Modeling The Dynamics of Software Reuse: An Integrating System Dynamics Perspective

Tarek K. Abdel-Hamid

Associate Professor of Information Systems
Naval Postgraduate School
Code AS/AH
Monterey, CA 93943
Tel: (408) 656-2686
Email: 3991p@vm1.cc.nps.navy.mil
Fax: (408) 656-3407

Abstract

An integrating System Dynamics approach for modeling software reuse is proposed. The approach has three unique features. First, it integrates the multiple functions of the software reuse process, including both the managerial activities (e.g., setting reuse production/consumption goals, allocating resources, staffing, etc.) as well as the technical activities (reusable component identification, generation, consumption, etc.). Second, the model uses the feedback principles of system dynamics as a conceptual lens to view and analyze the complex web of dynamically interacting variables. Third, computer simulation is utilized to handle the dynamic complexity of the model and to conduct controlled experimentation.

The model serves as a "learning laboratory" to study the software reuse phenomenon and gain a better understanding of the interactions and trade-offs that characterize reuse policy decisions, as well as serve as a management support tool to evaluate/design organizational reuse policies.

Keywords: reuse modeling, reuse management, system dynamics

Workshop Goals: Advancing state of the art in reuse dynamic modeling; Learning.

Working Groups: Reuse management, organization, and economics; reuse process models; education.

1 Background

My work on modeling the dynamics of software reuse is an extension of an ongoing research program to study, gain insight into, and make predictions about the dynamics of the entire software development process. This is being accomplished through the development of dynamic process models using the techniques of System Dynamics. System Dynamics is the application of feedback control systems principles to model, analyze, and understand the dynamic behavior of complex systems. Its origins trace back to the pioneering work of Jay W. Forrester [1].

To-date, the focus of my work has been on the study of software project management [2, 3]. For example, my colleagues and I have modelled/investigated the dynamics of software project staffing [4]; cost/schedule estimation [5]; quality assurance economics [4]; and organizational learning [6].

The current research effort extends the application of the System Dynamics technique to model the dynamics of software reuse. In contrast to our project management models which focus on the individual project lifecycle, the scope here is broadened to capture the operations of a software development organization as multiple software products are developed, placed into operation, and maintained over a long period of time.

2 Position

Experience from working with managers in many environments indicates that they are generally able to specify the detailed relationships and interactions among managerial policies, resources, and performance. However, managers are usually unable to determine accurately the dynamic behavior implied by these relationships. Human intuition, studies have shown, is ill-suited for calculating the consequences of a large number of interactions over time, especially when cause and effect are distant in time and space [7]. This can, and often does, lead to the adoption of policies that prove dysfunctional in the long run.

Thus, one motivation for developing computer-based models of complex socio-technical systems (such as software reuse), is to have a capability to reliably and efficiently trace through time the implications of a complex web of system interactions. By utilizing computer simulation techniques we, thus, combine, the strengths of the software manager with the strengths of the computer. The manager aids by specifying relationships within his/her software reuse environment (e.g., effect of staff experience on reuse rate), the computer then calculates the long-term dynamic consequences of these relationships (e.g., on productivity, quality, etc.).

A second reason for constructing computer-based models, is for experimentation. Engineers turn to laboratory experiments to learn about the behavior of complex engineering systems. Our social systems are far more complex and harder to understand than our technological systems. Why, then, do we not use the same approach of making computer models of social systems, such as a software development organization, and conduct laboratory experiments on these models?

> (quote) The answer is often stated that our knowledge of social systems is insufficient for constructing useful models. But what justification can there be for the apparent assumption that we do not know enough to construct models but believe we do know enough to directly design new social systems? ... I am suggesting that we now do know enough to make useful models of social systems. Conversely, we do not know

> enough to design the most effective social systems directly without first going through a model-building experimental phase [8].

Indeed, over the last three decades the work of Forrester and others in the System Dynamics field demonstrated both the feasibility and utility of constructing computer-based models to study complex social systems. More recently, the technique has been successfully applied to the software project domain [2]. Using these tools, the software reuse manager, like the engineer, can have a laboratory in which he/she can simulate organizational behavior and learn quickly and at low cost the answers that would seldom be obtainable from trials on real projects.

