Lessons Learned in Framework-Based Software Process Improvement

Pankaj Jalote
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
Kanpur, India – 208016
jalote@cse.iitk.ac.in

Abstract

Software process improvement (SPI) has emerged as a critical area for organizations involved in software development. There is now considerable evidence that SPI can provide substantial gains in quality, productivity, and cycle time. Currently, most organizations that embark upon a SPI program tend to use a framework like the Capability Maturity Model for their process improvement. In this article we discuss some of the lessons learned in using these frameworks for software process improvement. First, three critical success factors are discussed. The rest of the lessons have been grouped into three categories – framework related, process related, and SPI management related. For each category we discuss a few key lessons. These lessons are based on the experience of the author in implementing a CMM-based SPI program in a large software organization in India, and helping many organizations in India and other countries (primarily the US and Mexico) with their SPI programs.

1. Introduction

Technically, a process for a task comprises a sequence of steps that should be followed to execute that task. For an organization, however, the processes it recommends for use by its engineers and project managers are much more than a sequence of steps—they encapsulate what the engineers and project managers have learned about successfully executing projects. These processes help managers and engineers emulate past successes and avoid the pitfalls that lead to failures. Software process is the collection of processes for the various tasks involved in executing projects for building software systems.

As a result of changes in technology, knowledge, and people's skill, the processes for performing different tasks change. In other words, processes evolve with time. With knowledge and experience, processes can, and should, be "fine tuned" to give better performance. Software process improvement is concerned with this tuning of the software process. There are many published reports showing the benefits SPI can bring to quality, productivity, and cycle time (for example [1, 3, 5, 6, 9, 11, 19]).

For an organization that wishes to embark upon a SPI program, there are two clear approaches – totally internal SPI and framework-based SPI. In the internal SPI, the current processes of the organization are analyzed and depending on the shortcomings discovered and the goals of the SPI, initiatives are taken for improvement. A framework-based SPI, on the other hand, uses an external framework against which a process is analyzed and which may be used to determine the course of action in the SPI initiative.

Though SPI has been around for a long time, in the recent years framework-driven SPI has gained a lot of momentum. The most influential frameworks for SPI are the Capability Maturity Model (CMM) for software [18] and ISO9001 [12, 13]. Currently, CMM is, by and large, the most widely used framework for software process improvement. Some reports describe how the CMM based initiatives were implemented and the key success factors [4, 9, 10]; a detailed description of an implementation of the CMM in an organization is given in [14].

Though the CMM and ISO have been around for many years, organizations still find it hard to "implement" these frameworks – i.e. improve the processes so that they also satisfy the requirements of these frameworks. A survey done by SEI suggests that over two-thirds of the people responsible for SPI know what needs to be improved, but needed more guidance on how to do it [10].

In this article we share some of the lessons learned while using CMM or ISO as the framework for SPI, though the article focuses more on the CMM as it is currently more widely used and many people believe that it is better suited for software. This article is based on the author's experience in implementing CMM in a large software house in India, and then helping some others move up the maturity ladder. These lessons should help the organizations wanting to move up the maturity ladder in making their SPI programs more effective.

First we discuss the three key success factors of an SPI program**:** a dedicated group for performing process management activities, a delivery system to deploy the processes, and senior management sponsorship. Then we

group the other lessons learned in three categories – framework related, process related, and SPI management related. In each of the categories, we discuss a few key lessons.

2. Critical Success Factors

For a framework-based SPI to deliver results there are three critical success factors. In some sense, these form the basic lessons we have learned – without them the SPI initiative is not likely to succeed. These success factors are (a) having a dedicated group responsible for the SPI initiative with some full-time members, (b) having a suitable delivery system in place for deploying processes, and (c) senior management involvement and commitment.

The basic premise in SPI is that processes can be improved. This clearly implies that the processes are not static and will change (improve) over time. Improvement generally requires analysis and evaluation of the existing processes leading to identification of areas of improvement. To facilitate analysis, it is important to have a clearly defined definition of the processes being analyzed. Analysis is then followed by enhancing appropriate processes. This cycle of defining, analyzing, and refining repeats itself. These tasks in the software process improvement, which form the process management process, need to be executed by some group. As these tasks are fairly involved and require commitment over a substantial period of time, a critical success factor is to have a dedicated group to perform these tasks related to process definition and management. This group is frequently called the Software Engineering Process Group (SEPG).

