Requirements Engineering

Studies such as the CHAOS reports indicate that about half of the factors associated with project or product success are requirements related.

Recently, researchers have reported on studies showing that project success is directly tied to requirements quality. With such overwhelming evidence that requirements engineering is a cornerstone of software systems engineering, one could ask, why is it still a relatively neglected topic in university training?

It is quite rare, for example, that a new Computer Science (CS) university graduate might be asked to participate in the development of a compiler or operating system, yet nearly every graduate working in the industry will, sooner or later, be asked to participate in creating the requirements specifications for a product or service.

1.1 Why Has Requirements Engineering Become So Important?

For years, many products were successfully created without the participation of professionals who specialized in requirements creation or management. So, why is requirements engineering (RE)so important today? The answer lies in the changing nature of industry and society in general.

1- **The pace of product development has picked up drastically.**

   Whereas just a few decades ago, product improvements would be a slow process, today customers often demand new versions of a product in less than one year. For example, Siemens estimates that approximately 20 years ago, 55 percent of sales were from products that were less than 5 years old. Today, 75 percent of sales are from products that were developed less than 5 years ago (Figure 1.1).

2- **Turnover and technology change have impacted the experience levels of professionals engaged in the development of products.**

   Just a few short years ago, engineers might expect to spend their entire careers with a single company, whereas today job change is more common.

3- Outsourcing and offshoring have dramatically changed the product life cycle.

Specifications must now be created for implementation or manufacturing by organizations with potentially limited or no domain expertise. Imagine, for example, having to create a product specification for a washing machine, dishwasher, or luxury automobile to be built by staff who may have never even seen one! Under such circumstances specifications must be exact and detailed.

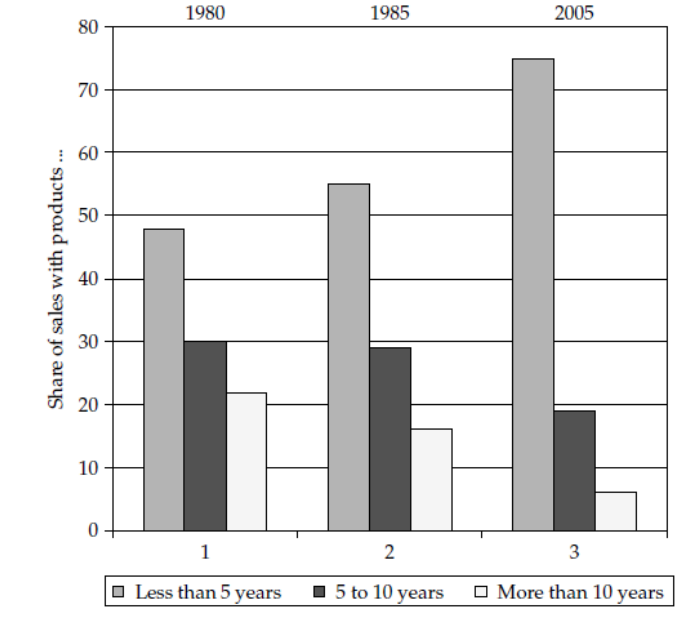


FIGURE 1.1 Acceleration of new product creation

Product innovations can be more easily implemented in software than hardware because of the lower engineering investment and modification costs. This results in domain-specific, complex software for which high-quality requirements specifications are essential.

Requirements engineering is extremely important when a product, service, or industry is regulated. For example, the U.S. government's Food and Drug Administration (FDA) and Federal Aviation Administration (FAA) both mandate specific activities and work products (e.g., hazard analysis) where there is the potential for injury or death. Sarbanes-Oxley regulations mandate traceability for certain types of financial software used by companies doing business in the United States. Good requirements engineering practices are essential for companies that must comply with government regulations.

1.2 Misconceptions about Requirements Engineering

Misconceptions about requirements engineering can strongly influence a company's processes. Many companies and organizations have a solid understanding of requirements processes, but some do not. Some of the more common misconceptions are listed under the headings that follow.

Misconception 1: Any Subject Matter Expert Can Become a Requirements Engineer after a Week or Two of Training

Requirements engineers need strong communication and knowledge of engineering skills, the ability to organize and manage a data set of requirements, high-quality written and visual presentation skills, and the ability to extract and model business processes using both text and graphical (e.g., Integration DEFinition [IDEF], Unified Modeling Language [UML]) techniques. First and foremost, to elicit requirements from stakeholders requires the ability to interact with a variety of roles and skill levels, from subject matter experts (detailed product requirements) to corporate officers (elicitation of business goals).
Moreover, people have to be trained to write good specifications. High school and university training tends to teach a style of writing that is antithetical to the techniques needed to create unambiguous and complete documents. Requirements analysts typically need significant training, both classroom and on the job, before they can create high-quality specifications.

Misconception 2: Nonfunctional and Functional Requirements Can Be Elicited Using Separate Teams and Processes

The subject domains for nonfunctional and functional requirements are related, may impact each other, and may result in iterative changes as work progresses. Team isolation may do more harm than good.

Misconception 3: Processes That Work for a Small Number of Requirements Will Scale

Requirements engineering processes do not scale well unless crafted carefully. For example, a trace matrix is an N × N matrix, where N is the number of requirements of interest. In each cell, a mark or arrow indicates that there is a trace from requirement Ri (row i) to requirement Rj (column j). It is relatively easy to inspect, say, a 50-

requirement matrix, but what happens when five to ten thousand requirements are needed to define a product? Filtering and prioritization become important in order to retrieve results that can be better understood, but the requirement annotations necessary to provide such filtering are often neglected up front because the database is initially small.

1.3 Industrial Challenges in Requirements Engineering

Many RE challenges have been identified as potentially impacting project performance. We have observed that problems tend to be exacerbated by three critical factors:

1- Being a decision to outsource the implementation.
2- Being a significant change in technology.
3- Being the introduction of new products (e.g., entering a market where the company has minimal prior experience).

When a decision is made to outsource, changes must take place in all processes, especially in the area of requirements engineering. The implementation may be done by staff with minimal domain knowledge and, because of customs, logistics, time, or distance, with limited access to subject matter experts. Attempts to use the same processes and techniques used for in-house development for the development of specifications for subcontracting or outsourcing may lead to significant delays in delivery, sometimes even resulting in project cancellation.

When technology changes rapidly, domain experts may no longer be “experts.” Techniques and solutions that worked for many years may become obsolete or irrelevant. Such technological discontinuities may require substantial new training, or the experts in the older technologies may make poor decisions for new product designs.

1.4 Key Success Factors in Requirements Engineering

Most of the factors can be evaluated prior to project initiation, the key success factors are:

1- The Project Has a Full-Time, Qualified Chief Architect

On many large projects the only senior technical role that spans the requirements process through delivery is that of the chief architect. He provides technical continuity and vision, and is responsible for the management of the nonfunctional requirements (e.g., scalability, quality, performance, environmental, etc.) and for the implementation of the functional requirements. In our experience having an experienced, full-time architect on a project contributes significantly to its success.

2- A Qualified Full-Time Architect Manages Nonfunctional Requirements

The architect is responsible for managing nonfunctional requirements and the relationships among requirements analysis, development, and management.

3- An Effective Requirements Management Process Is in Place

The critical success factors in a requirements management process are well defined by the Capability Maturity Model Integration (**CMMI**), specifically those addressing change management and traceability. A change control board (CCB) performs an impact analysis and conducts cost/benefit studies when feature changes are requested. The **CCB** acts as a gatekeeper to prevent unwanted "scope creep" and ensures properly defined product releases.

4- Requirements Elicitation Starts with Marketing and Sales

The marketing and sales organizations and the project's requirements engineering staff must establish strong bonds to enable accurate definition of product and/or product line features. Incorrect features and requirements may be carried over into the requirements development activities and create downstream problems.

5- Requirements Reviews Are Conducted for All New or Changed Requirements or Features

Requirements must be reviewed, and the review must occur at the right level. Since it typically takes one hour to review four to ten requirements (e.g., for the first review—follow up reviews may go faster), reviews must be conducted at a high enough level to avoid "**analysis paralysis**" and yet low enough to catch significant feature level defects.

6- Requirements Engineers Are Trained and Experienced

Requirements engineering is like any other scientific or engineering endeavor in that the basic skills can be learned through training. But without experienced staff, the project may "stall" or "churn" in the requirements definition stage. If the staff is new, and the team has more than four members, RE mentors should be used to improve the skills of the team.

7- Requirements Processes Are Proven and Scalable

When processes are defined at the start of a project, they should be bootstrapped from prior successful efforts, not just based on "textbook" examples. As the size of a project increases, or the number or size of work products increases, the methodologies must be scaled to match.

8- Subject Matter Experts Are Available as Needed

Arrangements must be made early on to access the experts needed to assist in defining requirements. For example, during tax season, tax accountants and attorneys may be unavailable. Schedules cannot be defined unless the experts are available during requirements development. Accountants and attorneys may be unavailable. Schedules cannot be defined unless the experts are available during requirements development.

