22nd IPMA World Congress
"Project Management to Run"
9 - 11 November 2008, Roma, Italy

Improving management effectiveness and overall performance of software development projects through a system dynamics approach

Stefano Armenia[1], Massimiliano M. Schiraldi[2], Diego Falsini[3]
("Tor Vergata" University of Rome, Dept. of Enterprise Engineering, Rome, Italy)

Abstract
While existing research has mainly focused on project management's static view, our work investigates the impacts of projects' structure and behavioural dynamics on their performance, with a specific focus on the influence of some peculiar development processes. A dynamic simulation model of a single phase project was built using the system dynamics methodology. The model integrates several previously developed and tested project structures and adds a separate structure for the negotiation process. Simulations describe the behaviours generated by the interaction of customized development processes in single-phase projects. Project performances are measured in terms of time, quality and cost. Our research aims to show that development processes, as well as shared resource levelling techniques, significantly impact the dynamic behaviour of projects through the feedback, delays and nonlinear relationships which are usually omitted in traditional project management practice, as well as in methods, tools and models, but are very important descriptors of project complexity. Expanding the models used to manage projects to include dynamic features requires a change of focus by researchers and practitioners. The system dynamics methodology provides some of the tools for developing and implementing such an expansion in project models.

Keywords
Project Management, System Dynamics, Resource Allocation, Decision Support Systems, Collegial Decision Making

1. Introduction
The combination of feedback, time delays, and nonlinear relationships in project structures have been shown to reduce performance and cause them to be very difficult to manage (Thomas and Napolitan, 1994; Reichelt, 1990; Cooper, 1980). The dynamic nature of project behaviour precludes the generation of a single set of decision rules which are robust in the face of all possible project conditions. As a matter of fact, both complexity and dynamic features of projects seems to be poorly understood by managers (Diehl and Sterman, 1995; Sterman, 1994; Paich and Sterman, 1993; Rechtin, 1991). In addition traditional tools are inadequate for dealing with the dynamic complexity of projects.
A project should be really considered as a man-made goal-oriented open system and, thus, it tends to be unpredictable and unstable. The complexity of projects and of their environment has increased the disruptive effect of subjective human factors. Personal judgement based on past experience is no longer sufficient to cope with this problem. There is a need to understand better the strategic issues of project management and to learn effectively from past failures; this can only be achieved through a more formal systemic analysis.
System Dynamics (SD) may be useful in describing causal and dynamic complexity arising in software projects and organizations, thus eventually allowing for the building of a new set of tools which will support management in decisions as well as allow them to experiment in learning environments and checking out their hypotheses without implementing them first and wait for the, rather often, catastrophic consequences.

2. Literature review

2.1 Dependencies and feedback problems of development projects in literature
Traditional project management models based on the Critical Path Method (CPM) and PERT (Moder et al. 1983; Halpin and Woodhead 1980) provides several tools for trading away good performance in one measure for improved performance in another. For example durations of activities along the critical path can be shortened by adding more resources (Ulrich and Eppinger, 1994; Wheelwright and Clark, 1992; Moder et al., 1983). The effects of altering activity dependencies among activities to shorten the critical path can be investigated (Barrie and Paulson, 1984; Moder et al., 1983). These methods are limited by their use of an indirect project measure (time) and by bundling the characteristics of and relationships among scope, resources, and processes in each activity into a single duration

[1] Corresponding author. Postal address: "Tor Vergata" University of Rome, Department of Enterprise Engineering, Via del Politecnico, 1 – 00133 Rome (Italy), tel. +39.06.7259.7805 – fax +39.06.2021351.
E-mail address: stefano.armenia@uniroma2.it

[2] E-mail address: schiraldi@uniroma2.it

[3] E-mail address: diego.falsini@uniroma2.it

estimate. They also tend to ignore iteration or require that iteration be implicitly incorporated into duration estimates and precedence relationships.
More sophisticated models based on a joint CPM/PERT paradigm address some of these limitations, but cannot fully model development processes. Other research approaches identify some dynamic consequences of different project structures on project performance (Smith and Eppinger 1997; Eppinger et al. 1994; Steward 1981).
Large reductions in cycle time can be realized by applying concurrent development (Wheelwright and Clark, 1992, Womack et al., 1990; Nevins and Whitney, 1989). But the cycle time reduction comes at the cost of increased complexity. Steward (1981) and Eppinger et al. (1990) developed the Design Structure Matrix to investigate the iterative nature of product development. Design Structure Matrices have been used to map (Smith and Eppinger, 1991) and predict (Morelli and Eppinger, 1993) information flows among activities. But the Design Structure Matrix cannot directly model the structure of a development process over time. Other model structures based on project characteristics have been suggested (Rodrigues and Williams, 1996) and described conceptually (Cooper 1980). However, the specific features and characteristics that distinguish different development processes have not been described at a formal model level of detail. Excluding phase-specific development process structures from project models implicitly assumes that those development processes have no impact on project performance. Yet the availability of work as described by the precedence relationships within and between phases is an important constraint on project performance (Rosenau and Moran 1993; Clark and Fujimoto 1991; Wheelwright and Clark 1992; Moder et al. 1983).

