Design for Electrical and Computer Engineers

Theory, Concepts, and Practice

Ralph M. Ford and Christopher S. Coulston

This document was prepared with LaTeX.

0.1 About the Authors

Ralph Ford obtained his Ph.D. and M.S. degrees in Electrical Engineering from the University of Arizona in 1994 and 1989 respectively. He obtained his B.S. in Electrical Engineering from Clarkson University in 1987. He worked for the IBM Microelectronics Division in East Fishkill, NY from 1989-1991, where he developed machine vision systems to inspect electronic packaging modules for mainframe computers. Ralph also has experience working for IBM Data Systems and the Brookhaven National Laboratory. He joined the faculty at Penn State Erie, The Behrend College in 1994. Ralph has experience teaching electronics and software design, as well as teaching the capstone design course sequence in the electrical, computer, and software engineering programs. His research interests are in engineering design, image processing, machine vision, and signal processing. Ralph is currently Director of the School of Engineering at Penn State Behrend. He also serves as a program evaluator for ABET. He was awarded a Fulbright Scholarship to study at the Brno University of Technology in the Czech Republic in 2005.

Chris Coulston obtained his Ph.D. in Computer Science and Engineering from the Pennsylvania State University in 1999. He obtained his M.S. and B.S in Computer Engineering from the Pennsylvania State University in 1994 and 1992 respectively. Chris has industry experience working for IBM in Manassas, VA and Accu-Weather in State College, PA. He joined the faculty at Penn State Erie, The Behrend College in 1999. He has experience teaching design-oriented courses in digital systems, embedded systems, computer architecture, and database management systems.

Chris' research interests are in Steiner tree routing algorithms and artificial life. He is currently an Associate Professor of Electrical and Computer at Penn State Behrend and also serves as Chairperson of the program.

Contents

0.1 About the Authors . . . iii
0.2 Preface . . . vi

1 Concept Generation and Evaluation 1
1.1 Creativity . . . 2
1.1.1 Barriers to Creativity . . . 3
1.1.2 Vertical and Lateral Thinking . . . 5
1.1.3 Strategies to Enhance Creativity . . . 7
1.2 Concept Generation . . . 9
1.2.1 Concept Evaluation . . . 13
1.2.2 Initial Evaluation . . . 13
1.2.3 Strengths and Weaknesses Analysis . . . 15
1.2.4 Analytical Hierarchy Process and Decision Matrices . . . 15
1.2.5 Pugh Concept Selection . . . 18
1.3 Project Application: Concept Generation and Evaluation . . . 19
1.4 Summary and Further Reading . . . 20
1.5 Problems . . . 21

2 System Design I: Functional Decomposition 23
2.1 Bottom-Up and Top-Down Design . . . 24
2.2 Functional Decomposition . . . 25
2.3 Guidance . . . 27
2.4 Application: Electronics Design . . . 28
2.5 Application: Digital Design . . . 34
2.6 Application: Software Design . . . 37
2.7 Application: Thermometer Design . . . 40
2.8 Coupling and Cohesion . . . 46
2.9 Project Application: The Functional Design . . . 48
2.10 Summary and Further Reading . . . 49
2.11 Problems . . . 51

0.2 Preface

This book is written for undergraduate students and teachers engaged in electrical and computer engineering (ECE) design projects, primarily in the senior year. The objective of the text is to provide a treatment of the design process in ECE with a sound academic basis that is integrated with practical application. This combination is necessary in design projects because students are expected to apply their theoretical knowledge to bring useful systems to reality. This topical integration is reflected in the subtitle of the book: Theory, Concepts, and Practice. Fundamental theories are developed whenever possible, such as in the chapters on functional design decomposition, system behavior, and design for reliability. Many aspects of the design process are based upon time-tested concepts that represent the generalization of successful practices and experience. These concepts are embodied in processes presented in the book, for example, in the chapters on needs identification and requirements development. Regardless of the topic, the goal is to apply the material to practical problems and design projects. Overall, we believe that this text is unique in providing a comprehensive design treatment for ECE, something that is sorely missing in the field. We hope that it will fill an important need as capstone design projects continue to grow in importance in engineering education.

We have found that there are three important pieces to completing a successful design project. The first is an understanding of the design process, the second is an understanding of how to apply technical design tools, and the third is successful application of professional skills. Design teams that effectively synthesize all three tend to be far more successful than those that don't. The book is organized into three parts that support each of these areas.

The first part of the book, the *Design Process*, embodies the steps required to take an idea from concept to successful design. At first, many students consider the design process to be obvious. Yet it is clear that failure to understand and follow a structured design process often leads to problems in development, if not outright failure. The design process is a theme that is woven throughout the text; however, its main emphasis is placed in the first four chapters. Chapter 1 is an introduction to design processes in different ECE application domains. Chapter 2 provides guidance on how to select projects and assess the needs of the customer or user. Depending upon how the design experience is structured, both students and faculty may be faced with the task of selecting the project concept. Further, one of the important issues in the engineering design is to understand that

systems are developed for use by an end-user, and if not designed to properly meet that need, they will likely fail. Chapter 3 explains how to develop the Requirements Specification along with methods for developing and documenting the requirements. Practical examples are provided to illustrate these methods and techniques. Chapter 4 presents concept generation and evaluation. A hallmark of design is that there are many potential solutions to the problem. Designers need to creatively explore the space of possible solutions and apply judgment to select the best one from the competing alternatives.

