Astronomical Data Analysis Software and Systems IX
ASP Conference Series, Vol. 216, 2000
N. Manset, C. Veillet, and D. Crabtree, eds.

Managing the Software Development Process

J. Lubelczyk

NASA Goddard Space Flight Center, Greenbelt, MD 20771

A. Parra

Computer Sciences Corporation, Lanham-Seabrook, MD 20706

Abstract. The goal of any software development project is to produce a product that is delivered on time, within the allocated budget, and with the capabilities expected by the customer and unfortunately, this goal is rarely achieved. However, a properly managed project in a mature software engineering environment can consistently achieve this goal. In this paper we provide an introduction to three project success factors, a properly managed project, a competent project manager, and a mature software engineering environment. We will also present an overview of the benefits of a mature software engineering environment based on 24 years of data from the Software Engineering Lab, and suggest some first steps that an organization can take to begin benefiting from this environment. The depth and breadth of software engineering exceeds this paper, various references are cited with a goal of raising awareness and encouraging further investigation into software engineering and project management practices.

1. Introduction

The goal of any software development project is to produce a product that is delivered on time, within the allocated budget, and with the capabilities expected by the customer and unfortunately, this goal is rarely achieved. However, Edward Yourdon argues that "a properly managed project, in a mature software engineering environment, managed by a competent manager, can repeatedly deliver a software system on time, within cost, and satisfactory to the user" (Yourdon 1997). Thus, success is a product of only three abstract variables, a properly managed project, a competent manager, and a mature software engineering environment. In this paper we will address the three variables from the perspective of "Can it really be that simple?" We want to emphasize that the simplicity of the variables does not imply that implementing them is easy. Software is a growing industry; as it matures finding methods to increase productivity and quality while keeping cost under control will justify investing some effort in implementing these simple concepts. The corresponding ADASS '99 presentation can be found on-line at http://seam.gsfc.nasa.gov.

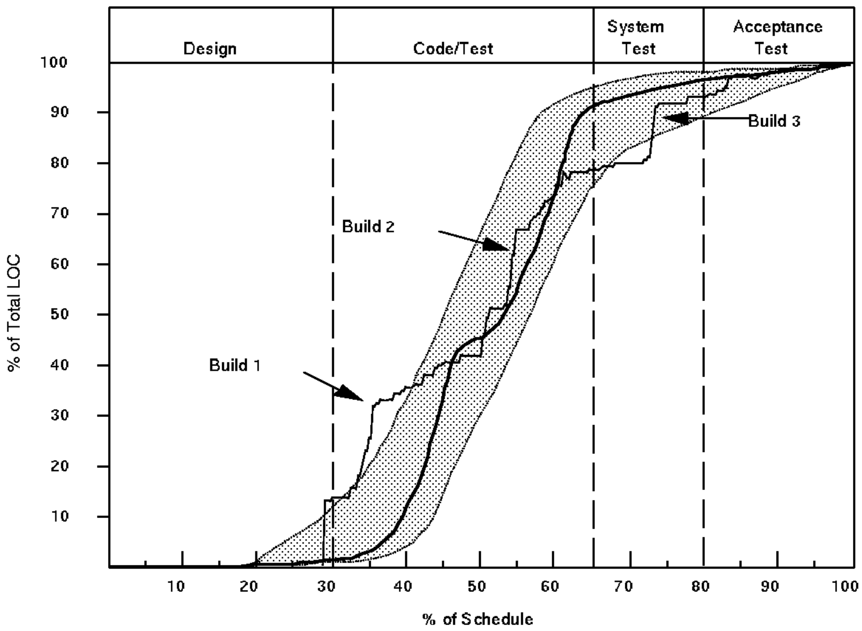

Figure 1. Tracking system size in lines of code (LOC) for a hypothetical project.

2. A Properly Managed Product

A properly managed project has a clear, communicated, and managed set of goals and objectives, whose progress is quantifiable and controlled and whose resources are used effectively to efficiently produce the desired product. When properly managed, a project usually has a communicated set of processes that cover the daily activities of the project, forming the project framework. As a result, every team member understands their roles and responsibilities and how they fit into the big picture, thus promoting the efficient use of resources.

The processes also provide quantifiable feedback (metrics) with respect to process goals and objectives. Metrics provide the information necessary to assess a project's progress against a baseline project plan. Particular attention should be paid to the cost, schedule, scope, and quality aspects of a project. One method of managing these areas is to apply earned value techniques that provide planned verses actual feedback. This approach can be used on a variety project processes such as code development which can be tracked as shown in Figure 1 (SEL 1995).

