IT Project Management, Concept Modeling, and Blind Dates

Eliot Rich
University at Albany
State University of New York
School of Business
1400 Washington Avenue
Albany, NY
rich@acm.org

Mark Nelson
Digital Content Strategist
National Association of College Stores
500 East Lorain Street
Oberlin, OH 44074
mnelson@nacs.org

Andrew Whitmore
University at Albany
State University of New York
College of Computing and Informatics
1400 Washington Avenue
Albany, NY
ajw367@gmail.com

Abstract

This paper describes a recent activity where scholars from two disciplines met to consider the reasons behind a three-decade long attempt to modernize a critical IT application in the US government. From the system dynamics perspective, the problem appears to be related to dynamic and repeating management failures with an embedded project management model. We decided to use a simple project model to develop this perspective. Our initial attempt shows some of the anticipated dynamics of project escalation. More important, though, was the discussion of the problem itself that was launched by the use of a formal model. We believe that this approach has provided insight into how to approach a more focused and grounded analysis of the problem domain.

Keywords: Project Management, Government Information Systems, Project Commitment

Introduction

This paper describes a recent meeting where scholars from two disciplines met to consider the phenomenon of IT software project escalation and de-escalation. We thought of it as a "blind date," as the authors agreed to get together to discuss the problem, not being sure what was being brought to the table, and not knowing what would result. From the system dynamics perspective, the problem appears to be related to a dynamic and repeating decision and management process with an embedded project management model. We decided to use a simple project model to develop this perspective. It has been advocated that such simple models may be used as a mechanism to introduce the concepts of stock and flow to unfamiliar audiences (Richardson & Andersen, 1995) and establish grounds for future communications. Here we present our initial attempt to adapt an existing model to this end.

The problem environment, a very large and complex government software development project, has borne several launches and crashes, each accompanied by investigations, management inquiries, and little progress. The problem environment is reminiscent of a well-behaved and simple concept model of projects (Richardson & Pugh, 1981) with two new structures: The influence of "scope creep," a pressure to add functionality to the project, as well as a decision rule for project termination. As anticipated, adding changes to the requirements set increases the resources needed during the life of the project. This in turn triggered the decision rule that caused premature termination.

The use of a simple and illustrative concept model helped in the formulation of a more thorough set of research objectives and establishes a basis for dialog between scholars of organizational behavior and IT project management. It also identified critical model conceptualization questions that might be addressed by a detailed examination of the existing archival dataset. There is a downside, however, as the chosen concept model captured only one important dynamic behavior, and will likely need significant revision to address other important problems. Thus, the more that is expected of the concept model, the harder it is to keep it simple and elegant.

The ultimate goal of this research stream is to integrate descriptive research of escalating projects with the feedback based insights available from dynamic modeling. This combination may provide understanding of the effects of firm capabilities, policy changes, and scheduling conflicts on project success. Working with simple project models is the first step towards an elaborated model, grounded through retrospective analysis of project materials, to learn more about IT development in organizations under stress.

Background

Our research examines a case of extended failure to complete a critical and complex IT modernization effort in a US government organization. This modernization project, still incomplete, has spanned three decades and has cost over $14 Billion in its various incarnations (see, for example, Varon, 2005). The project has been revamped, stopped and restarted several times, reflecting a continuing problem that until recently appeared intractable (Holmes, 2006).

This cyclical escalation and de-escalation of the development effort is of great interest to scholars of organizational behavior and decision-making. This may be evidence of the development of organization rigidity in decision-making, causing a commitment of resources well after the project's goals are actually unachievable, followed by a disengagement, restructuring and re-starting of a new attempt to achieve the old or revised goals.

Multiple studies have called for longitudinal investigation of escalating and de-escalating commitment in a field-study context (Brockner, 1992; Staw, 1997; Staw & Ross, 1987), however, few studies have taken a longitudinal perspective focused on the complex interaction processes involved. In experiments with individuals, Staw and Fox (1977) found that once engaged, de-escalation processes should prevent further escalation as investment continues. However, in long-term, large-scale IT projects where the project goal or objectives persist, redirection may change commitment to the short-term course of action, but not to the long-term course of action. When such projects that involve high levels of investment are redirected, conditions exist that may produce multiple cycles of escalation and de-escalation (Montealegre & Keil, 2000). Supporting this notion, Drummond (1994) found that escalation of commitment is different in situations where individuals inherit unsuccessful and long-running projects, and that such situations experience a cyclical escalation effect. However, Drummond's study was not conducted in an IT-project context. Newman and Sabherwal (1996) provide perhaps the best evidence as to what happens to these projects over multiple decision points. In their longitudinal analysis of a large-scale IT project that repeatedly failed they demonstrated how determinants of commitment among individuals shift from escalation to withdrawal of commitment and back over time. In doing so, they found the dynamics among escalation determinants to be different from much of the prior research. These latter two studies demonstrated that escalation and withdrawal of commitment share a more complicated and dynamic relationship over time and provide support for conducting additional longitudinal field studies of these processes.

