Making Big Decisions Better

Integrating System Dynamics with other program management tools to make smarter & faster decisions

Thomas W. Mullen
Ritu Sharma
PA Consulting Group
One Memorial Drive
Cambridge, MA 02142
Main Switchboard: 617-225-2700
Fax Number: 617-225-2631
Tom.Mullen@paconsulting.com
Ritu.Sharma@paconsulting.com

Abstract:
Most large development projects suffer overruns and delays, despite substantial effort spent on systems tracking risks and projecting performance. Managers have an especially difficult time making big decisions such as major project re-plans. Typical project management systems have key blind spots that limit their value for comprehensive decisions. Most project management tools do not address project dynamics – variations in productivity and quality over time under different conditions. System Dynamics models have been used to address this weakness and capture project dynamics, but typically these models have their own blind spots as they omit key details. With many pressing decisions and little time, managers rely on intuition to supplement the limitations of management tools. The combination of little time for major decisions, limited tools, and unreliable intuition is a key contributor to the poor results often achieved on major projects. This paper offers perspective on the challenges of making major decisions and describes a case using an integrated management tool -- a System Dynamics model linked to a database of project details. This management system was used to restructure a multi-billion dollar development program with detail and rigor – examining dozens of different options, sensitivities, and leverage points in one month.

Key Words: "System Dynamics", "Project Management", "Risk Analysis", "Decision-Making"

1. Major Decisions on Large Projects:

1.1. Situation:

A vast amount of effort is expended on formal risk management systems and project planning tools. Yet programs frequently suffer large cost and schedule overruns with little advance warning. In 1995 the Standish Group conducted a study finding that 53% of the software development projects in the US cost an average of 89% more than their original estimates. Recent updates have shown small improvements some years and worsening in others with a continued unhappy track record. In 2003 nearly 70% of projects were cited as unsuccessful, and as the size of a project increases so does the failure rate. Why are our management systems failing to give us the information and decision-making power we need to run projects successfully? What are the issues that make managing programs successfully such a challenge?

The reasons for project failures are complex and varied, and entire books have been written attempting to diagnose the reasons. However, one factor that is especially important on larger projects has rarely been addressed – getting the big decisions right. The predominant tools and systems for large projects – such as critical path schedules, Earned Value systems, project budgeting and expenditure tracking systems – are designed for everyday, more tactical use and reporting of progress. They are of little help in addressing larger, more strategic issues and problems that often emergeon large and complex development projects and programs.

1.2. Why Is Making Big Decisions So Difficult on Large Development Programs?

A set of factors comes together to create a "perfect storm" for big decisions on large projects. First, managers are so busy with all the complex program details and meetings that there is little time to devote to big-picture thinking and strategic issues. Second, there are many formal risk registers tracking risks, but strategic issues are rarely included on these lists, lulling managers into thinking that they are systematically tracking all risks when in fact they are not. Finally, even if they had time to consider the issues and they were being tracked, their tools to analyze performance do not address many of the key dynamic factors that determine long-term performance on complex projects.

1.2.1. Issue 1 – Little Time for Strategic Thinking:

Project managers are frequently absorbed by the many important program details that must be handled for a project to succeed and much time is spent in standing meetings, responding to questions and crises, and drilling into deep dives on critical technical issues. Projects are traditionally planned or bid with aggressive schedules and as new issues emerge there is little time for extended analysis of decisions – delaying a decision is tantamount to delaying the project. The relentless march of time is a further deterrent to stepping out of day-to-day details and thinking about the larger context – projects are nearly always characterized as having a planned completion date, which tends to keep managerial attention focused on getting tasks accomplished. All of these factors lead to managers often taking quick decisions with little analysis of the strategic and long-term implications, with poor results.

1.2.2. Issue 2 – Untracked Major Issues:

On large programs in particular, strategic issues beyond the technical challenges often arise to cause performance problems and delays. These strategic issues are often overlooked either due to success-oriented thinking or since they can be uncomfortable to recognize and track publicly (e.g. What if our bid was overoptimistic and we can't deliver? What if our customer changes their priorities or redirects the project? What if our funding is cut? What if we have to divert some of our best people to another project?). As shown below in Figure 1, there are 3 broad types of major issues that tend to go untracked: Performance (e.g. efficiency challenges, aggressive plans), Contractual (e.g. late changes, funding cuts) and Workforce Related Issues (e.g. hiring policies, experience levels).

