

Requirements Engineering Tasks

Donald Firesmith, Software Engineering Institute, U.S.A.

Many managers and others who are not professional requirements engineers tend to greatly over-simplify requirements engineering (RE). Based on their observations that requirements specifications primarily contain narrative English textual statements of individual requirements and that all members of the engineering team are reasonably literate, there is a common myth that practically anyone with little or no specialized training or expertise can be a requirements engineer. After all, what is there to do but ask a few stakeholders what they want (requirements elicitation), study the resulting requirements to make sure they are understood (requirements analysis), write the requirements down in a document (requirements specification), and then ask the customer if they're right (requirements validation). Just give the team a short class in use case modeling, and they are ready to go.

Unfortunately, the preceding is a misleading, if much too prevalent, myth. While these four RE tasks (*not sequential phases!*) are commonly performed with varying degrees of completeness, rigor, and success on most projects, a list of tasks containing only these four is far from complete. The purpose of this paper is to provide a brief introduction to all of the major tasks comprising RE, as well as to three essential and highly related tasks from the management, configuration management, and quality engineering disciplines. Depending on the top-most goals of the system development project or product line development projects, the RE teams need to ensure that the actual RE method to be used contains all of the essential and cost-effective RE tasks, tailored to meet the specific needs of the endeavor.

Although not the primary topic of this paper, a brief word must be said about the makeup of the RE teams that will be performing the requirements engineering tasks. Because of the criticality of the requirements and the breadth of their scope and impact, each RE team clearly must be cross-functional to be effective. In addition to requirements engineers, the RE teams need to either include or collaborate closely with domain experts and a representative sample of key stakeholders such as customer representatives, marketing, business analysts, user representatives, system architects to ensure requirements feasibility, system testers to ensure requirements verifiability, etc.

Otherwise, important requirements will be missed (not specified), will be specified incorrectly, or will be ambiguous.

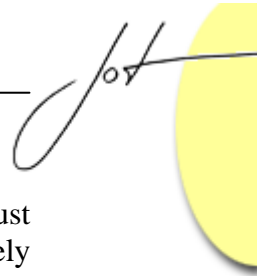
Having well-engineered requirements is critical. This is not just due to their major positive impact on project costs (both development and life-cycle) and schedule, which are largely due to the extreme costs of fixing requirements defects once the system is built and fielded. Also critical is having well-engineered functional and quality requirements because of their positive impact on system acceptability by its many stakeholders. These are some of the reasons why businesses gain such a high return on investment from good requirements engineering practices. Therefore, it is typically important to perform some or all of each of the following RE tasks, although the amount of effort and formality will naturally vary due to many factors such as system criticality and organizational maturity. The following tasks do not represent process for process sake, but rather good engineering discipline for the sake of the business and the system's many stakeholders.

The following list of reusable RE tasks can be used to develop an endeavor-specific RE method for a single project, a program of related projects (e.g., product line development), or an entire business enterprise. It can also be used as a checklist to ensure completeness during the evaluation of an existing RE method. The following specific set of tasks comes from the OPEN Process Framework (OPF) Repository Organization (www.opfro.org), the world's largest repository of free, open-source, reusable method components. Other methods and frameworks may divide RE into a different set of tasks or include tasks that more logically belong to other disciplines such as scope management from project management and requirements verification from quality engineering (specifically quality control).

BUSINESS ANALYSIS

During this task, the business strategy team, technology strategy team, and cross-functional requirements engineering team(s) collaborate to analyze the business context in which the system shall be developed and exist. In some organizations, this task is performed prior to the involvement of the RE teams by the other teams; in other cases, it is incorporated into the requirements identification task and performed solely by the RE teams. This task typically may include the following subtasks:

- *Analyze the customer organization's business enterprise* to understand the:
 - Business model.
 - Organizational structure and relationships.
 - Technology currently being used.
 - Relevant planned improvements.
- *Analyze the competitor organizations* that produce competing systems to:
 - Identify, profile, analyze, and understand the competitors.
 - See how the planned system [upgrade] will improve the customer organization's business enterprise and help it compete.



-
- *Analyze current and potential/planned marketplaces* in which the system must compete to determine system properties needed to enable it to effectively compete.
 - *Analyze critical technologies* to determine their readiness for use in the system in terms of their level of maturity and their compatibility with other requirements and the proposed system architecture.
 - *Analyze current and intended future user communities* to understand their needs and desires and determine how the system might improve their tasks and workflows.
 - Analyze the stakeholders to:
 - Identify different stakeholder persons, roles, organizations, and systems.
 - Profile them including categorizing them into well-defined and well-understood groups.
 - Understand their needs, desires, responsibilities, and tasks.
 - *Develop a business case* to determine whether the system [enhancement] should be developed by:
 - Determining its costs and benefits
 - Comparing its merits relative to those of competing systems.