3. A Competent Project Manager

A competent project manager is someone with the knowledge, skills, and experience to lead a project to completion effectively and efficiently. A non-exhaustive list of the characteristics of a competent project manager can be found in Appendix A. A project manager uses the project framework as a tool to organize and control the project. A project manager must have the ability to operate within a wide and diverse set of roles and responsibilities. Roles and responsibilities that involve communication with the customer or stakeholders can be considered project boundary management, others are internal to the project and not seen or of interest to the customer(s).

Boundary management involves building positive relationships with customers and stakeholders (Parker 1994). A prime responsibility of the project manager is to serve as a conduit for information exchange between the stakeholders and the project staff, which involves balancing their competing demands. Changes in project schedule, cost and scope (requirements) must be approved and negotiated. Agreements must be struck with stakeholders with differing needs and expectations in order to establish a common set of requirements and goals for the project to be working toward. Open communication is essential and assuring that both the project members and stakeholders are speaking the same language is key, which usually means speaking in terms that the customer understands and eliminating much of the technical jargon.

Internally, a project manager provides support and leadership for the development team, and manages the software development effort by ensuring that the defined processes are tailored, communicated, and followed. Unless the project is small, a project manager will usually not be the technical lead. Their primary role is one of risk mitigation. They identify project problem areas based on process feedback, investigate the root cause of project problems and determine if they are technical, process, or people related, and apply resources to areas that will have the biggest return on investment.

In any development effort, people put the project framework in motion. As such, staffing and team building are critical roles for the project manager, it is no longer enough to simply work cost and schedule issues. A project manager must get the most out of their human resources and a key to getting the most out of people is to accept the obvious fact that they are different. They have different backgrounds, experiences, and ideas and it is this diversity that adds a great deal of value to a project and a great deal of work for the project manager.

One of the best ways to dramatically increase the chance of project success is to have a team where the whole is greater than the sum of the part, a jelled team (DeMarco & Lister 1987). One good way to help a team to jell is to allow the team to tailor the project framework to support the unique needs of their project. A wonderful project framework won't provide much help to a project if the team doesn't believe in it. Tailoring is also a way to improve productivity and the current prescribed baseline project framework. By allowing teams to perform "controlled experiments," an organization promotes an environment for building jelled teams and puts the Hawthorne affect to work for it (DeMarco & Lister 1987).

Having team members agree on a common set of processes is not an easy job. It requires keen skills in facilitation, conflict resolution, and communication. A good way to help a diverse team define a common set of processes is to first develop a common set of goals for the project. Another perhaps not so obvious fact is that teams don't attain goals; people on the teams attain goals. Thus, the purpose of a team is not goal attainment but goal alignment (DeMarco &

Lister 1987). So when the occasional project "Holy War" develops, i.e. "What is an object," these goals can be used to facilitate a negotiated peace and help the parties work toward agreement on a common approach.

One way to help a team stay focused on its goals is to generate a weekly status report. It serves as a means to remind them of approaching scheduled milestones, provide them with thoughts on how well the project is doing in the eyes of the stakeholders, communicate issues by including a verbatim copy of each subsystem's status report, and motivate the team by supplying a "quote of the week." A common mistake made by organizations is to have information only flow up the management chain. This omission of bi-directional communication discourages team ownership because only the managers have a complete understanding of a project's status and issues, resulting in a top down decision-making paradigm. Keeping a team informed of a project's status encourages them to jell and be part of the solution. This simple tool improves a team's productivity by removing a decision making bottleneck, thus improving process and team efficiency.

4. A Mature Software Engineering Environment

The concepts of a mature software engineering environment are easily described in comparison with an immature environment. In the area of processes, an immature environment's processes are ad hoc (or chaotic) and the individual projects are independently defining and improving their processes, resulting in unrelated processes and metrics from project to project. This environment lends itself to poor and often optimistic project cost and schedule estimates because these estimates are usually not based on quantifiable historical data. Product quality will be inconsistent across projects and may not be improving over time. Process improvement will be limited at best and usually will not take place across an organization. This lack of coordination and communication of corporate knowledge produces an organization that may have concurrent successful and unsuccessful projects.

An immature software engineering environment offers little support from the organization, which means that project success must solely rely on the skills, talent, and heroic efforts of the personnel on the project. A chaotic environment forces the manager to be reactive to problems as they occur, because the process feedback is unavailable. This information vacuum severely limits their ability to control and mitigate the risks associated with a project. This absence of information further inhibits an organization from improving by not providing the historical documentation needed. This fire fighting method of management may be useful in the short run to solve immediate problems, but results in a myopic short term perspective which doesn't promote the efficient use of resources. This also makes the repeatability of an experience for future work or additional builds difficult if there is project turn over, because success depended on internal processes that heroic and talented individuals naturally follow. Replacing people requires a long learning curve in training personnel before they are fully able to contribute to the project.