These studies led us to consider a systems perspective, where the underlying structures of project management under dynamic requirements contributes to both escalation of commitment and de-escalation. Studies of project dynamics often focus on unrecognized misconceptions of project progress and subsequent rework cycles as the key to understanding budget and schedule overruns (Cooper, 1980; Reichelt & Lyneis, 1999). Application of this perspective to the specific issues of software has been documented and extensively tested by Abdel-Hamid and Madnick (1990, 1991). Their ideas have been disseminated widely in the software development literature, with several hundred citations since its original publication two decades ago[1]. Their work is quite well developed, but is challenging for those new to dynamic modeling. Therefore we chose to begin this analysis with an illustrative approach using a simpler project model, looking for a broad conceptual understanding.

The empirical grounding for this work comes from examination of IT modernization efforts at the U.S. Internal Revenue Service (IRS) spanning a period over thirty years. Using archival analysis techniques we examined documents from a variety of sources both inside and outside the IRS to adhere to guidelines for gathering historical evidence. Primary source material included over 750 historical documents, including audits, reports, memos, project documentation, media documents, congressional testimony, as well as previous academic studies. Photographs of early operations provided contextual artifacts of the operational conditions within the agency. After completion of the archival analysis, our findings were reviewed and refined through a small number of confirmatory interviews with individuals familiar with the relevant IRS projects and public sector IT project management.

[1] A recent Google Scholar search found almost 400 references to the Abdel-Hamid and Madnick text.

The IRS efforts at modernization provide a good context for studying large-scale IT projects in the public sector for several reasons. First, the IRS is considered to be a very well-managed agency by IT researchers (Bozeman, 2002) and if such a project were to be successful anywhere, one might expect that to happen at the IRS. Second, the IRS made four distinct major attempts to modernize since the late 1960s. The first three attempts clearly failed and were abandoned at a combined cost over $4 billion (in nominal value): the Tax Administration System Project from 1969 to 1977; the Service Center Replacement System project from 1978 to 1986; and the Tax Systems Modernization project from 1987 to 1997. Along the way, there were over 40 other projects aimed at initiating or furthering agency modernization, most of which were also abandoned as failures (*ibid*) The fourth attempt, the Business Systems Modernization Project, began with the Blueprint Project around 1998 and is still underway with an estimated cost in excess of $10 billion (Varon, 2004). While experiencing some success, the current attempt is already significantly over budget and behind schedule (Johnston, 2003). Third, the various iterations of the IRS modernization effort received wide-scale attention in the media, in oversight organizations, and within the agency itself. The ease in acquiring longitudinal historical data on the project at a relatively low cost, and the nature of the IT project itself, made this an ideal case to study to learn more about abandonment processes in large-scale public-sector IT projects.

Problem Context

The Internal Revenue Service received 228 million citizen and corporate tax returns in 2006, somewhat larger than the 202 million received in 1995 (U.S. Department of The Treasury, 1995, 2006). Taxes on personal incomes, including social insurance programs, represent the bulk of resulting revenue. Along with population pressure, this growth reflects the increasing complexity of the tax code, as tax brackets have been modified and new programs implemented to capture income streams for governmental operations.

The current information systems supporting the IRS date back to the late 1950's, and reflected the architecture and data processing methods available at that time. Much of that architecture is still in place, relying on obsolete hardware and software of vertigo-inducing complexity. Each modernization effort was pledged to untangle the code and bring modern approaches and technology to bear.

As is often the case, the IRS modernization efforts are deeply enmeshed in a changing context that is not in their control (Figure 1). Congressional and executive branch policymakers introduce innovations that reflect their concerns about the current and future state of the economy. Changes to tax laws and modification in enforcement priorities engender alterations to the same software systems that are undergoing modernization. The wrapping of new policy requirements with those needed to replicate the processing already in place greatly increases the complexity of change. Even the simplification of the tax code during 1980s did not result in compliance savings from the taxpayer perspective (Slemrod, 1992) or the IRS, as the concurrent presence of old and new tax laws in the same system was required.

