# Requirements & Specifications

Carnegie Mellon University
18-849b Dependable Embedded Systems
Spring 1999
Author: Eushiuan Tran

## Abstract:

Defining requirements to establish specifications is the first step in the development of an embedded system. However, in many situations, not enough care is taken in establishing correct requirements up front. This causes problems when ambiguities in requirements surface later in the life cycle, and more time and money is spent in fixing these ambiguities. Therefore, it is necessary that requirements are established in a systematic way to ensure their accuracy and completeness, but this is not always an easy task. This difficulty in establishing good requirements often makes it more of an art than a science. The difficulty arises from the fact that establishing requirements is a tough abstraction problem and often the implementation gets mixed with the requirements. In addition, it requires people with both communication and technical skills. As requirements are often weak about what a system should not do, this poses potential problems in the development of dependable systems, where these requirements are necessary to ensure that the system does not enter an undefined state. The development of dependable embedded systems requires even more complicated requirements as the embedded system not only interacts with the software but also with the outside world. Therefore, the importance of establishing good requirements is even greater in embedded systems design.

---

## Contents:

---

# Introduction

Requirements and specifications are very important components in the development of any embedded system. Requirements analysis is the first step in the system design process, where a user's requirements should be clarified and documented to generate the corresponding specifications. While it is a common tendency for designers to be anxious about starting the design and implementation, discussing requirements with the customer is vital in the construction of safety-critical systems. For activities in this first stage has significant impact on the downstream results in the system life cycle. For example, errors developed during the requirements and specifications stage may lead to errors in the design stage. When this error is discovered, the engineers must revisit the requirements and specifications to fix the problem. This leads not only to more time wasted but also the possibility of other requirements and specifications errors. Many accidents are traced to requirements flaws, incomplete implementation of specifications, or wrong assumptions about the requirements. While these problems may be acceptable in non-safety-critical systems, safety-critical systems cannot tolerate errors due to requirements and specifications. Therefore, it is necessary that the requirements are specified correctly to generate clear and accurate specifications.

There is a distinct difference between requirements and specifications. A requirement is a condition needed by a user to solve a problem or achieve an objective. A specification is a document that specifies, in a complete, precise, verifiable manner, the requirements, design, behavior, or other characteristics of a system, and often, the procedures for determining whether these provisions have been satisfied. For example, a requirement for a car could be that the maximum speed to be at least 120mph. The specification for this requirement would include technical information about specific design aspects. Another term that is commonly seen in books and papers is requirements specification which is a document that specifies the requirements for a system or component. It includes functional requirements, performance requirements, interface requirements,

design requirements, and developement standards. So the requirements specification is simply the requirements written down on paper.

---

# Key Concepts

## Establishing Correct Requirements

The first step toward developing accurate and complete specifications is to establish correct requirements. As easy as this sounds, establishing correct requirements is extremely difficult and is more of an art than a science. There are different steps one can take toward establishing correct requirements. Although some of the suggestions sound fairly obvious, actually puttting them into practice may not be as easy as it sounds. The first step is to negotiate a common understanding. There is a quote by John von Neumann that states "There's no sense being exact about something if you don't even know what you're talking about." [Gause89] Communication between the designer and customer is vital. There is no point in trying to establish exact specifications if the designers and customers cannot even agree on what the requirements are.

Problem stems from ambiguities in stating requirements. For example, say the requirement states that we want  to create a means that would transport a group of people from Boston to Washington D.C. Possible interpretations of this requirement includes building a bus, train, or airplane, among other possibilities. Although each of these transportation devices satisfy the requirement, they are certainly very different. Ambiguous requirements can be caused by missing requirements, ambiguous words, or introduced elements. The above requirement does not state how fast the people should be transported from Boston to Washington D.C. Taking an airplane would certainly be faster than riding a bus or train. These are also missing requirements. "a group of people" in the above requirement is an example of ambiguous words. What exactly does "group" imply? A group can consist of 5 people, 100 people, 1000 people, etc. The requirement states to "create a means" and not "design a transportation device". This is an example of introduced elements where an incorrect meaning slipped into the discussion. It is important to eliminate or at least reduce ambiguities as early as possible because the cost of them increases as we progress in the development life cycle.