Performance Issues:	*Contractual Issues:*	*Workforce Issues:*
• Low productivity • Inability to achieve planned concurrency • Delays in a critical workstream	• Funding changes • Unplanned design changes • Changes in workscope	• Insufficient or inexperienced resources • Unexpected attrition • Production gaps (layoffs and rehiring)

Figure 1 -- Examples of Strategic Issues That Often Emerge on Large and Complex Projects

1.2.3. Issue 3 – Inability to Quantify Important Strategic Dynamics:

Managers tend to use linear, 'left-to-right' project management tools to assess risks and manage performance (i.e. Earned Value Systems, Critical Path Models, Technical Performance analyses). These tools cannot readily address nonlinearities (ie adding more and more people to a task leads to diminishing incremental efficiency), or feedback over time (see below). Strategic issues often involve feedback and are difficult to evaluate fully without capturing project dynamics. Adding unplanned work to a project is a typical strategic issue involving important feedback that is not captured fully through static analyses. Adding work to a project often has far greater cost and schedule impact than the initial direct estimates and analyses anticipate. Figure 2 below shows a simple feedback loop of how adding work increases the need for resources through overtime, hiring, and/or diversion of other resources, slowing progress on existing work and further raising the need for resources.

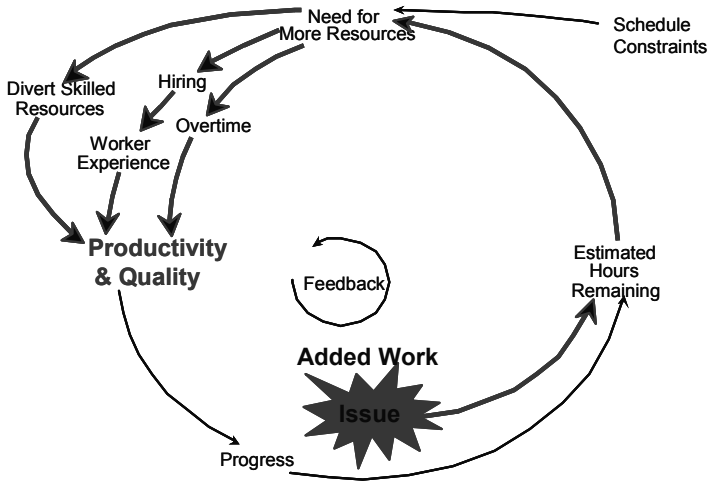

Figure 2 -- An Example of Feedback: Adding work causes a need for resources and then an even further need for resources as progress slows

The net effect of these challenges is often inadequate warning of major program problems, the need to take major decisions to respond to the problems, having little time available to make such decisions, and limited help from the available decision-making tools. Managers thus have to fall back on instincts forged by experience and make 'gut' decisions. A (very) few managers

are blessed with very accurate intuition and instincts. Most managers, however, have limited ability to intuit how the many complex factors driving a large project (interconnected work stages, team experience, morale and human factors, key project requirements, etc) will behave over time under different conditions. Moreover, since many managers gain experience on smaller and less complex projects, some of what they learn is in fact only appropriate in these settings and can actually be counterproductive on large projects. Indeed, they have "learned" the wrong lessons. For all these reasons, it should be no surprise that the track record of strategic decisions (and project outcomes) is worst on large projects.

1.3. Can We Do Better?

What is needed to improve the state of strategic decision-making on large projects, to increase the likelihood of handling major program challenges more effectively and achieve better outcomes?

1.3.1. A Potential Solution: Integrate System Dynamics with detailed program tools and systems

An integrated approach combines both the important detailed and dynamic complexities that drives program performance to help program managers do better at spotting and evaluating problems early enough to take rapid action and reduce their severity or possibly even eliminating them. Figure 3 below shows the advantages of capturing both the detailed complexities from project tools and systems along with the dynamic complexities from system dynamics models.