2.2 Contribution of System Dynamics to Project Management issues in literature

A first model was proposed by Roberts (1974) to explore the basic dynamics of R&D projects where the concepts of perceived progress and real progress were first introduced. This model was further improved by Kelly (1970) to consider the management of concurrent projects. The model develop by Cooper (1980) at Pugh-Roberts Associates was the first major practical application of SD to Project Management. Richardson and Pugh (1981) presented a model for the management of R&D projects, which summarises the basic feedback structures of the project management process. Abdel-Hamid and Madnick (1991) applied SD in particular to the software development process for the first time. The models proposed by Lin and Levary (1989) considers an explicit breakdown of the project work into the classic life-cycle stages, providing a more detailed analysis for the schedules, budgets and staff allocation to the project. The above developments represent important contributions for the application of SD to software project management. They introduce valuable concepts and ideas that should be considered in the future. However, most of the reported cases refer to post mortem analysis. Here, the model is used to reproduced the behaviour of completed projects and helps to investigate the causes for deviations.
From the review of the literature we identified several key feedback structure on which we developed our model: the rework structure (Figure 1), the labor structure, the schedule structure, the available work structure, the quality structure, the scope structure.

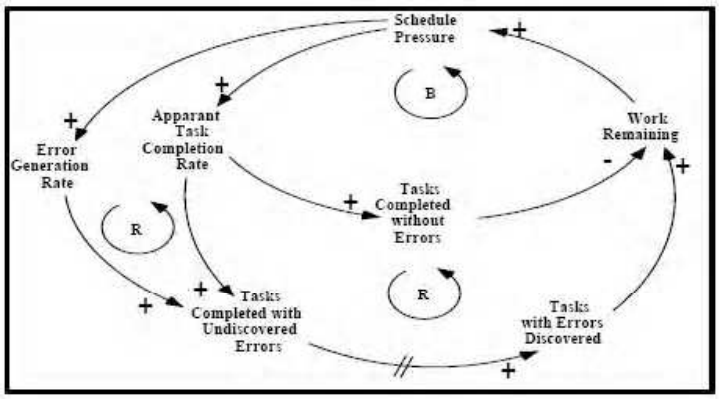

Figure 1: Key feedback structures – Rework (from Ford 1995)

The balancing loop in Figure 1 represents the intended impact of a management response to an increase in schedule pressure - reduce the work remaining. The two reinforcing loops represent the impacts of the unintended side effects of the structure - the generation of additional errors which require correction.
SD traditional notation, symbols and lexicon are not recalled in this paper, referring to Sterman (2000).

3. Proposal of a model

3.1 Setting the context

For our goal, which is to show and understand the major dynamics which affect software project management, we will limit our analysis to a single phase (coding) of a mono-product software project, that is a portion of its life

cycle, spanning over a maximum of 12-24 months (a medium time-dimension for software projects), and we will assume the development team is fully (100% of time) committed on a single project. We will design our model by taking into considerations the main performance indicators of project management, i.e. related to Cost, Time and Quality issues of the released finished product (or a "work in progress" element released to the subsequent phase), and other aspects like Project Complexity, Uncertainty and Risk.

3.2 The model structure: the coding phase of a software project

The model is composed by four subsystems: development processes, resources, control and productivity. The development processes subsystem is the focus of this work. Ford (1995) describes the other subsystems in detail.

3.2.1 The development process structure and Scope subsystems

Our model uses three features to describe the development process in a single phase: circular iteration, multiple development activities, and dynamic concurrence. Circular iteration is described with the stock and flow structure (Figure 3-10).

