A Reengineering Process Framework

John Bergey
William Hefley
Walter Lamia
Dennis Smith

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
Tel: (412) 268-6850
E-Mail: jkb@sei.cmu.edu

Abstract

This paper describes a unifying structure for the full spectrum of activities that constitute a reengineering project. Part of the framework's "value added" is its usefulness as a reference model for planning a reengineering project and for identifying, describing, understanding, assessing, or evaluating the technical, managerial, and project-related factors that characterize reengineering efforts. The framework can serve as a tool for planning, implementing, and assessing reengineering projects. It also provides a context for establishing common terminology and nomenclature to describe reengineering activities, a basis for more detailed investigations of specific activities within the reengineering process, and as a mechanism for reporting and presenting reengineering "lessons learned."

The framework can be used to guide the motivation, planning, and decision-making aspects of reengineering along with structuring the software practices and supporting technologies for implementing the full spectrum of reengineering activities. It is an outgrowth of the Software Engineering Institute (SEI) involvement with and support of reengineering projects in DoD and industry and the SEI's collaborative effort with the reengineering community to develop a reengineering "best practice" guide [1].

Examples are provided to illustrate common reengineering experiences, pitfalls, and lessons learned from both a technical and management viewpoint. These examples reflect actual experiences and data derived from recently completed reengineering projects in DoD, Industry, and academe.

Since the framework corresponds to the current state of practice in reengineering, it more closely reflects what can best be described as "opportunistic" reengineering practices. A follow-on effort is being initiated to produce a "forward-looking" reengineering framework. It will reflect a model-based software engineering
approach to reengineering based on an evolutionary systems development paradigm.

2. Definition and Scope of Reengineering

Understanding the scope of reengineering is foundational to the development of a reengineering process framework. Some of the most frequently asked questions are: "What is reengineering?" "What is the difference between reengineering and development?" "Between reengineering and maintenance?"

There are many reengineering definitions. One definition that is frequently cited is: "Reengineering, also known as both renovation and reclamation, is the examination and alteration of an existing system to reconstitute it in a new form and the subsequent implementation of the new form [2]." A broader definition is: "Reengineering is the systematic transformation of an existing system into a new form to realize quality improvements in operation, system capability, functionality, performance, or maintainability and supportability at a lower cost, schedule, or risk to the customer." This definition emphasizes that the focus of reengineering is on improving existing systems with a greater return on investment (ROI) than could be obtained through a new development effort. If reengineering isn't less costly, can't be accomplished in a shorter time frame, isn't less risky, or doesn't offer better value to the
customer, then a new development effort should be considered.

Additional insight is obtained by comparing and contrasting reengineering with the traditional maintenance and development attributes shown in Figure 1. By placing reengineering on a continuum with maintenance and new development, as shown in Figure 2, the true nature of reengineering becomes apparent.

Maintenance entails making corrective, perfective, and adaptive changes to software, while development focuses on implementing new capabilities, adding functionality, or making substantial improvements—typically by using new computer resources and incorporating new software technologies. Reengineering, on the other hand, spans the gap between these two activities and exhibits characteristics of both as illustrated in Figure 2. Since these activities are not precisely bounded, there may be significant overlap. In the context of bridging the gap between maintenance and new development, the reengineering framework serves as a roadmap and provides insight into the various technical and management aspects of the reengineering process, much like the traditional life-cycle model for development.

3. A Lessons Learned Perspective

The following observations are excerpts from a longer list of "lessons learned" about the current state of the practice, which were developed from a series of site visits conducted by the SEI Reengineering Center [3]. These observations, which are highlighted below, influenced and shaped the development of the reengineering framework described in this paper:

- A systematic understanding of the legacy system architecture and its operation is key to successful reengineering. The major impediments are lack of documentation or documentation that has not been kept current with ongoing maintenance, loss of human system expertise, and inability to recover the total system architecture. In some cases, recovering the legacy system architecture and abstracting the design may not be feasible, or it may be more costly than new development. Furthermore, the system architecture—even if fully understood—is often very difficult to change, which may indicate that a new development strategy is a preferable approach.
- Estimating the cost and schedule for reengineering a system is a nontrivial task that requires substantial up-front resources. An in-depth assessment and analysis of the legacy system is a prerequisite for planning a reengineering effort and refining the specific goals and objectives.