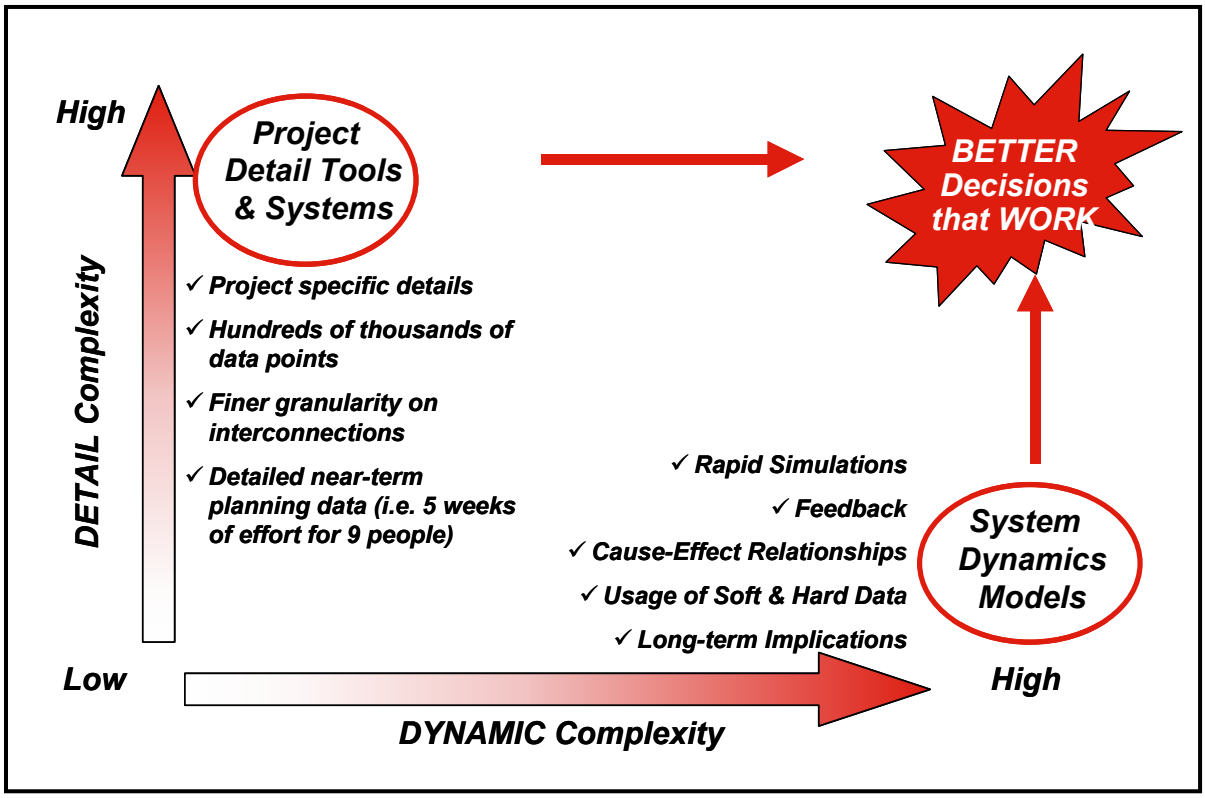

Figure 3 – Two Types of Complexity, Dynamic & Detail

Tracking a broader set of issues and bolstering analysis with System Dynamics models helps address traditional blind spots. System Dynamics cannot simply replace the more traditional tools, which are still essential for managing project details that System Dynamics generally does not address. Removing an EV system completely and replacing it with a System Dynamics model for all project teams is not realistic. The answer lies in integrating tools and connecting them where sensible to gain the best of both and save management time. Trying to analyze feedback with a typical detailed program management tool is usually done by drawing on lots of time from Subject Matter Experts (SMEs) and managers – leaving them with less time to accomplish the core work of the project. Though each individual decision or assessment may not consume inordinate resources, there are typically so many 'special' analyses during the life of a complex project, that drawing on so much SME and manager time is a huge detriment to project progress and productivity.

Dynamic simulation helps quantify and explain the key factors driving performance and helps identify the areas with greatest leverage. Detailed databases and information systems capture the technical information from SMEs on particular specifications and actions to ensure that solutions are feasible and will deliver the needed project results. Though all tools are imperfect, the combined power of dynamic simulation and detailed management tools is a large step forward. By reducing demands on SMEs and managers the time savings from key people is another benefit beyond the improvement in management information.

In the following section, we discuss an example of how we integrated a Systems Dynamic model with other tools addressing project details to solve a complex $1B problem and how the program continues to use the integrated solution on a regular basis to make smarter & faster decisions.

2. Case Example: Solving A Complex $1B Problem

2.1. A Major Development Program Facing Challenges

The (anonymized) decade-long program consists of dozens of hardware and software development areas with many hundreds of people working across the program to design, develop, integrate, and deliver multiple builds for varying users. The program was bid competitively and planned aggressively to meet user needs. In addition, the program faces the challenges of developing new technologies, working with undefined requirements, and bringing on board hundreds of new hires.

2.2. Project Team Introduction to System Dynamics

To provide the program with a strategic "what-if" tool, the program manager enlisted PA Consulting Group to develop a System Dynamics model of the program. The dynamic model would complement traditional program tools (risk tracking system, earned value, critical path project plan, financial spreadsheets, etc.) and provide rapid top-level analysis capability.