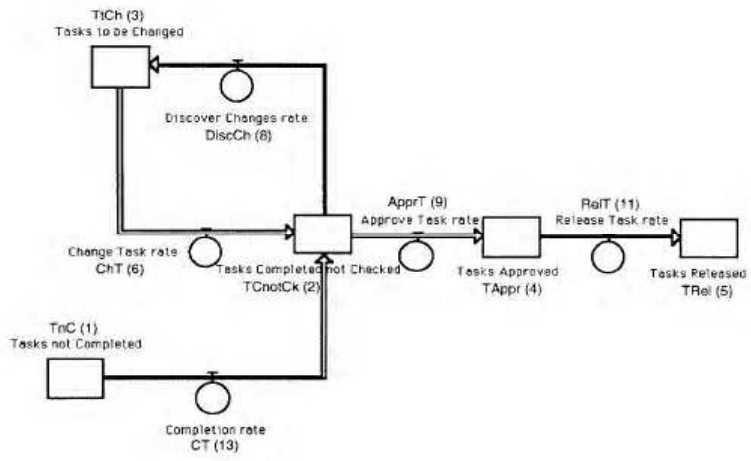

Figure 2: stock and flow structure of a single phase development process (from Sterman-Ford 2000)

We assume that development tasks go through five states: Tasks not Completed (*TnC*), Tasks Completed by not Checked (*TCnotCk*), Tasks to be Changed (*TtCh*), Tasks Approved (*TAppr*) and Tasks Released (*TRel*). Tasks initially reside in the Tasks not Completed stock. The first development activity is called "Complete Tasks" (*CT*). Completed tasks accumulate in the "Completed not Checked" stock. If no tasks require changes or those changes are not discovered during quality assurance, the tasks leave the Completed not Checked stock and pass through the Approve Tasks (*ApprT*) flow into the stock of Tasks Approved (*TAppr*). Approved tasks are subsequently released through the Release Tasks (*RelT*) flow to the stock of Tasks Released (*TRel*). This represents delivering tasks to the managers of downstream phases or to customers. Tasks needing changes are discovered through the Quality Assurance (*QA*) activity.

These tasks move through the Discover Changes (*DiscCh*) flow from the Completed not Checked stock to a stock of Tasks to be Changed. These tasks are corrected or improved through the Change Tasks (*ChT*) activity and returned to the Completed not Checked stock. Changes can be generated during both completion and correcting or improving tasks.

We formally model the process structure for our single phase with five equations represented in Figure 2 and other 13 equations represented in Figure 3.

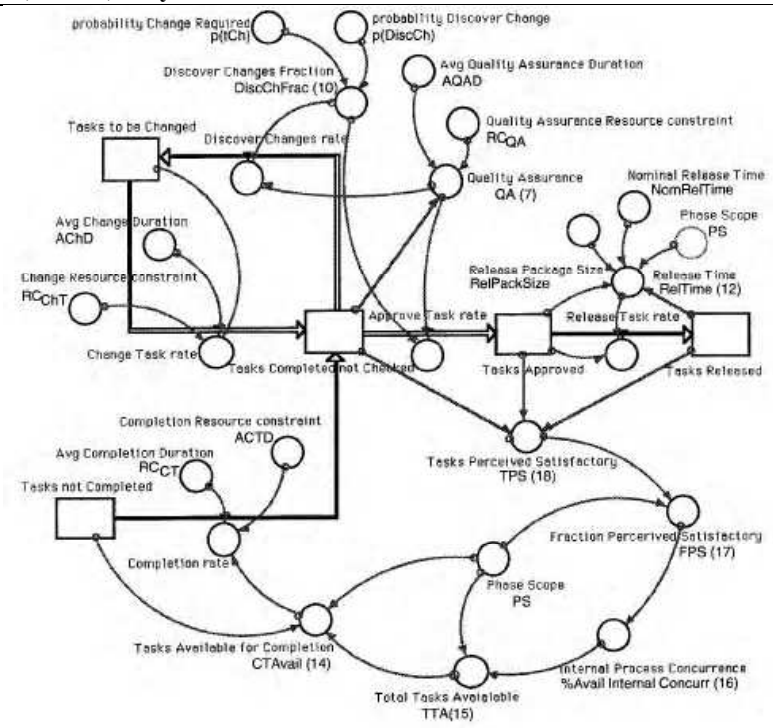

Figure 3: extended S&F structure with auxiliaries (from Sterman-Ford 2000)

3.2.2 The labor and productivity subsystem

In the previous section we have seen that development of software in a software project phase go through three main activities: (1) *Task Completion*, (2) *Task Review*, (3) *Task Rework*. For each of these activities we have seen in the development sector that there is one or more associated flows of tasks. Such a rate represents the activity process itself, that is represents the "speed" at which tasks are processed. In these sense, such activities processing is constrained by the resources allocated to them. Keeping things simple, we saw that the work rate for each activitiy is equal to the minimum between the speed allowed by the availability of work (divided by the average activity duration) and the speed allowed by the resources allocated to that activity. This means that resources determine the maximum speed at which an activity may be performed.