Our experience also indicates that having only part-time members in this group who volunteer to this "extra" job usually does not succeed. It is highly desirable to have a core group of full-time people (whose strength is of the order of 1% to 2% of the engineering staff), which should be supplemented by volunteers across the organization who participate on a part-time basis for specific tasks during various SPI initiatives.

Frequently, it is assumed that process definition is the main challenge in an SPI initiative. This stems from an assumption that defining "good" processes is the tough part, and once such processes are defined, people will follow them automatically because they are "good". This, unfortunately, is far from the reality on the ground. The most difficult task in an SPI initiative is deployment of the defined processes. A survey of some high maturity organizations also supports the view that process deployment is perhaps the most demanding task during SPI initiatives [15].

So, besides putting effort into process definition, it is imperative to have *deployment mechanisms* in place for deploying the processes. If effective deployment mechanisms are in place, then the task of changing processes becomes considerably easier. And without an effective deployment mechanism, regardless of how "good" the processes are, they are more likely to remain on-paper only. Hence, for a successful SPI, existence of an effective delivery mechanism is a critical success factor. If it does not exist, it has to be formed (and this is what the Software Quality Assurance key process area at level 2 of CMM and the internal audit clause of ISO9001 try to do.)

Deployment frequently requires a three-pronged approach: training, consulting in use of processes, and a formal audit system. Training involves educating practitioners on the need for change and on the new processes to be used. Consulting is needed since, when time comes to deploy the processes, practitioners sometimes run into difficulties. Having consulting help available as and when needed is of great value for deploying processes. And finally, despite best of intentions, processes are sometimes not followed. A formal audit system which checks for process compliance is therefore essential to validate that processes are being properly used.

The final success factor is commitment and involvement of the senior management. SPI initiatives typically involve a large number of people besides the SEPG. And, as mentioned above, deploying processes across the organization is difficult and requires support from across the organization. Overall, an SPI initiative does not get this support unless the senior management shows strong commitment to the initiative.

The senior management commitment has two components: providing the resources (both people and financial) for the initiative, and spending their own time in participating, monitoring, and resolving issues. Without this commitment, the message goes that though the organization desires the SPI, they are not serious about it. Active involvement of senior management helps provide visibility to the people participating in the initiative, which is a strong motivator, particularly for those people who are participating part-time on a voluntary basis. In the author's experience, in almost all organizations that were very successful with their SPI, the initiative had the full backing from the CEO, who actually personally monitored the initiative. And in the situations where the SPI initiative continues to linger on, the CEO does not regularly monitor and drive the initiative, and does not allocate adequate resources to the initiative, relying more on voluntary and heroic effort of people.

3. Framework Related Lessons

When a framework like the CMM is used, sometimes people take it literally and as "Gospel" and then try to "implement" the key practices of the CMM (or the clauses of ISO). Out experience is that to get the bet results in SPI, the model being used has to be properly interpreted for the organization. Some of the key lessons we learned relating to the framework revolve around this concept.

Don't work for the framework, let it work for you. Frameworks like the CMM provide considerable flexibility to effectively support the business goals of the organization. For example, in the CMM, goals of each Key Process Area (KPA) need to be satisfied, but not the detailed key practices mentioned for that KPA. Even in goals, sufficient flexibility is possible in interpretation. (E.g. CMM defined for large defense systems but now used by smaller organizations – the analyses of SEI's data reveals that most of the users are not defense based [17].) Given that there is sufficient flexibility in the model, it is a poor approach to start treating the model literally without keeping the larger picture in mind. A simple-minded implementation of CMM will start implementing the key practices, which is likely to result in complicated processes that may not be suitable for the organization. The processes should be designed to solve the problems faced in the organization and projects. The framework should be used to guide this effort, but should not be used as the main basis for designing the processes.

Once a framework is accepted, don't argue with the framework. A lot of energy is sometimes spent within an organization in arguing for or against a model or regarding the suitability of a particular model. Given that there is sufficient flexibility in a model like the CMM, it matters less which model you are using – most models will allow reasonable practices. It is better to accept a framework and focus the effort on how to leverage and interpret the framework to provide benefits to the organization. If the energies can be focused on how to use the framework best, the result is likely to be better.

4. Process Related Lessons

Of course, the processes to be used by the organization have to be properly defined. To achieve a certain maturity level, these processes will have to satisfy some properties required by the framework. In our eagerness to implement the framework we may miss out on the benefits the processes are supposed to provide. Some of the process related lessons are given below.