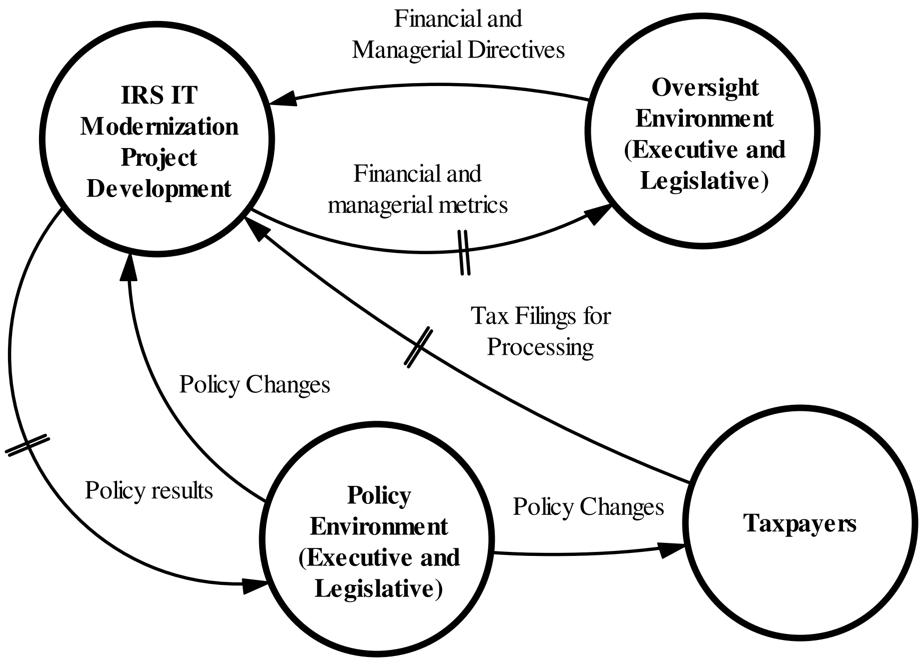

Financial and Managerial Directives
IRS IT Modernization Project Development
Oversight Environment (Executive and Legislative)
Financial and managerial metrics
Tax Filings for Processing
Policy Changes
Policy results
Policy Environment (Executive and Legislative)
Policy Changes
Taxpayers

Figure 1 IRS IT Modernization Project and Context

Framing the Hypothesis

The recurrent nature of failure has led some critics to brand the IRS's modernization efforts as characteristic of a failing organization that fails to learn from its experience. In this paper we raise an alternative perspective: To what extent does the interaction among the various components of the IRS modernization context affect the likelihood of project success? Does the complexity of the problem, in tandem with a changing policy and oversight process, create a situation that is impossible to resolve?

The changing policy and oversight processes to which the IRS modernization efforts are subjected help produce one of the familiar banes of project management: scope creep. Scope creep, or the altering or adding of requirements to an ongoing project, has been shown to adversely affect the likelihood of a project finishing on time and on budget (Wallace & Keil, 2004). Scope creep can arise through factors such as poor requirements definition and is often addressed by a change control board that establishes change policies and the approval process for those changes (Wiegers, 2003). However, the IRS modernization efforts differ from private sector IT projects in terms of the nature and source of the scope creep. While project managers employed by private corporations perceive requirements and scope risks as highly controllable through good project management (Wallace & Keil, 2004), public organizations, like the IRS, often have requirements changed or new requirements introduced exogenously from higher level federal bodies or changing legislation. Under these conditions, project managers at the IRS will perceive the scope creep problem as uncontrollable which will further increase the project's execution complexity and difficulty (Wallace & Keil, 2004) We argue that the deleterious effects of repeated, uncontrollable scope creep combined with an already complex project position the IRS for failure in its attempts to modernize.

As a first step, we consider one of the areas of complexity that arose in the IRS projects: The need to implement changes to tax policy in software while attempting to replace existing applications. This can be phrased as a simple dynamic hypothesis the effects of changing requirements on the characteristics of project success.

Hypothesis 1: Increasing the number of user requirements introduced during the progress of an IT project will increase resource use and likelihood of project termination.

This hypothesis combines the insights from project management models about project outcomes (Abdel-Hamid & Madnick, 1990, 1991; Cooper, 1980) with the provision of information about the future of the project. This provides a linkage to thinking about project escalation and de-escalation.

What defines success in a large-scale implementation project? It is not on time and under budget completion. At the outset of a project, while managerial expectations near absolute certainty of success based on experience, historical analysis says that projects still have an a priori risk of failure from unforeseen causes. The Standish Group reported in 1994 that only 16.2% of projects meet that exacting criterion, with a smaller fraction in large projects (The Standish Group, 1994). 31.1% of projects are terminated before they reach completion. It appears that the respondents to their survey measured success pragmatically, with "complete" projects that have only 42% of the original features, and most projects running 80% or more over budget. While this report is dated, and there can be some concern about its current applicability, it presents a measuring stick for evaluating project success through the eyes of IT managers.