9- All Stakeholders Are Identified

All the relevant stakeholders must be identified if requirements are to be properly defined and prioritized. The later key requirements are identified during the project, the greater the risk that major changes to the in-progress implementation will be necessary. Furthermore, the success of a product may be jeopardized by failure to validate key requirements.

10- The Customer Is Properly Managed

Customer management includes rapid feedback during prototyping, minimizing the number of points of contact between project staff and stakeholders, and maintaining strict control of feature change requests. It also includes using good techniques to elicit product features that are correct and unambiguous.

11- Progress and Quality Indicators Are Defined

The CMMI has a measurement and analysis practice area that overlaps with both requirements development and requirements management. Sometimes, a methodology (such as the Rational Unified Process [RUP] techniques for capturing text use cases) doesn’t include progress or work product quality measures. These indicators must be defined in advance, or project management will find it difficult to gauge project progress and make appropriate corrections.

12- The RE Tools Increase Productivity and Quality

Any software tools used must enable a process (increasing productivity and CMMI compliance), rather than hinder it. Positive outcomes may require tool integration, customization, or, in rare cases where there is a justifiable cost benefit, creating a new tool from scratch.

13- The Core Project Team Is Full Time and Reports into a Single Chain of Command

Studies have shown that a full-time core team is essential to the success of a large project. Without the continuity provided by a committed full-time core of people, issues may “fall through the cracks” or not show up until problems are revealed at integration testing time.

1.5 Definition of Requirements Engineering

Requirements engineering: involves all lifecycle activities devoted to identification of user requirements, analysis of the requirements to derive additional requirements, documentation of the requirements as a specification, and validation of the documented requirements against user needs, as well as processes that support these activities.

” Note that requirements engineering is a domain neutral discipline; e.g., it can be used for software, hardware, and electromechanical systems. As an engineering discipline, it incorporates the use of quantitative methods.

Whereas requirements analysis deals with the elicitation and examination of requirements, requirements engineering deals with all phases of a project or product life cycle from innovation to obsolescence. Because of the rapid product life cycle (i.e., innovation→ development→ release→ maintenance→ obsolescence) that software has enabled, requirements engineering has further specializations for software.

Thayer and Dorfman, define software requirements engineering as “the science and discipline concerned with establishing and documenting software requirements.”

1.6 Requirements Engineering’s Relationship to Traditional Business Processes

It is extremely important to tie requirements activities and artifacts to business goals. For example, two competing goals are “high quality” and “low cost.” While these goals are not mutually exclusive, higher quality often means higher cost. Customers would generally accept the higher cost associated with a car, known for luxury and high quality, but would likely balk at paying luxury car prices for a car expected to compete in the low-cost automotive market.

Unfortunately, some organizations may tend to decouple business and requirements activities. For example, business goals may drive marketing activities that result in the definition of a new product and its features. However, the business goals may have no clearly defined relationship to the artifacts used and produced during requirements analysis and definition.

RE activities start at the very beginning of product definition with business goals and innovation. Requirements engineering techniques can add an element of formality to product definition that can improve communication and reduce the downstream implementation effort.

1.7 Characteristics of a Good Requirement

Requirements characteristics are sometimes overlooked when defining requirements processes. They can be an excellent source of metrics for gauging project progress and quality The characteristics of a good requirement, as defined by the IEEE, are listed next. It is important to distinguish between the characteristics of a requirement and the characteristics of a requirements specification (a set of related requirements). In some cases a characteristic can apply to:

1- A single requirement.

2- Requirements specification.

3- The relationship of two or more requirements.

Furthermore, the meaning may be slightly different when referring to a requirement or a specification. Care must be taken, therefore, when discussing the characteristics described here to define the context of the attributes.

1- Feasible

A requirement is feasible if an implementation of it on the planned platform is possible within the constraints of the program or project.

For example, the requirement to handle 10,000 transactions per second might be feasible given current technologies, but it might not be feasible with the selected platform or database manager. So a requirement is feasible if and only if it can be accomplished given the resources, budget, skills, schedule, and technology available to the project team.

2-Valid

A requirement is valid if and only if the requirement is one that the system shall (must) meet.

Determination of validity is normally accomplished by review with the stakeholders who will be directly responsible for the success or failure of the product in the marketplace. There can be a fine line between "must" and "nice to have." Because the staff of a development team may be mainly focused on technology, it is important to differentiate between stakeholder requests that are wishful thinking and those that are actually needed to make the project or product a success. The inclusion of requirements that are nice but not valid is called "**gold plating**." As the name implies, having requirements on a project that are not valid will almost certainly:

1- Add cost without adding value.
2- Delaying project completion.

The Use of the Terms "Valid" and "Correct"

The IEEE Standard 830 uses the term "correct." We use the term "valid" instead because "correct" can be misleading. Something that is "correct" is said to be "without error," or mathematically provable. However, in the context of a requirement, "valid" is more appropriate, as the requirement may be exactly what the customer wants, but it may still contain errors or be an inappropriate solution.

3- Unambiguous

A requirement is unambiguous if it has only one interpretation.

Natural language tends toward ambiguity. When learning writing skills in school, ambiguity can be considered a plus. However, ambiguity is not appropriate for writing the requirements for a product, and care must be taken to ensure that there is no ambiguity in a requirements specification. For example, consider this statement:

"The data complex shall withstand a catastrophe (fire, flood)."

This statement is ambiguous because it could mean "The data complex shall withstand a catastrophe of type fire or flood," or it could mean "The data complex shall withstand any catastrophe, two examples being fire and flood." A person skilled in writing requirement specifications would rephrase as:

"The data complex shall be capable of withstanding a severe fire. It shall also be capable of withstanding a flood."

An example of an ambiguous statement is :

"**The watch shall be water resistant**." An unambiguous restatement is

"**The watch shall be waterproof to an underwater depth of 12 meters**."

A measure of the quality of a requirements specification is the percent of requirements that are unambiguous.

A high level of ambiguity could mean that the authors of the specification likely need additional training. Ambiguity often causes a project to be late, over budget, or both, because ambiguity allows freedom of interpretation. It is sometimes necessary to take a holistic view of ambiguity; e.g., a requirement may be ambiguous, but when placed in the context of the background, domain, or other related requirements, it may be unambiguous. Product features found in marketing literature (e.g., shock resistant) are typically ambiguous. However, when placed in the context of the detailed specifications used by manufacturing, the ambiguity is no longer present. On the other hand, a requirement may be unambiguous, but when placed in the context of related requirements, there may be ambiguity.

When two requirements conflict with each other or create contextual ambiguity, they are said to be inconsistent (see the later section “Consistent”).

4- Verifiable

A requirement is verifiable if the finished product or system can be tested to ensure that it meets the requirement.

Product features are almost always abstract and thus not verifiable. Analysis must be done to create testable requirements from the product features. For example, the requirement “The car shall have power brakes” is not testable, because it does not have sufficient detail. However, the more detailed requirement:

“The car shall come to a full stop from 60 miles per hour within 5 seconds”

Is testable, as is the requirement:

“**The power brake shall fully engage with 4 lbs. of pressure applied to the brake pedal.”**

As we have noted, product features lack detail and tend to be somewhat vague and not verifiable. However, the analysis of those features and the derived requirements should result in a specification from which full coverage test cases can be created.

5- Modifiable

A requirements specification is modifiable if its structure and style are such that any changes to a requirement can be made easily, completely, and consistently while retaining the structure and style.

The characteristic modifiable refers to two or more interrelated requirements or a complete requirements specification. Modifiability dictates that the requirements specification has:

1- A coherent.

2- Easy-to follow organization.

3- Has no redundancy (e.g., the same text appearing more than once), and that it keeps requirements distinct rather than intermixed.

A general rule is that information in a set of requirements should be in one and only one place so that a change to a requirement does not require cascading changes to other requirements. A typical way of ensuring modifiability is to have a requirement either <u>reference</u> other requirements specifically or use a <u>trace mechanism</u> to connect interrelated requirements.

6- Consistent

A requirement is consistent if it does not contradict or is not in conflict with any external corporate documents or standards or other product or project requirements.

In general, consistency is a relationship among at least two requirements
Contradiction occurs when the set of external documents, standards, and other requirements result in ambiguity or a product is no longer feasible to build. For example, a corporate standard might require that all user interface forms have a corporate logo in the upper-right corner of the screen, the bottom center of the screen. There are now two conflicting requirements, and even though a requirements specification may be internally consistent, the specification would still be inconsistent because of conflict with corporate standards. Creating documentation that is both internally and externally consistent requires careful attention to detail during reviews.

7- Complete

A requirements specification is complete if it includes all relevant correct requirements, and sufficient information is available for the product to be built. When dealing with a high-level requirement, the completeness characteristic applies holistically to the complete set of lower-level requirements associated with the high-level feature or requirement. Completeness also dictates that:

• Requirements be ranked for importance and stability.
• Requirements and test plans mirror each other.

A requirements specification is complete if it includes the following elements”

1. Definition of the responses of the system or product to all realizable classes of input data in all realizable classes of situations. Note that it is important to specify the responses to both valid and invalid input values and to use them in test cases.