Keep the processes simple. There is no point in having detailed processes, as their implementation cannot be verified. And as we have discussed, a key aspect of deploying the processes is to be able to verify their implementation on projects through audits. It is a mistake, often due to over-enthusiasm, on the part of process designers, to have a process that specifies even the smallest task that has to be performed – such processes are virtually impossible to verify. How do you check if the small step mentioned in the process was done? Each process step should have clear outputs which can be used for validation. In this regard, it is best to separate guidelines and checklists from processes. For example, detailed steps for design are better kept as guidelines or checklists, which are used by the designer but whose execution is not necessarily to be validated by an auditor. This makes the processes themselves simple and more stable, and at the same time provides flexibility in methodologies at the lowest level by providing different checklists and guidelines. Keeping the processes simple makes them verifiable and minimizes the desire by practitioners to "fake it".

Have the executors of the process define the process. It is a bad idea for someone other than the users of the process to define the process. So, to define a process, it is perhaps best to have a task force that consists primarily of users of the process, and which is given some high-level guidelines and requirements by the SEPG. Processes defined by the users themselves have a much better chance at being accepted by the users. While defining processes, wherever possible, it is best to "standardize" processes that are being practiced by some in the organization. In other words, it is best to leverage the experience within the organization to define processes. Frequently, for many of the problems, solutions have already been found within the organization. In these situations, the task of process definition becomes identifying the solutions and then "packaging" them properly for a wider use. If the processes being proposed are substantially different from what has been practiced in the organization, then it is best to pilot the process on some project before "standardizing" it.

Keep metrics simple and with value for project management. When organizations and people learn to use metrics better, they can evolve their own metrics. However, in the start, keeping the metrics program simple and standard helps adoption of the metrics. The basic metrics that should suffice in most cases are effort, defects, schedule, and size [7, 16]. It is also important that the metrics collected provide value for the project in which they are being collected. Collecting metrics that are primarily for historical analysis or for organization-level

analysis tends to be viewed as a "burden" by the project, and may be resisted. If the metrics provide direct value to the project, there is added incentive to collect them. At the very least, they should help the project manager by providing good visibility in how the project is progressing on the three critical dimensions of cost, schedule, and quality – for project management this is the main use of metrics [2]. Tool support for metrics collection and analysis should also be built early, so metrics collection and analysis is not an overhead but an advantage for the project

5. SPI Management Related Lessons

In a framework-based SPI initiative, one of the objectives is to reach a certain level in the framework. This initiative can be quite extensive. Perhaps most SPI initiatives that do not succeed, like software projects that fail, do so due to improper management of the initiative. Here we give some of the lessons learned regarding managing the CMM initiative in an organization [14].

Treat each SPI initiative as a project. Conceptually, process improvement is an ongoing activity, with most models for process improvement being cyclic (e.g. the IDEAL model [8]). However, in practice, software process improvement, at least in the initial stages, can be treated like projects, with each process improvement initiative being a project. The objective of the project can be to satisfy some requirements of the framework, like achieving a level I of the CMM. With this objective, the SPI is a project which should end with an assessment demonstrating the achievement of the objectives. This project should be planned and managed just like any other (software) project. The project should have a project manager, who is assigned resources that are sufficient for the scope of the project. It is best if the goal of the project is articulated in terms of a maturity level – it is a clear and measurable goal and provides an "icon" around which the organization can be rallied.

Have a schedule of one year or less. Once the SPI initiative is treated as a project with a well-defined goal, it makes sense to achieve the goal quickly. A shorter time span for the SPI project will help keep the attention of the senior management as well as the people who are participating in the initiative, and will provide early feedback on the initiative. It is best to target a level that can be achieved in a shorter time span; a schedule of about one year is perhaps best.

Manage the risks to the SPI project. If the SPI is treated as a project with a well-defined goal, then there will be risks. Use of risk management, helps in achieving the SPI goals. The risk management in an SPI project will deal with those conditions which can cause the project to not achieve the desired goal. With SPI projects, most of these risks will be internal organizational issues. Through risk management they are prioritized, their impacts understood, and suitable measures taken to mitigate them. Like in software project management, risk management is perhaps one of the most important techniques that can help the project succeed.