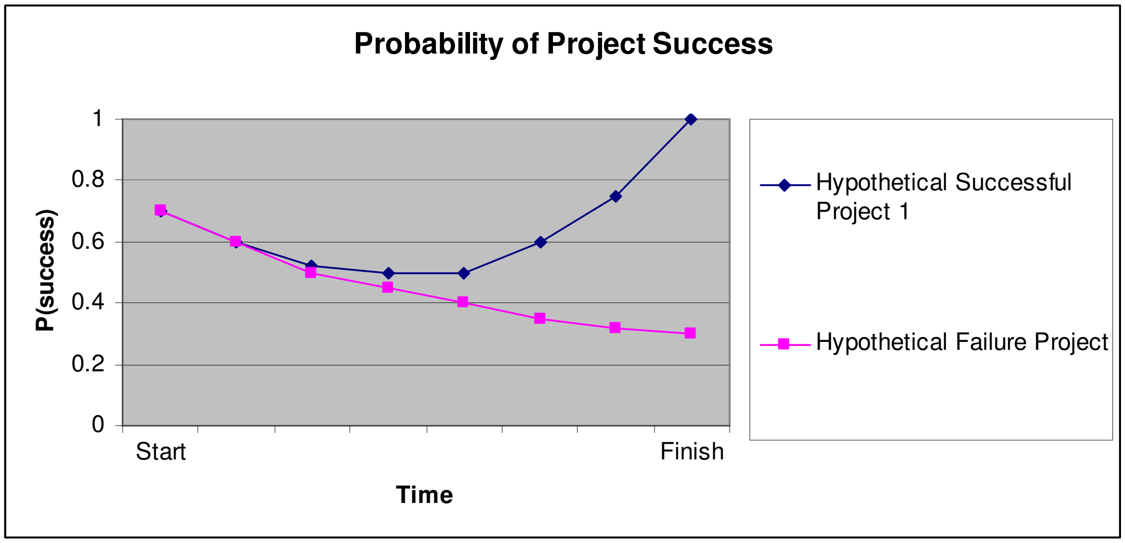


Figure 2: Project Risk Profiles

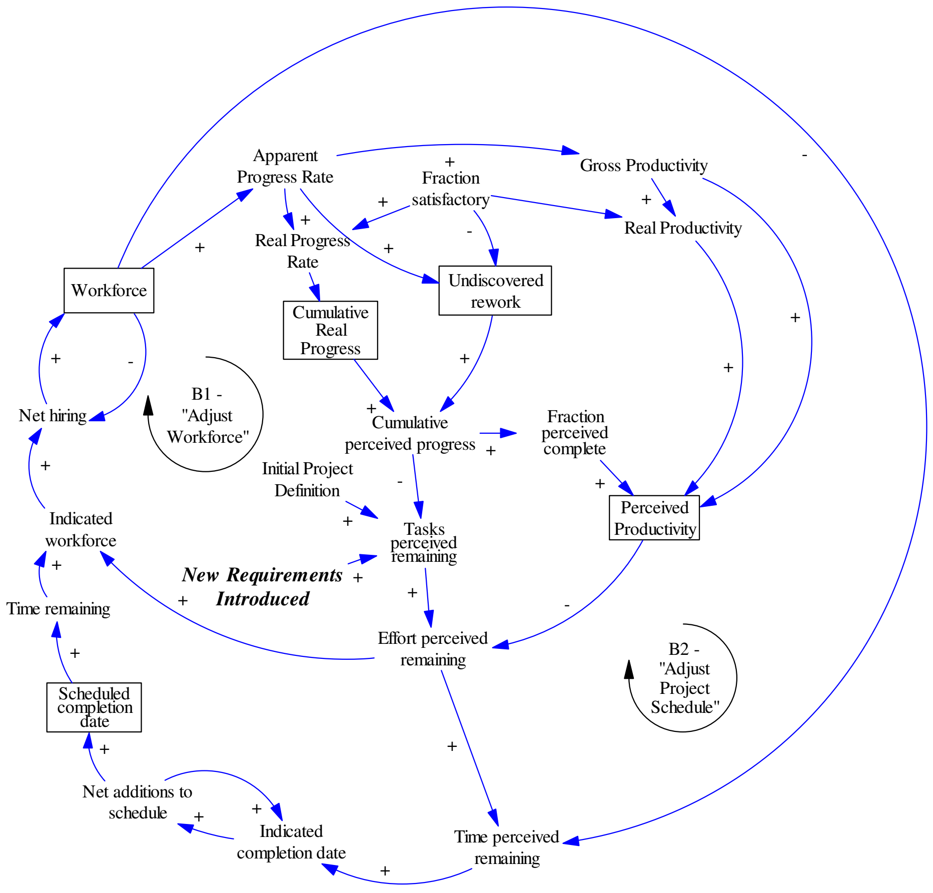
The construction of a metric for success or failure is a necessary linkage between the project models and the oversight environment. We would expect to find that even successful projects run into risky periods and persevere through to completion, while unsuccessful projects find themselves in situations where risk compounds without resolution (Figure 2). While efforts are being made to predict project complexity at its beginning (Xia & Lee, 2004), the effects of complexity often emerge during the details of software design and integration. Projects may well tip into a failure mode where insiders recognize the inevitable, but the news has not reached their managerial or oversight counterparts. In addition, even with timely reporting of problems, managers may continue their escalatory behaviors in the face of psychological and social pressures as well as structural issues among stakeholders .

The definition of organizational tolerance for failure is not cut and dry, as it is quite possible to imagine situations where organizations see the slim chance of project success better than the consequence of stopping, or where the achievement of substantial portions of the work is considered sufficient for its purpose. For this work we assume that a project that reaches 80% of the established requirements within 150% of the initial timetable or 150% of the initial resource estimate (in staff hours) is successful.

Model Structure and Simulation

The baseline model for this experimentation is the project model presented in Richardson and Pugh (1981). The model is a simple and elegant illustration of feedback, used throughout the text as a vehicle for introducing various concepts relating to the development, testing, and documentation of a model. It consists of two major balancing loops with some minor delay structures (Figure 3). The model starts out of equilibrium, as there are fewer staff members than needed to complete the volume of tasks; as the model runs staffing is adjusted to meet task demand, which in turn is affected not only by the staff effort but by the growth of undiscovered rework. Its simplicity makes it a good candidate for extension into a model of project failure.