Using a spiral development approach (first building/structuring an initial model based on a database of past programs we modeled, then calibrating it to the current program plans, then expanding and refining the model gradually over time as it is used and as actual data is available), PA developed a 70+ phase (with each phase being a major release of a software build, or design of a major subsystem, or a group of production units) dynamic program model building off of prior similar models. The model represents each major area of work – capturing the major stages and activities within design, build, and test – across several companies and locations, and

identifies a dozen drivers of productivity and quality changes over time and the feedback loops driving each. It includes the interdependencies among the various phases for work product handoffs and sharing of information, staff, and other resources. The model is calibrated and refined on a quarterly basis using actual program data on labor hours spent, progress achieved, design changes, schedule slips and other key information (program changes, unplanned efforts, new management decisions on resourcing, etc.). Managers across the project have identified issues that have potential program-wide implications, which are prioritized for analysis with the program simulation model. More than two-dozen senior managers from across the different teams have been involved with model analyses to assess a broad set of program issues such as:

- What would be the impact of a reduction in funding? Which areas are the most sensitive?
- How can we build margin for the future and still achieve targets? How can we cut cost?
- Where should we allocate our resources? What level of overtime should be applied?
- What are the implications of potential delays in requirements and tools?
- What are the cost and schedule implications of a specific design change?
- What are the implications of gaps in build or design?
- What are the cost/schedule implications of different levels of concurrency across builds?
- How mature is the product? How many hours (and calendar time) are likely to be needed to mature design further?
- How much rework is left in the pipeline? What should be our planning assumptions for rework levels going forward? What are the implications of different rework priorities?

Using the dynamic model, we quantified and explained how issues in one team can cascade through feedback to impact other teams at the same time and into the future (a phenomenon often referred to as "delay and disruption").

2.3. The System Dynamics Model Helped Us Spot A Problem Early

As we combined the various program challenges and assessed the long-term program-wide implications of issues, analyses with the dynamic model increasingly highlighted a risk of major budget overruns and schedule delays. Simulations highlighted common project management challenges of inadequate resources, disconnects between teams, unrealistic plans, design uncertainty and technical feasibility issues. At this time, Earned Value reports showed SPI (Schedule Performance Index) of .99 and CPI (Cost Performance Index) of very close to 1.0 (i.e. indicating performance to date and likely future performance was within 1% of plans), and managers remained unaware of major program challenges and optimistic about future performance.

After only a few more months of additional program experience each bit of new progress data supported the dynamic analyses indicating serious future problems. Though many managers had greater faith in the Earned Value tools that they were more familiar with (and liked the optimistic view), a critical group of senior managers found our analyses credible and asked for further analysis of potential options. Using simulations, we showed that continuing to target unrealistic plans would increase program disruption and delay milestones further, and that early action would improve actual performance.

After reviewing the outlook and need for action with program managers, the decision was made to take action. A cross-disciplinary team was formed comprising project analysts, dynamic modelers, and subject matter experts, chaired by a key manager. ***We were charged with developing, in one month, a new, executable program strategy and plan, with customer buy-in.***

2.4. Solving The Problem Was A Big Task And Would Require More Than The System Dynamics Model

As part of this mandate, we would need to re-plan the program, re-negotiate requirements with customers, understand the direct and indirect impacts of many mitigations on the overall program, and select the best combination of mitigations to give the customer the most bang for their program buck. Neither the simulation model nor detailed traditional planning methods alone would be able to address all the questions and arrive at a good solution. Simulation would provide top-level insights without the ability to assure executability or prioritize specific mitigation areas according to customer preferences. Detailed project management systems alone would not be able to simulate the feedback impacts of different paths. Indeed, even attempting to develop a fully detailed plan with multiple options with high accuracy would take many months of substantial effort requiring far more person-hours and calendar time than our team had available. Dynamic analyses had highlighted the benefits of taking rapid action by reducing uncertainty, program churn, and unnecessary delays and rework. A pretty good plan in one month would be better than an excellent plan in six months … but there were many key details that needed to be addressed to ensure the plan would be at least 'pretty good'.

Never on the project had a program-wide re-plan been taken in one month; most decisions required many months of analysis and management review given the program and organization complexity and high stakes. The current program requirements and plans were still in flux and uncertain with missing details. The ground-rules for the initial strategic plan had changed, so there were many potential options. It was a daunting task to try to accomplish such a rapid re-plan with all this uncertainty, and there was need for rigor given the scope of the decision. How would we: conduct a more detailed assessment of the program outlook; get user buy-in to reduce specifications; and develop a new baseline that would meet cost and schedule targets all in one month with numerous crucial design details, interdependencies, and incomplete data.

We faced 5 key challenges that we had to work to resolve:

2.4.1. A lot of unknowns

So far the program was only 10% of the way through the overall Software development process and workscope. With sparse information on actual progress to date, how can we say with reasonable confidence what the program can feasibly achieve? What is the appropriate confidence bound? How do we know what risks lie ahead and predict "unknown unknowns"?

2.4.2. Involve multiple stakeholders

We had to get internal experts, multiple management layers, & customer buy-in. All of them had different agendas and perspectives on the problem and some saw it to their advantage to delay acknowledgement of problems and maintain that the program was still on-track, even if that meant that the re-plan effort failed. How can we coordinate activities across such a large multi-location project? How do we involve different parties and have a constructive working session to solve the problem?

