Allocating Resources in Multi-Project Programs: Lessons Learned from the Trenches

Edward Lari, Dr. Jeffrey Beach, Dr. Thomas A. Mazzuchi, and Dr. Shahram Sarkani
George Washington University

There have been significant strides in the use of project management methods for resource allocation in single-project programs in the DoD environment. Unfortunately, when it comes multi-project programs, inter-project resource allocation techniques are inadequate. The objective of this article is to shed light on some of the challenges program managers are confronted with when allocating resources among multiple project programs, and to highlight some of the practical solutions that have been proven useful in resolving these issues.

The past two decades have seen an increase in the size and complexity of DoD software implementation programs. This has been accompanied by an exponential growth in the number of multi-project programs in program managers' portfolios. Yet the literature on program management has tended to be written with the underlying assumption that the programs managed are made up of single projects as opposed to multi-project programs. In reality, this is not the case. Most programs involve the simultaneous management of multiple projects or a portfolio of projects managed to ensure the aggregate results at the end of the program.

Multi-project programs further add to the level of complexity of the resource management equation. In managing large-scale software implementations, the success of any program is largely dependent on the effective utilization of its resources across multiple projects. Right from the initial stages of program initiation to the final stages of delivery, the program manager is often faced with the complexities of resource allocation and its negative impact on performance results.

The Traditional Process for Resource Allocation

In multiple-project programs, demand for project resources typically exceeds its supply. This is usually the first lesson learned by the fledgling manager involved in any aspect of the resource allocation process. This has given rise to one of the basic principles of project portfolio management: By defining the relative priority of projects and by applying resource leveling and skill-set capacity constraints, a method which allocates resources in a manner that optimizes their enterprise utilization can be derived. While this theory has many proponents—and the principle may, in fact, be true—its practical implementation is usually very challenging.

A review of traditional methods of resource allocation would illustrate how it falls short in determining the true enterprise demand or in identifying the key resource conflicts at the program level. From a conceptual standpoint, there is a great deal of commonality in the processes utilized by organizations to allocate resources across multiple projects. The tools used for this purpose, be it customized software, spreadsheets, tables, or even roundtable discussions, are all different. Yet all traditionally begin with an algorithm for assessing and categorizing demand, and creating the proposed book of work. Considering that demand will exceed supply—and that the pool of resources will run dry long before the list of initiatives are exhausted—the initial step is to give relative rankings for the different projects through a prioritization process. The prioritization process is followed by building the master project schedule, capacity planning, and high-level resource assignments.

The prioritization process doesn't require complexity, needing only an estimate of expected benefits and initial costs to be effective. For example, one small division of an intelligence organization used the very simplistic yet effective *Is it* process, which sought answers for questions such as *Is it a project?*, *Is it a realistic estimate of benefits?*, and *Is it a valid approach to solving a problem?*. Also used are resource estimate descriptions such as *Small*, *Medium*, and *Large* to facilitate the definition of a project's cost/benefit. These resource estimates—which would inevitably be tied to a project's initial priority—were developed after a 15-minute roundtable review of the proposed solutions by technology platform managers. Whatever the method is, the net results of a prioritization process are either an ordinal ranking of projects or a grouping of projects into *Critical*, *High*, *Medium*, and *Low* categories.

Once the initial prioritized list of projects is created, the next step is to build the preliminary enterprise master program schedule. The master schedule establishes time boundaries for projects that are ready to go and need to be staffed. It may also create placeholders for those projects that should remain in the queue until the next round of prioritization occurs. Projects that do not make the cut are sent to the *project graveyard* where they wither away from lack of attention or are resurrected for another crack at prioritization. Within all project portfolio management systems is an embedded project evaluation process used to assess the health of the projects at various stages in their life cycle. It is critical to ask whether the project is still relevant to the program's goals and objectives. If the answer is no, then the project should be stopped. This way, the program manager ensures that all resources are deployed where they will offer the best return to the program as a whole. In practice, this is one of the most difficult decisions program managers have to make, as reshuffling a project portfolio will likely displease the managers of those projects. However, the program manager must always seek to globally optimize the program performance—as opposed to trying to optimize each project.

