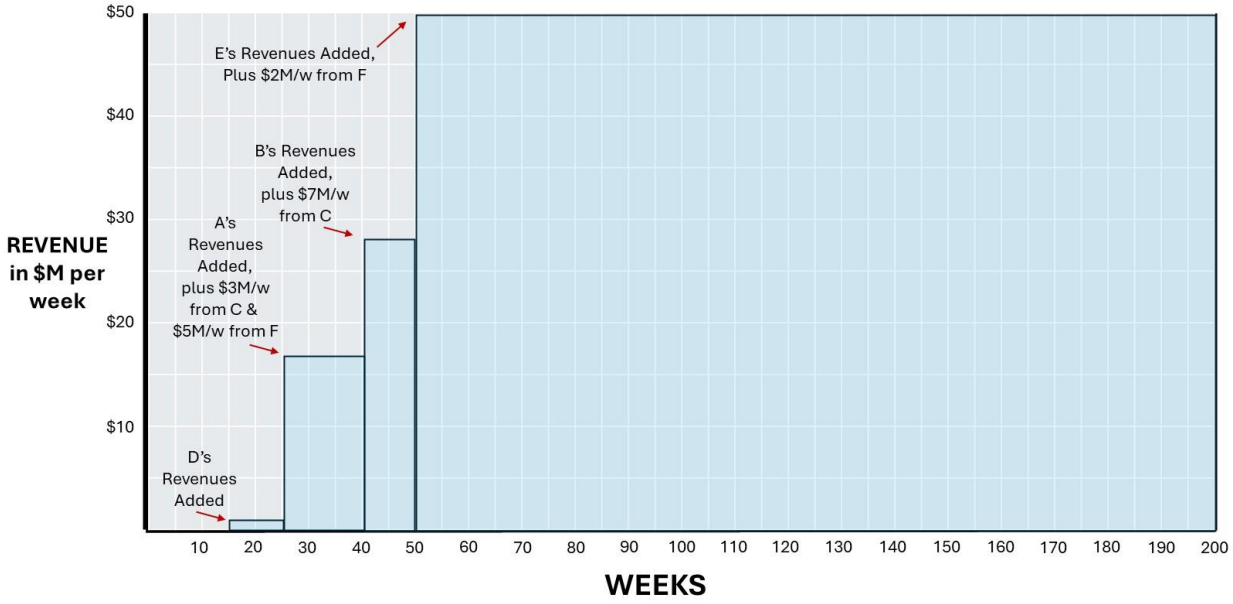

Figure 12: Graphical illustration of how program revenues will accrue with the value-optimized schedule.

To illustrate the way that we have improved our sample program's value generation based on the PgVBS data, Figure 13 shows the serial schedule revenue generation profile from Figure 9 on the same graph as the value-optimized revenue generation in Figure 12:

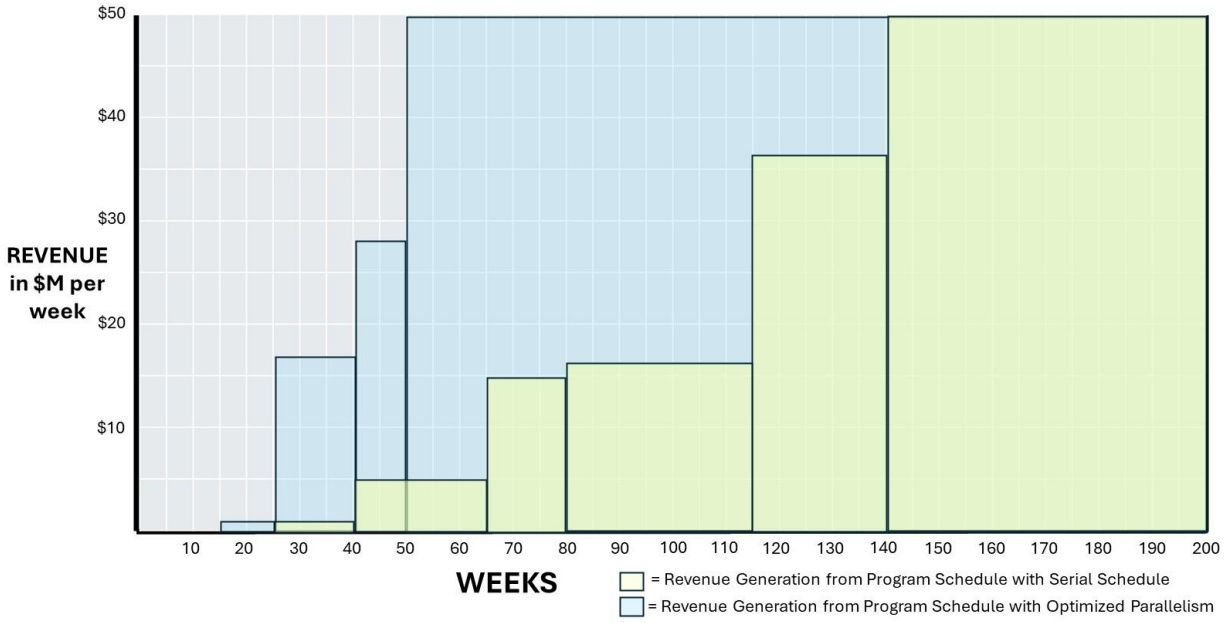

Figure 13: Revenue accumulation of both serial and value-optimized schedules on one graph.