VISIONING

During this task, the main RE team collaborates with key stakeholders to produce a vision of the new system (or next version of the existing system) to be developed. Although logically distinct, some organizations incorporate this task into the following requirements identification task. This task typically includes the following subtasks:

- Define the system's mission.
- Determine the business problems and opportunities to be solved by the system.
- Determine the most important stakeholder needs to be fulfilled by the system.
- Determine the most important business, functional, and quality goals of the system.
- Determine any major business, technical, and legal/regulatory constraints on the system.
- Use this information to build a consensus among key system stakeholders regarding the vision of the system to lay a foundation on which the system requirements can be engineered.

REQUIREMENTS IDENTIFICATION

During this task, the RE teams identify potential requirements. This task typically includes the following subtasks:

- *Identify sources* of requirements (e.g., stakeholders, documents, legacy systems, problem reports, etc).

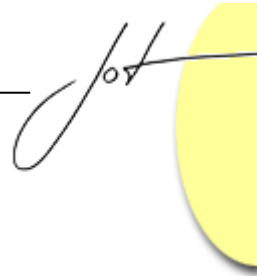
- *Elicit* needs, goals, desires, and requirements from a representative sample of all major stakeholder types (e.g., customers, users, maintainers, operators, subject matter experts, marketers, and certifiers). Note the use of the more general term “requirements identification” for this overall task and the relegation of “requirements elicitation” to a single subtask. Elicitation is only one of the useful techniques for identifying requirements and should not be relied on totally to identify requirements.
- *Gather* potential requirements from existing documents describing legacy or competing systems, problem reports, marketing surveys, and other sources.
- *Invent* new requirements so that the system will be truly better than the legacy systems it will replace and therefore worth building. Invention is a critically important, though underutilized, technique for identifying requirements, and is often the difference between a highly successful system and a marginally successful system.
- *Transform* stakeholder desires, expectations, and needs into informal, textual, potential requirements.

This task may produce two inconsistent sets of requirements if both the customer and developer organizations perform this task on the same endeavor. In this case, a consensus as to the correct set of requirements must be achieved, typically by a combination of this task, the requirements analysis task, and the requirements validation task.

REQUIREMENTS REUSE

During this task, the RE teams reuse all or part of preexisting requirements work products. This task typically includes the following subtasks:

- *Identify* any potentially relevant reusable requirements work products (e.g., individual requirements, requirements templates, requirements diagrams, requirements models, and requirements specifications). This includes both complete work products (e.g., an individual requirement) as well as parts of work products (e.g., one use case or use case path out of an entire use case model).
- *Evaluate* the identified reusable requirements work products for relevancy to the current endeavor.
- *Tailor* the relevant identified reusable requirements work products to meet the needs of the current endeavor.
- *Reuse* the tailored reusable work products by incorporating them into the current endeavor’s requirements work products.



REQUIREMENTS ANALYSIS

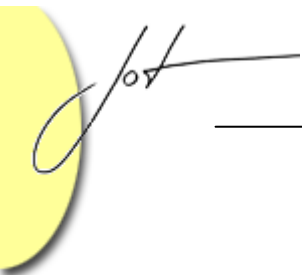
During this task, the RE teams analyze the identified and reused requirements. Note that new requirements are often identified during the analysis of previously identified requirements. This task typically includes the following subtasks:

- *Study, categorize, decompose and organize, model, quantify, refine, prioritize, justify, and trace* each requirement to its source(s). It is important to note that different types of requirements require different modeling techniques. For example, whereas use case modeling is very good for analyzing functional requirements, it is not very good for analyzing other types of requirements. Data modeling (e.g., logical data models, object models, and information engineering models) are useful for data and interface requirements. Different kinds of quality requirements also need different kinds of analysis techniques such as:
 - Performance modeling for analyzing performance requirements.
 - Asset, accident, hazard, and risk analysis for safety requirements.
 - Asset, attack, attacker, threat, and risk analysis for security requirements.
- *Transform* informal textual requirements into semiformal or formal requirements (if formal methods are used).
- *Negotiate* the prioritization of requirements with the requirements stakeholders, and use the negotiated prioritization to help schedule the implementation of the requirements. Note that this subtask is often performed concurrently with the requirements validation task.
- *Verify* any related assumptions.
- *Transform* potential raw requirements and related information into real requirements that have the necessary quality characteristics such as clarity (i.e., lack of ambiguity), completeness, consistency, correctness, feasibility (e.g., technical, financial, schedule, etc.), verifiability, and understandability.
- *Ensure* that the requirements are sufficiently well understood that they can be properly specified.