After the master program schedule is created, the next step is to undertake capacity planning. Capacity planning, normally performed by the platform and skill set, is what makes the master schedule believable. Here, the total resource hours are deducted from the overall capacity. Project managers are given their projects and—with the staff they believe is required to complete those projects—go through their initiation processes with management's final instructions regarding the triple constraints: time, budget, and quality.

High-level resource assignment follows capacity planning. During the resource assignment stage, the project managers provide resources to the scheduling groups. While doing this, the possibilities of over-allocation and under-allocation arise. The result of a specific resource shortfall means that the team will be asked to do more to either cover the deficit or be forced to get by with less.

High-level resource assignments are analogous to a hotel reservation. In software engineering, the department or platform (e.g., Web development, AS/400, testing) supplying the resource is the hotel. The room and room type correspond to position (i.e., developer) and attributes (e.g., manager, senior developer, etc.) of the resource. Length of stay (duration of assignment) is allocated from available inventory, but the actual room number (resource name) is not yet known. It will not be determined until the guest (project manager) arrives (kicks off their project). The difference between the two comparisons is that in the field of IT, the project manager can't be upgraded to a better room or compensated with a free meal for any inconvenience. As mentioned, specific resource shortfalls mean that the team will be asked to do more to cover the deficit or be forced to get by with less; hotel guests, however, will never be asked to share a room with others if the hotel over-books.

Challenges With the Traditional Approach: A Case Study

So if the resource allocation process—with stages such as prioritization, scheduling, capacity planning, and high-level resource assignment—takes place under constant scrutiny and monitoring, then why do so many programs fail? And why do most analyses of program and project failures[1] list poor resource scheduling and planning as primary causes?

To plainly illustrate this point, we present the following scenario from a case study encountered by the authors in consulting assignments.

A software requirements analyst (SRA) was told that for the next five weeks their time will be spent on a project. Assuming a 40-hour work-week, this equates to an initial resource request of 200 hours. At the project kickoff meeting, the SRA is given a high-level schedule that mapped out the standard project management life-cycle milestones. According to the project manager, the SRA's role on the project would be limited to three weeks developing the functional requirements and two weeks assisting in the creation of the system design document. The project manager assured the analyst that none of these issues would take more than 20 hours a week during that time period. The analyst examined the project schedule and reported back that the SRA could make those dates. The SRA then needed to meet with three different business units to solicit their input into the document: This required first arranging times to meet with the units individually, then finding a location for a comprehensive meeting, and then using the knowledge to prepare documentation on the existing process. All of those activities can be done in a week. The analyst estimated two weeks for creating the functional requirements document, including time for walkthroughs (that need to be scheduled), revisions, and signoffs seemed reasonable. As for the time required for assisting system design, the analyst wasn't sure—but two weeks seemed realistic.

Two days into the project, the SRA realized the task estimates were off significantly. There was no existing documentation for the system: That meant two weeks of sitting down with developers and creating it. Scheduling the meetings turned into a nightmare as well. There actually turned out to be 10 business units that were stakeholders in the process—and located throughout the country. Including travel time (even meeting face-to-face with half the units and speaking to the others via conference call), this made crafting an initial draft of the requirements much more than a two-week job: four weeks if everything went well, six weeks with coordination issues. To make matters worse, one of the developers came up with a new idea on how to implement the functionality. Even though the SRA had to wait until the business requirements were complete to make sure, the developer confidently projected that his time would drop from 12 weeks to three. The project manager thought any lost requirement time could be made up during the development, and continued to report the project's overall status as *green* (on schedule within an acceptable variance of planned targets). Needless to say, the project did not move on as per the previously planned project schedule. This example clearly shows that in spite of thorough planning, something had gone wrong somewhere.

An analysis of this case study presents the following questions:

- What numbers should be reflected in capacity planning and the enterprise project schedule?
- While the aggregate hours for this project may decrease, what will be the impact to the SRA's pool of resources for the program?
- Where are the extra analysis hours supposed to come from?

