Improving Software Project Management Through System Dynamics Modeling Abstract

Doug Sycamore
Motorola
Scottsdale, Arizona

Managing a project and understanding the many system dynamics and feedback loops associated with a project is a formidable task. Creating schedules and tracking progress are two important activities for managers. These activities become exponentially more complex and difficult for larger projects. Good managers possess an intuitive talent understanding how a system will behave when modifications are performed and make decisions using these skills and experience. However, when wrong decisions are made and implemented into a project, disastrous results could occur, reducing the probability of success. The larger the project, the more feedback loops, the greater the dynamics, and the reduced probability of accurately predicting an outcome from modifications. Changing only one variable could effect the dynamics of a project with an unpredictable outcome. This is particularly true when considering all the system feedback loops associated with a project.

In today's market there are plenty of Commercial Off The Shelf (COTS) project management tools available. Many will perform basic features such as scheduling, budget tracking, man-hour reporting, resource allocation, and capital expenditure tracking. Some of the more advanced project management tools have graphical user interfaces with pull-down menus or pop-up windows. These more advanced tools often have the graphical capability to display Gantt charts, PERT charts, critical paths, pie graphs, or other types of charts, graphs, and histograms. However, they all lack the ability to accurately model the system dynamic parameters that influence a project's schedule, budget, completion percentage, and quality. For example, when a project begins to fall behind schedule, then productivity pressure increases. None of the tools available in today's market implement this type of feedback.

Very few COTS project management tools have the capability to provide a manager with "What-If" scenarios. None of the tools available in today's market take into consideration how these system dynamic variables affect the outcome of a project. What is needed is a tool that gives accurate foresight into the dynamics of a system based upon intuitive managerial decisions.

The following Masters Degree thesis statement was proposed:

> *Using the "iThink" tool available from High Performance Systems, a new tool can be developed such that a software manager may plan, track, and predict the outcome of a software project better than the more traditional approaches by incorporating system dynamic feedback loops into the tool.*

The tool developed to support the Masters Degree Thesis models system dynamics describing the relationships among the various components of a software system. This is accomplished through equations that represent causal influences rather than statistical correlation. When a software manager determines some type of corrective action plan must take place to a project, this tool can be used to simulate the modifications before implementing the new idea. As the tool executes the simulation, a software manager can visually observe the results in a real-time fashion. The

results are calculated using the system dynamic model developed which takes into consideration areas like communication overhead, schedule pressure, and rework hours generated by defects. The flexibility exists to pause the simulation at any time and make multiple modifications to controllable parameters. This tool also allows the ability to resume after being paused and incorporates the modifications into the simulation.

The tool has four basic feedback loops comprised of non-linear system dynamic variables (see Figure 1). All four of these feedback loops begin and end at the object labeled, Schedule and Effort, which is the nucleus of the system. This object represents a project's schedule and the effort in staff hours to complete the project.

