System Dynamics Group
Sloan School of Management
Massachusetts Institute of Technology

System Dynamics II, 15.872
Professors John Sterman and Hazhir Rahmandad

Assignment 4

Understanding Cost & Schedule Overrun In Product Development Projects*

Assigned: Sunday 1 December 2013; Due: Wednesday 11 December 2013
Please do this assignment in a team totaling three people and submit digitally.

Projects to design and build new products or services, whether large, one-of-a kind projects such as HealthCare.gov or Boston's Big Dig, or smaller, more routine projects such as designing a new car or smartphone operating system, are notorious for cost and schedule overruns, poor quality, bugs and defects, and failure to meet customer requirements—they are Late, Expensive and Wrong (LEW). Such projects are increasingly important in our economy. For example, almost every new product or service today involves development of new software. Success in the market often depends on being on schedule and budget. This assignment builds your understanding of the feedback structures driving success or failure in projects.

Case Background: You have been hired as a consultant to the IT department of a major financial services institution. The department is responsible for the development, extension, and maintenance of software to support the growing range of products and services offered by the institution. The department has not been doing a very good job. All of its software development projects over the last few years have significantly exceeded both their original budgets and schedule. The Vice President in charge of the IT Department put it this way: "Our projects start out okay, but about halfway through we begin to discover additional work, often to correct errors made on earlier work. When we realize we are behind we add additional resources as they are freed from other assignments. But we barely keep pace with the workload, and end up throwing lots of staff at the job. And we still don't finish on time. I am beginning to wonder if there isn't something to Brooks Law.[1]"

You interview some of the key project managers from the department, and come up with a long list of factors that might be contributing to the problems in the IT department:

- Uncertain specs from the operating departments (the customers for the projects)

*Originally prepared by James Lyneis, April 2000. Last modified November 2013.

[1] "Adding manpower to a late software project makes it later." Brooks, Frederick P. Jr. The Mythical Man-Month. Reading, MA, Addison Wesley, 1995.

- Late customer specification changes
- Delays in getting additional staff
- New staff unfamiliar with the project
- Design errors discovered late in the project
- Overwork and fatigue among the programming staff
- Time spent getting people brought on a project up to speed
- Project staff not doing a thorough job checking their work
- Pressures to promise delivery earlier than is reasonable
- Building design errors into coding, and not discovering the problem until testing

There is a lot of pressure on you and your team to provide implementable solutions for these problems. Some suggest you should immediately begin developing a comprehensive model of the IT department. However, you know that the most effective way to develop a model, and to engage and educate the client along the way, is to build it in small steps, making sure you and the clients understand each step fully before adding additional complexity. With this in mind, you work through the four steps below, and at the end have some significant preliminary insights for your client.

⇒ Brevity is a virtue in your write up. Unless specifically requested, it is not necessary to hand in complete sets of output (graphs, tables) for each test and simulation you do. A summary table will suffice. For example, you might construct a table showing the date of project finish and cumulative effort expended. However, as always, you must explain the changes you make in the equations so that an independent third party can replicate your simulations.