The "big bang" approach to deployment can work. If there are multiple changes that are desired to achieve some level, then it may be best to define methods for all of these and then deploy them all together. This method makes the task of training, hand holding, etc. for the SEPG a one-time, though intense, task. By doing it in increments, the deployment mechanism (which will generally involve training, consulting etc.) will have to be activated each time. The incremental approach can also create resentment among practitioners due to the "stream" of changes they are requested to implement. By implementing all changes at once, the change may be substantial, but the end state is reached quickly. However, the amount of change should be such that it can be deployed in the span of about a year, as discussed above.

6. Conclusion

There is a considerable interest in software organizations in software process improvement (SPI) to improve the quality and productivity of the organization. The capability maturity model (CMM) and ISO9001 are currently the most widely used framework for process assessment and improvement. This article discusses some of the lessons learned in framework-based SPI.

It is our experience that SPI is not very hard or complex. It basically requires commitment from the senior management of the organization and the will to do it. We also believe that a framework-based improvement makes the task of process improvement considerably easier and provides the desired results. Whether or not the framework is suitable for the business is frequently not a very pertinent question as frameworks like the CMM provide sufficient flexibility to match the business goals of the organization. Hence, the organization has sufficient flexibility in setting its processes while still satisfying the requirements of the framework.

Our experience is that a suitable level of CMM should be set as the target for SPI, and the duration for achieving the goal should be kept reasonably short. Then the SPI project should be planned and managed like a regular

project. Proper risk management for the SPI project can help tremendously in achieving the goals.

In the end, we must keep in mind that SPI or the CMM do not solve all the problems related to projects. Processes and process improvement have limitations. For overall improvement, it is clear that one has to consider the other two aspects that determine the quality and productivity of an organization, namely people and technology.

7. References

[1] L. J. Arther, "Quantum improvements in software system quality", *Commn. Of the ACM*, 40:6, June 1997, pp. 47-52.

[2] N. Brown, Industrial-strength management strategies, *IEEE Software*, July 1996, pp. 94-103.

[3] K. L. Butler, "The economic benefits of software process improvement", *Crosstalk*, July 1995, pp. 10-19.

[4] M. Daskalantonakis, "Achieving higher CMM levels", *IEEE Software*, July 1994, pp. 17-24.

[5] M. Diaz and J. Sligo, "How software process improvement helped Motorola", *IEEE Software*, vol. 14, no. 5, Sept/Oct 1997, pp. 75-81.

[6] R. Dion, "Process improvement and the corporate balance sheet", *IEEE Software*, July 1993, pp. 28-35.

[7] R. Grady and D. Caswell, *Software Metrics: Establishing a Company-Wide Program*, Prentice Hall, 1987.

[8] J. Gremba and C. Myers, *The IDEAL model: A practical guide for improvement*, http://www.sei.cmu.edu/ideal/ideal.bridge.html, 1997.

[9] T. J. Haley, "Software process improvement at Raytheon", *IEEE Software*, Nov. 1996, 33-41.

[10] J. D. Herbsleb and D.R. Goldenson, "A systematic survey of CMM experience and results", *18th Int. Conf. On Software Engineering*, Berlin, 1996, pp. 323-330.

[11] W. Humphrey, T. R. Snyder, and R. R. Willis, "Software process improvement at Hughes aircraft", *IEEE Software*, July 1991, pp. 11-23.

[12] *ISO9001, Quality Systems – Model for Quality Assurance in Design/Development, Production, Installation, and Services*, Intl. Standards Organization, Geneva, 1987.

[13] *ISO9000-3: Guidelines for the application of ISO9001 to the development, supply and maintenance of software*, International Standard, 1991.

[14] P. Jalote, *CMM in Practice – Processes for Executing Projects at Infosys*, Addison-Wesley (SEI Series on Software Engineering), 1999.

[15] P. Jalote, Use of metrics in high maturity organizations, *Software Quality Professional*, Vol 4, No 2, March 2002.

[16] L. H. Putnam and W. Myers, *Industrial Strength Software – Effective Management Using Measurement*, IEEE Computer Society Press, 1997.

[17] "SEMA – Maturity Profile", www.sei.cmu.edu/sema/profile.html.

[18] Software Engineering Institute, contributors: M. Paulk, et. al., *"The Capability Maturity Model for Software, Version 1.1"*, Addison Wesley, 1995.

[19] H. Wohlwend and S. Rosenbaum, "Schlumberger's software process improvement program", *IEEE Tran. On Software Engg.*, 20:11, Nov. 1994, pp. 833-839.