The second part of the book, *Design Tools*, presents important technical tools that ECE designers often draw upon. Chapter 5 emphasizes system engineering concepts including the well known functional decomposition design technique and applications in a number of ECE problem domains. Chapter 6 provides methods for describing system behavior, such as flowcharts, state diagrams, data flow diagrams and a brief overview of the Unified Modeling Language (UML). Chapter 7 covers important issues in testing and provides different viewpoints on testing throughout the development cycle. Chapter 8 addresses reliability theory in design, and reliability at both the component and system level is considered.

The third part of the book focuses on *Professional Skills*. Designing, building, and testing a system is a process that challenges the best teams, and requires good communication and project management skills. Chapter 9 provides guidance for effective teamwork. It provides an overview of pertinent research on teaming and distills it into a set of heuristics. Chapter 10 presents traditional elements of project planning, such as the work breakdown structure, network diagrams, and critical path estimation. It also addresses how to estimate manpower needs for a design project. Chapter 11 addresses ethical considerations in both system design and professional practice. Case studies for ECE scenarios are examined and analyzed using the IEEE (Institute of Electrical and Electronics Engineers) Code of Ethics as a basis. The book concludes with Chapter 12, which contains guidance for students preparing for oral presentations, often a part of capstone design projects.

Features of the Book

This book aims to guide students and faculty through the steps necessary for the successful execution of design projects. Some of the features are listed below.

- Each chapter provides a brief motivation for the material in the chapter followed by specific learning objectives.

- There are many examples throughout the book that demonstrate the application of the material.
- Each end-of-chapter problem has a different intention. Review problems demonstrate comprehension of the material in the chapter. Application problems require the solution of problems based upon the material learned in the chapter. Design problems are directly applicable to design projects and are usually tied in with the Project Application section.
- Nearly all chapters contain a Project Application section that describes how to apply the material to a design project.
- Some chapters contain a Guidance section that represents the author's advice on application of the material to a design project.
- Checklists are provided for helping students assess their work.
- There are many terms used in design whose meaning needs to be understood. The text contains a glossary with definitions of design terminology. The terms defined in the glossary (Appendix A) are indicated by ***italicized-bold*** highlighting in the text.
- All chapters conclude with a Summary and Further Reading section. The aim of the Further Reading portion is to provide pointers for those who want to delve deeper into the material presented.
- The book is structured to help programs demonstrate that they are meeting the ABET (accreditation board for engineering programs) accreditation criteria. It provides examples of how to address constraints and standards that must be considered in design projects. Furthermore, many of the professional skills topics, such as teamwork, ethics, and oral presentation ability, are directly related to the ABET Educational Outcomes. The requirements development methods presented in Chapter 3 are valuable tools for helping students perform on cross-functional teams where they must communicate with non-engineers.
- An instructor's manual is available that contains not only solutions, but guidance from the authors on teaching the material and managing student design teams. It is particularly important to provide advice to instructors since teaching design has unique challenges that are different than teaching engineering science oriented courses that most faculty are familiar with.

- PowerPoint™ presentations are available for instructors through McGraw-Hill
- There are a number of complete case study student projects available in electronic form for download by both students and instructors and available at. These projects have been developed using the processes provided in this book.

How to Use this Book

There are several common models for teaching capstone design, and this book has the flexibility to serve different needs. Particularly, chapters from the Professional Skills section can be inserted as appropriate throughout the course. Recommended usage of the book for three different models of teaching a capstone design course is presented.

- **Model I.** This is a two-semester course sequence. In the first semester, students learn about design principles and start their capstone projects. This is the model that we follow. In the first semester the material in the book is covered in its entirety. The order of coverage is typically Chapters 1–3, 9, 4–6, 10–11, and 7–8. Chapter 9 (Teams and Teamwork) is covered immediately after the projects are identified and the teams are formed. Chapters 10 (Project Management) and 11 (Ethical and Legal Issues) are covered after the system design techniques in Chapters 5 and 6 are presented. Students are in a good position to create a project plan and address ethical issues in their designs after learning the more technical aspects of design. Chapter 12 (Oral Presentations) is assigned to students to read before their first oral presentation to the faculty. The course concludes with principles of testing and system reliability (Chapter 7 and 8). We assign a good number of end-of-chapter problems and have quizzes throughout the semester. By the end of the first semester, design teams are expected to have completed development of the requirements, the high-level or architectural design, and developed a project plan. In the second semester, student teams implement and test their designs under the guidance of a faculty advisor.
- **Model II**. This two-semester course sequence is similar to Model I with the difference being that the first semester is a lower credit course (often one credit) taught in a seminar format. In this model chapters can be selected to support the projects. Some of the core chapters for consideration are Chapters 1–5, which take the student from project

selection to functional design, and Chapters 9–11 on teamwork, project management, and ethical issues. Other chapters could be covered at the instructor's discretion. The use of end-of-chapter problems would be limited, but the project application sections and example problems in the text would be useful in guiding students through their projects.

- **Model III**. This is a one-semester design sequence. Here, the book would be used to guide students through the design process. Chapters for consideration are 1–5 and 9–10, which provide the basics of design, teamwork, and project management. The project application sections and problems could be used as guidance for the project teams.