Figure 3: Structure of Simple Project Model (Adapted from Richardson and Pugh, 1981)


Apparent Progress Rate
Gross Productivity
Fraction satisfactory
Real Productivity
Real Progress Rate
Workforce
Undiscovered rework
Cumulative Real Progress
B1 - "Adjust Workforce"
Net hiring
Cumulative perceived progress
Fraction perceived complete
Initial Project Definition
Perceived Productivity
Indicated workforce
Tasks perceived remaining
New Requirements Introduced
Time remaining
Effort perceived remaining
B2 - "Adjust Project Schedule"
Scheduled completion date
Net additions to schedule
Indicated completion date
Time perceived remaining

The published model presents a small project of 1200 tasks to accomplish and a "fraction satisfactory" of 0.7, representing the unforeseen errors that crop up during development. To simulate the introduction of incremental requirements during the project we add "*New Requirements Introduced*" to the original model, a structure that pulses an additional 5% task load every six months. This is a rough attempt to consider the generation of changes as driven by a policy or legislative cycle, rather than as a purely random process. A second structure (Figure 4) represents when a project is identified as a failure based on the decision rule presented earlier. Once the threshold for project failure has been reached, resources are removed from the project. In this very simple model the decision rule is treated as a discrete event, rather than as a more gradual development of consensus.

Simulation runs present the effects of this initial change in requirements. The *Base* run replicates the results of the published model, and the *More Requirements* run adds the task load (Figure 5) to the project definition. The *No Cancel* run depicts the same parameters as the *More Requirements* run, if the project decision rule was not active. As one might anticipate, the increase in requirements changes the total resources anticipated for the project. In this model, the endogenous adjustment to requirements change increases staffing rather than adjust the project schedule, where staffing levels off at 53 persons, up from the 43 needed with the original project requirement set.

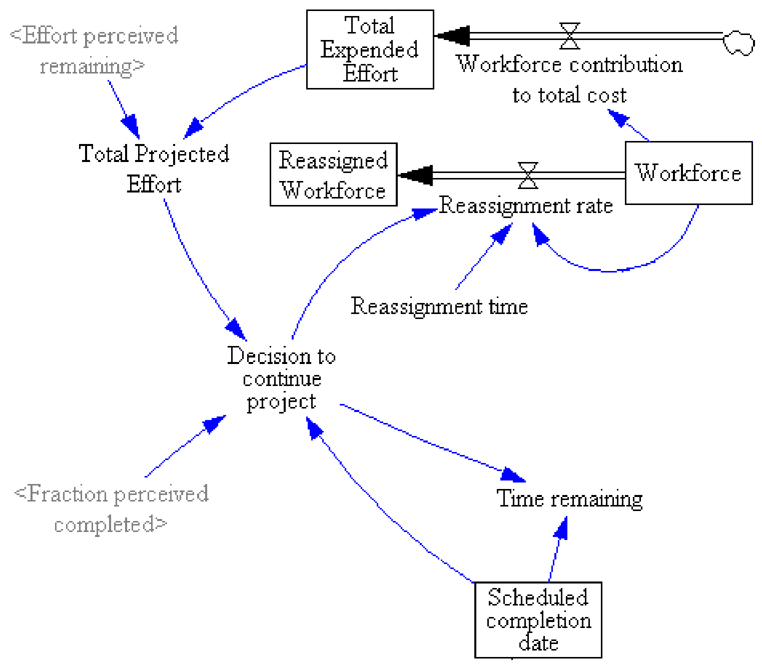

<Effort perceived remaining>
Total Expended Effort
Workforce contribution to total cost
Total Projected Effort
Reassigned Workforce
Workforce
Reassignment rate
Reassignment time
Decision to continue project
<Fraction perceived completed>
Time remaining
Scheduled completion date