Often the problem one has in establishing correct requirements is how to get started. One of the most important things in getting started is to ask questions.

Context-free questions are high-level questions that are posed early in a project to obtain information about global properties of the design problem and potential solutions. Examples of context-free questions include who is the client?, what is the reason for solving this problem?, what environment is this product likely to encounter?, and what is the trade-off between time and value?. These questions force both sides, designer and customer, to look at the higher issues. Also, since these questions are appropriate for any project, they can be prepared in advance. Another important point is to get the right people involved. There is no point in discussing requirements if the appropriate people are not involved in the discussion. Related to getting the right people involved is making meetings work. Having effective meetings is not as easy as it sounds. However, since they play a central role in establishing requirements it is essential to know how to make meetings work. There are important points to keep in mind when creating effective meetings, which include creating a culture of safety for all participants, keeping the meeting to an appropriate size, and other points. [Gause89]

Exploring the possibilities is another important step toward generating correct requirements. Ideas are essential in establishing correct requirements, so it is important that people can get together and generate ideas. Every project will also encounter conflicts. Conflicts can occur from personality clashes, people that cannot get along, intergroup prejudice such as those between technical people and marketing people, and level differences. It is important that a facilitator is present to help resolve conflicts.

In establishing requirements, it is important to specifically establish the functions, attributes, constraints, preferences, and expectations of the product. Usually in the process of gaining information, functions are the first ones to be defined. Functions describe what the product is going to accomplish. It is also important to determine the attributes of a product. Attributes are characteristics desired by the client, and while 2 products can have similar functions, they can have competely different attributes. After all the attributes have been clarified and attached to functions, we must determine the constraints on each of the atrributes. Preferences, which is a desirable but optional condition placed on an attribute, can also be defined in addition to its constraints. Finally, we must determine what the client's expectations are. This will largely determine the success of the product.

Testing is the final step on the road to establishing correct requirements. There are several testing methods used, as listed below. [Gause89]

- **Ambiguity poll** - Used to estimate the ambiguity in a requirement. This involves asking questions such as how fast?, how big?, how expensive?, and then determining if there is ambiguity between the high and low values.
- **Technical review** - A testing tool for indicating the progress of the requirements work. It can be formal or informal and generally only deals with technical issues. Technical reviews are necessary because it is not possible to produce error-free requirements and usually it is difficult for the producers to see their own mistakes.
- **User satisfaction test** - A test used on a regular basis to determine if a customer will be satisifed with a product.
- **Black box test cases** - Constructed primarily to test the completeness, accuracy, clarity, and conciseness of the requirements.
- **Existing products** - Useful in determining the desirable and undesirable characteristics of a new product.

At some point it is necessary to end the requirements process as the fear of ending can lead to an endless cycle. This does not mean that it is impossible to revisit the requirements at a later point in the development life cycle if necessary. However, it is important to end the process when all the necessary requirements have been determined, otherwise you will never proceed to the design cycle.

Establishing good requirements requires people with both technical and communication skills. Technical skills are required as the embedded system will be highly complex and may require knowledge from different engineering disciplines such as electrical engineering and mechanical engineering. Communication skills are necessary as there is a lot of exchange of information between the customer and the designer. Without either of these two skills, the requirements will be unclear or inaccurate.

It is essential that requirements in safety critical embedded systems are clear, accurate, and complete. The problem with requirements is that they are often weak about what a system should not do. In a dependable system, it is just as important to specify what a system is not suppose to do as to specfiy what a system is suppose to do. These systems have an even greater urgency that the requirements are complete because they will only be dependable if we know exactly what a system will do in a certain state and the actions that it should not perform. Requirements with no ambiguities will also make the system more dependable. Extra requirements will usually be required in developing a dependable embedded system. For example, in developing a dependable system for  non-computer-literate people, extra requirements should be specified to make the system safe even in exceptional or abusive situations.