Characteristic features of proposed modeling approach. The proposed approach has three unique features. First, the model integrates the multiple functions of the software reuse process, including both the managerial functions (e.g., setting reuse production/consumption goals, allocating resources, staffing, etc.) as well as the technical activities of reuse (e.g., reusable component identification, generation, consumption, etc.). This contrasts with most of the research to-date on software reuse which tended to emphasize one aspect or problem area in isolation e.g., domain analysis, organizational rewards, identification/ qualification of reusable components, economics, etc. Clearly such micro-oriented research does make a useful contribution to our understanding of the software reuse phenomenon. However, before we can say that we have a complete understanding, it is necessary to show that our knowledge of the individual components can be put together in a total system i.e., an organization can be synthesized, which allows for the interactions of all the relevant variables and of all the structural components [9].

> There is a need to learn what the relationships are between the technical and non-technical aspects, how one influences the other, and what makes a reuse program a success or a failure. Reuse should be seen as a whole, as an integrated system and therefore addressed as such. To see reuse as a separate and independent collection of tools, techniques, or methods will only perpetuate the current status [10].

The second unique feature of the proposed modeling approach is the use of the feedback principles of system dynamics to structure and clarify the complex set of dynamically interacting variables. Feedback is the process in which an action taken by a person or thing will eventually affect that person or thing. A feedback loop is the closed sequence of causes and effects, from an initial cause, its series of ripples through an entire chain of causes and effects, until the initial cause eventually becomes an indirect effect of itself. Circular feedback processes are universal in social systems, the software engineering domain being no exception [2].

The third feature, is the reliance on computer simulation to handle the dynamic complexity of such a model, and to serve as a "laboratory" for controlled experimentation.

> By using a (simulation) model of a complex system, more can be learned about internal interactions than would ever be possible through manipulation of the real system. Internally, the model provides complete control of the system's organizational structure, its policies, and its sensitivities to various events [1].

Overview of model structure and behavior. The model is quite detailed, containing more than 200 difference equations. Providing a full description of the model's structure is, thus, beyond the scope of this paper. Instead, I provide an overview of the model's five major sectors, which are shown in Figure 1. At the top of the figure is the software development and maintenance sector. The function of this sector is to provide the overall organizational setting within which software reuse is conducted and managed. It provides a characterization of the modeled organization's unique software development environment (e.g., size of project portfolio, average size of projects, type of software, maintenance backlog, etc.).

Having defined a particular organizational setting, we can next model the specific software reuse activities of interest. These are captured in two highly interdependent sectors, a reusable components production sector, and a reusable components consumption sector. The production sector models the reusable components repository and the production process that populates it. Factors affecting the reusable component deposition rate include the degree of functional overlap between applications in the domain, organizational reuse support, learning, schedule pressures, perceived costs, and organizational goals. Two things to note here. First, that most of these factors are dynamic, that is, they can change over time. And second, the distinction between reality and the perception of reality (e.g., as in "perceived" costs).

Similarly, the reuse consumption sector captures the reuse consumption rate and its determinants. These include: perceived benefits of reuse, repository size, reuse support, learning, overlap between applications, and schedule pressure.

The software production, maintenance, and reuse activities are all very much affected, and they themselves affect, the size and characteristics of the organization's human resource. The human resource sector captures the hiring and firing of staff, staff resource allocation, training, turnover, etc.

Finally, the management policy sector provides the leverage points in the model through which management can affect the software reuse process. Possible managerial interventions include setting reuse goals, allocating resources, organization, etc.

As suggested above, the software reuse process, like most organizational systems, is characterized by a complex network of interconnected feedback loops. As an example, two simple feedback loops are depicted in Figure 2. The lower loop is a positive self-reinforcing feedback loop. It shows how reuse can increase an organization's software development productivity, which would increase the organization's software delivery rate, this in turn can lead to the generation of more reusable components, which ultimately can lead to even higher reuse rates.

This self-reinforcing loop is counterbalanced by the negative feedback loop that is also shown. This second loop shows that a high level of reuse would decrease the amount of new/original software components developed, which, in turn, decreases the opportunities for developing reusable components. A reduction in the production of reusable components could then slow down or even decrease reuse.

Which of the two loops is dominant? To answer this question, we rely on the simulation capability of the model. A simulation run is shown in Figure 3. It shows that the two feedback loops of Figure 2 interact in a non-linear fashion. A linear-type interaction would have led to constantly increasing or decreasing reuse consumption and production rates (depending on which of the two loops is more dominant). Instead, the simulation shows that initially the positive feedback loop dominates. That is, in the early phases of a software reuse effort, when the reuse rate is relatively low, the reuse process "feeds on itself" to produce a period of accelerating growth.