REQUIREMENTS PROTOTYPING

During this task, the RE teams generate requirements engineering prototypes. This task typically includes the following subtasks:

- *Produce* one or more requirements prototypes (e.g., paper or wireframe prototypes of user interfaces or executable models).
- *Evaluate* these prototypes. This may involve analysis of static prototypes or execution and evaluation of dynamic prototypes.
- *Use* these prototypes to:
 - Help identify new requirements such as functional, data, and quality requirements regarding user interfaces.

- 
- Better understand existing requirements.
 - Identify defects in the existing requirements that drove the development of these prototypes.
 - Support the analysis of these requirements.

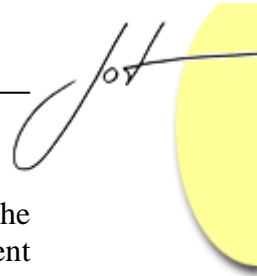
REQUIREMENTS SPECIFICATION

During this task, the RE teams generate and publish analyzed and/or validated requirements in paper or electronic requirements specification documents. This task typically includes the following subtasks:

- *Generate* requirements documents specifying appropriate sets information for different audiences at different times during development:
 - System, subsystem, software, and hardware requirements specifications containing the individual requirements and associated ancillary information. Except for interface requirements in interface requirements specifications, I do *not* generally recommend banishing any class of requirements to separate documents. For example, it is usually a mistake to specifying quality requirements in supplementary specification documents and specifying safety and security requirements in separate safety and security policy documents. This typically causes such requirements to be inadequately specified, specified too late during development, and largely ignored during the development of the architecture, even though quality requirements often should have the biggest impact on the architecture and are the most expensive to retroactively implement if overlooked.
 - Operational concept documents (OCDs) containing use cases, misuse or abuse cases, and usage scenarios.
 - Glossary and Domain Object Model to properly define the meaning of the terms used in the requirements.
- *Distribute* the requirements specifications to their audiences or make access available to them.
- *Iterate* the requirements specifications as a result of informal feedback. Note that more formal feedback will come as part of the requirements verification subtask of quality engineering.

REQUIREMENTS MANAGEMENT

During this task, the RE teams manage all requirements, regardless of their status. Note that as with any other work product, the configuration management of requirements and other requirements work products is actually a part of the configuration management discipline. This task typically includes the following subtasks:



-
- *Record and store* the requirements and their attributes (i.e., metadata about the requirements) in an appropriate repository, database, or requirements management tool.
 - *Control access* (e.g., create, read, update, delete) to the requirements (e.g., based on metadata such as authorization to create/read/update/delete requirements by role, requirement state, requirement ownership, requirement responsibility, date of last change to the requirement, etc.).
 - *Negotiate* with the stakeholders to eliminate any inconsistencies between requirements and their priorities.
 - *Report the status* of the requirements (e.g., the number, percentage, and state of the requirements and requirements categories).
 - *Trace* the requirements (e.g., to the associated architecture, design, implementation, and test work products).

REQUIREMENTS VALIDATION

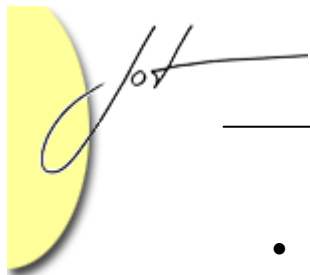
During this task, the RE teams validate the correctness of the analyzed requirements with their stakeholders and make any necessary corrections. This is an ongoing task that typically includes the following subtasks:

- *Identify* a representative sample of all major stakeholder types (e.g., customers, users, maintainers, operators, subject matter experts, marketers, and certifiers) to validate the requirements.
- *Ensure* these stakeholders validate the correctness of the requirements.
- *Iterate* to fix any requirements problems.
- *Certify* that the requirements are an acceptable description of the system, software application, or component to be implemented.