## Requirements and Specification's Role in System Design

Systems exist everywhere in the universe we live in. The universe can be considered a system, and so can an atom. A system is very loosely defined and can be considered as any of the following definitions. [Blanchard90]

- A combination of elements forming a complex or unitary whole (i.e. river system or transportation system)
- A set of correlated members (i.e. system of currency)
- An ordered and comprehensive assemblage of facts, principles, or doctrines in a particular field of knowledge (i.e. system of philosophy)
- A coordinated body of methods, a complex scheme, or a plan of procedure (i.e. system of organization and management)
- Any regular or special method of plan of procedure (i.e. a system of marking)

The important characteristic of a system is that there is unity, functional relationship, and useful purpose. Systems engineering is not a technical specialty but is a process used in the evolution of systems from the point when a need is identified through production and construction to deployment of the system for consumer use. [Blanchard90] Systems engineering requires knowledge from different engineering disciplines such as aeronautical engineering, civil engineering, and electrical engineering. The development of embedded systems also requires the knowledge of different engineering disciplines and can follow the techniques used for systems engineering. Therefore, it is appropriate that the steps used in establishing system requirements also be applicable to requirements for embedded systems.

The conceptual system design is the first stage in the systems design life cycle and an example of the systems definition requirements process is shown in Figure 1. Each individual box will be explain below.

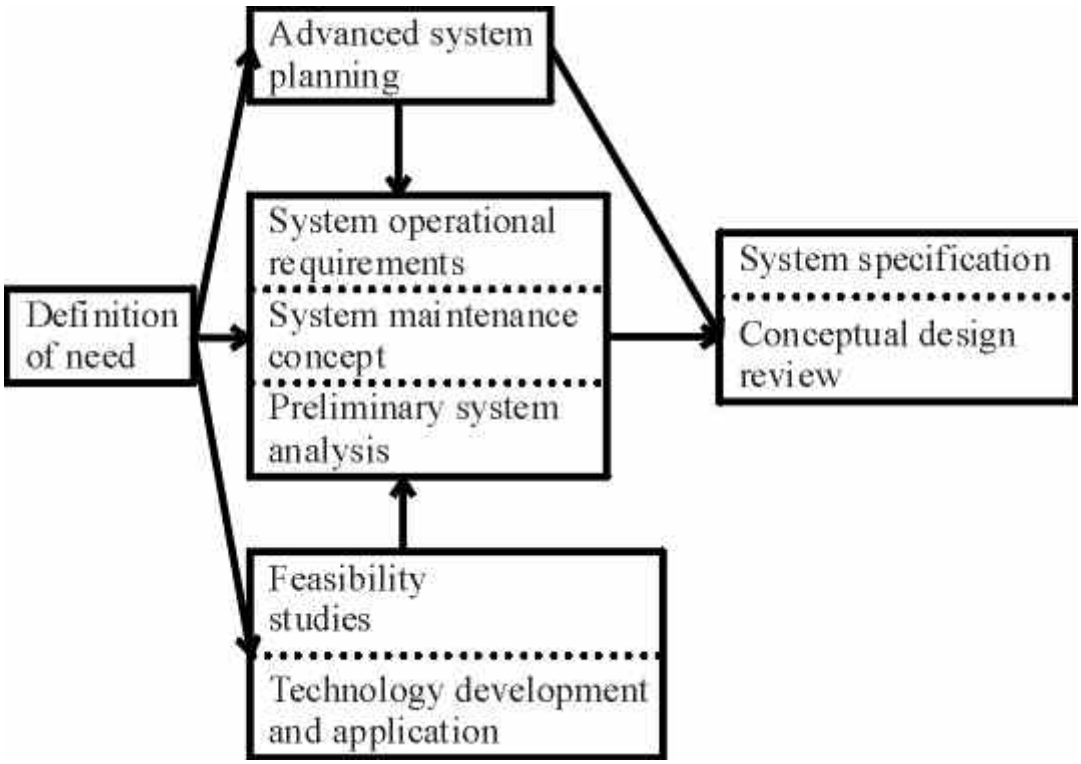


Figure 1: Example of system requirements definition process [Blanchard90]