Acknowledgements

Undertaking this work has been a challenging experience and could not have been done without the support of many others. First, we thank our families for their support and patience. They have endured many hours and late evenings that we spent researching and writing. Melanie Ford is to be thanked for her diligent proofreading efforts. Bob Simoneau, the former Director of the Penn State Behrend School of Engineering, has been a great supporter of the book and has also lent his time in reading and providing comments. Our school has a strong design culture, and this book would not have happened without that emphasis; our faculty colleagues need to be recognized for developing that culture. Jana Goodrich and Rob Weissbach are two faculty members with whom we have collaborated on other courses and projects. They have influenced our thinking in this book, particularly in regard to project selection, requirements development, cost estimation, and teamwork. We must also recognize the great collaborative working environment that exists at Penn State Behrend, which has allowed this work to flourish. Our students have been patient in allowing us to experiment with different material in the class and on the projects. Examples of their work are included in the book and are greatly appreciated. John Wallberg contributed the disk drive diagnostics case study in Chapter 11 that we have found very useful for in-class discussions. John developed this while he was a student at MIT. Thanks to Anne Maloney for her copyediting of the manuscript. The following individuals at McGraw-Hill have been very supportive and we thank them for their efforts to make this book a reality – Carlise Stembridge, Julie Kehrwald, Darlene Schueller, Craig Marty, Kris Tibbetts, and Mike Hackett.

Finally, we would like to thank the external reviewers of the book for their thorough reviews and valuable ideas. They are Frederick C. Berry

(Rose-Hulman Institute of Technology), Mike Bright (Grove City College), Geoffrey Brooks (Florida State University Panama City Campus) Wils L. Cooley (West Virginia University), D. J. Godfrey (US Coast Guard Academy), and Michael Ruane (Boston University).

We hope that you find this book valuable, and that it motivates you to create great designs. We welcome your comments and input. Please feel free to email us.

Ralph M. Ford,

Chris S. Coulston,

Chapter 1

Concept Generation and Evaluation

> *Creativity is a great motivator because it makes people interested in what they are doing. Creativity gives hope that there can be a worthwhile idea. Creativity gives the possibility of some sort of achievement to everyone. Creativity makes life more fun and more interesting.—Edward DeBono*

When developing a design, it is important to explore many potential solutions and select the best one from them. Too often a single concept is generated and is the only one pursued, the unfortunate result being that potentially better solutions are not considered. When confronted with a problem, engineers must explore different concepts, critically evaluate them, and be able to defend the decisions that led to a particular solution. Two key thought processes employed are creativity and judgment. Creativity involves the generation of novel concepts, while judgment is applied to evaluate and select the best solution for the problem. Creativity and judgment appear to be inherent individual qualities that can't be taught. That is to some extent true, but with practice and application of formal techniques, they can be improved.

It is important to distinguish between innovation and creativity. Creativity refers to the ability to develop new ideas, while innovation is the ability to bring creative ideas to reality. Innovation is valued by companies since new products and services are often their lifeblood. That is why many make it a priority to hire engineers who can bring creativity to the design process. This chapter addresses creativity, concept generation, and evaluation in design. The first part describes barriers to creative thought,

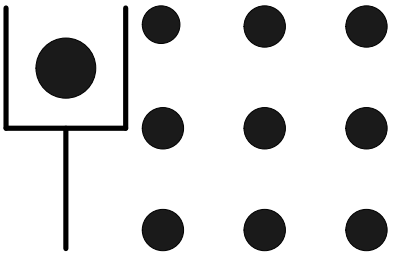


Figure 1.1: (a) The shovel problem. Think of this as a shovel with a coin on the spade. The objective is to move two lines so that the coin is no longer in the spade, but there is still a shovel. (b) The nine dot problem. Draw four connected straight lines that pass through all nine dots.

followed by strategies for overcoming them and enhancing creativity. Next, methods for concept generation are presented, followed by techniques for concept evaluation.

Learning Objectives

By the end of this chapter, the reader should:

- Understand the importance of creativity, innovation, concept generation, and concept evaluation in engineering design.
- Be familiar with the barriers that hinder creativity.
- Be able to apply strategies and formal methods for concept generation.
- Be able to apply techniques for the evaluation of design concepts.

1.1 Creativity

Is creativity something that is inherent in the individual or something that can be learned? It appears that both are true; some individuals are naturally more creative than others, yet people can enhance their creativity with conscious effort and practice. This section examines barriers to creativity, different thinking modes, and strategies for enhancing creativity. One of the ways to spark creativity is to solve puzzles. To get into the creative spirit the reader can try to solve the puzzles presented in Figure 3.

1.1.1 Barriers to Creativity

James L. Adams, an engineer and former professor at Stanford University, has researched innovation in technical domains. He examined the barriers to creativity and classified them into the following four types: 1) perceptual blocks, 2) emotional blocks, 3) cultural and environmental blocks, and 4) intellectual and expressive blocks [Ada01].

Perceptual blocks are those that prevent people from clearly seeing the problem for what it is. A common perceptual block is the tendency to delimit the problem space, or in other words, to put constraints on the problem that don't exist. Have you solved the puzzles shown in Figure 3 yet? If not, it is possible that you are placing constraints on the problems that don't exist. Knowing that this is the case, you may want to go back and try again. Another example of a perceptual block is the tendency to stereotype or see a solution to a problem that one is biased to see. This occurs because we have used similar techniques for solving the problem in the past. For example, if you have used a microcontroller to solve a certain type of problem, chances are that you are going to consider using a microcontroller in all related problems in the future. Another perceptual block is the difficulty of isolating the true problem. Three pictures that illustrate this are shown in Figure 1.2. When examining these images, people tend to form a conclusion as to what the content of each one is. Look carefully, as each picture has two equally valid interpretations.