2. Full labels and references to all figures, tables, and diagrams in the specification and definitions of all terms and units of measure.

3. Quantification of the nonfunctional requirements. That is, testable, agreed-on criteria must be established for each nonfunctional requirement.

Nonfunctional requirements are usually managed by the project’s chief architect. In order for the completed product to be correct and complete, it must include the testable requirements that have been derived from the high-level nonfunctional requirements. It is difficult to create complete specifications, yet complete specifications are mandatory under certain circumstances; e.g., where the implementation team has no domain knowledge, or where communication between subject experts and developers will be problematic. We have seen projects where the requirements definition phase was shortened for schedule reasons. The general consensus when doing a risk analysis, it was nearly always quite clear that having the developers complete the requirements was not an appropriate process, due to:

• Limited access to subject matter experts

• Lack of experience or bias when defining product requirements.

At the back end of the project, the failure to properly define the requirements almost always caused a greater delay than would have happened by allowing the requirements specification to be completed with the appropriate level of detail up front.

8- Traceable

Requirements traceability is the ability to describe and follow the life of a requirement, in both a forward and backward direction, i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases".

Traceability is required for proper requirements management and project tracking.
A requirement is traceable if the source of the requirement can be identified, any product components that implement the requirement can easily be identified, and any test cases for checking that the requirement has been implemented can easily be identified.

Tracing is sometimes mandated by a regulatory body such as the Federal Aviation Administration (FAA) or Food and Drug Administration (FDA) for product safety. Furthermore, there are some rare situations where failure to create the appropriate traces between requirements can have legal repercussions.

Other Project- or Product-Specific Characteristics

Occasionally, the requirements for a specific project or product have characteristics that do not apply to all the projects or products. While it can be argued that an attribute that crosscuts all other requirements is just another requirement, when treated as a characteristic it is more likely that the requirement will be fulfilled. For example, if a new system is being built that must be downward compatible with an older system; it could be argued that the need for downward compatibility is just a nonfunctional requirement.

However, we have found that having such all-encompassing requirements converted to characteristics makes it more likely that the completed system will be in compliance. A similar approach can be used for other "umbrella" requirements such as

- Compliance with Sarbanes-Oxley regulations
- Meeting all corporate security requirements
- Meeting electrical safety requirements

Characteristics of a Good Requirements Specification

As was stated in the definition of consistency, the definition of a characteristic may be different when applied to requirements and to a specification. A requirements specification is a filtered compendium of requirements. Having the requirements in a document rather than a database permits holistic views and allows the addition of history, a rationale, etc. There are certain characteristics that apply to specifications as opposed to individual requirements as listed here:

- A requirements specification is feasible if building the product specified is feasible given the state of technology, the budget, and the allotted time.
- A requirements specification is unambiguous if there is no pair-wise ambiguity in the specification.
- A requirements specification is valid if every requirement in it is valid.
- A requirements specification is verifiable if every requirement in it is verifiable.
- A requirements specification is modifiable if there is no redundancy and changes to requirements are easily and consistently made; e.g., a change to one requirement does not require cascading changes to other requirements.
- A requirements specification is consistent if the requirement set is internally consistent.
- A requirements specification is complete if it provides sufficient information for complete coverage testing of the product or system.
- A requirements specification is traceable if every requirement in it can be traced back to its source and forward to test cases.

• A requirements specification is concise if the removal of any requirement changes the definition of the product or system.

project time constraints. Consequently, it is important to prioritize and identify risks when defining requirements. For example, “If this nonfunctional requirement is not completely analyzed, what are the risks to the project, the company, and/or the user?” By doing a risk analysis, the effort associated with fully defining a requirement set can usually be balanced against the needs of the project.

1.8 Requirements and Project Failure

It must be remembered that most systems under development are not new; i.e., only a fraction of the requirements in the product are new or unique. Yet issues of requirements maintenance and long-term support are often missing from project plans; e.g., the project plan is created as though the requirements will be discarded after project completion.

When long-term requirements management is not planned, requirements creep can cause significant problems late in a project. Furthermore, Capers Jones reports that the defect rate increases significantly in requirements that are injected late over those that are created prior to the start of implementation, and the most egregious defects in requirements defined or modified late in a project can sometimes show up in litigation.

1.9 Quality and Metrics in Requirements Engineering

As was mentioned in connection with the success factors for projects, project indicators need to be defined in order to have some measure of project transparency. It is important to be able to answer the questions:

“**Am I making progress**?” and “**What is the quality of my work products**?”

How does one, for example, determine that a requirements specification is of high quality? Requirement characteristics or quality indicators are extremely important for determining artifact quality. <u>They can be measured by inspection (metrics), and the</u>

reported metrics can then be used to determine the quality of individual requirements and requirements specifications. Furthermore, metrics summaries tracked over time can be used to identify potential problems earlier to permit corrective actions, and provide guidance as to what type of corrective actions to take. For example, a high level of ambiguity in a requirement set might indicate that the analysts creating the requirements may need additional training in requirements writing.

Function Point Metrics as Leading Indicators

A function point is used to estimate the complexity and effort necessary to build a software product. Capers Jones has published extensively on this topic Function point metrics are:

1- An excellent way of identifying potential problems with requirements prior to the implementation of a project.

2- There is a clear correlation between function points and requirements; that is, function points can be used as an indicator of requirements creep and quality.

3- Function point analysis (FPA) can be effective in determining requirements completeness.

1.10 Summary

- We've introduced some of the key challenges for requirements engineering and some of the success factors to achieve good RE.
- We've provided a definition of requirements engineering.
- We've described the characteristics of a good requirement and a good requirements specification.

Requirements Engineering Artifact Modeling

"Without goals, and plans to reach them, you are like a ship that has set sail with no destination." —Fitzhugh Dodson

2.1 Introduction

In order to successfully reach a destination, travelers needs to know where they are going. For most of the software and system development life cycle, the work products are well understood, and professionals generally have a reasonable understanding of how to create them.

Requirements engineering is somewhat different, since it is a relatively new field in which fewer have worked; sometimes the objectives can be a bit obscure or hard to define. A lack of well defined work products may result in ill-defined RE artifacts and processes, with repercussions felt in the downstream phases of the life cycle.

This chapter discusses an important aspect of requirements engineering work; that is, fully and accurately defining RE work products and their relationships. While the examples shown here are specific to RE, many of the techniques could (and in some cases should) be extended to the entire project life cycle.

The purpose of requirements engineering artifact modeling is to:

- Define a reference model for RE that provides the core set of RE artifacts (work products) and their interdependencies.

- Guide the establishment and maintenance of product- and project-specific RE processes.

Thus, early requirements engineering activities include

- Analyzing marketing information, stakeholder, and user needs to derive the functional and nonfunctional requirements to be met by the system's design.

- Understanding the effect of these requirements on the business that creates the Product

- Consolidating these requirements into consistent and complete requirements and systems specifications as defined in the Requirements Engineering Artifact Model (REAM).

RE artifacts are used to **support product design** and **project management decisions** throughout the entire product life cycle. The quality and appropriateness of these artifacts is a key factor for successful system development. Developing consistent and comprehensive specifications of the "desired" system is an important objective of RE.

Thus, **the key components of requirements engineering artifact modeling** are:

• An RE artifact model as a measurable reference model that can be used to support interdisciplinary communication and specifications development.

• A process tailoring approach that specializes the RE artifact model to specific organizational or project needs

• RE artifact-centered process guidelines that define completion levels of the RE artifact model. The specified completion levels form a baseline for measuring project progress and artifact quality.

2.2 RE Taxonomy

It is important that all stakeholders and process participants in the development of products understand the meaning of each requirements engineering term to represent the same thing.

If, for example, customers, product managers, and manufacturing understand the term "feature" to mean different things, there may be difficulty with quality assurance tasks and related productivity. While universal definitions exist for many terms in requirements engineering, there is still disagreement within the RE research community as to the meaning of some terms such as "nonfunctional requirement." Consequently, it may be necessary for an organization or project to create its own set of definitions wherever there is the potential for misunderstandings.

We recommend that a project or product team have a **glossary** of terms. An enterprise-wide dictionary is always preferable but may not be feasible; e.g., different parts of an organization may be working in different domains.

A **taxonomy** is a collection of controlled vocabulary terms organized into a hierarchical structure. Taxonomies are commonly used to classify things; e.g., a taxonomy of the insect world.

An example of a taxonomy for requirements is given in Figure 2.1. In well-structured taxonomies, each term has only one parent. However, depending on need, it is possible to have poly-hierarchies where a term can have more than one parent. Figure 2.2 illustrates the difficulty of creating taxonomies; i.e., there may be multiple ways of representing concepts. Note that a term can appear in more than one place in a taxonomy.