In establishing system requirements, the first step is to define a need. This need is based on a want or desire. Usually, an individual or organization identifies a need for an item or function and then a new or modified system is developed to fulfill the requirement. After a need is defined, feasibility studies should be conducted to evaluate various technical approaches that can be taken. The system operational requirements should also be defined. This includes the definition of system operating characteristics, maintenance support concept for the system, and identification of specific design criteria. In particular, the system operational requirements should include the following elements. [Blanchard90]

- **Mission definition** - Identification of the primary operating mission of the system in addition to alternative and secondary missions.
- **Performance and physical parameters** - Definition of the operating characteristics or functions of the system.
- **Use requirements** - Anticipation of the use of the system.
- **Operational deployment or distribution** - Identification of transportation and mobility requirements. Includes quantity of equipment, personnel, etc. and geographical location.
- **Operational life cycle** - Anticipation of the time that the system will be in operational use.
- **Effectiveness factors** - Numbers specified for system requirements. Includes cost-system effectiveness, mean time between

maintenance(MTBM), failure rate, maintenance downtime, etc.
- **Environment** - Definition of the environment in which the system is expected to operate.

Basically, the system operational requirements define how the system will be used in the field by the customer.

Usually, in defining system requirements, the tendency is to cover areas that are related to performance as opposed to areas that are related to support. However, this means that emphasis is only placed on part of the system and not the whole system. It is essential to take into consideration the entire system when defining system requirements. The system maintenance concept basically describes the overall support environment that the product is supposed to exist in.

After the system operational requirements and system maintenance concept are defined, the preliminary system analysis is performed to determine which approach for system development should be adopted. The following process is usually applied. [Blanchard90]

**1. Define the problem** - The first step always begin with clarifying the objectives, defining the concerned issues, and limiting the problem so that it can be effectively studied.
**2. Identify feasible alternatives** - All the alternatives should be considered to make sure that the best approach is chosen.
**3. Select the evaluation criteria** - The criteria for the evaluation process can vary considerably, so the appropriate ones must be chosen.
**4. Applying modeling techniques** - A model or series of models should be used.
**5. Generate input data** - The requirements for appropriate input data should be specified.
**6. Manipulate the model** - After data is collected and inputed, the model may be used. Analysis after using the model will lead to recommendation for some kind of action.

After the preliminary system analysis, advanced system planning will be done. Early system planning takes place from the identification of a need through the conceptual design phase. The results from these planning will be defined as either technical requirements included in the specifications or management requirements included in a program management plan. The documents associated with these requirements are shown in Figure 2. The system specification includes information from the operational requirements,

maintenance concept, and feasibility analysis. The System Engineering Management Plan(SEMP) contains three sections. The technical program planning and control part describes the program tasks that have to be planned and developed to meet system engineering objectives such as work breakdown structure, organization, risk management, etc. The system engineering process part describes how the system engineering process applies to program requirements. Finally, the engineering specialty integration part describes the major system-level requirements in the engineering specialty areas such as reliability, maintainability, quality assurance, etc.