2.4.3. Short Timeframe

We needed to make a rapid decision and provide the teams direction quickly to avoid inefficiencies, churn, and rework. The teams were still unclear on the end product and how to compare all the different types of functionalities being delivered.

2.4.4. Many important details

There are many important details to consider when replanning a program: Workscope, resources available, design complexity, testing cycles, technical interdependencies, future bottlenecks. All these program details existing in varying forms and in multiple spreadsheets and databases across the various teams – at different levels of detail, in different formats, and with some important information missing or out-dated. How do we combine detailed inputs and data from several project teams to get the full program picture? How to compare very different functions?

2.4.5. Many trade-offs

There were many trade-offs to consider – what should get higher priority: cost vs. schedule vs. efficiency vs. customer preference. How to deal with synergy – our customer prefers A, then B, then C, but it's much more efficient to do C before B … what's the right tradeoff? Which mitigation strategies would give the most traction both technically and politically?

2.5. Program Details And Dynamics Were Combined

A critical decision was made to connect the many important program details with the dynamic model. Data gathered would not directly feed into the dynamic model but would be held in a related system to help us select which options to input and examine with the dynamic model, and help us assess the feasibility of each option after simulating cost and schedule with the dynamic model. To accomplish all this rapidly and systematically, we decided on a four-phase approach to the solution (each phase being allotted ~1 week) that is explained in further detail below.

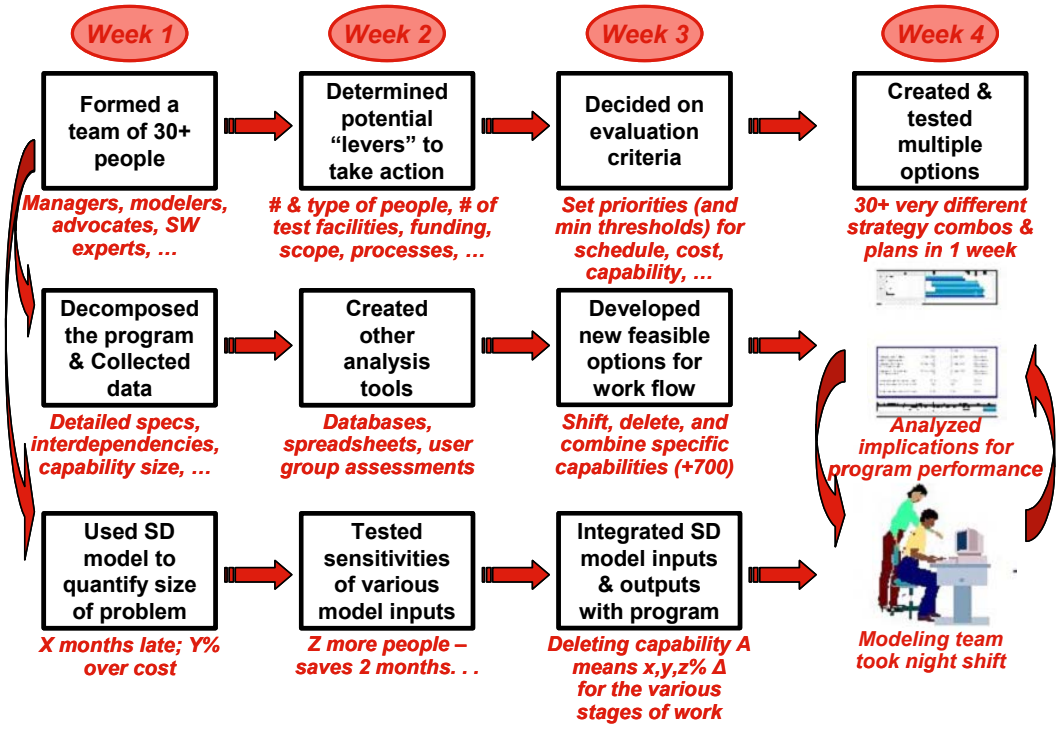

Figure 4 –Steps Involved In The Four-Phase Approach To The Final Solution

2.5.1. Phase 1 – Formed a team of 30+ people and collected critical data

We started by forming a team of thirty people with a mix of managers, system dynamics modelers, and subject matter experts (software developers, test script writers, etc.). For phase 1, the team broke into two groups: a program details group & a system dynamic modeling group.

