

Lecture 02

Investment in the Requirements Process

The industry average investment in the requirements process for a typical system is 2% to 3% of total project cost. It should be evident from the information already presented that this amount of investment is inadequate and in fact is the root cause of the failure of many projects. Data from the U.S. National Aeronautics and Space Administration (NASA) described in [2] provide a clear and powerful message: projects that expended the industry average of 2% to 3% of total project cost/effort on the (full life cycle) requirements process experienced an 80% to 200% cost overrun, while projects that invested 8% to 14% of total project cost/effort in the requirements process had 0% to 50% overruns [2, p. 9]. (Obviously, our goal is not to have overruns at all; however, a smaller overrun is preferable to a larger one!) This book describes how to achieve an appropriate level of investment in the requirements process and the associated benefits.

A Process Approach

Over the past two decades, there has been considerable discussion of the value of a “process approach.” By a process approach, I mean developing and using a documented description—a process flowchart and an accompanying process description (PD)—of a set of activities that results in the accomplishment of a task or achievement of an outcome. Based on my experience, there is great value to using a process approach:

Those who support the activity document the actions or activities involved in getting something done.

Once documented, there is a common (shared) understanding of what is involved.

The documented process can be understood by all who are involved.

Those involved, having a common understanding, can suggest improvements to the process (enabling *continuous improvement* and empowering those who are involved to contribute ideas for making the process better).

Several general process models have been developed. For example, the Capability Maturity Model (CMM[®]) [3] developed by the Software Engineering Institute (SEI) at Carnegie Mellon University in the late 1980s provides an industry standard framework for assessing the maturity/capability of a development process. The current version of this model is called the Capability Maturity Model Integration (CMMI[®]) [4]. Its success is due to the model’s capability to discern whether software is being developed more effectively. One can tell whether the development effort is “better” or “worse” over time. Some PMs may question the value of process improvement, believing that it diverts resources from their main responsibility of satisfying the customer

needs and that process improvement costs too much money. Industry data maintained by the SEI reflect a 7:1 return on investment (ROI) from process

improvement.² Other industry data consistently report 40% to 50% rework on development projects. Reducing rework is a lucrative target for process improvement efforts. Reducing rework can provide the resources to undertake process improvement initiatives.

Also, requirements process models are available; for example, one is provided in my earlier book and available on my Web site (www.ralphyoung.net); the spiral model for requirements engineering; and a model is provided in *Mastering the Requirements Process* [5].

The Requirements Plan

A requirements plan should be developed by the RA early—either during the proposal preparation phase or soon after a decision is made to proceed with a development project or task. The purpose of the requirements plan is to determine and document how the real requirements will be evolved and how the requirements-related activities in the system life cycle (listed and described above) will be addressed. Following is a list of suggested topics for this plan and a description of each topic:

Purpose (of the requirements plan): This was defined in the preceding paragraph.

Contract/project summary: A high-level summary of the objectives of the system or software should be provided. This section can be extracted from other documents such as a *vision and scope document* that may have been written previously to describe the overall intent.

Background: This section should describe the situation that led to the decision to develop the system or software. It should identify the major stakeholder groups—those who have an interest in the system, such as the customer (the person or organization providing the funds to pay for the project or its end products), various categories of users, developers, and major suppliers.

Evolution of the requirements: A *mechanism* should be agreed upon between the customers/users and the development team to review the stated requirements and evolve the real requirements. Customers may resist this effort, believing that they already have a “good” set of requirements. The RA should be familiar with industry experience concerning how many projects have failed and how many more have been seriously and negatively affected by a failure to invest in this critical step [1, p. 48]. A *mechanism* is a way to get something done or to achieve

2. See B. K. Clark’s “Effects of Process Maturity on Development Effort,” Center for Software Engineering, University of Southern California, 1999, at www.ralphyoung.net/goodarticles, for an excellent summary of

the benefits of process improvement.

a result. The recommended mechanism to evolve the real requirements is a cooperative or joint team composed of one or a few representatives of the users and a similar number of technically proficient developers. The members of the *joint team* should review the requirements to ensure that they meet the criteria of a good requirement provided in Table 1.1. Also the rationale for each requirement (why it is needed) should be documented. Industry experience is that by taking this one step, up to *half* of the requirements can be eliminated.