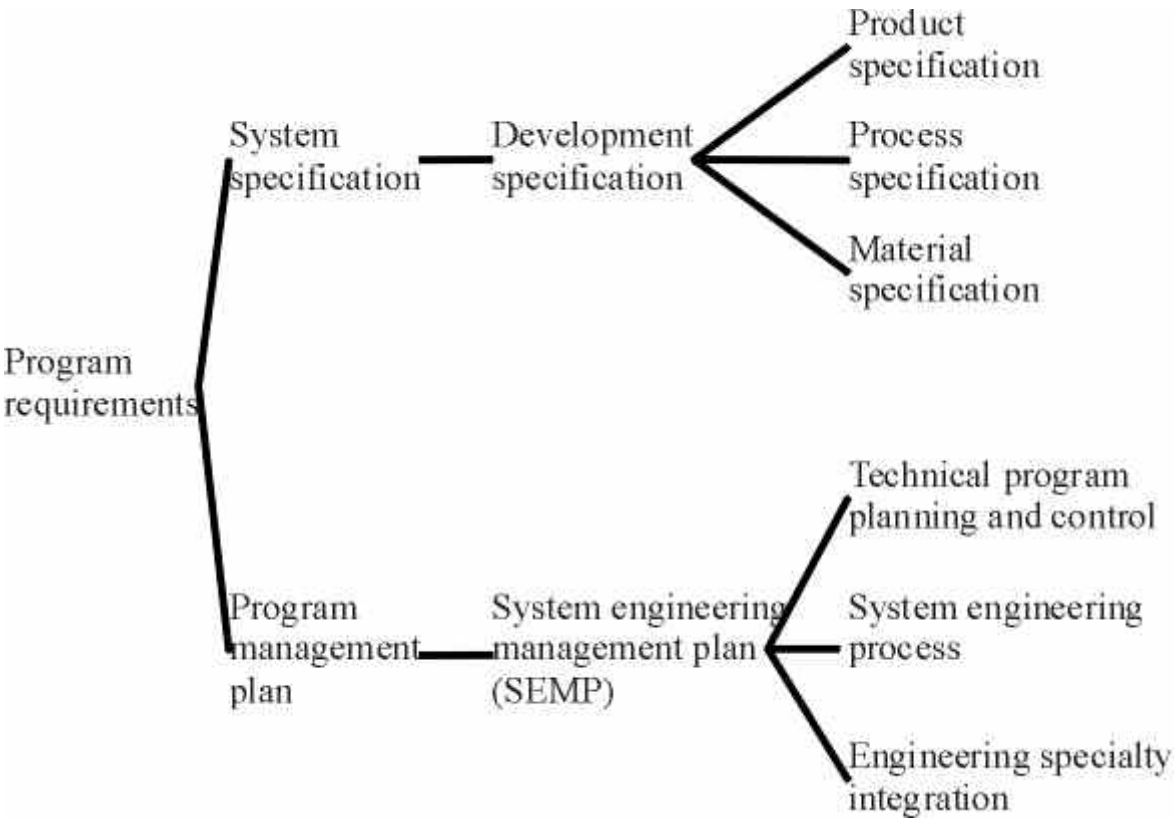


Figure 2: Program documentation [Blanchard90]

Finally, the conceptual design review is also performed during the conceptual design stage. It usually occurs early in the system engineering development life cycle after the operational requirements and the maintenance concept have been defined. [Blanchard90]

## Requirements Traceability

It is very important to verify that the requirements are correctly implemented in the design. This is done with requirements traceability which is usually refered to as "the ability to follow the life of a requirement, in both forwards and backwards direction (i.e. from its origins, through its development and specification, to its

subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases.)" [Ramesh95] Requirements traceability captures the relationships between the requirements, specifications, and design. Standards for systems development such as the one from the U. S. Department of Defense (standard 2167A) require that requirements traceability be used. Although requirments traceability has been around for more than 2 decades, there has been no consensus as to what kind of information should be used as part of a traceability scheme. The problem is that the definition of traceability differs when taken from different points of view of the system.(i.e. the view of the system is different for customer, project manager, test engineer, etc.) Each organization has a different purpose and methodology for requirements tracing. While it is not the purpose of this paper to dwell into a long discussion about requirements traceability, a short example of the methodology used at one organization will be given. [Ramesh95]

The projects typically involved at Abbott Laboratories Diagnostics Division are real-time, embedded, in vitro diagnostic instruments approaching 200,000 lines of code. They have found that traceability aids project managers in verification, cost reduction, accountability and change management. Traceability helps in verifying that software requirements are satisfied in the design process and that they are tested for during the verification process. Traceability allows the allocation of product requirements early in the development cycle thereby reducing costs of correcting defects (due to untraceable components) in the integration and system test phase. Providing quantitative traceability analyses also allows for accountability in making sure that project milestones are approved, deliverables are verified, and customers are satisfied. The documentation from traceability also keeps information organized during changes in staff or management.