One of the most common *emotional blocks* is the fear of failure. People often have creative ideas, but are afraid to express them since they may be criticized or may not have the "correct" answer. It is cliché to hear that you must fail often to succeed, but true. The highly successful product design company, IDEO, that was examined in Chapter 2 takes the approach in concept generation to "*fail early and often*" in order to succeed [Kel01]. Their design teams are encouraged to develop many seemingly outlandish ideas that are often discarded, but sometimes lead to innovative solutions. Another emotional block is a fear of chaos and disorganization. The creative process challenges engineers, since it is disorganized and not a neat scientific approach to which they are accustomed. Another block is the tendency to critically judge ideas, rather than generate and build upon them. Finally, it takes time for creative ideas to incubate. Most of us can relate to the experience when we could not solve a problem that nagged us for a period of time, followed by that unexpected *"Aha!"* moment when we identified the solution.

Environmental blocks refer to those things in our environment that limit

Figure 1.2: Each of the images shown above has two different interpretations. Can you determine what they are?

creative ability. This could be in the form of poor teamwork where members distrust each other and criticize each other's ideas. In the workplace, this could be due to autocratic management that resists new ideas. There are also cultural biases against creativity. There is a bias against creativity as an approach to problem solving in the engineering field. This is usually based upon the reasoning that there is a single correct solution to a problem and creativity is an excuse for poor engineering. It is true that creativity and brainstorming alone do not solve engineering problems—the concepts generated need to be scrutinized using engineering principles to become viable innovations.

The final block that Adams identified is that of *intellectual and expressive*. In an engineering context, this means that the designer needs to have an understanding of intellectual tools that are applied to solve problems. For example, mathematics is a universal language for expressing and solving scientific problems. Specific examples in ECE are languages that describe the characteristics of systems such as functional, logical, and state behaviors. Examples in digital design are truth tables (input, output behavior) and state diagrams (stimulus-response). In the domain of electronics design, a functional approach (input, output, and function) is commonly used. Chapters 5 and 6 present tools for modeling the behavior of ECE systems.

1.1.2 Vertical and Lateral Thinking

Edward DeBono is the father of a field known as ***lateral thinking***, which offers a different perspective on the barriers to creativity. Lateral thinking is contrasted to what is known as the vertical thinking process [Deb67, Deb70]. Engineers tend to be vertical (or convergent) thinkers, meaning that they are good at taking a problem and proceeding logically to the solution. This is typically a sequential linear process, where the engineer starts at the highest level and successively refines elements of the design to solve the problem. This is usually based upon experience solving similar problems and conventional tools that are employed in that particular area.

The objective of lateral (or divergent) thinking is to identify creative solutions. It is not concerned with developing the solution for the problem, or right or wrong solutions. It encourages jumping around between ideas. In the words of DeBono "*The vertical thinker says: 'I know what I am looking for.' The lateral thinker says: 'I am looking but I won't know what I am looking for until I have found it.'* " The field of lateral thinking is characterized by puzzles of the following type found at Paul Sloane's Lateral Thinking Puzzles website [`http://dspace.dial.pipex.com/sloane`]:

A body is discovered in a park in Chicago in the middle of summer. It has a fractured skull and many other broken bones, but the cause of death was hypothermia.

A hunter aimed his gun carefully and fired. Seconds later, he realized his mistake. Minutes later, he was dead.

A man is returning from Switzerland by train. If he had been in a non-smoking car he would have died.

The objective in these puzzles is to develop plausible scenarios that explain how each of the above situations could have happened. A solution for the first example is that a person stowed away in the wheel compartment of a jet airliner. While in flight he froze and died of hypothermia. When the plane prepared to land, it lowered its landing gear, causing the body to fall to the park, fracturing his skull, and breaking his bones. Can you develop plausible scenarios that describe each of them?

Vertical thinking focuses on sequential steps toward a solution and tries to determine the correctness of the solution throughout the process. This is very different from lateral thinking where there is nonlinear jumping around between steps and there is no attempt to discern between right and wrong. As such, lateral thinking is more apt to follow least likely paths to a solution, whereas vertical thinking follows the most likely paths. The goal in lateral thinking is to develop as many solutions as possible, while vertical thinking tries to narrow to a single solution.

Lateral thinking is appropriate for the concept generation phase. So should concept generation and brainstorming be done by the individual or by a team? DeBono and Osborn [Osb63] conclude that creativity is more effective by individuals than by teams. However, Osborn also points out that there is great value in applying creativity in teams, since it provides a place for the team to work together on problems and see other perspectives. Our anecdotal observations of student design teams supports this—group brainstorming is effective for developing concepts, new product ideas, new features, and different ways to combine technologies. This is because in groups, ideas are readily built upon by other team members. More mathematical, technical, and theoretical breakthroughs tend to be the work of the lone genius. Examples of this are the Theory of Relativity (Einstein),

Boolean Logic (Bool), and Shannon's Sampling Theorem. We have also observed that novice designers, who do not have much experience in concept generation, can benefit greatly from group brainstorming techniques.