3.2.3 The resources subsystem

The project's total Workforce is assumed to be composed by two different workforce elements, namely *NewPeople* and *Experts*. In particular, the *Average Assimilation Delay* in the model has been set to 17 wwk (working week), so almost 85 working days (wday). We divided the workforce into these two categories first because New Hirees almost always pass through an orientation during which they are less than full productive (their weekly overall productivity has been assumed, as it will be seen in the following, half of that of an expert), both if they have been recruited from outside the company and also if they have been transferred from another project. Second, because we wanted to capture also the training overhead involved in adding new members to a software project. The training of newcomers is often carried out directly "on the job" by veterans, which carries away part of the time that an Expert may in effects more productively (in terms of instant project productivity) allocate on development activities.

3.2.4 The control and plan subsystems

Any control function has at least three elements:

1. *Measurement* of what is happening in the activity being controlled
2. *Evaluation* by comparison with expected values
3. *Communication* of the gap for behaviour control if the need arises for doing so

Progress in a software project (and thus also in a single phase) is measured by the number of resources consumed (effort and/or time), tasks completed or both.

3.2.4 The performance subsystems

As has been previously stated, a project performance is mainly evaluated through 3 parameters: Time, Quality and Costs. We have developed an index factor for each one of these three elements, which altogether contribute to an overall Project Performance Index:

Project Performance = (Budget Index + Final Quality + Time Index) / 3

3. A case study

Original case study:

- Project Size: 94.100 SLOCs (Single Lines Of Code)
- Estimated effort: 5.000 mandays (Actual: 7.000)
- Estimated Duration: 85 wwk (actual: 110)

We initialized the model with the data from the original case study, taking also into considerations an added factor as the work obsolescence. The latter was considered to provide an added work, over time, of 25% the total initial project size.

- Project Size: 94.100 sloc + 23.000 sloc from the obsolescence factor
- Initial estimated Effort: 3.000fte (full time equivalent)
- Actual Effort Spent: 8942 fte
- Estimated project duration: 85 wwk (with a 30% safety pct, released in only one burst at 90% of the project completion)
- Actual simulated duration: 110 wwk

Overall Performance was attested on a 61%, taking into account costs, time and quality, with the following indexes:

- Time Index: 77,27%
- Budget Index: 35,99%
- Quality Index: 71,00%

The obtained results are in accordance to those which were obtained in the original case-study.
A set of input parameters relating to policies for human resource and allocation to activities management, development process management, costs, quality and time management have been identified:

a. *Costs section*: Safety Cost, Budget effect on Basework
b. *Human Resource section*: Willingness to change workforce, Workforce needed calculation method (accounting for effort/time), Maximum/Minimum desired WF levels, Training
c. *Development section*: Internal Precedence Relationship (Concurrency Function – technology related), Package release size (release policy)
d. *Quality section*: Target Quality of the project, Effect of perceived quality on QA allocation (quality gap)
e. *Allocation section*: initial percentages to activities allocation, effect of Schedule Pressure, Quality Gap and Budget on QA and Basework, Desired Rework delay
f. *Time section*: Safety time, Forecast of completion date calculations, Release slippage to staff function
g. *Client Relationship Management section*: Communication of Project progress, Negotiation on change requirements, Client's trust.

Experimentation with most of them have showed interesting results which may help management in understanding the underlying dynamics which are affecting project performance as well as allow management to experiment with different policy parameters in order to correct their mental models according to the way they manage projects.

4. Conclusions

This research addressed the important issue of the causes of dynamic behaviour in software development projects by building, testing and applying a dynamic simulation model of a single phase project.
Feedback, delays and nonlinear relationships were found useful in describing the drivers of dynamic behaviour. The concept of software development as a set of interactive demand-driven activities was used to build rich descriptions of causal relationships based on previous research. The strong direct and indirect influences of development processes were identified by explicitly separating development processes from resources, scope, and targets.
The research identifies a gap between current project models used for management and the complexity of project structures. A failure to bridge this gap is expected to limit project performance improvement. Expanding the knowledge and understanding of project dynamics is a critical part of meeting this need. The development of new or improved tools, as simulation (dynamic, discrete and/or hybrid) for communication and management practice is also expected to be essential to translating improved knowledge and understanding into improved project performance.
This research has contributed insights concerning the dynamics of projects, a tested framework for modelling projects based on demand for development activities, a tool for future research and a tool for improving the understanding of product development practitioners. This work has created opportunities for expanding the study of project dynamics in several potentially valuable directions. This research has pushed project management toward a broader image of projects and its role in project performance. It points to ways of improving performance through improved understanding of project structure and behaviour. Future research will expand and refine the understanding and use of dynamics to manage projects. In particular, the author proposes the extension of the model to a multi-phase software project and to a multi-project environment; the development of standard SD-based project evaluation metrics; the integration at the operational level of our SD model with traditional project modelling approaches