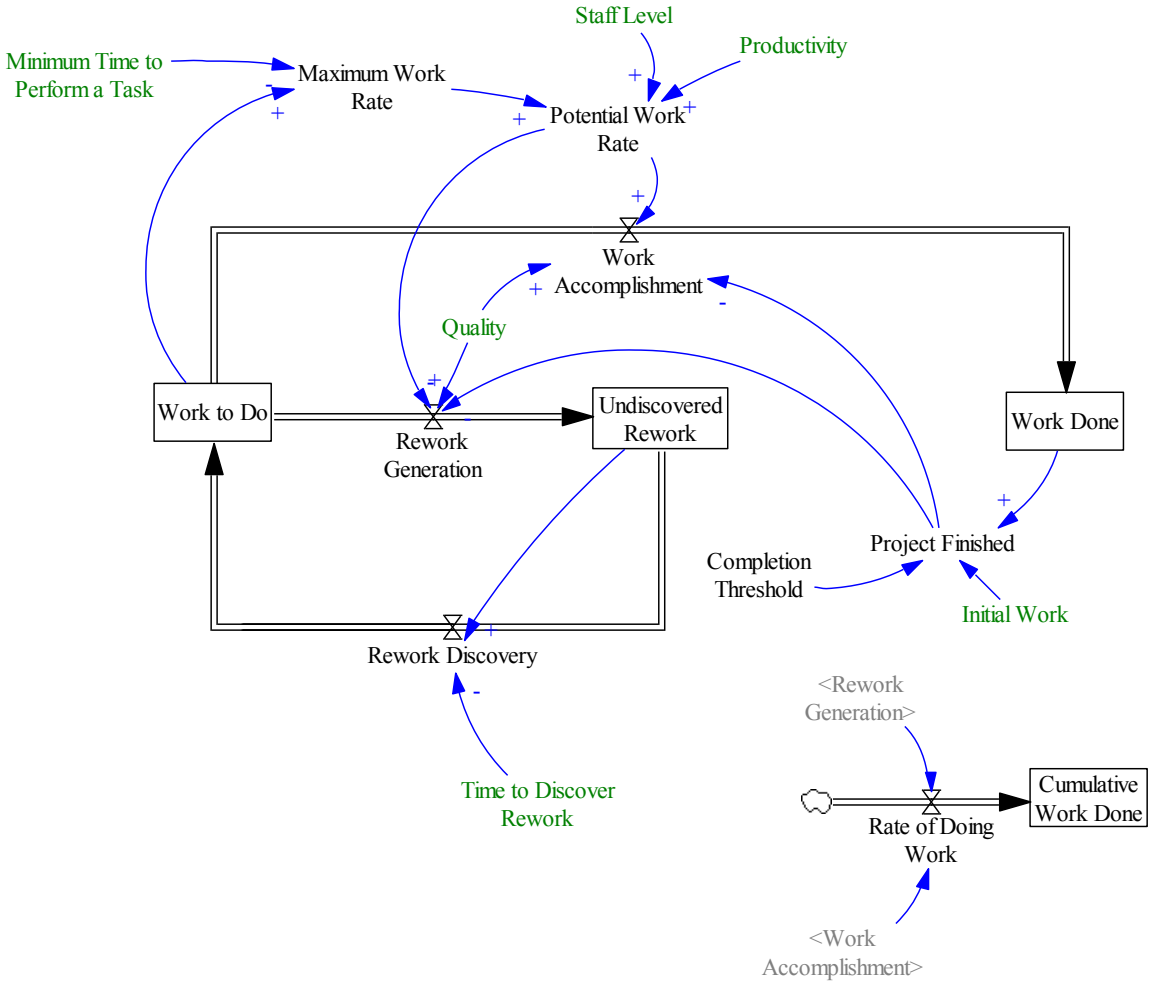
A. Step 1: The Rework Cycle

To begin your analysis, you hypothesize that the "rework cycle" is likely to be at the heart of the IT department's project problems. You therefore construct a rework cycle model of a typical department project, as illustrated in the following figure. Your interviews indicated the following:

- A typical project involves 100 tasks
- Under optimal conditions, each programmer accomplishes 1 task per month
- Normally, programming error rates are 25%
- It takes about 4 months to discover design errors
- A typical project starts with 4 staff

- Define the completion of work (Project Finished) when work done is 99% of original work to do. Use Vensim's IF THEN ELSE function. Stop further work accomplishment and rework at this point.

The Rework Cycle:



❑ A1. Create your model from scratch based on the diagram above and complete the equations for the rework cycle. You do not need to add any variables to the diagram. Select a sufficiently long time horizon for simulation and adequately small TIME STEP (See Appendix A in *Business Dynamics*). Hand in your fully documented and dimensionally consistent model (.mdl file).