Success is not a term often associated with an immature software engineering environment. Optimistic cost and schedule estimates, undefined processes and metrics, and undefined product quality are huge obstacles that must be overcome by the project personnel in order to achieve project success. Basic PERT analysis predicts that a project in this type of organization has only a 16.7% chance of meeting cost and schedule estimates with inconsistent product quality, not a recipe for success.

Now let us look at these organizational properties from the perspective of the Mature Software Environment. A mature software engineering environment provides the historical support and feedback for the project management process. This support results in improvements in several key project processes such as planning by providing a basis for realistic cost, schedule, and staffing estimates; controlling by supplying metrics that quantify project progress and product quality; and product development by defining the base-line project framework and product standards. It also serves as the basis for long term improvement by supplying improvements to the current baseline set of development processes.

Repeatability comes naturally when processes are both defined and documented. The reuse of processes, or project framework, increases the accuracy of the predicted product quality. Measures are taken to understand and monitor products and processes. These measures are also captured and analyzed by the software engineering group and provide a historical basis for the estimation of project cost, staffing, and schedule. Improving estimates based on historical data dramatically increases the chances of project success, since estimates provide the baseline for the project. Using an organization's historical data further improves estimates by tailoring a generic model as shown in Figure 2 (SEL 1995).

The project manager is involved in the process as the coordinator; planning, organizing, staffing, leading, controlling, and communicating (Mackenzie 1969). Roles and responsibilities are clearly defined, the manager and the developers are aware of what is expected of them. Reactionary management occurs less often, allowing for the proactive allocation of resources to mitigate identified risks. Crisis is not a way of life for the mature software environment.

These mature characteristics are not merely for the large organization. Watts Humphrey's Personal Software Process is a formal example of how an individual can measure, monitor and improve their individual performance (Humphrey 1996). A more simplified example would be for individuals to monitor their own performance using time management tools, such as the Franklin Planner, which results in their ability to estimate future work based on documented past performance. At the small group level, an organization that uses configuration management, analysis of metrics, and project management is well on it's way to becoming a mature software engineering environment. In the case of Computer Sciences Corporation at the NASA Goddard Space Flight Center, the use of mature software engineering characteristics has resulted in an organization that is both ISO 9001 compliant and CMM Level 5 (Optimizing). These two tools can be used to guide an organization toward maturity in software engineering.

While building a mature software engineering environment, the CSC organization has seen a reduction in errors per KSLOC from 6.5 in 1985 to 0.74 in 1996. This was accompanied by a reduction in Mission Cost by a factor of two and an increase of reuse from 22% to 78%. This is baselined data that is averaged

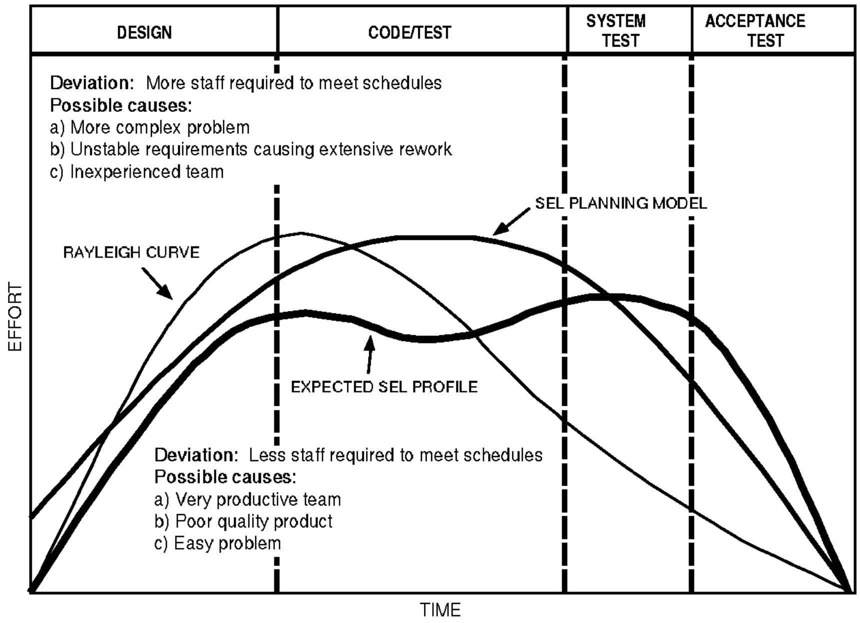

Figure 2. Improving estimates based on historical data dramatically increases the chances of project success, since estimates provide the baseline for the project. Using an organization's historical data further improves estimates by tailoring a generic model.

over typical projects. Building a mature environment requires some investment but the benefits and improvement of the software product are quantifiable.