Roles and responsibilities of the project's personnel involved in requirements-related activities: Even on a small project, it's likely that more than one person will be involved with requirements-related activities. It's helpful to clarify and document these roles, so that everyone understands his or her unique and common responsibilities. For example, someone should be designated to provide requirements training (the content of this training is described in Chapter 5). Another person will be responsible for the automated requirements tool. Yet another person may have responsibility for the key processes to be utilized on the project, including the *requirements process*. Still another may be responsible for designing the architecture (the underlying structure of the system or software). Since the requirements and the architecture impact each other, a recommended requirements practice is to iterate the requirements and the architecture repeatedly—this results in stronger requirements and a more robust architecture [1, pp. 131–158].

Table 1.1 Criteria of a Good Requirement

<i>Each Individual Requirement Should Be</i>
<i>Necessary:</i> If the system can meet prioritized real needs without the requirement, it isn't necessary.
<i>Feasible:</i> The requirement is doable and can be accomplished within budget and schedule.
<i>Correct:</i> The facts related to the requirement are accurate, and it is technically and legally possible.
<i>Concise:</i> The requirement is stated simply.
<i>Unambiguous:</i> The requirement can be interpreted in only one way.
<i>Complete:</i> All conditions under which the requirement applies are stated, and it expresses a whole idea or statement.
<i>Consistent:</i> It is not in conflict with other requirements.
<i>Verifiable:</i> Implementation of the requirement in the system can be proved.
<i>Traceable:</i> The source of the requirement can be traced, and it can be tracked throughout the system (e.g., to the design, code, test, and documentation).
<i>Allocated:</i> The requirement is assigned to a component of the designed system.
<i>Design independent:</i> It does not pose a specific implementation solution.
<i>Nonredundant:</i> It is not a duplicate requirement.
<i>Written using the standard construct:</i> The requirement is stated as an imperative using "shall."
<i>Assigned a unique identifier:</i> Each requirement shall have a unique identifying number.
<i>Devoid of escape clauses:</i> Language should not include such phrases as "if," "when," "but," "except," "unless," and "although." Language should not be speculative or general (i.e., avoid wording such as "usually," "generally," "often," "normally," and "typically").

Definition of the requirements process to be used: As noted above, a documented requirements process is essential. A process may be thought of as a flowchart (indicating the steps performed and the person or organization that performs each step) accompanied by a narrative PD that indicates, for example, the name of the process, its customers, inputs to the process, outputs from the process, tasks performed in the process, the person or organization performing each task, and some measures (metrics) that can be used to evaluate the quality of the products produced by the process and the performance of the process. Experience shows that it's a good practice to involve the major stakeholders of a process in its construction. This approach encourages understanding, completeness, and *buy-in* to the defined process, as well as commitment to using it.

Mechanisms, methods, techniques, and tools to be utilized: Several examples of each category will be described throughout this book. Obviously, some are more appropriate in some cases than others, and some are particularly useful in specific situations. The specific mechanisms, methods, techniques, and tools should be determined and documented, and the project team should be familiarized with those selected and the rationale for their selection.

Integration of proven effective requirements practices: Experience has shown that use of a set of proven effective requirements practices can make a huge difference on a project [1]. For example, the practice of investing time and effort to define the real customer needs has already been recommended. Recommended “best” requirements practices will be described throughout this book and are summarized in Chapter 6. Select and document a set of requirements practices that will serve your project well.

References: There will be a set of documents that are key references for the requirements process. Examples include documents that describe system goals and objectives, lists of requirements of different users, standards that the customer has specified be applied, policies that are applicable, and so forth. These references should be listed, and the location where each can be accessed should be indicated.

Recommended strategy: Based on analysis of the above information, a strategy should be developed and set forth to optimally leverage requirements-related aspects of the project. Elements of the strategy might include the following:

The partnering strategy;³

3. The term “partnering” is often used to suggest a close, coordinated, effective working relationship. Here I refer to a defined process of partnership effort in a project. I encourage you to familiarize with the references at [6] and to consider use of the partnering process. You may find (as I have) that it holds one of the secrets to project success.