The theme echoed in this scenario—one repeatedly encountered by project and program managers—is that the root cause of failing to perform the resource allocation process properly can be traced back to two specific issues:

1. The inability of program managers and the systems that support them to determine true capacity.
2. The failure of project and program managers to comprehend the impact of the program's critical path across multiple projects.

Alternative Solution Approaches

While these problems are solvable, the solutions, like most management problems, are neither linear nor discrete. Experience indicates, however, that a combination of approaches will go a long way in solving the problem.

Determining True Capacity

A common error often made by programs is failing to perform true program capacity in the initial stages. For program capacity planning to work, both the supply of and demand for available resources must be well-estimated. What may not be apparent in performing this exercise is how to account for the margin of error inherent in both estimates. This margin of error may or may not be factored in when performing the initial resource assignments, but it has to be accounted for while undertaking capacity planning at both the project and program levels. Not only do calculations have to be performed at the project levels, but also at program levels so as to adequately estimate program-level risks. By changing the way the future demand is calculated and by using the magnitude of project tasks as an additional input, managers can perform resource allocation more effectively and resolve these conflicts before they become full-blown program risks.

Efficient Program Time-Tracking

Assuming that the initial resource supply has been calculated correctly, the real availability of personnel is determined by the projects and program level. An employee working on multiple projects should have the ability to track their inter-project usage statistics. Unfortunately, this is not often the case, even though organizations have deployed various resource tracking systems and rigorously track hours at the project and development life-cycle phase levels. A typical employee only receives guidance on the percentage of time allotted and the overall number of hours the platform has budgeted for the project in question; no consideration is given to the inter-project utilization. In order to perform program-level time-tracking, systems and processes that can track resource usage both at the

Software Defense Application

DoD program managers are increasingly being tasked with managing complex software programs. As such, the number of multi-project programs in the manager's portfolio is not only increasing but constantly changing—a challenge accompanied by a set of unique resource allocation problems not typically encountered in single-project programs. Unfortunately, the program management literature has tended to treat programs as if they are all composed of a single project, thus ignoring these challenges. The objective of this article is to address the challenges faced by program managers in DoD software development programs as they allocate resources across multiple projects. This article will be especially useful to DoD program managers who manage portfolios of projects, project managers who work in multi-project programs, and software engineers and analysts who support programs in the DoD software community.

project and program-level must be implemented. This may be as simple using a spreadsheet or a more integrated enterprise-resource planning system.

Program-Level Task Tracking

A specific project can have many tasks. Therefore, it is necessary to track the tasks at the different levels, project and program alike. Project managers typically conduct task-level resource tracking, but others involved with the program rarely get that opportunity. The program manager is usually conditioned to report on the resource pools at the summary level and does not normally request information about them on the task level. While the program manager rarely has the need, both the project and program managers should be equally aware of the task requirements and tracking to be undertaken (down to the task level) to avoid unnecessary resource conflicts.

As for accounting for different types of outcomes, program/project evaluation and review technique (also known simply as PERT) formulas can become an effective tool to account for variances in resource estimates at the program. For all resources assigned to a task:

- Effort remaining on a task per project = Σ Estimate of Hours to Completion = Σ (Pessimistic* + 4 x Most likely* + Optimistic*) ÷ 6

  * *Estimated remaining hours for any task.*
- The total effort remaining for a project resource at the program level = Σ Effort remaining on all project tasks.
- The total remaining effort for a resource = Σ Efforts on all projects in the program.