1.1.3 Strategies to Enhance Creativity

There are valuable strategies that can be employed to enhance the creative process. The body of research on the subject is very large and key points are summarized as follows:

- *Have a questioning attitude.* One of the keys is to have a questioning attitude and challenge assumptions. The willingness to do this generally decreases as people age. Young children are highly creative and are constantly questioning everything, with questions such as "*Why do trees have leaves?*" or "*Why is the sky blue?*" Asking basic questions stimulates creativity and is applicable to technical designs. When examining a design with a microcontroller, ask questions such as "*Is there a way to replace the microcontroller?*", "*Are there other features that I can achieve with the microcontroller?*", and "*Is there a better microcontroller that can be used?*"

- *Practice being creative.* Research shows that people can improve their creative ability through conscious effort. For example, try solving the puzzles presented in this chapter and in the end of the chapter problems. Be conscious of things that bother you ("pet peeves") in your everyday life and try to develop new solutions for them.

- *Suspend judgment.* It is easy to criticize and immediately dismiss ideas, so it is important to defer judgment and be flexible in thinking. Seemingly outlandish ideas can lead to other concepts that are valuable solutions. The opportunity for new solutions is curtailed if ideas are immediately judged and discarded. Creative concepts can be developed by taking a concept and modifying it or combining it with other seemingly unrelated concepts.

- *Allow time.* The creative process needs time for incubation. The human mind needs time to work on problems, so set aside time to reflect on the problem and to allow it to incubate so that the "*Aha!*" moment of discovery can happen.

- *Think like a beginner.* New solutions often come from novices. The reason is that novices don't have preconceived ideas as to the solution

for a problem. Experience is a double-edged sword—it allows one to quickly solve problems by drawing upon pre-existing solutions, but can inhibit creativity. If confronted with a new problem that bears similarity to one encountered in the past, then it is likely that the new solution will bear similarity to the old one. If everyone else is it doing it one way, consider the opposite.

Many creative ideas arise from novel combinations and adaptations of existing technology. SCAMPER, an acronym for Substitute, Combine, Adapt, Modify, Put to other use, Eliminate, and Rearrange/Reverse, can be used as a guide to systematically generate creative concepts. The SCAMPER principles are valuable in brainstorming and are described below:

- *Substitute.* Can new elements be substituted for those that already exist in the system?
- *Combine.* Can existing entities be combined in a novel way that has not been done before?
- *Adapt.* Can parts of the whole be adapted to operate differently?
- *Modify.* Can part or all of a system be modified? For example, size, shape, or functionality.
- *Put to other use.* Are there other application domains where the product or system can be put to use?
- *Eliminate.* Can parts of the whole be eliminated? Or should the whole itself be eliminated?
- *Rearrange or Reverse.* Can elements of the system be rearranged differently to work better? This is different from substituting in that the elements of the system are not changed, but rearranged or ordered differently to create something new. In terms of reversal, are there any roles or objectives that can be reversed?

SCAMPER is a modification of a set of questions that was originally posed by Osborn [Osb63] and was modified to its form above by Michalko [Mic91].

1.2 Concept Generation

After the problem is defined, the next step is to explore concepts for the solution. It is unlikely that a design team will have reached this stage without some ideas for solving the problem, but it is important to fully explore the design space. Ullrich and Eppinger [Ull03] identify the following phases of concept generation – search internally, search externally, and systematically explore. Each is considered in turn.

External searching was covered to a great extent in Chapters 2 and 3, which addressed conducting background research and benchmarking. Methods of external searching are:

- Conduct literature search.
- Search and review existing patents.
- Benchmark similar products.
- Interview experts.

Internal searching is done by the team members via methods such as brainstorming. The team members need have to have a common problem definition for this to be effective. Understanding the tradeoffs using requirement analysis methods in Chapter 3 is also valuable, as overcoming tradeoffs leads to innovative solutions. Furthermore, the team should decompose larger problems into sub-problems and the attack the sub-problems individually. Chapter 5 addresses the process of problem decomposition.

DILBERT® by Scott Adams

The most well-known method of internal searching is ***brainstorming***. Group brainstorming is effective for generating many concepts in a short period of time. Experienced design teams are known to generate hundreds of concepts in an hour. Traditional brainstorming is not highly-structured—though a facilitator helps—and employs five basic rules:

- No criticism or judgment of ideas.
- Wild ideas are encouraged.
- Quantity is stressed over quality.
- Build upon and modify the ideas of others.
- All ideas are recorded.

Figure 1.3: Wally brainstorming. (Dilbert © United Feature Syndicate. Reprinted by permission.

Many novice design teams struggle with unstructured brainstorming and more formalized approaches, such as brainwriting and the Nominal Group Technique, can be of benefit. The steps of ***brainwriting*** are:

1. The team develops a common problem statement that is read out loud.
2. Each team member writes their ideas down on a card and places it in the center of the table.
3. Other team members then take cards from the pile and use other's ideas to generate new ones or build upon them, keeping in mind the principles of SCAMPER. Alternatively, members can each generate an idea, write it on a card, and then pass it to another team member. Each member then builds upon the idea passed to them.

Brainwriting 6-3-5 is a variation where the objective is to have six people, develop three ideas in five minutes. The optimal number of people for the exercise is thought to be six, although it is not necessary. Each person generates three ideas in five minutes, and clearly describes it using sketches and written descriptions on paper. At the end of five minutes, each team member passes their ideas to another team member. The next person reviews the ideas of their teammate and adds three more by building on them, developing new ones, or ignoring as necessary. This process continues until all members have reviewed all papers.