- Objective assessment, measurement, analysis, and evaluation should be done on the legacy code and other artifacts. Several automated tools exist to perform static analysis of code to extract useful size and structural metrics from extant code. Reengineering vs. redevelopment decisions should include objective appraisal of these findings, to avoid unwarranted commitment to the legacy system.

- One of the most overlooked aspects of reengineering concerns deploying and transitioning the system into operational use. Upward compatibility is frequently treated on an ad hoc basis or not adequately addressed.

- Most reengineering efforts represent a "reengineering point solution" in that the reengineering of the system is not an integral part of an evolutionary system development process. At the completion of reengineering, the delivered system may not be any more amenable to being reengineered than the original legacy system – apart from the fact that initially there is a better system understanding.

This information suggests the importance of several factors: 1) a systematic understanding of the current set of assets as part of the decision-making process, 2) the difficulty of developing accurate cost and schedule estimates, 3) the important role of deployment and transitioning to operational use, and 4) planning in terms of the long-term evolution of the system.

As a result, our framework integrates the decision-making components with the technical reengineering task and takes account of the operational support and integration testing activities.

![Figure 3. A Systems Perspective of Reengineering](image-url)
4. A Systems Perspective of Reengineering

Many treat reengineering as being synonymous with software reengineering, but its true scope is broader. In reengineering a system, it is important to understand that the legacy, or operational system, is only one part of the total system environment. Although the software (of the legacy system) is often the focal point of reengineering efforts – and properly so – it alone is only one element of the "total system" that must be considered when reengineering. Any of the system elements has the potential to be a significant cost driver, technology problem, or management concern. Accordingly, this paper emphasizes a total systems perspective as an essential first step toward ensuring a successful reengineering effort. The elements forming the total system environment are shown in Figure 3. The five generic types of system elements that should be carefully considered when reengineering are identified below. They can be characterized as follows:

1. **Operational system** – the operationally deployed system that typically is the focal point of the reengineering effort and comprises the core functionality of the online system that is currently in operational use.

2. **Operational support systems** – the other ancillary online systems that interface or communicate with the deployed system in its operational environment, but are not physically part of the operational system being reengineered. Examples include another interconnecting system, a monitoring or tracking system, an online communications system, a remotely located preprocessing or post-processing unit, etc.

3. **Logistical support systems** – the other offline systems that support the readiness of the operational system (e.g., data preparation facility, mission loading device, data analysis or report generation facility, specialized test equipment, calibration or recording device, trainers, etc.).

4. **System integration and testbed facilities** – the supporting systems (new or modified) required to integrate and test the reengineered software (e.g., instrumented mockups of the operational system, stimulation and simulation facilities, etc.).

5. **Software development and maintenance facilities** – the supporting systems (new or modified) required to reengineer the software (e.g., the software engineering environments used to adapt the legacy system software and develop new software components).

As illustrated in Figure 3, it is the amalgamation of the systems described above that constitute the "total system" to be reengineered. However, some of them may be null; i.e., not all of them may be present in a particular system configuration or affected by the reengineering effort. And in some cases, the focus of the reengineering effort may not be the operational system itself, but rather one of the other system elements. An example would be the reengineering of an automatic test equipment unit (a logistical support element) for use on the tarmac of an airport for testing an avionics system (the operational system) on board an aircraft.

All of these system elements may require significant reengineering, and any one of them may play a major role in the decision-making process and in the reengineering tradeoffs that have to be made. If any of these elements are not fully considered or are overlooked, the effort will be at risk in the reengineering transition phase.

5. A Reengineering Process Framework

The reengineering process framework, shown in Figure 4, encompasses four major phases:

1) Reengineering assessment
2) Reengineering decision analysis
3) Reengineering development
4) Reengineering transition

These four phases constitute a process architecture for reengineering. A process architecture provides a framework for integrating descriptions of all the tasks that, when properly ordered, constitute a full engineering process for producing a product [4]. The four phases of the framework apply equally to each of the five generic types of system elements described in Section 4 as well as the aggregation of all the system elements affected by the reengineering effort.