The program details group started off by decomposing the program and collecting data:
To create a new plan that would bring the program back on track, we first had to identify and decompose the key work products (packages of software, increments of tests, and other core chucks of work involved in developing the end functionality for the users). We broke down the overall design work into over 700 core capabilities. A capability is a specific functionality of the end product (e.g design of a windshield wiper system for a car would be one capability). Then we asked all the project teams to estimate effort for each capability (usually by drawing directly on existing data systems), connected with a team of users to have them prioritize the list of capabilities, and finally engaged the SMEs (Subject Matter Experts) to identify key constraints and dependencies across capabilities (e.g. some capabilities required other capabilities to be developed first, some required special facilities or labor groups, …). We collected all this detailed information from the various teams into a database. Having all this fundamental information in one place made it possible for the project teams and users to rapidly analyze different options and understand the full scope of work and spotlight potential bottlenecks. We also asked questions by capability to understand cross impacts between teams that would help us create scenario inputs for analyses.

- How much of your team's work is related to this capability? (# of people, labor hours)
- How would delaying this capability impact your team? (added hours, increased rework)
- How uncertain are you about the scope and requirements of this capability?

This information helped us translate the workscope in the model into a form that users and SMEs could understand and use. If a new plan involved deferring two capabilities, we could rapidly calculate the resulting change in workscope in different work areas, and the implications for related teams. This generated rapid inputs that could be used in the dynamic model.

Modeling team quantified more explicitly the size of the problem we were trying to solve:
We knew from our dynamic analyses that the program was facing major performance issues and would be well over budget and delayed if it attempted to accomplish the full workscope. The solution would require removing some requirements to focus the limited resources on the most critical functionality. Using the simulations, we had assessed how much scope in aggregate would need to be removed in order to get the program back on track. At a top-level, we could explain that because productivity continues to be X% lower than plan with ongoing risks, we needed to defer Y% of the scope to achieve planned schedules and stay within annual cost budgets. But to create a specific new plan we needed to get several levels of more specific detail and make the model insights actionable. Also, enough time had passed since the original functionality had been agreed that it was quite likely that customer preferences had changed and that an updated sequence of capabilities would give the customer benefit, helping to offset the pain of deferring some functionality to the future – if we could find a way to address such detailed questions in our team and within the needed timeframe.

2.5.2. Phase 2: Determine potential levers to take action

Using the model as a framework, we discussed the various possible types of management actions and decisions to bring the program back into the permitted cost and schedule constraints (in the

team we called this "getting in the box"). These working discussions involved a full team consisting of PA consultants, Project Managers, SMEs, and Users. The goal was to create a range of possible options that were feasible and would potentially help complete the program on schedule and on budget.

We started by looking at the key levers in our "rework cycle" model and brainstorming mitigations as a full team. This helped the program team and users talk through mitigations and understand implications jointly and it helped us (the modeling team) translate potential program changes and actions into model inputs. Figure 5 below shows a simplified diagram of the "rework cycle" that we refer to in project models (a wide range of literature exists describing System Dynamics models of projects).

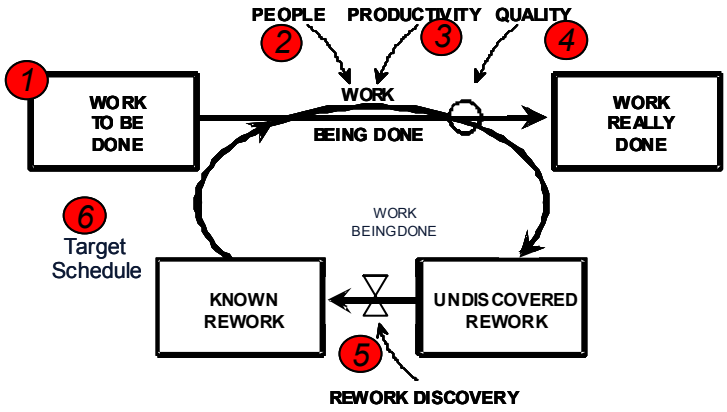

Figure 5 – Key Leverage Areas in the Project Rework Cycle

Program Model Levers:

There were 6 major categories of action that could be taken to improve the feasibility of the program plan. We examined potential changes and improvements in each area:

Lever 1: Work To Be Done

Mitigation: Reduce program workscope by deferring capabilities, institute measures to prevent/minimize typical "scope creep" and design changes

Challenge: Requires buy-in from users to reduce/refocus requirements

Lever 2: People

Mitigation: Increase resources applied through overtime, more shifts, or new hires

Challenge: Increases cost. Does not always speed progress as much as anticipated because of the negative consequences of fatigue from overtime, diluted skill base from hiring, increased congestion, less management oversight, etc.

Lever 3: Productivity

Mitigation: Increase productivity from resources (could assume either a step increase in productivity in near future or a gradual increase overtime)

Challenge: A powerful lever but very hard to achieve and deliver in short-term. Often requires investments and/or cultural change that creates initial disruption and hurts productivity. This can be easy to sign up to but is tough to deliver.