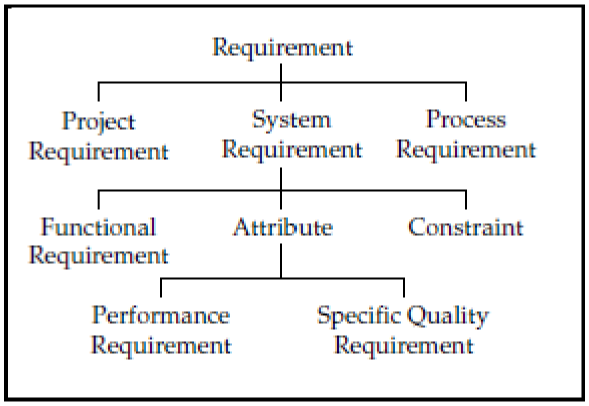


FIGURE **2.1** Requirements taxonomy suggested by Professor Glinz

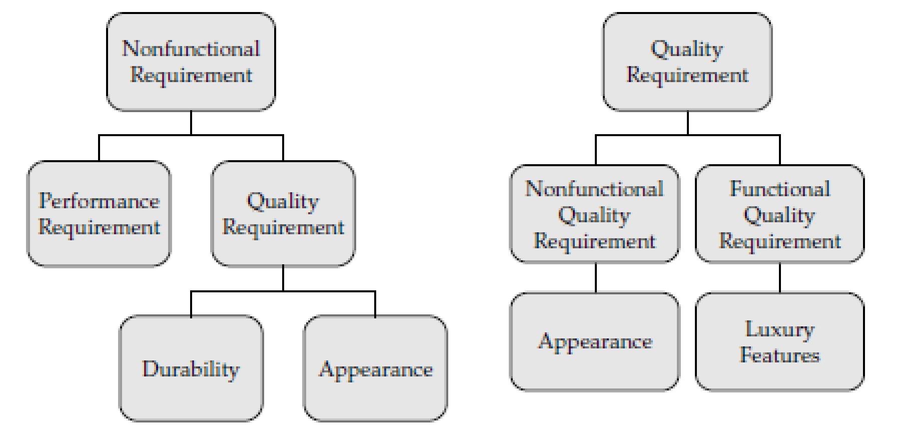


FIGURE 2.2 Two taxonomies illustrating differing representations of the same concepts

The difference between a glossary and a taxonomy is that in a glossary, terms are listed alphabetically and defined, whereas in a taxonomy, terms are grouped into classifications. To create a glossary, we recommend starting with a taxonomy of RE terms (e.g., Figure 2.1). The terms that would then go into the glossary are the leaves of the taxonomy tree plus any additional domain- or organization specific terms.

A complete RE taxonomy would include the classification of all artifacts associated with a requirements engineering process, not just the categorization of requirement types. Since the artifacts can change from organization to organization or project to project, any such taxonomy would have to be extensible. Taxonomies can be quite extensive. As an example, see the fragment of the taxonomy for security requirements given in Figure 2.3.

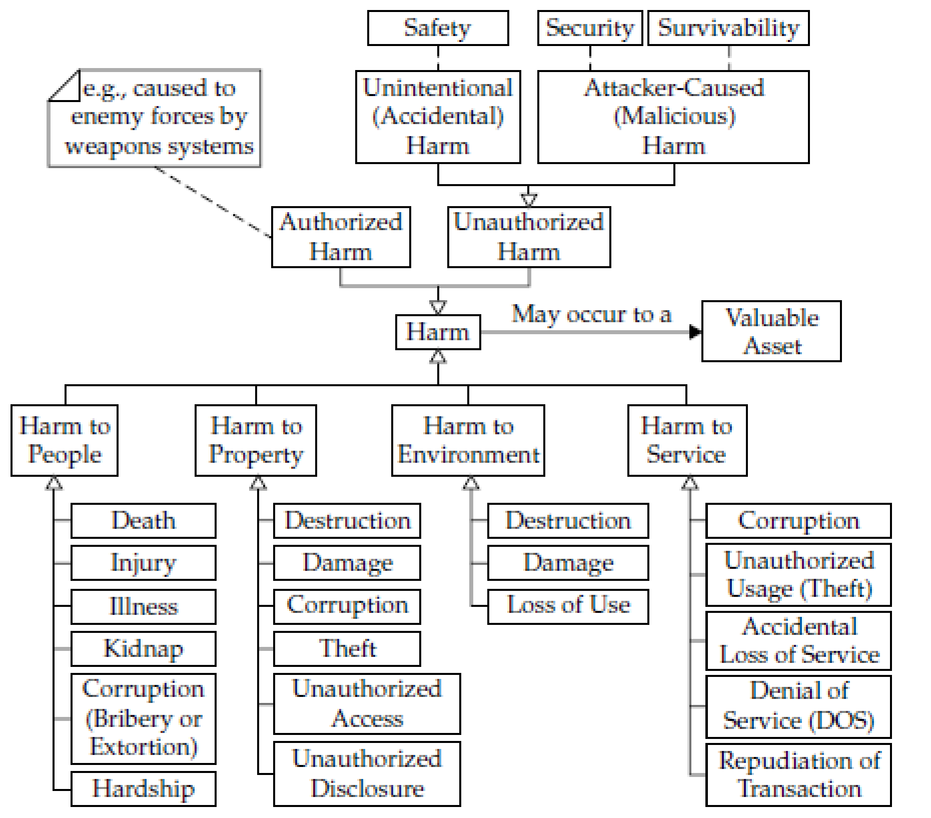
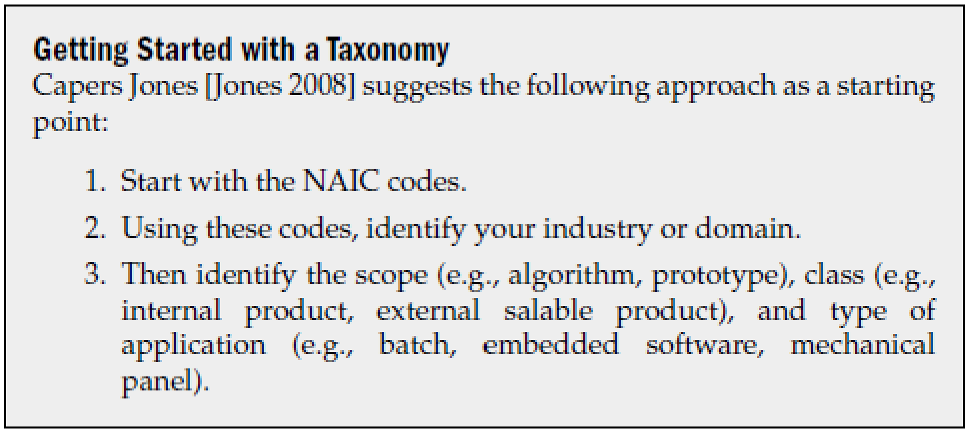


FIGURE 2.3 Sample taxonomy fragment for security requirements (Picture courtesy of Donald G. Firesmith, Software Engineering Institute, 2005)

A good starting point for creating a project- or product-specific taxonomy is the North American Industry Classification (NAICS) provided by the U.S. Government. This classification system is a great starting point for creating a domain/project/ product-specific taxonomy.



Getting Started with a Taxonomy
Capers Jones [Jones 2008] suggests the following approach as a starting point:

1. Start with the NAIC codes.
2. Using these codes, identify your industry or domain.
3. Then identify the scope (e.g., algorithm, prototype), class (e.g., internal product, external salable product), and type of application (e.g., batch, embedded software, mechanical panel).

Taxonomy Attributes

Any taxonomy for requirements engineering work products should, as a minimum, have the following attributes:

• **Complete** At the leaf level, include every requirement type that will be used by the organization or project. The categorization of requirements is critical when defining metrics. Without a proper categorization, it may not be possible to do a filtered query of a large requirements data store and return meaningful information.

• **Extensible** Companies should be able to take a core taxonomy and extend it. The sample fragment shown in Figure 2.3 is an example of a complex extension for security requirements.

• **Navigable** The taxonomy should be easy to navigate, possibly with hyperlinks on web pages.

• **Valid** There are many potential taxonomy sources; however, it is important that any such taxonomy used by an organization or on a product should be validated with other sources such as textbooks or experts.

• **Systematic** The categories should be well chosen and be at the same level.

Creation of an RE Taxonomy

There are many fine references and tools available to assist with the creation of taxonomies. We recommend the following simple steps (see the starting point suggested by Capers Jones in the sidebar on the previous page):

• Identify the tooling that will be used and how the taxonomy will be presented to project staff, keeping in mind that the taxonomy may have to be updated periodically, and there may be links to other tools; e.g., the taxonomy and the artifact model that will be described in the next section are interrelated.

• Collect all the requirement types that are currently in use or planned. Group them together.

• If the project is an incremental development, mine the requirements for classes. Note that Capers Jones estimates that as many as 75 percent of all new projects are incremental changes to an existing product.

- Categorize by grouping and create a draft taxonomy. For example, network performance requirements, UI performance (response time), query response times, etc., might all be grouped under *Performance Requirements*.

• Make sure that complete, agreed-upon definitions are available for every term that will be in the requirements taxonomy, including parent terms.

• Create a draft taxonomy and circulate to stakeholders for comments.

• Revise and publish (usually to the web).

• Provide feedback and maintenance mechanisms (including processes and identified roles) for keeping the taxonomy up to- date.