❑ A2. Which factors do you think are more important in determining project completion -- productivity, quality, or rework discovery time? Why? (*Please answer this before simulating your model.* Your grade is not affected by your answer to this question)**.**

❑ A3. As an extreme test case, if programmers did not make any errors, when would the project finish? What happens to work to do, work done, and undiscovered rework?

* You will want to create custom graphs to show all the important variables.

❑ A4. Now, set the value for normal quality to the value indicated in your interview notes. When does the project finish? What happens to undiscovered rework in this situation?

Provide a one paragraph, simple, and non-technical explanation of why the project takes more than 25 months (the length of a naïve estimate with 100 tasks completed by four people each having one task per month productivity).

- ❑ A5. Analyse the impact of productivity, work quality, and rework discovery time on project completion date and total work done. Perform this analysis by conducting sensitivity tests varying each parameter by plus-and-minus 33% (requiring 6 simulations in addition to the reference or Base Case in which the parameters are at their normal values). Explain your observations in simple managerial terms.
  - * Use a quality of 0.75 as the mid-point for your analyses
  - * Create a table reporting the completion rate and total work done for each condition to summarize the results of these tests.
  - * Your explanation should be written in none-technical language and understandable to any client team member who has never seen your model.

B. Step 2: Extending the Model: Adding the Quality on Quality Feedback and Variable Rework Discovery Time

After exploring the behavior of the rework cycle model with your client, you return to the office to expand the model to better reflect some important mechanisms revealed from the feedback you received from your client.

First, your discussions around the rework cycle indicate that Time to Discover Rework is not likely to be constant, but falls as progress on the project progresses. Your discussions indicate that rework discovery time is long initially, about 10 months, but once perceived project progress passes 50% complete and testing begins, rework discovery time falls below the initial value of 10 months. In the latter stages of the project, when almost everything is testing and error correction, the discovery time is approximately 1 month. The table below describes the best estimates of the people you interviewed.

Fraction Complete	Effect on Rework Discovery
0	1
.1	1
.2	1
.3	1
.4	1
.5	1
.6	.95
.7	.8
.8	.45
.9	.2
1.0	.1

Your client also highlights the phenomenon of errors building on errors. Specifically, the quality of prior work affects the quality of new work being done. New code to a code base containing errors will be more likely to have errors. For example, if the code base includes an erroneous label and pointer to data that needs to be retrieved from the firm's data warehouse, then code developed later using the same pointer will also be in error. Similarly, if programmers re-use a code object or "subroutine" (yes, the firm still uses some legacy code programmed in Fortran) containing errors then every instance of that re-used code will propagate the error. The fact that errors lead to more errors is a general challenge in software development, and is a particular problem in the IT department.

Your discussions indicate that while average quality may be 75% or less, management believes that "normal quality" would be about 85% were it not for these "quality on quality" and other effects (to be discussed in Step 3). You probe project personnel on the likely effect of prior quality on current quality. They estimate that normal quality would be achieved if the prior work were perfect. And they estimate that if all prior work were done incorrectly, then the quality of new work would be as low as 10% of normal. The table below describes their best estimate.

Average Work Quality	Effect on Current Quality
0	.1
.1	.25
.2	.35
.3	.45
.4	.55
.5	.65
.6	.725
.7	.8
.8	.875
.9	.95
1.0	1.0

* Assume that staff and scheduled completion date remain constant (although actual completion date may exceed the scheduled deadline).
* Include as a performance measure "cumulative person-years on the project."
* The fraction of the project perceived to be complete is a key performance measure used by management. You realize that perceived progress includes both work actually done and the stock of undiscovered rework. Because undiscovered rework is, by definition, not yet known to management, the project managers and team believe that the total amount done correctly is the sum of what has actually been done correctly and the stock of undiscovered rework.
* Consider reformulating quality as