Eventually, however, growth slows down. In this simulation run, this happens way before the organization reaches its theoretical reuse potential (which in this particular case is 60the balancing influence of the negative feedback loop. For as the reuse rate goes higher and higher, the amount of new/original code being developed decreases. This leads to smaller and smaller depositions to the repository. And because older reusable components are continuously being retired from the repository, the repository's size levels off, and even starts decreasing slightly (as shown in the figure). As this happens, the reuse rate follows.

Model Utility. The model will serve as a "learning laboratory" to study the software reuse phenomenon and gain a better understanding of the interactions and trade-offs that characterize reuse policy decisions, as well as servs as a management support tool to evaluate/design organizational reuse policies.

3 Comparison

Most of the research to-date on software reuse has tended to emphasize one aspect or problem area e.g., domain analysis [11], incentives [12, 13], identification/qualification of reusable components [14], economics [15], etc. In this research effort I build upon and extend what has been learned about the "pieces" of the software reuse phenomenon to construct a holistic dynamic model of the software reuse process that would allow us to study the complex set of interactions and trade-offs that characterize the process.

References

[1] J. Forrester, *Industrial Dynamics.* The MIT Press, Cambridge, Mass, 1961.

[2] T. Abdel-Hamid and Madnick, *Software Project Dynamics: An Integrated Approach.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1991.

[3] T. Abdel-Hamid and S. Madnick, "Lessons learned from modeling the dynamics of software development," *Communications of the ACM*, 1989.

[4] T. Abdel-Hamid, "The dynamics of software project staffing: A system dynamics based simulation approach.," *IEEE Transactions on Software Engineering*, Feb. 1989.

[5] T. Abdel-Hamid, "Adapting, correcting, and perfecting software estimates: A maintenance metaphor," *IEEE Computer*, Mar. 1993.

[6] T. Abdel-Hamid and S. Madnick, "The elusive silver lining: How we fail to learn from software development failures," *Communications of the ACM*, 1990.

[7] G. Richardson and G. I. Pugh, *Introduction to System Dynamics Modeling with Dynamo.* The MIT Press, Cambridge, Mass., 1981.

[8] J. Forrester, "Counterintuitive behavior of social systems," *Technology Review*, vol. 73, Jan. 1971.

[9] M. L. Griss, "A multi-disciplinary software reuse research program," in *Proceedings of the 5th Annual Workshop on Software Reuse, Palo Alto, CA., 1992* (M. Griss and L. Latour, eds.), pp. Griss–1:8, Department of Computer Science, University of Maine, Nov. 1992.

[10] R. Prieto-Diaz, "Making software reuse work: An implementation model," *Software Engineering Notes*, vol. 16, July 1991.

[11] R. Prieto-Diaz, “Domain analysis: An introduction,” *Software Engineering Notes*, vol. 15, no. 2, pp. 47–54, 1990.

[12] W. Tracz, “Software reuse: Motivators and inhibitors,” in *COMPCON87, San Francisco, CA*, Feb. 1987.

[13] R. Holibaugh and J. Morin, “A reuse incentive model,” in *The 5th Annual Software Technology Conference, HQ USAF/SC-USAF STSC*, Apr. 1993.

[14] G. Caldiera and V. Basili, “Identifying and qualifying reusable software components,” *IEEE Computer*, Feb. 1991.

[15] J. Gaffney and R. Cruickshank, “A general model of software reuse,” in *Proceedings of The 1992 International Conference on Software Engineering, Melbourne, Australia*, May 1992.

4 Biography

Dr. Tarek K. Abdel-Hamid is an Associate Professor of Information Systems at the Naval Postgraduate School. Before joining NPS, he spent two and a half years at the Stanford Research Institute. He is an advisor to NASA’s Jet Propulsion Laboratory (since 1988) on the development of computer-based tools for software project management. He received the B.S. degree in aeronautical engineering from Cairo University, Cairo, Egypt, in 1972, and the Ph.D. in management information systems from MIT in 1984. His research interests focus on software project management, software reuse, and system dynamics. He has authored or coauthored more than 30 papers on these topics and is the coauthor of Software Project Dynamics: An Integrated Approach (Prentice-Hall, 1991). Dr. Abdel-Hamid is a member of the ACM, SIM, IEEE-CS, and the System Dynamics Society.