Other Types of Taxonomies Useful in RE

In addition to a "generic" RE taxonomy covering the classification of requirements, there are other artifact taxonomies that may be useful. For example, a ***document classification taxonomy*** can be used to identify common templates, assist with planning processes such as version control and baselining, and aid in the training of staff.

The leaves of such a document classification taxonomy should all be real documents that are created by the organization or project staff. A partial requirements document classification taxonomy can be seen in Figure 2.4.

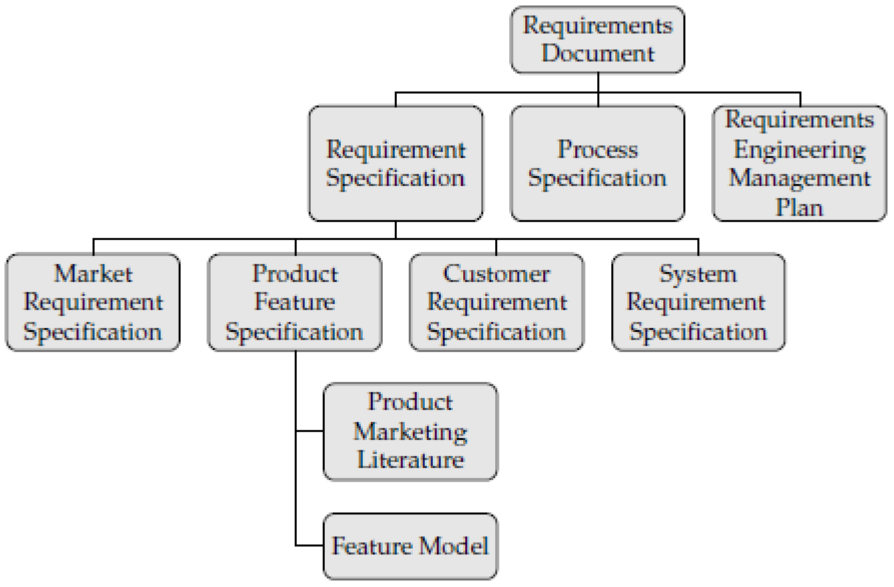

Requirements Document
Requirement Specification
Process Specification
Requirements Engineering Management Plan
Market Requirement Specification
Product Feature Specification
Customer Requirement Specification
System Requirement Specification
Product Marketing Literature
Feature Model

FIGURE 2.4 Sample partial taxonomy of requirement documents

Taxonomy Extension

To extend a taxonomy is a rather simple undertaking. The classification tree is extended with artifacts of the appropriate classification (see Figure 2.5). Figure 2.6 illustrates how detailed a taxonomy can become.

If templates with the appropriate attributes are filled in for each artifact, process definition is much simpler (see the section "Extending an Artifact Model to Augment Process Definition").

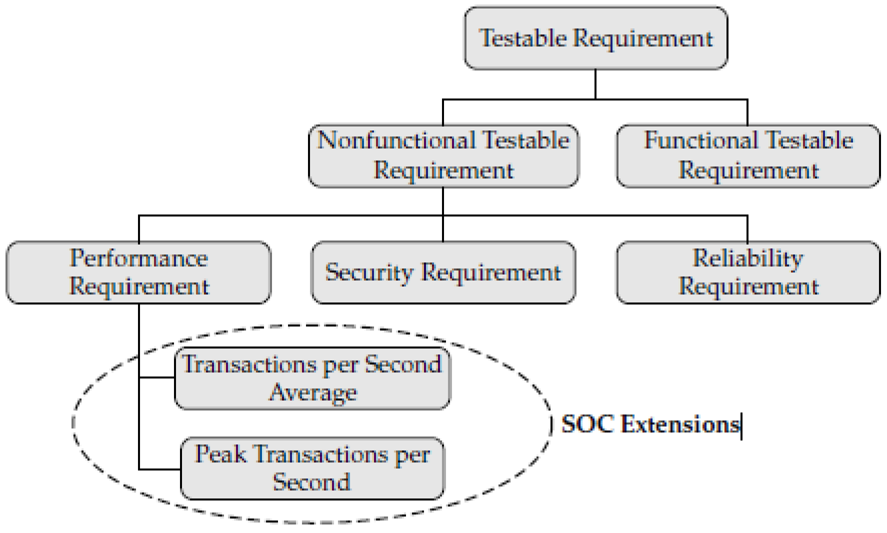

Testable Requirement
Nonfunctional Testable Requirement
Functional Testable Requirement
Performance Requirement
Security Requirement
Reliability Requirement
Transactions per Second Average
Peak Transactions per Second
SOC Extensions

Figure 2.5 Sample extension of a taxonomy

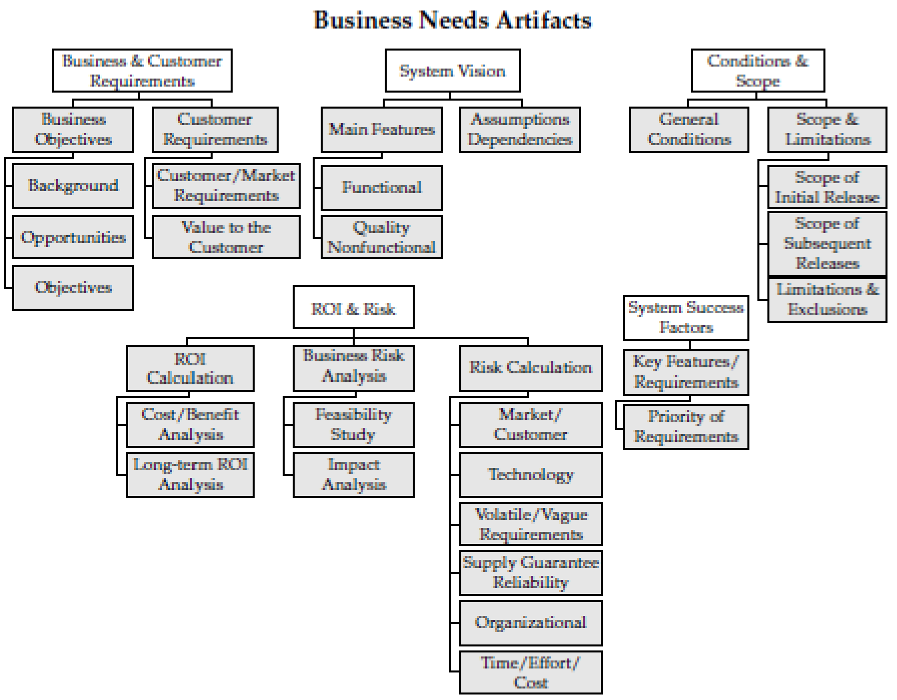


Figure 2.6 Taxonomy of business needs artifacts

2.3 RE Artifact Model

An RE artifact model (REAM): is a meta-model for the structuring of requirements engineering work products. A *meta-model* is an explicit model of the constructs and rules needed to build specific models within a domain of interest.

An RE artifact model contains all the artifacts referenced, modified, or created during requirements engineering activities. The artifacts shown on REAM diagrams are those that are actually used in a project, and they each have a name and definition.

Upon first glance, the REAM diagram may appear similar to a software class diagram. However, there are some significant differences. A software class diagram may show many different types of relationships between objects, whereas an RE artifact model only shows simple associations (a single solid line). For example,

• Classes shown on a class diagram may have methods and attributes; an RE artifact only has a name and description.

• A class diagram may show abstract classes, or classes for which there is no physical representation; an artifact model only shows real objects that will be used or created during a requirements engineering activity.

An artifact model is different than a taxonomy in that it is a graph rather than a tree, has many more artifacts than what would be in a taxonomy, and typically contains many domain specific extensions. Both a REAM and a taxonomy can be multitiered, so that selecting an object can open onto a different diagram. However, care must be taken in that while a taxonomy lends itself well to a hierarchical approach, artifact models tend to be flatter. For example, an object on one REAM diagram might have a relationship with an object on a different diagram.

Elements of an Artifact Model

A fragment of an artifact model is shown in Figure 2.7. It consists of

• **Artifact** A rectangular box with the name of the artifact. The definition of each artifact should be in a glossary or taxonomy accompanying the model.

• **Association** A line connecting two artifacts. The line indicates that there is a relationship between the artifacts. Every association must be labeled to indicate the relationship between the artifacts.

• **Cardinality** The cardinality indicates quantities. Any numbering convention can be used if appropriately defined;

However, the Unified Modeling Language (UML) notation3 is typically used. If the cardinality is not specified on an association, then unity is implied. Figure 2.7, showing a sample model fragment, can be read as follows: "One or more actors participate in one or more use cases, and an actor can initiate one or more use cases."

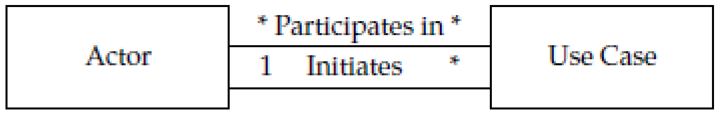


FIGURE 2.7 Simple artifact model

Creation of a Requirements Engineering Artifact Model

The actual creation of an artifact model is not difficult. What is important is to have a holistic understanding of the business processes used from product creation through maintenance. It may be necessary to identify an individual within an organization who can interact with stakeholders across the different organizational units. Across the entire organization and product life cycle, then, these questions must be asked:

- What are the artifacts that the roles use?
- How are the artifacts related?
- Who creates them?
- Who modifies them?
- How do they become obsolete?