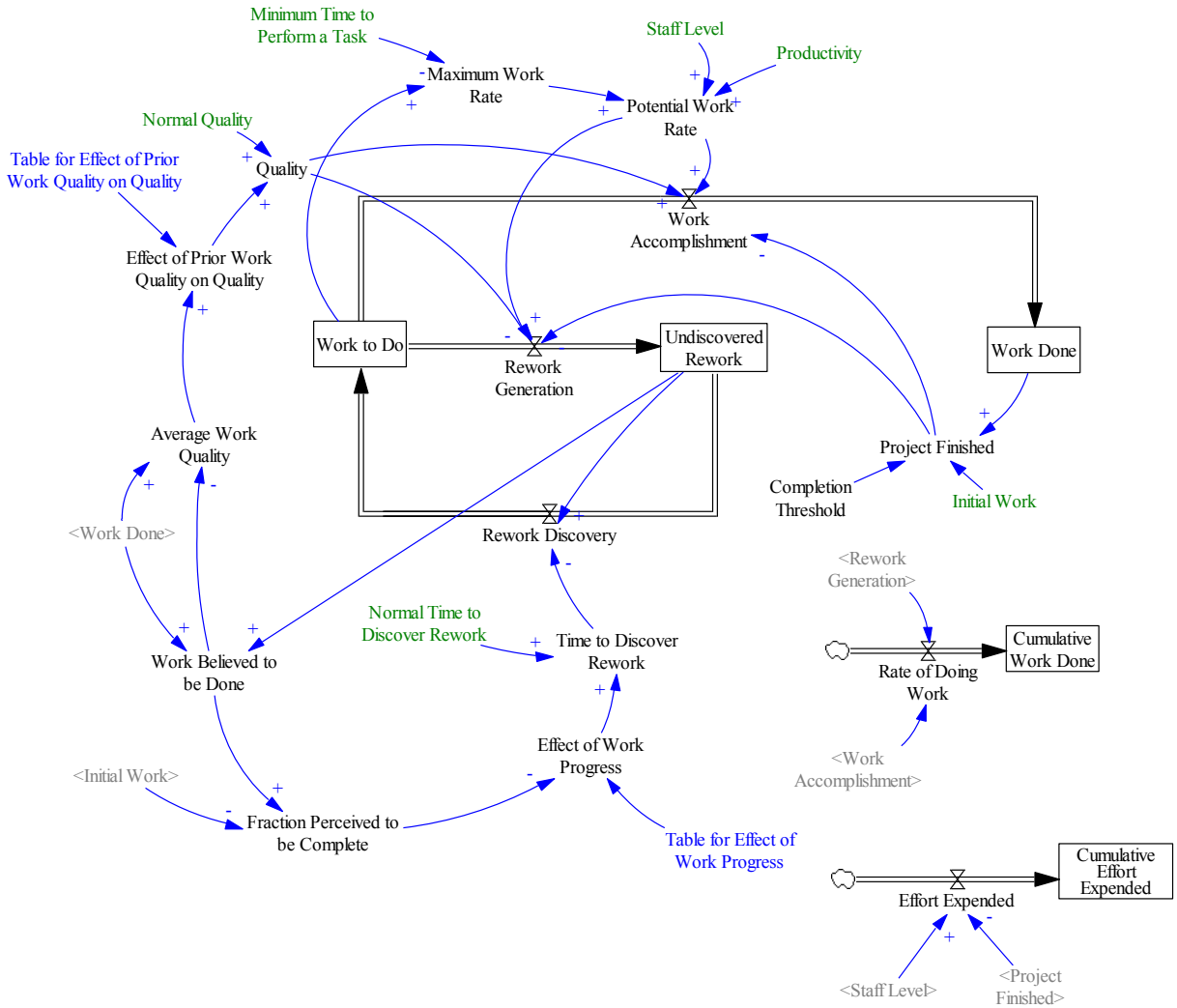
  Quality = Normal Quality * Effect of Prior Work Quality