Figure 4: Decision Rule for Project Cancellation

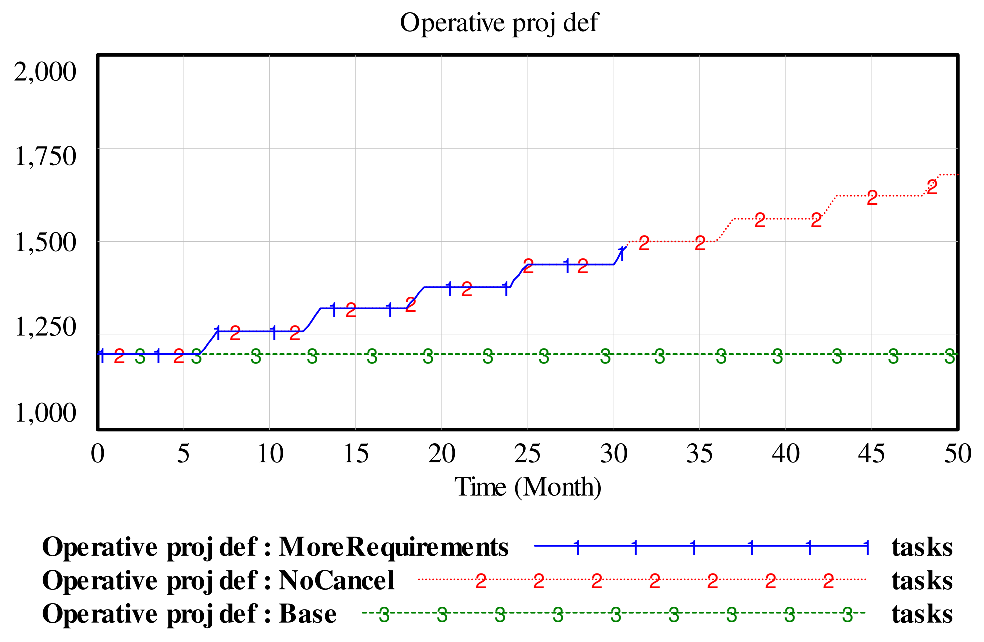


Figure 5: Project Size

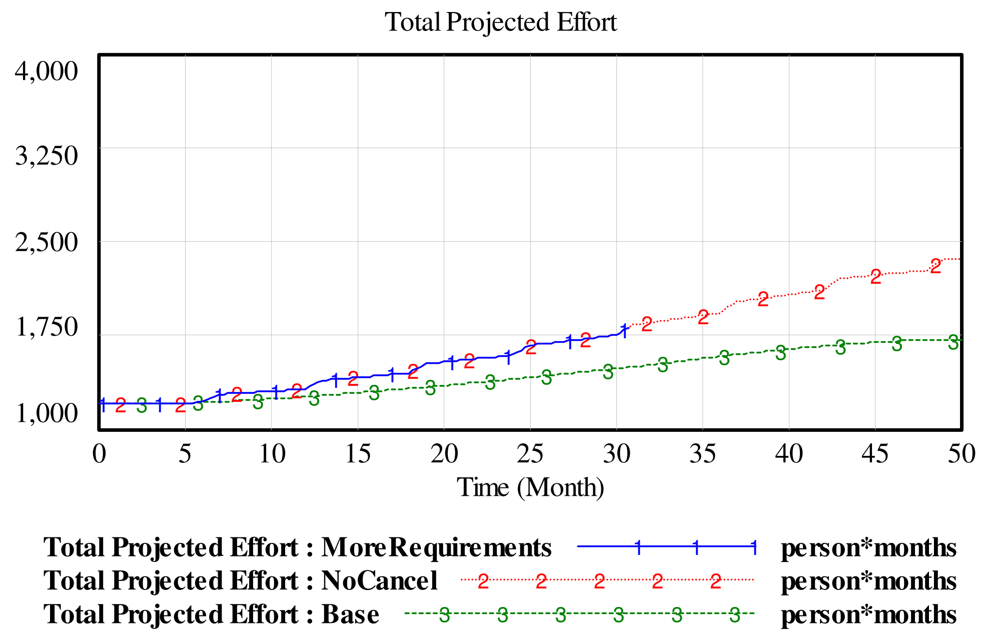


Figure 6: Total Projected Effort

The run-up in desired resources for the project is a harbinger of bad things, however, in that once the project staffing increases the projection for future costs goes up as well. Assuming that the new resources will remain on the project drives the total projected effort to 1,835 person months, versus 1,500 for the base case (Figure 6). This takes the *More Requirements* scenario beyond the decision threshold, and the run ends. If the run was not cancelled, the resource gap would continue to widen, as new requirements would continue to come into the project.

Observations and Next Steps

From a dynamic modeling perspective the results of this effort are not surprising, and are in line with intuition. Nevertheless, couching the problem in terms of a formal model is a useful starting point for examination of the repeated failures of the IRS project by showing in a dramatic way the rationale for consideration of endogenous effects and outcomes. Using an existing concept model gave substance to discussions about the effects of unforeseen rework on project outcomes.

The model did not always behave in ways that reflected the IRS reality. For example, these runs reflect a stiffness that emphasizes staff changes over schedule changes. Rather than tweak the model, though, we decided to use it as a basis for discussion of the underlying problem. This choice is illustrates the primary result of the effort. The discussion exposed several questions about the escalation – de-escalation phenomenon that the model does not address, regarding staffing, productivity, increasing project complexity, and information delays. We also learned a bit about ways to organize and examine the mass of documentary data that has already been collected. The list of questions is growing faster than we anticipated.

There are some cautions as well needed here. We need to be sure that we identify and address the problems seen in the IRS data, not just those that are present in the concept model. There are other alternative published formulations (e.g., those developed in Abdel-Hamid & Madnick, 1991) that might well serve as guideposts for modeling repeated project failures. The choice of concept model can prematurely impose an inappropriate conceptual lens on the problem. Like any good blind date, a successful discussion leads to a promise of things to come. The next step in this work is to consider how to proceed.