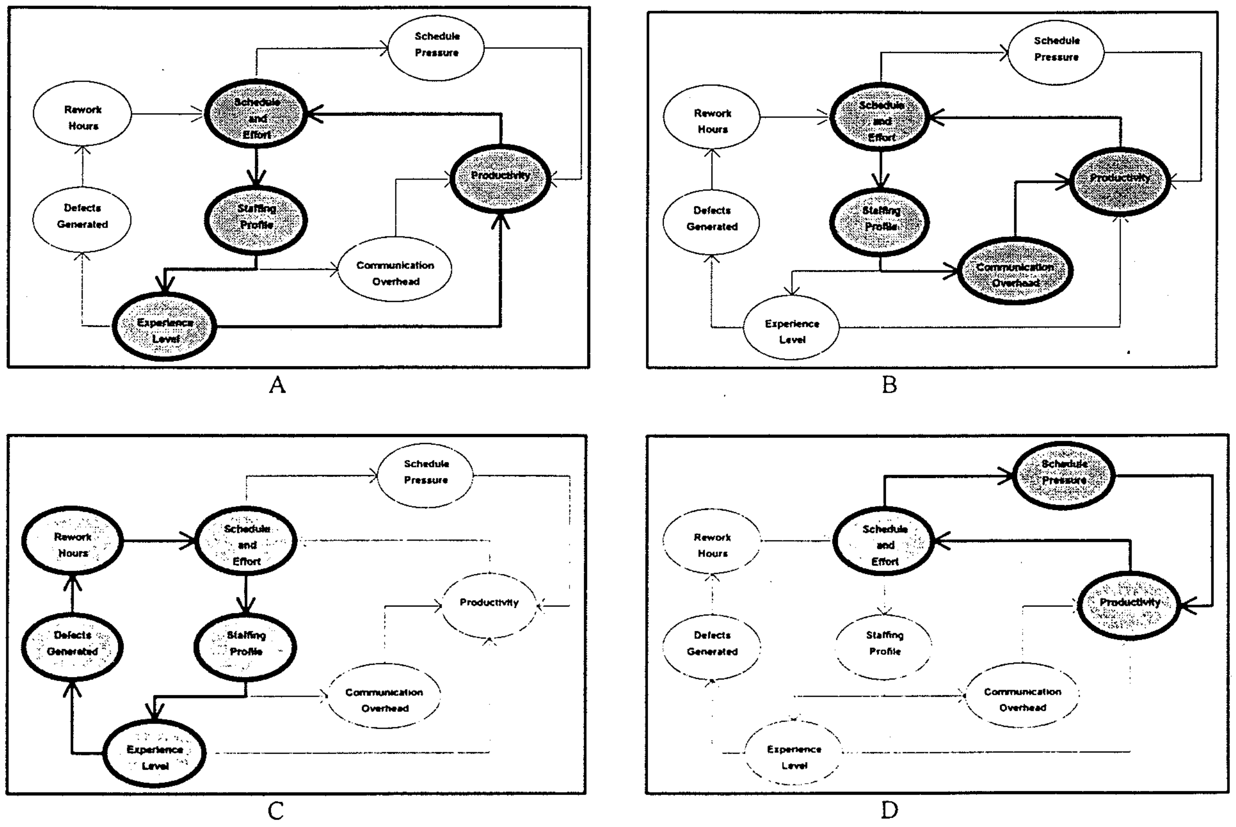

Figure 1

The first feedback loop represents the staffing profile of a project (see Figure 1:a). Depending on the schedule and the amount of effort it will take to complete a project, a software manager can adjust the staffing profile. The staffing profile effects the productivity based on the number of engineers working on a project, the domain expertise of the engineers, and the amount of time an engineer participates on a project. The tool is also configurable to reflect the impact of engineers often multitasked on more than one project.

The second feedback loop models the communication overhead (see Figure 1:b). If a project requires 30 engineers to complete the work, the communication overhead is much greater than a project requiring only five engineers. This concept seems obvious, but it is not trivial when modeling the effect it has on the productivity. For example, a project with 30 engineers will experience a productivity degradation of 54% due to communication overhead, whereas a project with five engineers only experiences a productivity degradation of 1.5%. It is important to note that the graph depicting the communication overhead is a non-linear curve.

The third feedback loop takes into consideration the amount of defects generated by the engineers during the design and code phases of a software increment, which translates into rework hours (see Figure 1:c). The more rework hours the greater the impact to the schedule. A software increment is defined as an event consisting of design, code implementation, and integration activities of a software system. Multiple increments can be performed in serial, parallel, or a combination of the two. The tool also models how different engineers with various domain expertise generate different amounts of defects. An engineer with less domain expertise will generate more defects than an engineer with a higher degree of domain expertise.

The fourth feedback loop models the schedule pressure associated with the percentage of work complete per the schedule (see Figure 1:d). The further behind schedule the greater the schedule pressure. As schedule pressure increases, engineers will work more hours and more efficiently becoming more productive. However, if schedule pressure remains high and engineers are working many hours of overtime, they begin to generate more defects and eventually an exhaustion limit is reached. Once the exhaustion limit is reached, productivity will decrease until a time period of recuperation is achieved. The transition from exhaustion to recouping is not a step function and is modeled by a curve that represents a practical situation. The converse also holds true. When a project is ahead of schedule, then schedule pressure decreases and less hours and efficiency is demonstrated by the engineers.

Each feedback loop is not conceptually complicated. However, when incorporating the four feedback loops with the non-linear system dynamic variables, it becomes almost impossible for a software manager to accurately predict the outcome of a project, especially when changes are made to the inputs during the middle of a project (See Figure 2). Imagine making a modification to a staffing profile, and then trying to understand exactly the effect it has on the project when considering all of the feedback loops it affects simultaneously.

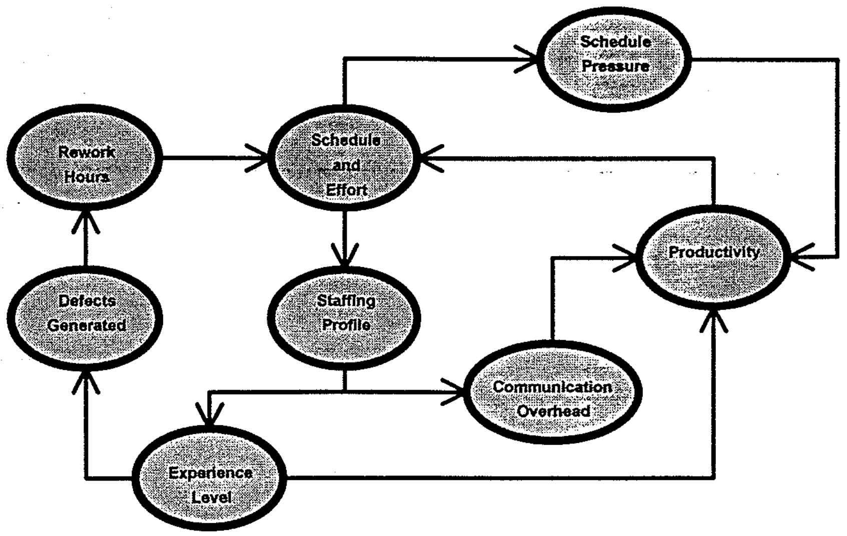

Figure 2

The tool developed to support the Masters Degree Thesis allows a software manager to configure a project and execute many "what if" scenarios. The ability to modify the staffing profile, the expertise of the staff, the dependencies of the software increments, and many other features before and during a simulation exists. As the simulation runs, many different outputs are updated in a real-time fashion to the screen and the final results can then be logged for later analysis.

Note: This tool was not developed for commercial use, but designed to prove the concept of the Masters Degree Thesis statement. It demonstrates how system dynamic behavior and feedback loops need to be incorporated into existing project management tools rather than depending on the limitations of current static processing currently being modeled.

Author's Biography: Doug Sycamore is a full time engineer working as a software task leader for Motorola, Government Space and Technology Group (GSTG) in Scottsdale, Arizona. He received a B.S. in Computer Science from Michigan State University and is currently finishing an M.S. in Computer Science from Arizona State University under the supervision of Professor James Collofello.