Each of the four phases are described in terms of the generic activities that are necessary to fulfill their intended function. In turn, a customized set of tasks must be specified to carry out the activities that are identified. For a particular project, reengineering practitioners would specify the set of tasks required to meet their specific goals and objectives. Ideally, these tasks would be specified in a work breakdown structure (WBS), or equivalent form, as part of their planning documents. Two generic types of planning documents are included as an integral part of the reengineering framework. They are an initial reengineering assessment plan and a more comprehensive reengineering project plan.

5.1 Reengineering Assessment Phase

The first phase of the framework is the reengineering assessment phase, which is expanded in Figure 5.

The initial activity of the reengineering assessment phase is to establish the scope and direction of the reengineering effort. Another activity is centered on developing a preliminary plan to guide and direct the initial phases of the reengineering effort. A parallel activity is

![Figure 4. A Reengineering Process Framework](image-url)
establishing the initial system requirements to reflect the desired quality improvements. The remaining activities include performing an inventory and analysis of the legacy system, formulating candidate reengineering strategies, and performing a risk assessment of the candidate approaches. The objective of the inventory and analysis phase, which is a precursor to reverse engineering of the system, is to quickly obtain a level of understanding of the system's architecture, design, operational characteristics, and functionality. In performing a reengineering assessment, consideration should also be given to formulating maintenance alternatives and new development options to insure that reengineering is the most viable and effective option.

Successfully completing the reengineering assessment phase is essential to establishing the following:

- The true scope and extent of the reengineering effort.
- Explicit reengineering goals and objectives.
- Quantitative and qualitative data describing the legacy system.
- The feasibility and practicality of reengineering the legacy system.
- Candidate reengineering strategies and technical approaches.
- Alternative solutions to reengineering.

5.2 Reengineering Decision Analysis Phase

Following the reengineering assessment phase is the reengineering decision analysis phase which is shown in Figure 6. A robust decision-making process is needed to account for all the technical, economic, programmatic, and political issues of the organization that can influence and affect reengineering. An integral part of the decision-
making process is performing a cost/benefit/risk analysis and trading off the candidate reengineering strategies against new development or maintenance alternatives.

Figure 6. Reengineering Decision Analysis Phase

The outcome of the analysis is a decision to reengineer, initiate a new development effort, or continue in a maintenance mode. If the decision to reengineer the system is affirmed, the final step is to prepare a comprehensive reengineering project plan for subsequent approval and implementation. This plan covers the tasks to be performed, the schedule for completing the tasks, the risks involved, the resources required, milestones and funding, and an itemization of the deliverable products that make up the reengineered system. While cost is often thought of as the predominant determinant in choosing one approach over another, quite frequently reengineering is selected because it affords the least risk, shortest implementation schedule, minimal transition problems, greatest upward compatibility, maximum usage of existing resources, or increased customer acceptance.

Successfully completing the decision analysis phase is essential to establishing the following:

- Comprehensive analysis data for the decision-making process.
- An approved and validated reengineering strategy and basic technical approach.
- The projected impact of reengineering the system.
- The cost, resources, schedule, and risk envelope for reengineering the system.
- The expected benefits of reengineering the system.

In essence, the combined effect of the reengineering assessment and decision analysis phases is to provide a systematic means for 1) obtaining a thorough understanding of all the technical, programmatic, and political ramifications of reengineering the system, 2) making an informed and judicious decision as to whether or not to proceed with the reengineering
of the system, and 3) developing a comprehensive project plan to guide and direct the reengineering effort (should it be elected).