In the ***Nominal Group Technique*** (NGT) [Del71] each team member silently generates ideas that are reported out in a round-robin fashion so

that all members have an opportunity to present their ideas. Concepts are selected by a multi-voting scheme with each member casting a predetermined number of votes for the ideas presented. The ideas are then ranked, discussed further, and voted upon again if necessary. The steps of NGT are as follows:

- *Read problem statement.* It should be read out loud by a team member (the facilitator).
- *Restate the problem.* Each person restates the problem in their own words to ensure that all members understand it.
- *Silently generate ideas.* All members silently generate ideas during a set period of time, typically 5–15 minutes.
- *Collect ideas in a round-robin fashion.* Each person presents one idea in turn until all ideas are exhausted. The facilitator should clarify ideas and all should be written where the entire team can view them.
- *Summarize and rephrase ideas.* Once the ideas are collected, the facilitator leads a discussion to clarify and rephrase the ideas. This ensures that the entire group is familiar with them. Related ideas can be grouped or merged together.
- *Vote.* Each person casts a predetermined number of votes, typically three to six, for the ideas presented. The outcome is a set of prioritized ideas that the team can further discuss and pursue.

To systematically generate concepts, the problem is decomposed sub-functions and solutions are sought for the sub-functions. A ***concept table***, demonstrated in Table 2.1, is a tool for identifying different combinations, arrangements, and substitutions. The table headings identify functions to be achieved in the design, while the entries in the corresponding column represent potential solutions. Novel products or solutions are generated by combining elements from each of the columns, which are identified in the table by circled elements. The solutions can be in the form of a single element selected from each column, or as in the example shown, multiple elements selected from each column.

Based on the concepts circled in Table **??**, one can imagine a personal computing system that has the following features: 1) is wearable with different credit card size components placed on the body and in clothing to make it comfortable to use; 2) is powered by a combination of solar cells, fuel-cells, and from thermal heat generated by a person's body; 3) has a microphone

Table 1.1: A concept table for generating ideas for a personal computing system. The potential solution is identified by the combination of circled elements.

User Interface	**Display**	**Connectivit & Expansion**	**Power**	**Size**
Keyboard	CRT	Serial & Parallel	Battery	Hand-held, Fits in pocket
Touchpad	Flat Panel	USB	AC Power	Notebook size
Handwriting Recognition	Plasma	Wireless Ethernet	Solar Power	Wearable
Video	Heads-up Display	Wired Ethernet	Fuel Cell	Credit Card Size
Voice	LCD	PCMCIA	Thermal Transfer	Flexible in shape
		Modem / Telephone		

and camera integrated in the user's clothing for interface to the system, as well as a flexible foldable keyboard for typing that is stored in a pocket; 4) has a heads-up display integrated with the user's eyeglasses or baseball hat; and 5) has a miniature earpiece microphone used for communication. While the above example focused on novel combinations and substitutions, the concept table can also be used to examine the possibility of eliminating ideas. For example, the table inherently assumes that a display will be used. However, it should also be asked if it is absolutely necessary in the design.

Another example is shown in Table 1.2, where the objective is to identify design concepts for a temperature measurement and display device. There are three main elements to the proposed solution: the thermal sensing method, circuitry that converts the sensor information (temperature) to a voltage, and a display unit that converts the voltage to a displayed temperature. Note that the table implies a three-stage architecture, thus concepts are generated within that framework. There may be completely different architectures that are better.

A related tool is a ***concept fan***, which is a graphical representation of design decisions and choices. An example concept fan for the temperature

Table 1.2: Concept table for a temperature measurement device.

Thermal Sensing	Conversion to Voltage	Display
Thermistor	Op Amp Design	Seven-Segment LEDs
RTD	Transistor Designs	LCD
Thermocouple		Analog Dial Indicator

measuring device is shown in Figure 2.9. Design decisions are identified by circles; solutions are indicated by squares. In this example, more options are shown than in Table 1.2. Concepts are generated by selecting among the different solution blocks.

1.2.1 Concept Evaluation

The concepts generated are evaluated to determine which are the most promising to pursue. The designer should exercise engineering judgment and use the customer needs and technical factors to drive the decision. This process is shown in Figure 2.9, where the user needs, concepts, and engineering consideration serve as inputs to a decision process to ranks the concepts. A point of caution—some of the methods presented generate numerical scores for comparing concepts, leading one to potentially believe that the quantitative results are infallible. Keep in mind that the inputs are based on qualitative and semi-quantitative assessments and can be geared to select a preconceived notion of the solution. It is important to maintain flexibility of thinking, to challenge assumptions, and ultimately determine the best concept.

1.2.2 Initial Evaluation

The concepts generated should be initially reviewed and those that are completely infeasible discarded. Some of the reasons a concept may be deemed infeasible are that it may be far too costly, will take too long to develop, or involve too much risk. In many cases it may be deemed that using cutting-edge technology represents an unacceptable risk. Concepts that clearly cannot meet the engineering requirements should also be discarded. Care should be taken not to completely eliminate ideas that may have merit, as conditions change and some concepts that were previously thought unrealistic may become viable in the future.