Appendix: Model Equations

```
Apparent progress rate=
        Workforce*Gross productivity
        ~       tasks/Month
        ~               |

Cumulative perceived progress=
        Cumulative Real Progress+Undiscovered Rework
        ~       tasks
        ~               |

Cumulative Real Progress= INTEG (
        Real Progress Rate,
                0)
        ~       tasks
        ~               |

Decision to continue project=
        IF THEN ELSE(((Fraction perceived completed<0.8):AND:(Scheduled completion
date>=60)\
                :OR: (Total Projected Effort>=1800)) :AND: (swCancelOverride = 0) ,
0, 1)
        ~       dmnl
        ~               |

Detection of undiscovered rework=
        Undiscovered Rework/Time to detect rework
        ~       tasks/Month
        ~               |

Effort perceived remaining=
        Tasks perceived remaining/Perceived Productivity
        ~       person*months
        ~               |

Fraction perceived completed=
        Cumulative perceived progress/Operative proj def
        ~       dmnl
        ~               |

Fraction satisfactory=
        0.7
        ~       dmnl
        ~               |

Generation of undiscovered rework=
        Apparent progress rate*(1-Fraction satisfactory)
        ~       tasks/Month
        ~               |

Gross productivity=
        1
        ~       tasks/person/Month
        ~               |

Indicated completion date=
        Time+Time perceived required
        ~       Month
        ~               |

Indicated productivity=
        Weight given real productivity*Real productivity+(1-Weight given real
productivity)*\
                Gross productivity
        ~       tasks/(Month*person)
        ~               |

Indicated Work Force=
        ZIDZ(Effort perceived remaining,Time remaining)
        ~       persons
        ~               |

Initial project definition=
        1200
        ~       tasks
        ~               |

Net additions to schedule=
        (Indicated completion date-Scheduled completion date)/Schedule Adjustment Time
        ~       dmnl
        ~               |

Net Hiring Rate=
        (Workforce sought-Workforce)/WorkForce adjustment Time
        ~       persons/Month
        ~               |
New requirements introduced=
	(PULSE TRAIN(Update period,1,Update period,50))*Requirements generation
	~	tasks/Month
	~		|

Noise=
	0
	~	dmnl
	~		~	:SUPPLEMENTARY
	|

Operative proj def= INTEG (
	New requirements introduced,
		Initial project definition)
	~	tasks
	~		|

Perceived Productivity= INTEG (
	(Indicated productivity-Perceived Productivity)/Time to perceive productivity,
		Indicated productivity)
	~	tasks/person/Month
	~		|

Real productivity=
	Fraction satisfactory*Gross productivity
	~	tasks/person/ Month
	~		|

Real Progress Rate=
	Apparent progress rate*Fraction satisfactory
	~	tasks/Month
	~		|

Reassigned Workforce= INTEG (
	Reassignment rate,
		0)
	~	persons
	~		~	:SUPPLEMENTARY
	|

Reassignment rate=
	IF THEN ELSE( Decision to continue project=0 , Workforce/Reassignment time ,
0*Workforce\
		/Reassignment time )
	~	persons/Month
	~		|

Reassignment time=
	3
	~	Month
	~		|

Requirements generation=
	60
	~	tasks/Month
	~	RANDOM UNIFORM(100,600,Noise)
	|

Schedule Adjustment Time=
	6
	~	Month
	~		|

Scheduled completion date= INTEG (
	Net additions to schedule,
		40)
	~	Month
	~		|

sw New Requirements Introduced=
	0
	~	dmnl [0,1,1]
	~		|

swCancelOverride=
	0
	~	dmnl
	~		|

Tasks perceived remaining=
	Operative proj def-Cumulative perceived progress
	~	tasks
	~		|

Time perceived required=
	Effort perceived remaining/Workforce sought
	~	months
	~		|

Time remaining=
	Decision to continue project*MAX((Scheduled completion date-Time),0)
	~	Month
	~		|

Time to detect rework=
	IF THEN ELSE( Decision to continue project=0 , 1e+009, TTDRWF(Fraction
perceived completed\
		))
	~	months
	~		|

Time to perceive productivity=
	6
	~	Month
	~		|

Total Expended Effort= INTEG (
	Workforce contribution to total cost,
		0)
	~	person*months
	~		|

Total Projected Effort=
	Effort perceived remaining+Total Expended Effort
	~	person*months
	~		|

TTDRWF(
	[(0,0)-(1,20)],(0,12),(0.2,12),(0.4,12),(0.6,10),(0.8,5),(1,0.5))
	~	months
	~		|

Undiscovered Rework= INTEG (
	+Generation of undiscovered rework-Detection of undiscovered rework-Detection
of undiscovered rework\
		,
		0)
	~	tasks
	~		|

Update period=
	IF THEN ELSE (sw New Requirements Introduced,6, 999999)
	~	months
	~	IF THEN ELSE (sw New Requirements Introduced,RANDOM \
		UNIFORM(5,15,Noise),999999)
	|

WCWFF(
	[(0,0)-(40,1),(0,0),(3,0),(6,0),(9,0.1),(12,0.3),(15,0.7),(18,0.9),(21,1),(40,1)],(0\
		,0),(3,0),(6,0),(9,0.1),(12,0.3),(15,0.7),(18,0.9),(21,1),(40,1))
	~	dmnl
	~		|

Weight given real productivity=
	WGRPF(Fraction perceived completed)
	~	dmnl
	~		|

WGRPF(
	[(0,0)-(10,10)],(0,0),(0.2,0.1),(0.4,0.25),(0.6,0.5),(0.8,0.9),(1,1))
	~	dmnl
	~		|

Willingness to change WorkForce=
	WCWFF(Time remaining)
	~	dmnl
	~		|

Workforce= INTEG (
	Net Hiring Rate-Reassignment rate,
		2)
	~	persons
	~		|

WorkForce adjustment Time=
	3
	~	Month
	~		|

Workforce contribution to total cost=
	Workforce
	~	persons
	~		|

Workforce sought=
	Willingness to change WorkForce*Indicated Work Force+(1-Willingness to change
WorkForce\
		)*Workforce
	~	persons
	~	Note:  Never goes negative, so don't need the max function.
	|

********************************************************
	.Control
```

```
********************************************************~
		Simulation Control Parameters
	|

FINAL TIME=
	IF THEN ELSE (Decision to continue project = 0, Time, 50)
	~	Month
	~	The final time for the simulation.
	|

INITIAL TIME  = 0
	~	Month
	~	The initial time for the simulation.
	|

SAVEPER  = 0.25
	~	Month [0,?]
	~	The frequency with which output is stored.
	|

TIME STEP  = 0.25
	~	Month [0,?]
	~	The time step for the simulation.
```