Figure 13 illustrates, in clear value-based terms, the benefits it is possible to generate from a program schedule optimized through the data in a value breakdown structure.

From Week 141 to Week 200, both schedules will generate $50M per week, or a total of $3,000M. But:

- In the optimized schedule, value generation will start at the beginning of Week 16 and peak at $50M per week at the beginning of Week 51; whereas
- In the serial schedule, the revenue generation will start later, accumulate more slowly, and not reach $50M per week until the beginning of Week 141.

At the end of Week 140:

- Figure 13 shows that the serial schedule will have generated $1,825M, whereas
- The optimized schedule will have generated $5,045M.

An extra $3,220M, or a 176.4% increase over 140 weeks seems like a reasonable return for the effort of developing a VBS and then using it to optimize the schedule.

And remember, we still haven't used the individual projects' critical path drags and drag costs to compress each of their respective durations. There are benefits yet to come!

Finally, even most moderately good schedulers would probably do better than the all-serial-projects example. But would they get to the revenue generation level of our second example? Even on a very simple six project program with all the value in revenues? How much value is being left on the table of every program, due to inefficient planning and scheduling?

And then imagine the situation on complex programs! That is where an AI system that can assign resources on the basis of the value interactions in a PgVBS could add tens or hundreds of millions of dollars to a big program!

As difficult as some of these innovations might be on a truly complex program, shouldn't it be worth the effort?

14. Conclusions

1. Activities, work packages, projects and programs have a fractal relationship, and are all investments, performed because their scope is intended to generate more value than it costs.

2. Yet, rare among investments, the investor can exercise a measure of control over programs through efficient planning, scheduling and managing programs and projects.

3. Since the motive is to generate value from an investment, a plan for programs and projects that explores how value will be generated, including the value interactions between work items, should guide all decisions.

4. The schedule by which the value will be generated is a vital output of a program's value generation process as some schedules can greatly enlarge the return on investment. But scheduling on a program is much less well understood than scheduling for a project.

5. There is inadequate agreement on the meaning of the terms "program" and "project". They are often used interchangeably and confusingly. Whatever terms are used, the one which generates value *only* after it is finished is usually called a *project,* and its finish represents the merging of activities and work packages into a product or service that generates value. By contrast, a *program* has many projects (and non-project work, like operations) within it, and each can start to generate value as soon as it's complete.

6. A critical path schedule is of great value on a project, as it determines when a project is completed and its value can start to be generated. Also, the sequence of activities

and work packages that drives the predecessor-successor relationships within a project is usually created by a physical requirement: we can't replace the pipe till we've turned off the water.

7. In contrast, a program may have many critical paths, each leading to a project finish that represents the moment that project's value can start to be generated. The sequence (and schedule) of projects within a program tend not to be driven by physical requirements. Instead, they should be sequenced in the way that maximizes the value of their interactions, with projects able to generate value for the program as soon as they are finished.

8. The program's scope, in the form of its projects and their value interactions, should be assembled by the program manager into a single document called the program value breakdown structure (PgVBS) through a collaborative effort among stakeholders and the program manager.

9. The PgVBS should not only estimate the value (in whatever coinage—value can sometimes, such as in public health programs, be quantified in saved human lives!) each project is expected to contribute to the program, but also the value interactions of each project with other projects (and non-project work).

10. Two types of important value interactions among projects are *enabler*, where the enabled project can either not be performed or will generate *zero* value without its enabler; and *kindler*, where the kindled project has its value increased by the output of the kindler project.

11. For scheduling purposes, an *enabler* is almost always a finish-to-start (FS) predecessor of its enabled project; whereas a *kindler* is almost always a finish-to-finish (FF) predecessor (or have a *mutual FF* relationship) with a kindled project, as the kindled value typically isn't generated until both the kindler and kindled finish.

12. Knowledge of the PgVBS, with its enabler and kindler projects and their value interactions, allows a skilled scheduler to greatly increase the value generation and ROI of a program through good value-based scheduling decisions.

13. On a large program, the complexities of the value interactions would likely overwhelm even the most skilled project scheduler. While the scheduler could improve value generation (especially through computation of critical path drag and drag cost on projects that precede high value interactions between projects), it is likely that an AI system that (a) uses a profit-based metric such as the DIPP, and (b)

operates in a similar manner to the Stockfish chess engine (i.e., giving a "score" based on the program's expected value generation every time a decision is implemented and/or the program is updated) would be extremely helpful in maximizing program ROI.

The author would like to thank Jan Willem Tromp, Alex Lyaschenko and David Pells for their help with and feedback regarding this article.

About the Author

Stephen Devaux

Boston, MA, USA

Stephen Devaux (also known as "Steve the Bajan") has been a project management theorist, consultant, author and educator for 36 years. His first theoretical article, "When the DIPP Dips", was published in *Project Management Journal* in 1992. Since then he has taught project management in graduate, undergraduate and executive ed programs at six universities, and training classes for scores of organizations including Siemens, Ford, Qatar Telecom, BAE Systems, the US Air Force, iRobot, and Rockwell International. He has consulted in planning and recovering projects ranging from Fashion Model Barbie to avionics on the F-35.

The innovative methods and metrics that he has introduced to the project management discipline include the DIPP, critical path drag and drag cost, the value breakdown structure (VBS), and the doubled resource estimated duration (DRED).

Mr. Devaux can be contacted at apm7@ix.netcom.com