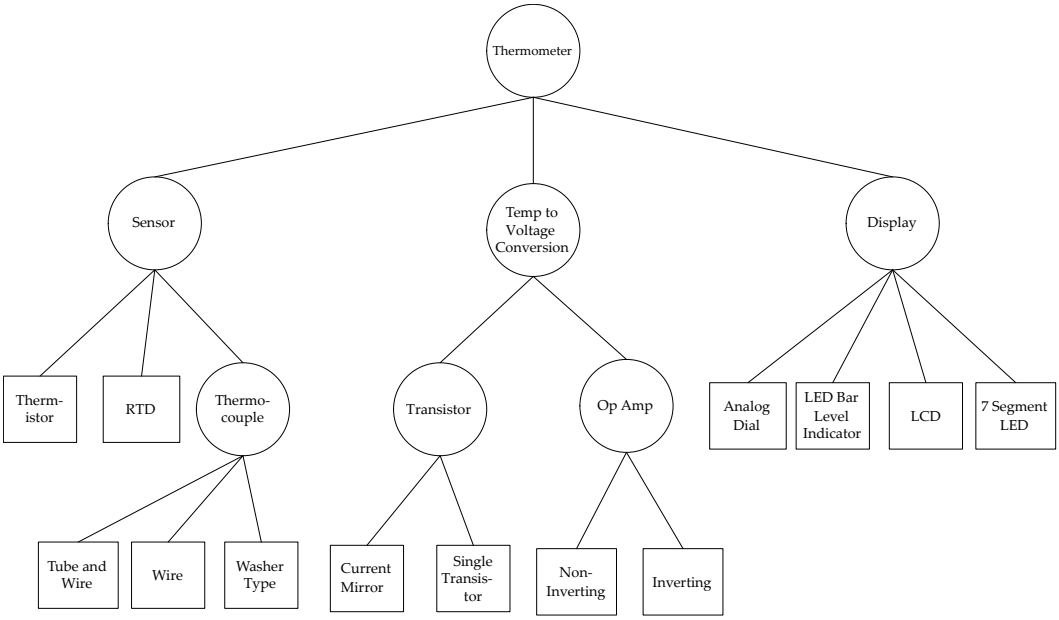


Figure 1.4: A concept fan for the temperature measuring device. The circles represent the choices to be made and the squares represent potential solutions to the choices.

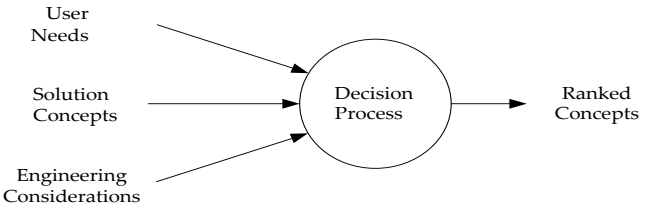


Figure 1.5: Process for concept evaluation.

Table 1.3: A strengths and weaknesses analysis of proposed methods for heating an Intel 1000XF card to be used in lifetime testing. [Ese03].

Method	Strengths	Weaknesses
Contact Heating	• Simplest design • Could be used internally to computer	• Does not create uniform temperature • Hard to control temperature
Temperature Chamber	• Uniform temperature • Greater control over temperature	• Must be external to computer • More difficult to design • Expensive

1.2.3 Strengths and Weaknesses Analysis

Another form of evaluation is to complete a ***strengths and weaknesses analysis*** of the potential solutions. Table 1.3 demonstrates the application of this analysis applied to an experimental design project for testing of the Intel 1000XF card (examined in Chapter 2 (Example 2.2) and Chapter 3 (Table 3.2)). In order to test the card under different operating temperatures, a method of heating the card and holding its temperature fixed during the experiment was needed. The two solutions compared were to use a contact heating element or to place the card in an environmental test chamber. In this particular example, the temperature chamber solution was ultimately selected due to the need for a uniform temperature distribution. The strength and weakness analysis is good for examining problems of moderate complexity. It suffers in that it does not require uniform criteria for comparison. To make the method more quantitative, relative scores for the strengths (plus factors) and weaknesses (minus factors) can be assigned and used to score the concepts.

1.2.4 Analytical Hierarchy Process and Decision Matrices

In the Analytical Hierarchy Process, design alternatives are compared against pre-selected criteria, such as the engineering or marketing requirements.

Table 1.4: A decision matrix for the Analytical Hierarchy Process.

		Design Option 1	**Design Option 2**		**Design Option n**
Criteria 1	ω_1	α_{11}	α_{12}	...	α_{1n}
Criteria 2	ω_2	α_{21}	α_{22}	...	α_{2n}
Criteria *m*	ω_m	α_{m1}	α_{m2}	...	α_{mn}
Score		$S_1 = \sum_{i=1}^{m} \omega_i * \alpha_{i1}$	$S_2 = \sum_{i=1}^{m} \omega_i * \alpha_{i2}$	...	$S_n = \sum_{i=1}^{m} \omega_i * \alpha_{im}$

Table 1.5: Pairwise comparison matrix.

	Accuracy	**Cost**	**Size**	**Availability**	**Weights**
Accuracy	1	5	3	$\frac{1}{4}$	0.42
Cost	1/5	1	2	$\frac{1}{4}$	0.12
Size	1/3	$\frac{1}{2}$	1	1	0.12
Availability	4	4	1	1	0.34

AHP is covered in detail in Appendix B and was first applied in Chapter 2 for project selection. The reader is encouraged to review Appendix B as necessary. The end result of AHP is a decision matrix is shown in Table 1.4, where the criteria are listed in the leftmost column with the associated weighting factors (ω_i) quantifying the relative importance of the criteria. The body of the matrix contains design ratings, α_{ij}, that reflect the technical merit of each of the j^{th} design options relative to i^{th} criterion. The total score, S_j, for each design option is computed as a weighted summation of the design ratings and weighting factors.