Lever 4: Quality

Mitigation: Improve quality (decrease the amount of rework generated, often by spending time to mature requirements and design).

Challenge: Even harder to achieve than productivity improvements and difficult to measure early on, but typically helps greatly (especially over the long term).

Lever 5: Rework Discovery

Mitigation: Reduce Delays in Action, Processes, and Decisions (remove bottlenecks)

Challenge: Improving speed of decision-making tends is difficult on large programs with slow bureaucratic processes. Discovering problems faster pays dividends but can require difficult and slow changes in culture (i.e. don't kill the messenger).

Lever 6: Target Schedule

Mitigation: Slip schedule now to enable more sensible resource allocation and plans.

Challenge: Not a good solution early in the program. Important to apply some pressure on project teams to perform and not provide too much schedule relief. Can be very helpful when projects are already disrupted and need to restore order. Requires customer buy-in.

We first tested each lever individually to understand the sensitivities and find out the maximum amount we would need to improve each lever alone in order to get the program on target. In every case, relying solely on a single mitigation was either too drastic or too unrealistic (i.e. too many capabilities to defer to get user buy-in). Therefore, we started creating combinations of mitigations to arrive at new feasible options. In the course of approximately two weeks, the project team created 35+ different options changing the actual capabilities deferred and assumptions about future performance. We had initially started out with half a dozen combinations but as we evaluated and discussed results more and more hybrid options were developed.

2.5.3. Phase 3: Decided on Evaluation criteria

With the full joint team, we selected criteria to evaluate program executability and compare options. This was a critical exercise for the project team and users because the tradeoffs that different parties were willing to make to get "back in the box" of project feasibility were very different (i.e. one group might be willing to increase resources while others would rather reduce functionality). The evaluation criteria also had to account for feasibility and acceptability of execution of the options and mitigations selected (i.e. are enough skilled/experienced people really available for us to hire). Even if we wanted, it would not be feasible to reduce costs, minimize delays, and keep all the functionality so we had to set a tolerance level for each criteria (i.e. we can accept up to a 5% cost growth to achieve schedule).

Using the model outputs and detailed data on user group priorities, we were able to then assess each option and provide an overall score based on critical factors to determine if we were getting close to being "back in the box". Outputs combined and summarized results covering both the program details (user group preference, execution risk) as well as results from the dynamic program model (program hours spent, program completion, software cost, etc.).

Examples of outputs used to summarize results from the dynamic program model and detailed program SMEs/tools for the various options are shown in Figures 6 and 7 below.

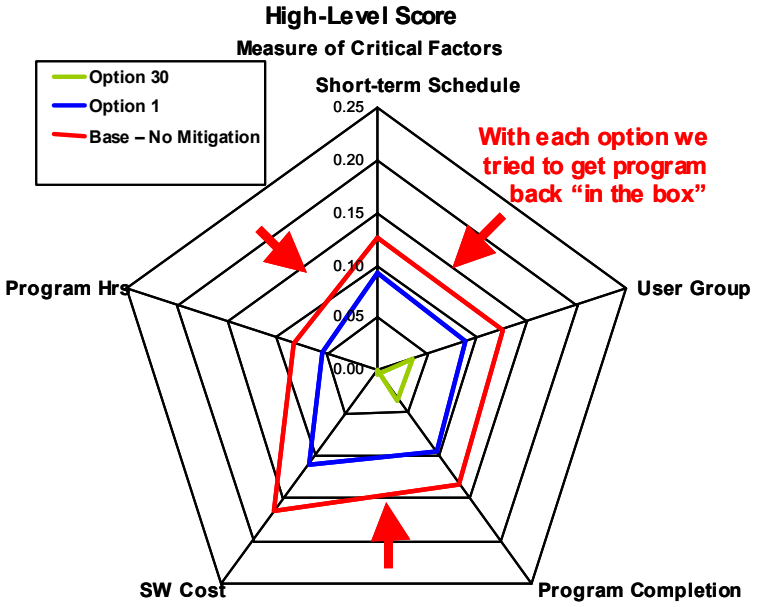

Figure 6 – Output Used in Evaluating Multiple Criteria for Each Option
(Scores closer to the center are best)

Options	Base No Mitigation	1 Option 1	2 Option 2	5 Option 5	10 Option 10	35 Option 35
Months of Delays (Green = X months or less; Yellow =X-Z months; Red = greater than Z months)						
Phase 1	Yellow	y	yy	yy	yy	yy
Build 1 Delivery	Red	y	Yellow	yy	Yellow	yy
Program End	Red	Yellow	Yellow	yy	yy	yy
% Hours Growth (green = X% or less; yellow = range X-Z%; greater than Z%)						
SW Hrs	Red	Yellow	z%	Yellow	z%	z%
Progran Hrs	Yellow	z%	z%	z%	z%	z%
Divergence from user priorities (1 or less; 1-1.25; greater than 1.25%)						
Normalized to Base	1.00	zz	zz	zz	Red	zz
Overall Scope Change (Green X% or less; Yellow = X-Y%; Red = greater than Z%)						
Normalized to Base	0%	Yellow	Yellow	Yellow	Red	Yellow
Execution risk						
Normalized to Base		Red				