References

Abdel-Hamid, T., Madnick, S.E., (1991), Software Project Dynamics An Integrated Approach, Prentice Hall (Englewood Cliffs, NJ)

Barrie, D.S., Paulson, B.C., (1984), Professional construction management, McGraw-Hill, (New York)

Clark, K.B., Fujimoto, T., (1991), Product Development Performance Strategy, Organization and Management in the World Auto Industry, Harvard Business School Press (Boston, MA)

Cooper, K. G., (1980), Naval ship production: A claim settled and a framework built. Interfaces 10(6), 20-36.

Diehl, E., Sterman, J.D., (1995), Effects of Feedback Complexity on Dynamic Decision Making, Organizational Behavior and Human Decision Processes, Elsevier

Eppinger, S.D., Whitney, D.E., Smith, R.P., Gebala, D.A., (1994), A model-based method for organizing tasks in product development, Research in Engineering Design, Springer

Ford, N.D., (1995), The dynamics of project management: an investigation of the impacts of project process and co-ordination on performance, Unpublished PhD thesis, Massachusetts Institute of Technology (Cambridge, MA)

Forrester, Jay W., (1961), Industrial Dynamics. Portland, Oregon: Productivity Press

Halpin, D.W., Woodhead, R.W., (1980), Construct Management, John Wiley & Sons (New York)

Kelly, J.C., (1987), A comparison of four design methods for real-time system, Proceedings of the 9th international conference on Software Engineering table of contents, Monterey, IEEE Computer Society Press (Los Alamitos, CA, USA)

Kerzner, H.,(2006), Project management: a systems approach to planning, scheduling, and controlling, Wiley

Moder, J.J., Phillips, C.R., Davis, E.W., (1983), Project Management with CPM, PERT and Precedence Diagramming, Van Nostrand Reinhold, (New York)

Morelli, M.D., Eppinger, S.D., Gulati, R.K., (1995), Predicting technical communication in product development organizations, Engineering Management, IEEE Transactions on

Nevins, J.L., Whitney, D.E.,(1989), Concurrent Design of Product and Process, McGraw Hill (New York)

Paich, M., Sterman, J.D., (1993), Boom, Bust, and Failures to Learn in Experimental Markets, Management Science 39(12), 1439-1458

Rechtin, E., (1991), Systems Architecting: Creating and Building Complex Systems, Prentice Hall

Reichelt, K. S., (1990), Halter Marine: A Case Study in the Dangers of Litigation, Technical Report D-4179, System Dynamics Group, MIT Sloan School of Management (Cambridge, MA)

Richardson, G.P., Pugh III A.L., (1981), Introduction to System Dynamics Modeling with Dynamo, MIT Press (Cambridge, MA)

Roberts, E.B., (1974), A simple model of R&D project dynamics. In E. B. Roberts, (ed.), Managerial Applications of System Dynamics, Productivity Press (Cambridge, MA)

Rodrigues, A.G., Williams, T.M., (1996), System dynamics in software project management: towards the development of a formal integrated network. European Journal of Information Systems 6, 51-56

Rosenau, M.D., Moran., J., (1993), Managing the Development of New Products, Achieving Speed and Quality Simultaneously Through Multifunctional Teamwork, Van Nostrand Reinhold (New York)

Senge P., (1994), The Fifth Discipline: The Art and Practice of the Learning Organization, Doubleday & Company

Smith, R.P., Eppinger, S.D., (1997), Identifying Controlling Features of Engineering Design Iteration, Management Science, JSTOR

Smith, R.P., Eppinger, S.D., (1997), Identifying Controlling Features of Engineering Design Iteration, Management Science, JSTOR

Sterman, J., (2000), Business Dynamics: Systems Thinking for a Complex World. Irwin/McGraw-Hill

Sterman, J.D., (1994), Learning in and about complex systems, System Dynamics Review 10 (2-3), 291-330

Steward, D.V., (1981), The design structure system- A method for managing the design of complex systems, IEEE Transactions on Engineering Management

Ulrich, K.T., Eppinger, S.D., (1994), Methodologies for Product Design and Development, prepublication draft, McGraw-Hill (New York, NY)

Wheelwright, S.C., Clark, K.B., (1992), Revolutionizing Product Development, Quantum Leaps in Speed, Efficiency, and Quality, The Free Press (New York)

Womack J.P., Jones, D.T., Roos, D., (1990), The Machine that Changed the World, Rawson Associates, (New York)