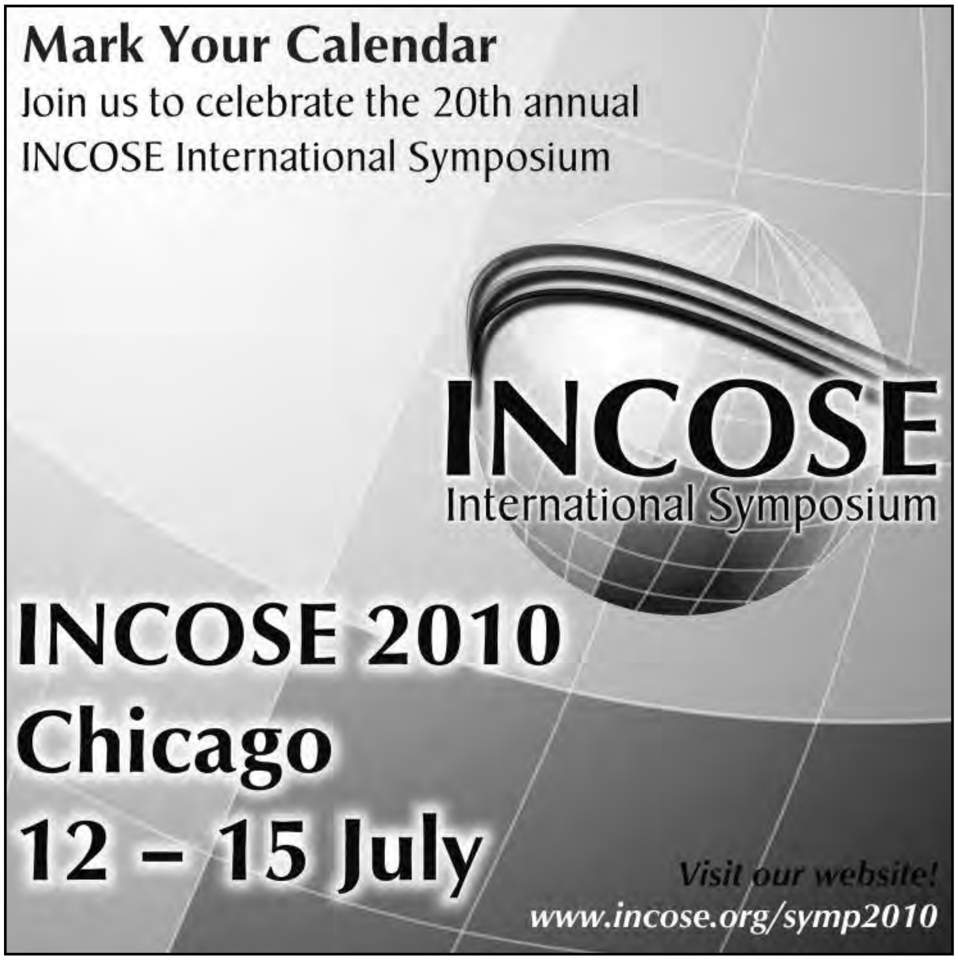
It should be noted that calculating remaining work estimates by resource, project, and program is a continuous exercise that should be included as a standard operating procedure whenever earned value analysis is performed on an ad-hoc basis. It may add some incremental effort to the program, but creating a task-level feedback loop ensures that both project and program managers know the true working capacity of their resource pools.

Program Resource Critical Path Analysis (CPA)

The classic definition for a critical path is the sequence of project network activities that add up to the longest overall duration in a project. CPA is a powerful tool that uses mathematical algorithms to schedule complex project activities. It is designed to provide visibility into potential project resource issues so as to develop risk-mitigation strategies.

Extending the critical path concept to a set of related projects is problematic to a lot of programs. This is because while the application of CPA in a single project is well understood, the ability to extend this concept to multi-project programs is challenging. This is because performing a CPA over multiple projects is not a simple activity to complete. It requires the ability to identify the critical path for each project at a low enough task level, a willingness to calculate the true effort for each resource assigned on the critical task, and the coordination of resources for critical path tasks across the program.

The coordination of resources for critical path tasks across the program is paramount to the success of program CPA. A minor oversight in coordination could lead to cascading resource risks across the entire program. One simple yet effective method is the frequency litmus test approach to assessing risks across a program's critical path. It states that a resource cannot be assigned to more than two critical path tasks in the same program. Our empirical observations confirm that most technical resources begin to lose focus when dealing with more than two crises at the same time. And, since tasks on the critical path have the highest probability of generating a true crisis, every effort must be made to avoid situations where a key resource—on critical path tasks—faces multiple crises simultaneously.