A specific in vitro diagnostic instrument contained approximately 175,000 lines of source code and approximately 1,600 software requirements that needed to be traced. While the division also has an automated traceability system (ATS) that allowed them to automate many of the tasks, it was the process and not the tool that led to their success. The main purpose of the traceability program is to identify links and determine that the links are complete and accurate. The traceability analysis consists of 4 aspects: forward requirements analysis, reverse requirements trace, forward test analysis, and reverse test trace. These steps are used to trace each software requirement through its design elements and test traces. The ATS can be used to design documentation matrices and test matrices that is used to perform the different analyses required. The ATS is also able to give feedback about the design components that are not yet implemented during

the life cycle. In the test phase, the ATS gives input to what requirements are covered by the test cases. [Watkins94]

### Requirements Standards

There are many requirements and specification standards. They are mostly military standards as opposed to "commercial" standards. In addition, most of the standards are in the systems engineering area, and in particular deals with the software aspects. A good reference to many of these standards is Standards, Guidelines, and Examples on System and Software Requirements Engineering from the IEEE Computer Society Press. [Dorfman90] This book is a compilation of international requirements standards and U. S. military standards. There is also a section on requirements analysis methodologies and examples. Listed below are several relevant standards, but the list is in no means exhaustive.

- IEEE Recommended Practice for Software Requirements Specifications (IEEE std 830-1998)
- British Standard Guide to Specifying User Requirements for a Computer-Based Standard (BS6719 - 1986)
- Canadian Standard, Basic Guidelines for the Structure of Documentation of System Design Information (Z242.15.4-1979)
- U. S. Military Standards

- System/Segment Specification (DI-CMAN-80008A, 2/29/1988)
- Software Requirements Specification (DI-MCCR-80025A, 2/29/1988)
- Interface Requirements Specification (DI-MCCR-80026A, 2/29/1998)

---

## Available tools, techniques, and metrics

The key tool for requirements traceability is the RDD-100 from Ascent Logic. In the past decade, RDD-100 has been mainly used in the aerospace, defense, and telecommunications industry. Today, its use is spreading to commercial industries engaging in systems engineering and information systems re-engineering projects. Among its many capabilities, RDD-100 can define requirements rigorously. In particular, the RDD-100 uses its Element/Relationship/Attribute repository to capture and trace complicated requirements. Each requirement is stored as a separate element, and graphical hierarchies show how the individual pieces of data relate to each other and trace back to their sources. [Ascent99]

There also exists requirements simulation tools. Foresight, produced by Nu Thena Systems, allows designers to capture system requirements and designs into graphical executable system models. These models can be analyzed and simulated to ensure requirements correctness. [Nu Thena99] The SES workbench is a system-level simulation software tool that focuses on behavioral and performance modeling. [SES99] Rational Rose is a visual modeling tool that among other features, allows UML modeling. [Rational99] The trend is towards executable requirements and simulations.

The Unified Modeling Language (UML) is an object-oriented modeling language for the specification of complex systems. UML represents a large effort from a number of methodologists to construct a common means of describing complex systems using object orientation. It is a specification language and not a development process so there are no details given on the features and how and when you link them together during systems development. UML is a relatively new language, and is still going through debate about the feasibility of using them in embedded systems. The main concerns about using an object oriented language for real-time embedded systems is about the speed and size of the application. Some points in support of object-oriented programming for embedded systems are that objects are effecient, developers can write larger systems with fewer defects in less time using OO methods instead of structured methods, and OO can be implemented in any language, including assembly language. [Douglass98] Other people have also stated arguments against using UML for embedded systems. In particular, restrictions in architectural modeling was cited as one shortfall. For example, there are no predefined node stereotypes to help improve standardization and the lack of these predefined stereotypes means there is no capability to capture information in depth to fully describe the operational properties of the system. Another shortfall deals with deficiencies in concurrency modeling and schedulability. However, UML is still relatively new and only time will tell if it is effective in the specification and development of embedded systems. [Moore98]