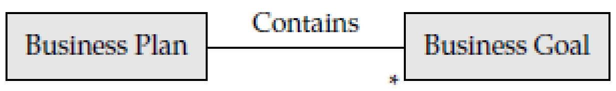
Consider, as an example, a small company creating a software product. They may have the following artifacts:

- Business plan
- Business goals

- Marketing brochure(s)
- Product features
- Customers
- Product definition
- Test plan
- Test cases
- System requirements
- Customer requirements
- Product design
- …

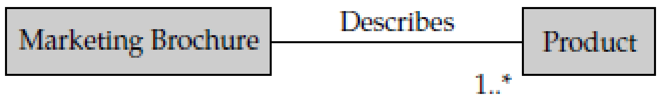
For creating the REAM, we first want to see how the products are related. We expect, for example, that a business plan will contain business goals (Figure 2.8). We can then model the artifacts and the relationships between them. We know that the business goals will be used as inputs to define the products. The products will be described in a marketing brochure (Figure 2.9).

FIGURE 2.8
Starting fragment



For creating the REAM, we first want to see how the products are related. We expect, for example, that a business plan will contain business goals (Figure 2.8). We can then model the artifacts and the relationships between them. We know that the business goals will be used as inputs to define the products. The products will be described in a marketing brochure (Figure 2.9).

FIGURE 2.9
Next model fragment



Various techniques are used to define the features that the product needs to have in the marketplace to meet the business goals. At this point, product features have to be tied to business goals. Since the model is a simple construct, any drawing tool can be used to create one. UML modeling tools work quite well, as do general purpose drawing tools such as Visio. However, it may be necessary to trace between different artifacts. Furthermore, it is important to have clear definitions of all the artifacts. This, in turn, may require stakeholder involvement. Also, if there is a taxonomy, all the leaves in the taxonomy should be in the artifact model. Since artifact models can be quite comprehensive, we may start with a subset. Let's say, for example, that, given the artifacts described, we wind up with an initial draft REAM as shown in Figure 2.10.

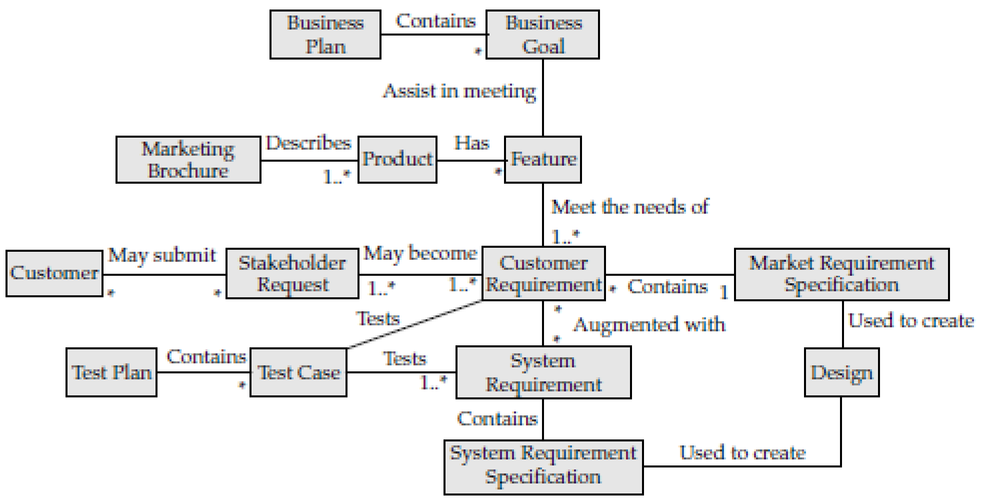


FIGURE 2.10 Example requirements engineering artifact model

There are some things missing from this draft REAM that would have to be added, including metrics, artifact reviews, project plans, standards and procedures, and so forth. Here are some other important things to consider that tend to be neglected until well into the project:

- Internal training standards and procedures
- Maintenance requirements (e.g., how will the product be maintained, what are the artifacts that will be needed to properly maintain the product after deployment?)
- Product documentation, including training manuals, marketing literature, internal maintenance manuals, and so on
- Holistic tool support that works across organizational boundaries (e.g., from the help desk to design)

2.4 Using the Artifact Model

The artifact model that is created prior to the start of a project is like looking at the X-ray of a patient prior to starting surgery. The model is used by most stakeholders (except possibly customers). Examples of the RE artifacts that will be used by the various roles of a development project are given in Table 2.1.

Extending an Artifact Model to Augment Process Definition

Artifact models can be extended to support process definition. For example, we may add artifacts such as completion status, decision gates, checklists, etc. (Figure 2.11). At the beginning of a project, a draft artifact model is created. The model is then used to define the product life cycle processes. After review, the artifact model and the defined processes are continually updated. While the upfront costs of creating a model may appear high, in our experience it is a very fast and cost-effective activity. Furthermore, just having project staff think about downstream artifacts, quality gates, and approval checklists can result in significant efficiencies.

2.5 Using Templates for Requirement Artifacts

A suggested way to get started creating a REAM is to use a template to fill in the information about each artifact in the model. A sample template is shown in Figure 2.12. The template is filled out for each artifact and then maintained with the same tool used to create the drawings. As mentioned previously, commercial tools are available; however, for a staff with developers it would be a relatively simple matter to extend a tool such as PowerPoint or Visio using the macro language. Once published to the web, the model and definitions are available to all the roles involved in the definition and creation of a product. A sample filled-in template for business and customer requirements is shown in Figure 2.13. If the template is for an artifact that will be created by the staff associated with the product, we recommend creating a checklist or set of quality indicators (see Figure 2.11) that can be used to determine:

• What is the quality of the artifact? Does it need rework?

• Has the artifact been completed? What are the criteria for completion?

• What is the status of the artifact; e.g., suggested, draft, completed, sunset?

Role (Cluster)	Objective	Functional Area	RE Artifacts
Product Management (ProdM)	Delivery of cost-effective products and solutions that meet customer needs	Planning and managing the entire life cycle of a product, including identifying customer needs, system vision, and scope	Business objectives, customer/user requirements, system vision, conditions and scope, product portfolio, return on investment (ROI), risks, system success factors
Requirements Engineering (RE)	Qualified and comprehensible/ reusable product decisions	Refinement and analysis of business objectives, reasonable and consolidated modeling of customer/user and business processes (functional, domain, quality goals, constraints)	Analysis models of customer and business needs (functional, domain, quality goals, constraints), user interface and system specification, acceptance conditions
Systems Architecture (SA)	High-quality and cost-effective system design that meets business requirements	Specifying system architecture according to quality and business requirements, defining the system structure, decomposing the system into functional interface specifications	Comprehensible functional system specification, system integration and interface specification, release planning, system test criteria
Project Management (ProjM)	Delivering the product solution within project constraints	Planning and managing the product development, process definition, measurement and control	System specification, design constraints, risk analysis, process requirements and constraints
Development (Dev)	Build to specifications	Implementation of product solution, including (hardware) design, coding, integration, testing	System/interface specification, design constraints, integration plan, system test criteria
Quality Assurance (QA)	Ensure verified product quality	Review and measurement of all specifications according to domain-specific quality standards	Measurable specifications, system integration, and (acceptance) test specification
Release Management (RelM)	Incremental release of product features	Release planning and execution according to market strategy, system structure, development sequence, and integration	Release strategy, system specifications, release planning, corresponding system interface, integration and test specification