5.3 Reengineering Development Phase

The third phase of the framework is the reengineering development phase, which constitutes the core element of the reengineering effort. This phase is expanded in Figure 7. The initial guiding activity of the reengineering assessment phase is to formally elicit and validate the detailed system requirements for the reengineered system. A reverse engineering activity, which is unique to reengineering, is focused on analyzing the existing system to obtain an in-depth understanding of the software system – usually with the help of automated software tools. Two associated activities include the actually recovery of reusable assets from the existing system and their adaptation and use in the reengineered system. There are also two generic activities to facilitate the reuse of externally developed (or procured) software components, referred to as NDI (non-developmental items). NDI may be COTS (commercial off-the-shelf) or GOTS (government off-the-shelf) software. The remaining activities are referred to as the forward engineering portion of the phase, since they reflect a more traditional development approach that encompasses a system design and development activity and culminates in a system integration and test activity.

Successfully completing the reengineering development phase is essential to meeting the following goals:

- A reengineered system that complies with the specified requirements.
- A tested and demonstrable system.
- Delivery of a reengineered system within the cost, schedule, and risk envelope.

5.4 Reengineering Transition Phase

As shown in Figure 8, the last phase of the reengineering framework is the reengineering transition phase, which focuses on deploying the reengineered system into its operational environment. Often overlooked, the transition phase is a key element in a successful reengineering project. The activities go beyond...
the normal site preparation and system installation and checkout. Ensuring that the user is properly trained and the organizational infrastructure is in a proper state of readiness are high-priority items. Another important aspect of the reengineering transition phase is allowing for a trial deployment to ensure that all the "glitches" in the system have been worked out. This not only alleviates and circumvents many of the problems normally encountered in deploying a system, but greatly facilitates achieving customer and user acceptance. Many lose sight of the fact that the job of reengineering is not complete until the reengineered system has been fully transitioned and is capable of replacing or superseding the legacy system in its operational environment. It should be noted that the sequence of the components in the framework is not intended to imply that all the transition activities can wait until the reengineering development is complete. On the contrary, the reengineering plan must not only specify the actual tasks to carry out the transition activities, but must also account for their proper scheduling. Consequently, even though the bulk of the transition activities are carried out in the latter phases of the reengineering project, many preparatory tasks (e.g., implementing provisions to ensure the readiness of the organizational infrastructure) may need to be accomplished during the time frame when reengineering development is taking place.

Successfully completing the reengineering development and transition phases is essential to meeting the following goals:

- A system that is compatible with the totality of its operational environment.
- An operationally validated system that merits customer acceptance.

The bottom line for a successful reengineering project is effectively using the quality improvements (afforded by the reengineered system) in the production or operational environment to the benefit and advantage of the customer and user. Such benefits may include reducing costs, increasing productivity, or gaining a competitive advantage by improving the quality of the products and services that are produced by the reengineered system.
6. Conclusions

The framework can serve as a reference model for guiding reengineering efforts and as a useful tool for planning, implementing, and assessing reengineering projects. Part of its value added is in identifying, describing, understanding, assessing, and evaluating such factors as

- The scope of a reengineering project.
- Techniques for gaining system understanding.
- Reengineering decision-making techniques.
- Methods and techniques for the reuse of legacy system assets.
- Reengineering processes and process models.
- The role of tools and technologies for supporting reengineering.
- Transition and deployment issues and concerns.

Because of the complexity of the total set of reengineering issues, the framework can serve as a guide for more detailed investigations of activities within the reengineering processes. Specific models can be developed to focus on selected aspects of the total framework, such as reverse engineering, risk assessment, decision-making, design recovery, adaptation of reuse components, or organizational readiness for operational deployment.

A number of organizations have developed reengineering models and processes that relate to components of this framework. The Defense Information Systems Agency (DISA) has developed a software systems reengineering process model [5] that addresses reengineering development. The Software Technology Support Center (STSC) and Air Force Cost Analysis Agency have sponsored the development of a Software Reengineering Assessment Handbook [6]. The handbook provides a technical and economic assessment methodology for reengineering decision analysis.

The framework also provides a context for establishing common terminology and nomenclature, for reporting and presenting reengineering "lessons learned," and for understanding the relationship of other reengineering contributions. It is anticipated that the framework will evolve over time and be used in reengineering developments, such as a measurement-based template for reengineering assessment.
References