Conclusion

The blinding pace of IT innovation and the growing complexity of DoD requirements has led to more challenging multi-project programs. Program managers are charged with managing programs that are more complex, thus requiring intra-project and inter-project resource utilization management. The ability to track and effectively utilize these resources, however, requires effective resource allocation methods.

While the field of project management has successfully developed techniques for managing intra-project resource allocation, its application in multi-project programs is still inadequate. It requires diligent project and program management skills and practices beyond those traditionally required in single-project environments. In addition, it requires the adaptation of best practices—including determining program-level true capacity, efficient program time tracking, program-level task tracking, and program resource-critical CPA. Finally, a concerted effort is required to integrate enterprise-level resource allocation and management tools for program management so as to enable the intra-project and inter-project resource allocation. Without this, effective program management is likely to fail.◆

Note

1. There has been significant analysis and discussion of program and project failures, including the Standish Group's Chaos Report (see the 1995 incarnation here: <www.projectsmart.co.uk/docs/chaos-report.pdf>), the Bull Report (1998), and the KPMG Canada Study (1997).

Additional Resources

1. Bainey, Kenneth R. *Integrated IT Project Management: A Model-Centric Approach*. Norwood, MA: Artech House, 2004.
2. Chin, Gary. *Agile Project Management: How to Succeed in the Face of Changing Project Requirements*. New York: AMACOM/American Management Association, 2003.
3. Gray, Clifford, and Erik Larson. *Project Management: The Managerial Process*. 2nd ed. New York: McGraw-Hill/Irwin, 2002.
4. Hill, Gerard M. *The Complete Project Management Office Handbook*. 2nd ed. Boca Raton, FL: Auerbach Publications, 2007.
5. Schwindt, Christopher. *Resource Allocation in Project Management*. Berlin/Heidelberg: Springer-Verlag, 2005.

About the Authors

Edward B. Lari is a senior principal systems engineer at General Dynamics, where his professional experience includes the application of operations research models in support of large-scale DoD resource allocation problems. He is a doctoral candidate in systems engineering at George Washington University (GWU), has a master's degree in manufacturing systems engineering from the Rensselaer Polytechnic Institute, and a bachelor's degree in manufacturing engineering from Central State University (Ohio).

5311-D Columbia RD
Columbia, MD 21044
Phone: (301) 537-7652
E-mail: laried@gwu.edu

Jeffrey Beach, D.Sc., is the head of the Structures and Composites Department at the Carderock Division of the Naval Surface Warfare Center, where he has worked since 1969. He earned his bachelor's and master's degrees in aerospace engineering from the University of Maryland, and his doctorate in engineering management and systems engineering from GWU.

School of Engineering and Applied Science
The George Washington University
Dept. of Engineering Management and Systems Engineering
20101 Academic WY
STE 227-A
Ashburn, VA 20147
Phone: (202) 994-5450
E-mail: beachje@gwu.edu

Thomas A. Mazzuchi, D.Sc., is chair of the Department of Engineering Management and Systems Engineering at GWU, where he is also professor of both operations research and engineering management. Mazzuchi earned his doctoral and master's degrees in operations research at GWU, and a bachelor's degree in mathematics at Gettysburg College.

School of Engineering and Applied Science
The George Washington University
Dept. of Engineering Management and Systems Engineering
1776 G ST NW
STE 101
Washington, D.C. 20052
Phone: (202) 994-7424
E-mail: mazzu@gwu.edu

Shahram Sarkani, Ph.D., is faculty adviser and head of GWU's engineering management and systems engineering off-campus programs office. He has served as a professor of engineering management and systems engineering since 1999, and is the founder and director (since 1993) of GWU's Laboratory for Infrastructure Safety and Reliability. Sarkani earned his doctorate in civil engineering from Rice University, and his master's and bachelor's degrees in civil engineering from Louisiana State University.

School of Engineering and Applied Science
The George Washington University
Dept. of Engineering Management and Systems Engineering
2600 Michelson DR
STE 750
Irvine, CA 92612
Phone: (949) 724-9695
E-mail: sarkani@gwu.edu