Table 2.1 Use of an Artifact Model by Project Roles

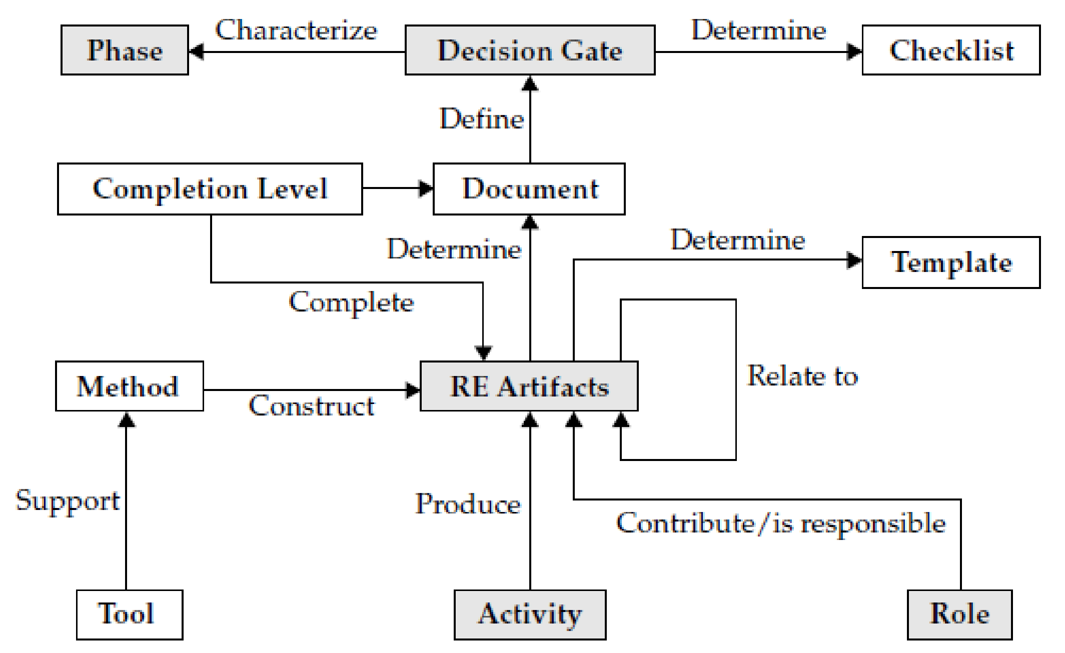
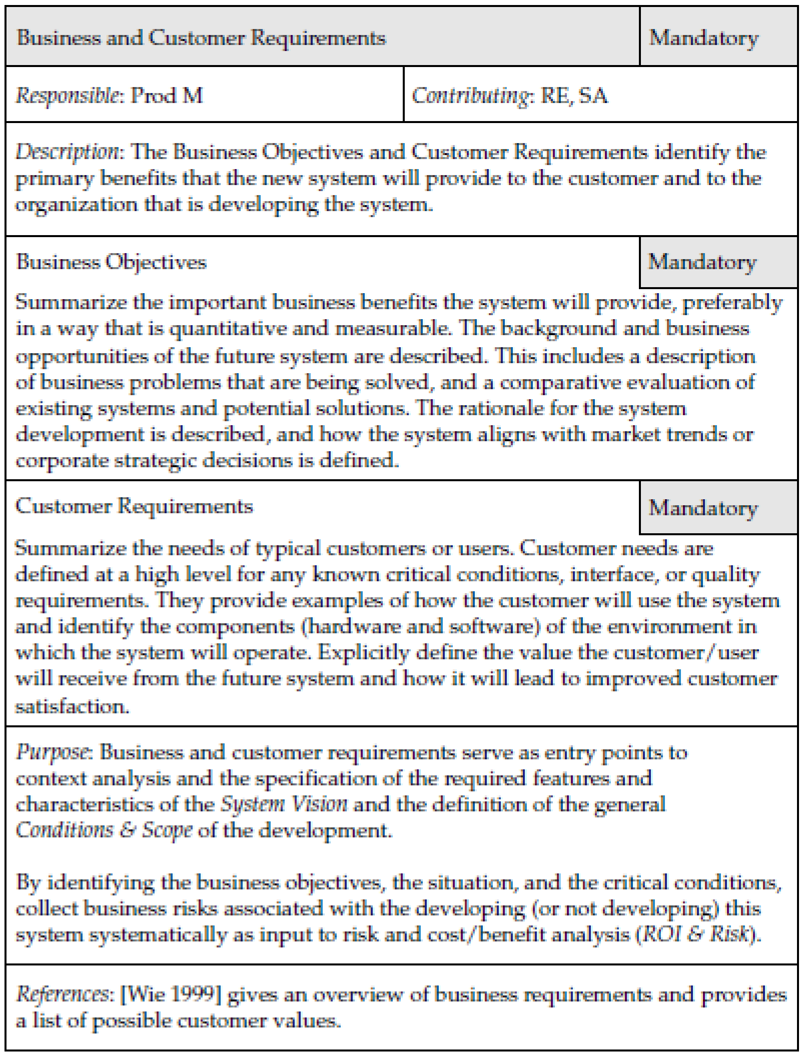


Figure 2.11 Process artifacts



Business and Customer Requirements	Mandatory
Responsible: Prod M	*Contributing*: RE, SA
Description: The Business Objectives and Customer Requirements identify the primary benefits that the new system will provide to the customer and to the organization that is developing the system.	
Business Objectives Summarize the important business benefits the system will provide, preferably in a way that is quantitative and measurable. The background and business opportunities of the future system are described. This includes a description of business problems that are being solved, and a comparative evaluation of existing systems and potential solutions. The rationale for the system development is described, and how the system aligns with market trends or corporate strategic decisions is defined.	Mandatory
Customer Requirements Summarize the needs of typical customers or users. Customer needs are defined at a high level for any known critical conditions, interface, or quality requirements. They provide examples of how the customer will use the system and identify the components (hardware and software) of the environment in which the system will operate. Explicitly define the value the customer/user will receive from the future system and how it will lead to improved customer satisfaction.	Mandatory
Purpose: Business and customer requirements serve as entry points to context analysis and the specification of the required features and characteristics of the *System Vision* and the definition of the general *Conditions & Scope* of the development. By identifying the business objectives, the situation, and the critical conditions, collect business risks associated with the developing (or not developing) this system systematically as input to risk and cost/benefit analysis (*ROI & Risk*).	
References: [Wie 1999] gives an overview of business requirements and provides a list of possible customer values.	

FIGURE 2.13 Filled-in artifact template for business requirements

2.6 Dynamic Tailoring of an Artifact Model

Software projects come in different sizes and use different methodologies. Large plan-driven projects can take years to implement and have staffs of well over 100 developers. Small, agile projects might have just two or three developers, and the project duration could be as short as a week or two.

When creating a REAM for a project, clearly one size does not fit all. If an organization has a range of projects on an ongoing basis, it is a good practice to provide some built-in tailoring facilities. An artifact, for example, could be mandatory on a "large" project, optional on a "medium-sized" project, and not used at all on a "small" project. If the project artifacts are tagged during the creation of the artifact model, it then becomes possible to filter and present the required information, or to couple it to a workflow used to reinforce the process. Tailoring techniques range from simple manual selection of artifacts to very sophisticated approaches such as the use of neural nets.

Regardless of the tailoring approach, it will not work unless the artifacts in the model have attributes that permit them to be evaluated based on type, size, and duration of the project. An example of a small model used to define the artifacts for a prototyping effort can be seen in Figure 2.14. An example table fragment for defining tailoring rules is shown in Figure 2.15.

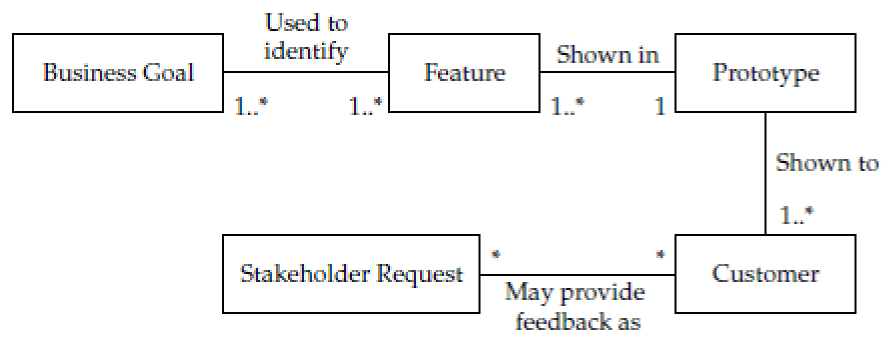
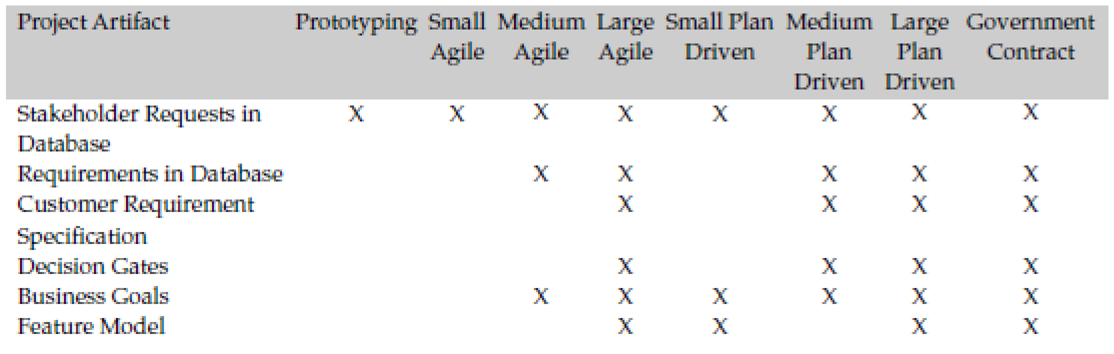


FIGURE 2.14 Artifact model for small prototyping project



Project Artifact	Prototyping	Small Agile	Medium Agile	Large Agile	Small Plan Driven	Medium Plan Driven	Large Plan Driven	Government Contract
Stakeholder Requests in Database	X	X	X	X	X	X	X	X
Requirements in Database			X	X		X	X	X
Customer Requirement Specification				X		X	X	X
Decision Gates				X		X	X	X
Business Goals			X	X	X	X	X	X
Feature Model				X	X		X	X

FIGURE 2.15 Sample table for tailoring RE processes

2.7 Organizational Artifact Model Tailoring

In addition to the tailoring of an artifact model for a specific project, high-level organizational models can be used as the starting point for the creation of project-specific models. An example is given in Figure 2.16. The starting point was a corporate-level model defining the core artifacts needed on any project. That model is then modified for the specific organization within the company, and finally the model is completed on a per-project basis.