5. Recommended First Steps

The goal of this presentation was to raise awareness and to encourage further investigation of the application of software engineering and project management techniques to your organization. We suggest that you become familiar with some of the basic software engineering project management concepts. The recommended reading list given here is small, and intended only as a very introductory list. For additional education there are formal classes in software engineering and project management as well as conferences.

Next, determine your organizational goals and find a way to measure (quantify) how your goals will be met. Improvements that do not relate to your goals will have little affect in the areas that you care about, reducing the effectiveness of your investment in the area of software engineering and project management.

Then start applying project management and software engineering techniques in small, focused, and goal oriented areas that will produce results quickly. This recommendation allows you to test out improvements as well as increase their value. The SEL for years has learned by conducting small experiments and then taking that information to a larger group once the value of the technique evaluated has been assessed. Finally, begin to understand the environment in which your organization is developing software. This understanding serves as the basis for process assessment and long term process improvement.

6. Suggested Reading List

Software Engineering Project Management (IEEE Computer Society) contains information to help you understand and successfully perform the unique role of the software project manager.

Peopleware (DeMarco & Lister 1987) gives quite a number of effective ideas for managing the human resource.

Cross Functional Teams (Parker 1994) gives practical, proven practices for working with teams.

A Guide to the Project Management Body of Knowledge by the PMI Standards Committee, Project Management Institute Publishing Division provides an overview and guide to project management concepts established by the Project Management Institute.

The *Software Measurement Guidebook* (Software Engineering Laboratory 1995) provides guidance for establishing a measurement program in an organization. Copies may be obtained on-line at http://sel.gsfc.nasa.gov/.

The *Software Management Guidebook* (Software Engineering Laboratory 1996) (NASA-GB-001-96) provides a list of core products and activities required of NASA software projects as well as providing guidance for managing software projects. Copies may be obtained on-line at http://sel.gsfc.nasa.gov/.

Recommended Approach to Software Development from the NASA GSFC Software Engineering Lab (SEL-81-305) provides a set of guidelines that constitute a disciplined approach to software development. Copies may be obtained on-line at http://sel.gsfc.nasa.gov/.

7. Appendix

Here are some important characteristics of a competent project manager. The manager's **values** are:

- Customer focused,
- Dedicated to success,
- Willing to accept a challenge,
- Values diversity and is able to obtain team agreement on a set of best practices,
- Understands that communication is essential to project success and supports it in every way possible,
- Understands the nature of the work being managed,
- A pessimist with an optimistic attitude (Plan for the worst, hope for the best),
- Understands the needs of others,
- Validates assumptions.

A manager's **knowledge and skills** include:

- Able to make timely decisions independently based on usually incomplete data,
- Able to lead, manage and administer (as needed) (Mackenzie 1969),
- Able to negotiate,
- Able to communicate and facilitate,
- Able to plan, organize, staff, lead, and control (Mackenzie 1969),
- Able to understand and influence the organization (PMI 1996),
- Able to endure an environment that offers lots of responsibility with little authority,
- Process improvement and reengineering,
- Team building,
- Conflict resolution,
- Project management tools, including:
    - Project organization charts and Role and Responsibility assignments,
    - Project meetings, status summaries and Web pages,
    - Work Breakdown Structure,
    - Scheduling and estimation techniques such as network diagrams and PERT,
    - Earned value analysis,
    - MS Project or similar application,
    - Project plan,
    - Risk analysis and mitigation strategies,
    - Requirement mapping and Quality Assurance.

References

DeMarco, T. & Lister, T. 1987, Peopleware, (New York: Dorset House)

Humphrey, W. 1996, Introduction to the Personal Software Process, Addison-Wesley Publishing Company

IEEE Computer Society 1997, Software Engineering Project Management, ed. Richard Thayer

Mackenzie, A. 1969, The Management Process in 3-D, Harvard Business Review

Parker, G. 1994, Cross Functional Teams, (San Francisco: Jossey-Bass Publishers)

PMI Standards Committee 1996, A Guide to the Project Management Body of Knowledge, Project Management Institute Publishing Division

Software Engineering Laboratory 1981, Recommended Approach to Software Development, SEL-81-305 (Greenbelt, MD: NASA Software Engineering Laboratory, GSFC)

Software Engineering Laboratory 1995, Software Management Guidebook, NASA-GB-001-96 (Greenbelt, MD: NASA Software Engineering Laboratory, GSFC)

Software Engineering Laboratory 1995, Software Measurement Guidebook (Greenbelt, MD: NASA Software Engineering Laboratory, GSFC)

Yourdon, E. 1997, Software Engineering Project Management (forward), IEEE Computer Society