* The Effect of Prior Work Quality should be driven by the average quality of work to date, used as an input into a Lookup function. Errors embedded in past work product can cause errors to be made on current work, and may lower productivity as ambiguities are sorted out. Some definitions

  Average Quality of Work done to date =
  Work Done/(Work Done + Undiscovered Rework)

* You may need to use the IF THEN ELSE, or the XIDZ function ("X if division by zero; see the Vensim Manual) function to avoid division by zero at the very beginning of the project when no work is completed.

The revised model diagram is shown below. Save a copy of the model you developed above with a different name. Using that copy, implement the diagram below and add the equations. You do not need to add any additional structure beyond the diagram:



❑ B1. Hand in your fully documented and dimensionally consistent model (the .mdl file).

- ❑ B2. How does the inclusion of a variable rework discovery delay affect the behavior of the simulated project? What happens to the completion date and the work backlog compared to the case when rework discovery time is constant? Compare the behavior of the model with and without this feedback, and provide a non-technical explanation of the mechanisms behind your observations.
- ❑ B3. How does the inclusion of the 'quality on quality' feedback affect the behavior of the simulated project? What happens to quality, completion date and the work backlog as compared to the prior? Compare the behavior of the model with and without this feedback, and provide a non-technical explanation of the mechanisms behind your observations.

C. Step 3: Extending the Model: Allowing for Increased Staff

Your model is beginning to behave like a real project, and now you can begin to use it to explore the consequences of the IT department's project staffing policies. To do so, you need to add the ability to increase and decrease staff on the project. You also need to include the consequences of adding staff on productivity and quality. Your interviews indicate:

- A typical project starts with a team of four experienced staff, but hiring/transfer can dynamically change staff levels.
- Because the IT department's programmers are fully engaged on other projects, or because hiring takes time, the time to increase staff via hiring or transfer averages 4 months.
- Excess staff is transferred out of the project or fired, with a delay of two months on average. When staff levels are reduced, the inexperienced members are cut first, and only then will experienced staff leave the project.
- New staff coming onto a project require 24 months to gain full experience
- Inexperienced staff has lower productivity and quality. Your interviews indicate that new staff produce at only 50% the productivity and quality of experienced staff.
- Inexperienced staff also reduces the productivity of experienced staff as a result of the need to train and oversee these new staff. Management estimates that each new staff needs about 40% of an experienced team member's time for training and oversight.
- Management estimates the staff level required to finish the project by dividing the estimated labor effort needed to complete (in person-months) by the scheduled time remaining. Since scheduled completion date is fixed, once the project passes the scheduled completion date, the scheduled time remaining can become negative, which Is not realistic. Therefore in staff planning, management sets the deadline one month from the current date once they pass the initial scheduled completion date. The typical project is scheduled to complete in 30 months.
- The IT department estimates cost to complete, which is the estimated labor effort, in person-months, needed to complete the project, by dividing the level of work remaining to be done by average net productivity on the project; average net productivity is estimated by dividing work believed to be done to date by cumulative person-hours used to date, and starts initially from the normal productivity of experienced staff.