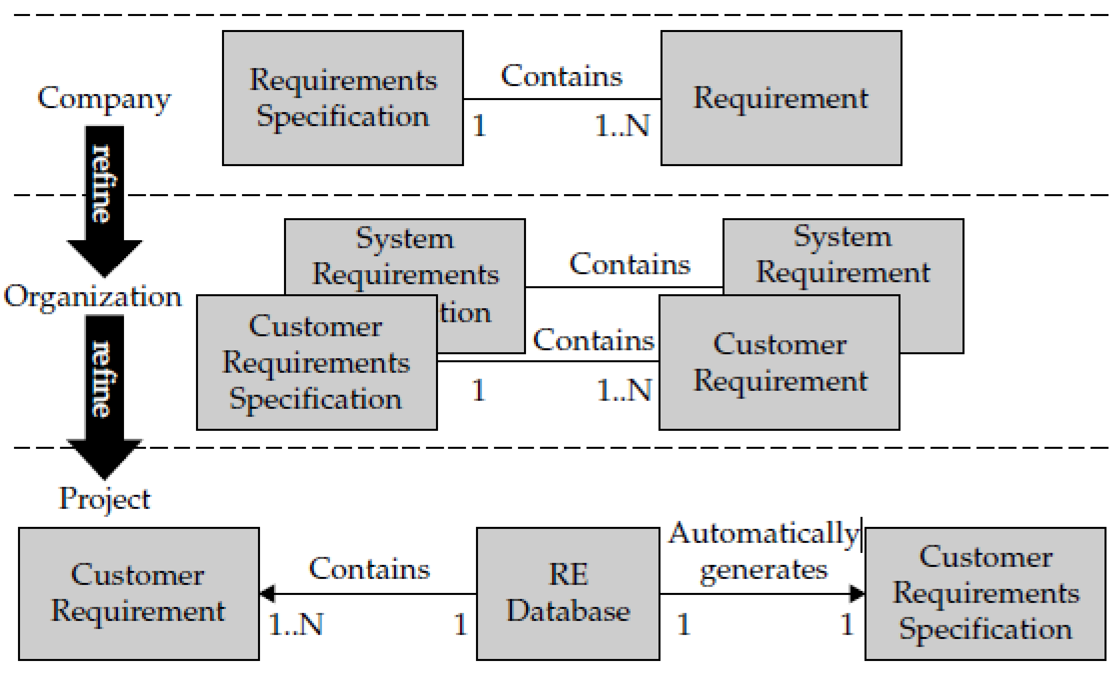


FIGURE 2.16 Organizational tailoring of an artifact model

2.8 Creating a System Life Cycle Process

As was mentioned earlier, both a taxonomy and an artifact model are useful in the creation of system life cycle processes. By adding attributes to the artifacts that specify when they are needed (based on the type and size of the project), a query will result in the production of a list of all the appropriate artifacts.

Project management can then use this list for planning, including the definition of decision and review points, work products needed, and quality artifacts needed to measure project quality and efficiency. An example process creation approach is illustrated in Figure 2.17. Process creation to some extent can be automated, depending on how much of an investment the organization is willing to make in tooling. Automation of process creation can include:

• Generation of selected project templates

• Assembly of standards and procedures from a library

• Drawing of a filtered, domain, and project-specific artifact model

• Population of a rule set for a workflow engine

In general, it is much better to have an **"active"** rather than a **"passive"** process. An ***active process*** is one where rules are used to prompt and inform staff about activities and provide templates for documents that have already been tailored based on the project type. A **passive process** is where documents (e.g., standards, procedures, templates) are stored containing process information, and the project staff has to download and read the relevant information.

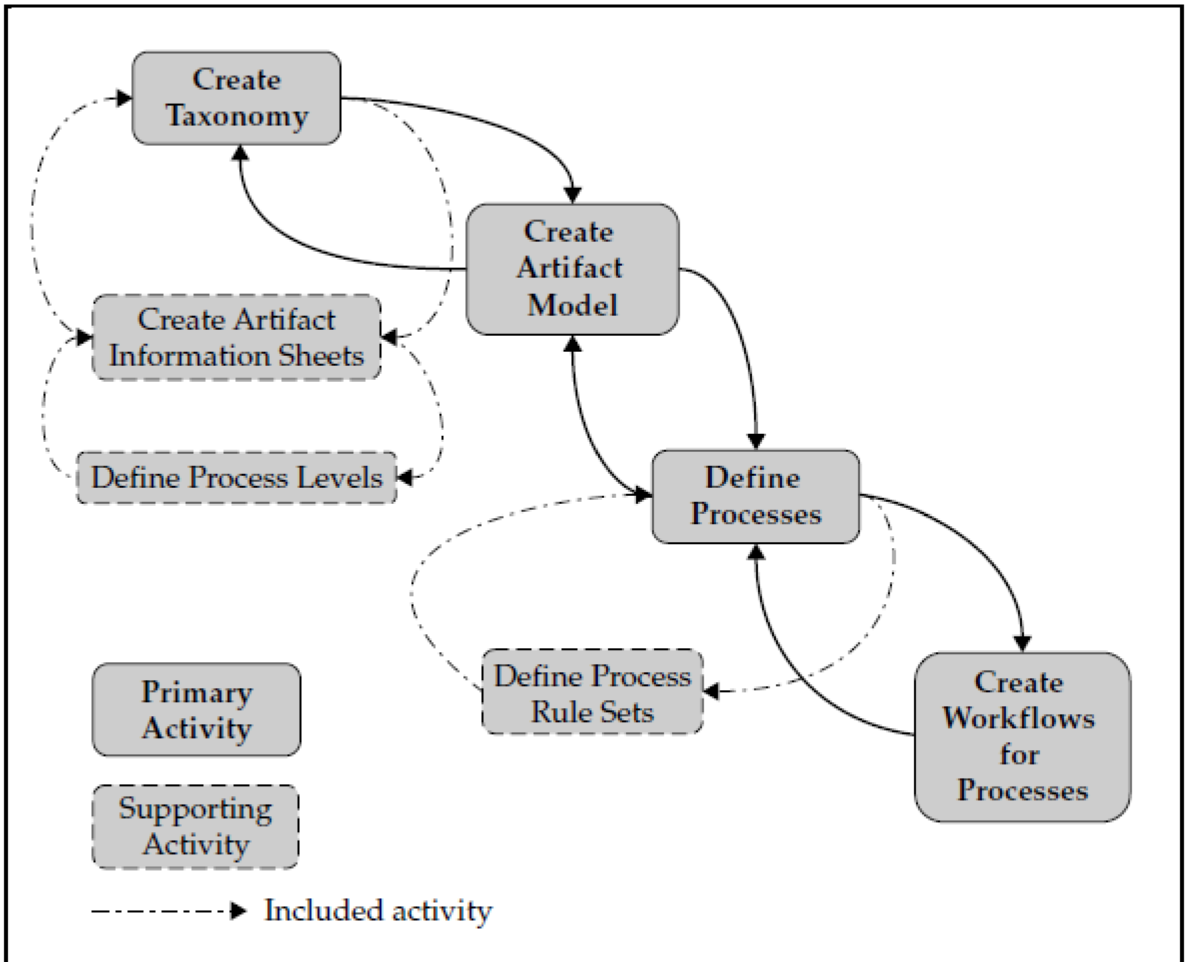


FIGURE 2.17 RE activities for process creation

2.9 Tips for Requirements Engineering Artifact Modeling

Some suggested practices for modeling requirements engineering artifacts are summarized below:

- Define a Glossary of Terms for your project or product.
- Create an RE Taxonomy while keeping in mind what tools will be used to maintain it and how it will be communicated to the project team (e.g., publish to a project web site).
- Develop an RE Artifact Model specific to your project.
- Communicate project roles to all team members and the artifacts they are responsible for as defined in the RE Artifact Model.
- Use templates to define RE artifacts.
- For scaling projects, provide tailoring information in the RE Artifact Model; e.g., a specific artifact may be mandatory, optional, or not used, depending on the project size.
- Tailor the RE Artifact Model for a specific project from any corporate-level models, if they exist.
- Create a system life cycle process by adding needed timing to the defined artifacts.

2.10 Summary

We have seen in this chapter how taxonomies are used to define and classify work products that are referenced, created, or modified during the requirements engineering process.
A taxonomy is typically the starting point for the creation of a project glossary and a Requirements Engineering Artifact Model. An artifact model for an organization is essential for the definition of requirements engineering processes; however, the same techniques can be extended to defining the entire life cycle model.

Organizations that have different types of projects need to be flexible in their approach to process definition, so that small projects will not be burdened with excessive bureaucracy and paperwork, while larger projects will have the infrastructure and tools necessary to succeed.