RELATED TASKS FROM OTHER DISCIPLINES

According to the OPEN Process Framework (OPF) Repository Organization (www.opfro.org), the preceding tasks clearly fall completely within RE. However, there are three other tasks that technically and logically belong to other disciplines, but which nevertheless are *absolutely critical* to the success of the requirements engineering effort. In some organizations and according to some development methods, this is why these tasks are included within RE. In either case, they must be properly addressed when developing a RE method. These tasks include:

- *Scope Management* is the *management* task that manages requirements changes that could significantly change the scope of the endeavor.
- *Requirements Verification* is the *quality engineering* task that controls the quality of the requirements and other requirements work products such as requirements models and requirements specifications.



- *Requirements Configuration Control* is the *configuration management* task that manages and evaluates the impact of proposed changes to baselined requirements and other requirements work products.

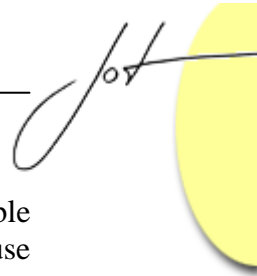
RELATIONSHIP BETWEEN THESE TASKS

On most projects, RE should not be thought of as the first phase of the waterfall development cycle. RE tasks and subtasks should typically be performed in an iterative, incremental, concurrent, and time-boxed manner:

- *Iterative* in the sense that the same tasks will typically need to be repeated on the same work products in order to fix defects and make other improvements. In practice, requirements are often of very poor quality, and iteration of the requirements and other requirements work products is absolutely essential.
- *Incremental* in the sense that most systems are too large and complex to engineer all requirements in a big-bang waterfall manner before beginning the tasks of other disciplines. For example, architecting cannot typically wait until requirements engineering is complete. Rather, requirements engineering is typically performed in a top-down manner, layer by layer in the system's hierarchical architecture.
- *Concurrent* in the sense that:
 - Requirements engineering tasks are performed simultaneously with the tasks of many other disciplines. The project should not and cannot stop until a complete set of perfect requirements are developed.
 - The requirements engineering teams rapidly cycle between tasks while different members of the requirements team concurrently perform different tasks on different sets of requirements.
 - When developing large and complex systems, different requirements engineering teams are concurrently performing different requirements engineering tasks on different components of the system architecture at different levels of the system architecture.
- *Time-boxed* in the sense that the completion of requirements tasks on increments and iterations of the requirements are scheduled to avoid analysis paralysis.

An important consequence of an iterative, incremental, and concurrent development cycle is that the RE teams must exist from initial conception through development and delivery. In systems being actively maintained with new versions and variants constantly being developed, the RE teams must also continue in operation, though the number and size of the teams may vary from phase to phase.

In addition to relationships between these tasks due to the development cycle, there are other relationships that must be considered. As described above, these tasks have been decomposed along logical lines to maximize their understandability. However, the way real projects work is never so logical in practice. A certain unavoidable amount of chaos is involved in the way teams actually work on real projects as people rapidly move from



task to task multiple times each day, or even each hour. Except in theory, it is impossible to successfully assign these tasks as lines in a project work breakdown structure because there is so much overlap between them. It is better, therefore, to manage by milestones (and inch pebbles) based on earned value associated with requirements work products than managing by requirements task completion, something that is rarely totally finished until the system is retired.

CONCLUSION

RE is logically comprised of many important tasks, not just the three or four that are most often cited. Depending on the specific needs of individual projects, the RE teams should ensure that their RE method contains all of the appropriate tasks and each of these tasks should be tailored appropriately. Once selected and tailored, the RE tasks should be performed in a manner that is consistent with the project's chosen development cycle, and this typically means iteratively, incrementally, concurrently, and constrained by appropriate time-boxes. Only by understanding all of the RE tasks can the development team ensure that the necessary RE tasks are appropriately staffed, scheduled, and performed. Finally, RE is a complex and often messy process in practice. Hopefully, the preceding summary of RE tasks will help you better perform requirements engineering and ensure that no important work slips through the cracks.

About the author



Donald Firesmith is a senior member of the technical staff at the Software Engineering Institute (SEI), where he helps the US Government acquire large, complex, software-intensive systems. Working in industrial software development since 1979, he has worked primarily with object technology since 1984 and has written 5 books on the subject. During the last four years, he has developed the world's largest (1,100+ webpage), free, and open source informational website of reusable process engineering components. Based on the OPEN Process Framework (OPF), it is located at <http://www.opfro.org>. Currently writing a book on the engineering of safety and security-related requirements, he can be reached at dgf@sei.cmu.edu.