It would also be very helpful to have a technique that can assist in a structured development of correct and accurate requirements. In particular, Quality Function Deployment (QFD) is a method for the structured product planning and development that enables a development team to specify clearly the customer's wants and needs, and then evaluate each proposed product or service capability systematically in terms of its impact on meeting those needs. QFD is based on matrices that show the relationships between, for example, a customer need and a feature of the system. Figure 3 shows an example of a matrix that gives the relationship for each row and column. For example, say the rows defines

customer wants in a car. Lets say A is the car looks cool and B is the car never breaks. The columns can be specific features of the car. Lets say 1 is good gas mileage and b is aerodynamic styling. Each box represents the relationship between a customer want and a feature of the car. A car with good gas mileage is not related to the car looking cool, so their is no relation. However, a car with aerodynamic styling would look cool so their is a strong relation. Similarly, a car with that never breaks has a possible relation with good gas relation but not much relation with aerodynamic styling. [Cohen95]

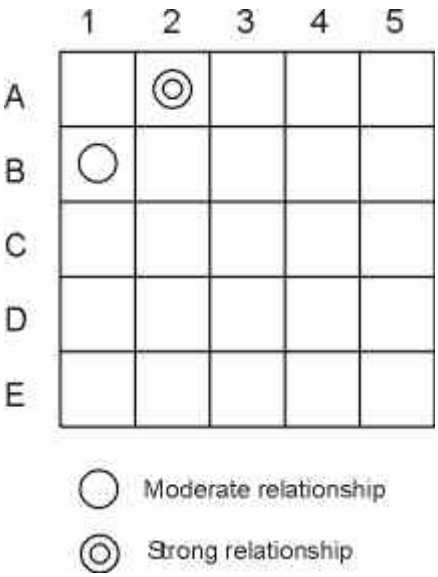


Figure 3: Example of matrix used in QFD

---

# Relationship to other topics

- **Standards** - Some requirements come from standards.
- **Verification/validation/certification** - It is necessay to be able to trace the implementation back to the requirements.
- **Social and legal concerns** - Whose responsibility is it when the cause of an accident traces back to the requirements and specifications?

---

# Conclusions

It is essential to establish correct requirements and specifications early in the development process to prevent errors later on in the system life cycle. However, we still need to realize the reality of requirements changing over time. There are several reasons why it is difficult to make good requirements. First, requirements

for embedded systems are even more complex than for other systems because they must interact with the outside world and not just other software components. Establishing correct requirements require people with both technical and communication skills. In addition, it is a tough abstraction problem and the implementation often gets mixed with the requirements. The complexity with establishing correct requirements makes it more of an art than an engineering skill. Often people have the attitude of "I'll know it when I see it," thus making it difficult to establish the requirements early. Over the past years, the area has mainly focused on tool development with the development of prototyping/execution tools, requirements traceability tools, and specifications/requirements "database" tools. The current trend is toward executable requirements/simulation tools such as Foresight.

---

# Annotated Reference List


- [Blanchard90] Blanchard, Benjamin S. and Fabrycky, Wolter J., *Systems Engineering and Analysis*, Englewood Cliffs, NJ: Prentice Hall, 1990.
- This book contains an overview of systems engineering. It contains information about the system design process, tools for systems analysis, design for operational feasibility, system engineering management, and system design applications. There is a useful section on the requirements definition stage.

- [Cohen95] Cohen, Lou, *Quality Function Deployment - How to Make QFD Work for You*, Reading, MA: Addison-Wesley, 1995.
- QFD is a structured planning process that systematically incorporates the voice of the customer into product design. This book explains QFD basics, advanced techniques, and examples. It gives guidance on implementing QFD from start to finish, and also covers its relationship to the product development cycle.