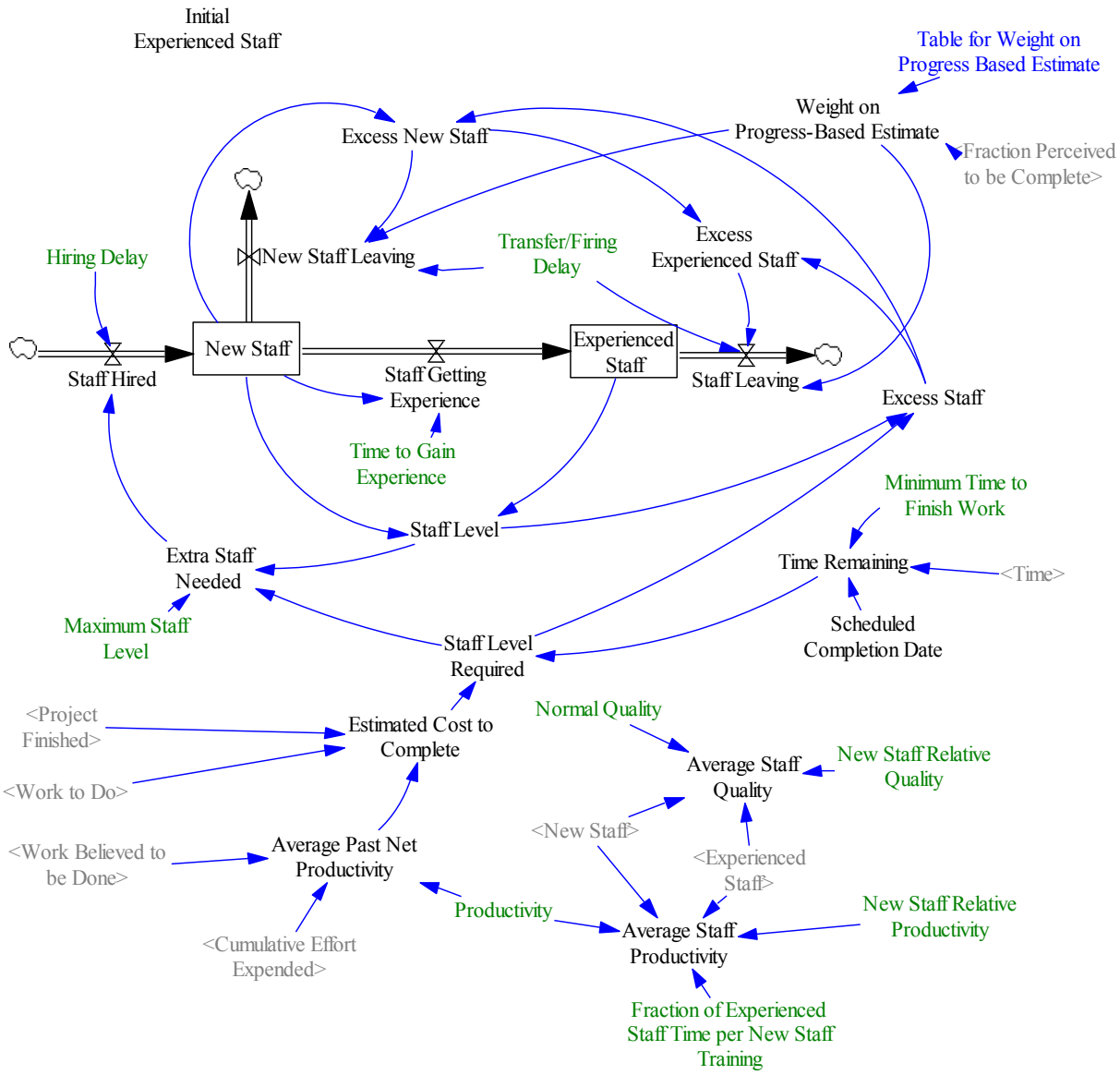
* Save your model from step B under a new name and expand it as follows:

* Create two stocks, new staff and experienced staff. Assume that all hires or transfers to the project enter the pool of new staff (since they don't have experience with this project), and that they take 24 months to become experienced.

* A simple weighted average can be used to calculate the average productivity and quality of the current staff. That is, average productivity depends on the productivity of new staff, the productivity of experienced staff, and the fraction of staff of each type. These averages should replace the normal quality and productivity values. Be sure you take into account the impact of new staff on the productivity of experienced staff created by on the job training and oversight.