The “upfront process” to be used (to understand real customer needs and the environment, understand and document the scope of the project, define external interfaces, define system components, and define the outline for specification of the system);

Determining what drives the requirements (regulations; higher- level specifications; standards; policies; existing systems and processes; constraints, such as cost, schedule, technical viability; customer and user needs and expectations);

Definition of a project requirements policy;

Definition of the requirements process (flowchart and PD) (A sample requirements process is provided in [1] and on my Web site (www.ralphyoung.net). You may be able to utilize it to tailor a requirements process for your environment or project.);

Mechanisms to be utilized (e.g., the joint team and others that are recommended in this book);

Training concerning requirements for the project team (including the customer);

Selection of an appropriate automated requirements tool and how it will be used;

Definition of the target architecture;

Plans to deal with new and changed requirements (e.g., use of a mechanism to control them, as well as versions, releases, and builds);

Understanding of risks inherent to the requirements, as it’s likely that lack of full understanding of some requirements creates major project risks;

Definition of guidelines for system development based on requirements considerations.

Appendixes: These might include the following:

Requirements process (flowcharts and PDs);

Partnering process approach [6];

Draft project requirements policy;

Action plans and timelines for needed efforts (e.g., selection of a requirements tool).

Factors Affecting Your Career Decisions

I recommend that you meet with your PM very early, perhaps even before your assignment to the project is finalized. Discuss with him or her perspectives concerning requirements. After digesting this book and my previous

one, you should have a sufficient understanding of requirements practices to allow you to conclude whether you can be effective in your role.

Does the PM believe that requirements, requirements practices, investing in the requirement process, controlling requirements changes and new requirements, and minimizing rework are important?

Do you sense that he or she will support you in the many roles in which you can potentially contribute to the project (see Chapter 2)?

Does he or she seem concerned about people, about motivating people, acknowledging their efforts, empowering them, and supporting them?

Does he or she have a good reputation in the organization as a PM?

Is he or she concerned about personal and professional growth?

Is he or she willing to delegate responsibility?

The point is that you are about to commit a portion of your professional life to a project. Take the time and effort to satisfy yourself that your time will be well spent. You should perceive that a new position will provide you with learning experiences, opportunities to make valued and needed contributions, to work with peers whom you respect, to derive self-satisfaction and fulfillment, and to have fun at work.

A Comment Concerning Small Projects

Many people feel that the approach that is used on medium and large projects is an inappropriate guide for small projects—that the practices, policies, mechanisms, methods, techniques, and tools can't be applied. My experience is that professional judgment can be used to scale down and apply key practices to achieve good results on small projects. I encourage members of small projects, tasks, or teams to benefit from what they can learn from the experiences of larger projects by tailoring the approach, rather than use smallness as an excuse for not taking advantage of industry lessons. See [7] for additional insights.

Summary

This chapter has focused on the importance of requirements and provided an introduction to the critical role of the RA (the roles of the RA are further detailed in the next chapter, and the skills and characteristics of an effective RA are described in Chapter 3). It should be apparent from the material presented already that there is great power and effect in leveraging requirements-related activities in engineering and computing. An alarming 53%

of industry's investment in technical development projects is a casualty

of cost overruns and failed projects. Major contributing factors are a lack of user input (13%), incomplete requirements (12%), and changing requirements (12%).⁴ The user community and particularly project management do not realize the value of investing in the requirements process. I suggest that it is not "okay" for an RA to be aware of this and not to discuss the implications with his or her PM. As a concerned professional, you have the responsibility to bring these facts and your recommended approach to your PM and to ask him or her to support an effective requirements process that incorporates effective requirements practices. RAs, engineers, and managers are in a strategic position to improve industry's performance. This book provides focused and specific guidance that can have a huge payoff. By applying the approach recommended in this book, you can have a very positive impact on your project and organization.

Case Study

This first case study reports on a workshop involving facilitated discussions among a group of PMs concerning the top reasons they believed systems and software projects had difficulties, based on their experience. Here are the top reasons that were reported by a set of PMs:

1. The requirements for the project are not explicit.
2. Requirements changes are made/accepted without addressing the concomitant cost, schedule, and quality impacts.
3. A requirements process is not used.
4. There is no mechanism (such as a joint team) to reach agreement on the definition of the requirements and to manage the requirements through the project life cycle.
5. The "real" customer needs are not defined.
6. There is no mechanism to maintain communication between the parties involved in the project.
7. Known, familiar, proven methods, techniques, and tools are not utilized.
8. The customer is not involved as a partner throughout the project life cycle.

I recommend that you keep these reasons in mind as you digest this book. Ascertain what you might be able to do or to recommend that will help overcome these problems.