- [Dorfman90] Dorfman, Merlin and Thayer, Richard H., editors, *Standards, Guidelines, and Examples on System and Software Requirements Engineering*, Los Alamitos, CA: IEEE Computer Society Press, 1990.
- This book is a collection of international requirements standards, U.S. military standards, and requirements analysis methodologies and examples.

- [Douglass98] Douglass, Bruce Powel, "Designing Real-Time Systems with UML - Part 1," *Embedded Systems Programming*, March 1998.
- This article is the first in a 3 part series on how the unified modeling language (UML) can be used to develop real-time embedded systems. It focuses on UML's notational and semantic features.

- [Gause89] Gause, Donald C. and Weinberg, Gerald M., *Exploring Requirements - Quality Before Design*, New York: Dorset House Publishing, 1989.
- This is an entertaining book that looks at the most challenging part of the development process - establishing requirements. It contains a collection of ideas developed over many years and is not only applicable to software development but to all types of other product developments.

- [Moore98] Moore, Alan and Beckwith, Tony, "UML's shortfalls in modeling complex real-time systems," *Computer Design*, November 1998, pp. 53-57.
- This article deals with several shortfalls with using UML for modeling real-time systems, including restrictions in architectural modeling, and deficiencies in concurrency modeling and schedulability.

- [Ramesh95] Ramesh, B., Powers, T., Stubbs, C., and Edwards, M., "Implementing Requirements Traceability: A Case Study," *Proceedings of the Second IEEE International Symposium on Requirements Engineering*, York, United Kingdom, March 1995, pp. 89-95.
- This paper presents a case study of an organization using requirements traceability as a component of implementing a quality system engineering program. There is a description of a model that describes the traceability practice, the benefits of using it, and the lessons learned from implementing it.

- [Watkins94] Watkins, Robert and Neal, Mark, "Why and How of Requirements Tracing," *IEEE Software*, July 1994, pp. 104-106.
- This article gives an example of how requirements traceability is used at Abbott Laboratories Diagnostics Division in developing an embedded real-time in vitro diagnostic instrument.

- [Ascent99] Ascent Logic website, http://www.alc.com, accessed May 6, 1999.
- Produces a tool for requirements traceability.
- [Nu Thena99] Nu Thena Systems website, http://www.nuthena.com, accessed May 6, 1999.
- Produced a requirements simulation tool.
- [Rational99] Rational Software website, http://www.rational.com, accessed May 6, 1999.
- Produced a requirements simulation tool.
- [SES99] Scientific and Engineering Software website, http://www.ses.com, accessed May 6, 1999.
- Produced a requirements simulation tool.


# Further Reading


- Gause, Donald C. and Weinberg, Gerald M., *Exploring Requirements - Quality Before Design*, New York: Dorset House Publishing, 1989.
- This is an entertaining book that looks at the most challenging part of the development process - establishing requirements. It contains a collection of ideas developed over many years and is not only applicable to software development but to all types of other product developments.
- Lattemann, Frank and Lehmann, Eckard, "A methodological Approach to the Requirement Specification of Embedded Systems," *Proceedings of the First IEEE International Conference on Formal Engineering Methods*, Hiroshima, Japan, November 1997, pp.183-191.
- This paper presents a systematic development process for describing requirement specifications of embedded safety critical systems.
- Patridge, Derek, "Where do Specifications Come From?," *Achievement and Assurance of Safety - Proceedings of the Third Safety-Critical Systems*

*Symposium*, Brighton, United Kingdom, February 1995, pp. 302-310.

- This paper presents different interpretations of how "specifications" is defined as there are various definitions and no one definition is the best.

- Ramesh, B., Powers, T., Stubbs, C., and Edwards, M., "Implementing Requirements Traceability: A Case Study," *Proceedings of the Second IEEE International Symposium on Requirements Engineering*, York, United Kingdom, March 1995, pp. 89-95.
- This paper presents a case study of an organization using requirements traceability as a component of implementing a quality system engineering program. There is a description of a model that describes the traceability practice, the benefits of using it, and the lessons learned from implementing it.

---

# Loose Ends

---

Go To Project Page