References

Abdel-Hamid, T. K., & Madnick, S. (1990). The elusive silver lining: How we fail to learn from software development failures. *Sloan Management Review*(Fall).

Abdel-Hamid, T. K., & Madnick, S. (1991). *Software Project Dynamics: An Integrated Approach*. Upper Saddle River, NJ: Prentice-Hall.

Bozeman, B. (2002). *Government Management of Information Mega-Technology: Lessons from the Internal Revenue Service's Tax Systems Modernization*. Arlington VA: The PricewaterhouseCoopers Endowment for The Business of Governmento. Document Number)

Brockner, J. (1992). The Escalation of Commitment to a Failing Course of Action: Toward Theoretical Progress. *Academy of Management Review, 17*(1), 39-61.

Cooper, K. G. (1980). Naval Ship Production: A Claim Settled and a Framework Built. *Interfaces, 10*(6), 20-36.

Drummond, H. (1994). Too little too late: A case study of escalation in decision making. *Organization Studies, 15*(4), 591-607.

Holmes, A. (2006, April 1). Baby Steps for IRS Upgrades. *CIO Magazine, 19*.

Johnston, D. C. (2003, Dec. 11). At I.R.S., a Systems Update Gone Awry. *New York Times,* p. C1,

Montealegre, R., & Keil, M. (2000). De-escalating Information Technology Projects: Lessons from the Denver International Airport. *MIS Quarterly, 24*(3), 417-447.

Newman, M., & Sabherwal, R. (1996). Determinants of Commitment to Information Systems Development: A Longitudinal Investigation. *MIS Quarterly, 20*(1), 23-54.

Reichelt, K. S., & Lyneis, J. M. (1999). The Dynamics of Project Performance : Benchmarking the Drivers of Cost and Schedule Overrun. *European Management Journal, 17*(2), 135-150.

Richardson, G., & Andersen, D. F. (1995). Teamwork in group model building. *System Dynamics Review, 11*(2), 113-138.

Richardson, G., & Pugh, A. (1981). *Introduction to system dynamics modeling with DYNAMO*. Cambridge, MA: MIT Press.

Slemrod, J. (1992). Did the Tax Reform Act of 1986 Simplify Tax Matters? *Journal of Economic Perspectives, 6*(1), 45-57.

Staw, B. M. (1997). The escalation of commitment. In Z. Shapira (Ed.), *Organizational Decision Making* (pp. 191-215). Cambridge, UK: Cambridge University Press.

Staw, B. M., & Fox, F. V. (1977). Escalation: The Determinants of Commitment to a Previously Chosen Course of Action. *Human Relations, 30*, 431-450.

Staw, B. M., & Ross, J. (1987). Behavior in Escalation Situations: Antecedents, Prototypes, and Solutions. *Research in Organization Behavior, 9*, 39-78.

The Standish Group. (1994). *The Chaos Report* o. Document Number)

U.S. Department of The Treasury. (1995). *Internal Revenue Service Annual Data Book*. Retrieved. from.

U.S. Department of The Treasury. (2006). *Internal Revenue Service Data Book*. Retrieved. from.

Varon, E. (2004, April 1). For the IRS There's No EZ Fix. *CIO Magazine, 17*.

Varon, E. (2005, May 1). The IRS Makes Progress. *CIO Magazine, 18*.

Wallace, L., & Keil, M. (2004). Software project risks and their effect on outcomes *Communications of the ACM, 47*(4), 68-73.

Wiegers, K. E. (2003). *Software Requirements*. Redmond, WA: Microsoft Press.

Xia, W., & Lee, G. (2004). Grasping the Complexity of IS Development Projects. *Communications of the ACM, 47*(5), 68-74.