Figure 7 – Matrix Comparing Multiple Options: *(Traffic lights – green is good, yellow is warning, red is trouble)*

2.5.4. Phase 4: Evaluating options and assessing impacts rapidly

The speed of simulation analysis was critical when the management team considered new options and mitigations. The project and user team would brainstorm options during the day and

the modeling team would run simulations off to the side and at night to test results. The model was used to rapidly assess both the cost (overall change in program labor hours) and schedule (months of schedule impact) implications of varying options. At times simulations showed counter-intuitive results and highlighted that typical management strategies such as increasing overtime and recruiting new resources had negative implications for team productivity and quality levels. The ability to quantify such factors, test ranges, and explain this clearly via simulation was critical to reconciling the differing opinions and intuitions across the team, especially given the different organizational priorities and politics represented.

The detailed information database was critical in figuring out which capabilities to defer and how that would impact the program. The user group could defer 10 easy capabilities and reduce overall program workscope by less than 1% or they could defer 1 complex capability and reduce actual workscope by greater than 5%. With over 700 capabilities and complex interdependencies between them it was critical to have the detailed information in an easily accessible manner as we narrowed down to a final solution.

2.6. Final Decision Was Made Quickly With Confidence and Rigor

After assessing various strategies, we decided on an option that got the program back on track by a combination of deferring and shifting capabilities, increasing productivity through organizational changes, investing in facilities to reduce bottlenecks, providing some schedule relief, increasing overtime slightly and hiring select experienced resources. Having involved the users, we had achieved buy-in along the way and jointly developed the final solution. Using the detailed data and SMEs (Subject Matter Experts), we strengthened our recommendations to be specific and content rich. With the simulation, we could back-up our strategy, show the structured framework and rigor, and explain to leadership and others the need for change.

The final solution was specific enough to be useful – cut Capability A by 15%, apply 80 more people in this specific area, move out B,C,& D capabilities 2 years from now, add a new testing capacity in this area, etc. It was comprehensive enough to address all key aspects of the program – design, development, test, support. Moreover the new plan was understood and owned by all the key parties – SW experts, test team, user group, management, customer, and analysis experts.

2.7. Proof of Success

Providing a rapid integrated solution that was accepted by all the parties and carried forward was the key battle to win. A rapid solution (only one-month) was critical in giving the project teams direction and making sure the program stayed on track instead of continuing on a downward spiral. Two other measures of success were that the process/techniques used for analysis have been institutionalized for ongoing use and the team received overall program recognition.

2.7.1. Institutionalized Process:

It was realized at the time that with time as new data comes in and user preferences change, our plans would need to be updated. So it was decided that the every year we would go through the same one-month exercise to assess the outlook and make any changes/refinements that are needed to the plan.

2.7.2. Team Recognition:

After all the hard work, the team was gratified to be thanked for their efforts and results. According to a letter from the overall leader of this multi-billion dollar project:

"I would like to recognize the dedication and outstanding performance of the ... [Project & PA Consulting Team] over the past month ...
... Working long hours and several all-nighters with the Program Office they were able to balance cost and schedule constraints while maximizing benefit to the [users].
The team made excellent use of the Program Simulation Model and disciplined systems engineering to ensure broad coverage of [our] complex design. Over 35+ different [project] options, varying in content, execution risk, and staffing constraints were simulated to obtain optimal solutions that would fit 'in the box'. ***Their work will undoubtedly play a critical role in shaping future ... capability …"***

3. Conclusions:

"Simplify as much as possible. But not more."
Albert Einstein

Projects are full of details – important details that must be addressed for a project to be successful. Thus project tools naturally focus on these details to help management teams get them all resolved. For strategic decisions, however, traditional tools are overcomplicated in some areas – such as including dependencies that are trivial when considering overall project performance – and oversimplified in others, as they omit the key dynamic details (such as lower productivity and quality when two work stages are done more concurrently than initially planned). Extracting the critical interdependency information from existing tools makes project detail more accessible and usable. Connecting these details to a System Dynamics model fills in the blind spots of dynamic oversimplification to add crucial insight on the long-term impacts of different options. Together, these tools provide a powerful, useful aid for strategic decision-making on complex projects. We see a bright future using integrated approaches to provide a new level of capability to strategic decision-makers to make faster and smarter decisions quickly to mitigate potential cost and schedule overruns.