* A diagram of the new staffing section is given below. A tricky formulation in this model regards the behavior early when little work has been accomplished. Because rework discovery takes time, early in the project work believed to be done, and average productivity, appear to be high. The resulting high productivity estimate might indicate that fewer staff are required, and transfers off the project would occur. However, a smart manager recognizes this possibility and will not reduce staff levels based on apparent progress until sufficient progress has been made to really determine where things stand,. You will note in the diagram below that the "weight on progress-based estimate" can slow down staff leaving. The weight on progress-based estimate is a function of the fraction perceived to be complete, and is estimated based on your interviews to approximately follow the table below:

Fraction Complete	Weight
0	0
.1	0
.2	0
.3	.1
.4	.25
.5	.5
.6	.75
.7	.9
.8	1
.9	1
1.0	1



- ❑ C1. Complete the model equations and document your new model. You can add the staff sector of the model to a new view to simplify navigation. Also you will benefit from a creating a control panel in which you include the key policy levers, tables, and parameters, along with key graphs that represent the summary of project's behavior. Hand in your documented model (.mdl file).
- ❑ C2. Describe the behavior of the simulated project when staff are endogenously increased in order to meet the originally scheduled date. Does the project finish on time? Does it finish sooner than when staff are not added to the project? Why or why not? (Note: while several computer runs may be necessary to get your model working properly, only one computer run is necessary to answer this question). Explain in plain English and support your argument with relevant graphs.
- * Look at the behavior of productivity and quality, and the factors that cause them to change over time. How does the addition of staff later in the project affect productivity

and quality? What does this do to the total amount of work done and to cumulative person-years spent on the project?

❑ C3. What policies would you recommend at this point? Explain briefly.

D. Step 4: Policy Analysis: When Does Adding Labor Make Sense

While the results thus far seem to be consistent with the behavior of the IT Department, you do not want to end with the recommendation that staff should never be added to a late software project. There must be some conditions under which adding staff makes sense. Conduct a series of analyses to determine when, given the model developed thus far, project performance is improved when staff are added. You should consider conducting a sensitivity analysis on the assumptions regarding:

Relative productivity of new staff
Relative quality of new staff
Effect of new staff on productivity of experienced staff
Time required for new staff to become experienced
Time required to increase staffing
Initial number of staff
Scope/length of the project

❑ D1. Prepare a summary of the results from your sensitivity experiments using a table which identifies each scenario and its impact on key performance metric(s). Besides sensitivity runs in which you increase/decrease one parameter at a time, conduct at least one simulation where multiple assumptions are changed in a realistic range and justify adjustment of staff during project life. Approximately 16 computer runs should be sufficient for this question. Under what conditions does it make sense to add staff to get a project completed on time? Why? Explain in simple English.

E. Synthesis of Policy Analysis and Recommendations

❑ E1. Prepare a recommendation for the management of the IT department regarding its project management and staffing policies. What would you suggest regarding initial staffing, their skills, and the schedule? How about adding staff as the project progresses? Provide a brief summary in non-technical language that synthesizes your understanding and offers management with clear guidance.

❑ E2. While your model is already pretty complicated, it still does not include several features that may potentially be relevant to IT projects in general. Provide management with a list of what you think are potentially valuable additions should they decide to continue with this analysis in future. You do not need to do any further modeling, only highlight what you think are important potential additions.

F. What to hand in and how to submit your work

Write up your responses to the questions above in a word (.docx) document. In addition, you need to submit three fully documented Vensim model (.mdl) files, one each for parts A, B, and C. Upload your team's assignment by 5 PM on the due date. Submit your assignment as a single .zip file including your response document and models.

Make sure you include your team-members' names in the document. Name files with your team's name, and for multiple files of the same type, the assignment section, e.g. "Team21.docx", "Team21-A.mdl", "Team21-B.mdl"and "Team21-C.mdl" all submitted as part of "Team21.zip".

15.872 System Dynamics II
Fall 2013