The application of AHP is demonstrated for the design of an electronic circuit for measuring temperature, by producing a voltage signal that is directly proportional to temperature.

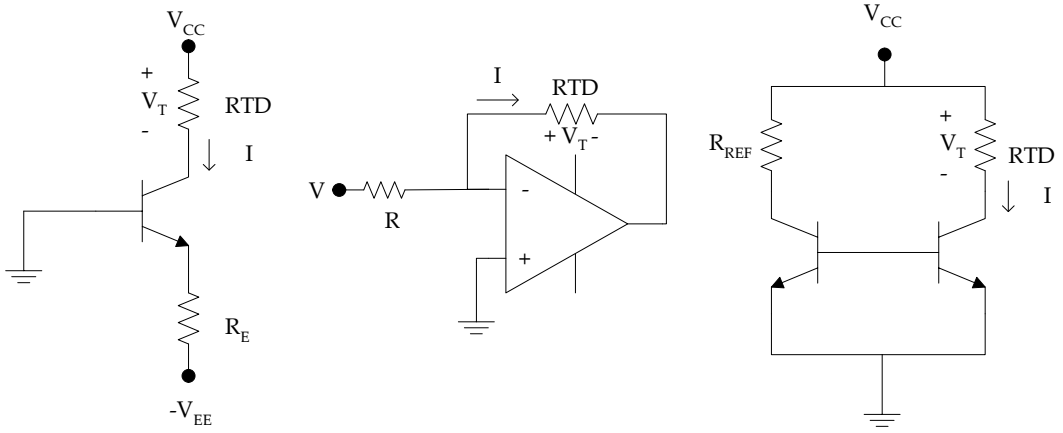
Step 1: Determine the Selection Criteria

Assume that the criteria for comparing the concepts are high accuracy, low cost, small size, and availability of parts for manufacture.

Step 2: Determine the Criteria Weightings

Assume that the criteria were ranked using pairwise comparison and weights computed (see Appendix B) as shown in Table 1.5.

Figure 1.6: Candidate solutions for temperature measurement.



1: Single Transistor 2: Inverting Op Amp 3: Current Mirror

Step 3: Identify and Rate Alternatives Relative to the Criteria

Three candidate solutions are shown in Figure 1.6. Each acts as a constant current source that drives a temperature measurement device (RTD). The resistance of an RTD varies with temperature, and when driven by a constant current, I, produces a voltage, V_T, that varies proportionally with temperature. Each circuit supplies a constant current of I=1mA.

The accuracy of each design was evaluated by a sensitivity analysis using a SPICE circuit simulation package assuming 10% resistors. The deviation of the output voltage (maximum deviation from nominal) for the three designs is 9.2%, 1.3%, and 1.9% respectively. Since the objective is to minimize the deviation, the following rating metric is used:

$$\alpha = \frac{min[deviation]}{deviation}$$

This produces the following normalize design ratings for accuracy: $\alpha 11 = 0.008$, $\alpha 12 = 0.55$, and $\alpha 13 = 0.37$.

The parts costs are the following: resistors = \$0.05, bipolar junction transistors (BJTs) = \$0.15, op amps = \$0.35, and RTDs = \$0.25. Using a measure for cost similar to (1) gives following normalized cost ratings for the three options respectively: $\alpha 21 = 0.31$, $\alpha 22 = 0.28$, and $\alpha 23 = 0.31$.

Assume that to manufacture each circuit on a printed circuit board requires the following dimensions: design 1 = 1 in^2, design 2 = 1.56 in^2, and design 3 = 2.25 in^2. The objective is to minimize size, and againusing a

Table 1.6: The decision matrix.

		Single BJT	**Op Amp**	**Current Mirror**
Accuracy	0.42	0.08	0.55	0.37
Cost	0.12	0.41	0.28	0.31
Size	0.12	0.48	0.31	0.21
Availability	0.34	0.35	0.40	0.25
Score		0.26	0.44	0.30

measure analogous to (1) for the required space to manufacture each produces the following normalized decision ratings: $\alpha 31 = 0.48$, $\alpha 32 = 0.31$, and $\alpha 33 = 0.21$.

Assume that the parts have an in-stock availability of 95%, 70%, 90%, and 80% of the time for the resistors, BJTs, RTDs, and op amps respectively. A measure for the overall availability of parts to manufacture each design is required. One way to measure this is to compute the probability that a design will be able to be manufactured based upon the past history of part availability. This is found by multiplying the availability of all individual components needed for the design:

P(design 1 can be produced) $= (0.95)(0.90)(0.70) = 0.60$

P(design 2 can be produced) $= (0.95)(0.90)(0.80) = 0.68$

P(design 3 can be produced) $= (0.95)(0.90)\ (0.70)(0.70) = 0.42$

This produces the following normalized decision ratings for availability: $\alpha 41 = 0.35$, $\alpha 42 = 0.40$, and $\alpha 43 = 0.25$.

Step 4: Compute Scores for the Alternatives

The decision matrix is built and the overall weighted scores for the alternatives are computed as shown in Table 1.6.

Step 5: Review the Decision

Remember that this is a semi-quantitative method. The final ranking indicates that design options 1 and 3 are quite similar, while both are inferior to option 2.

1.2.5 Pugh Concept Selection

Pugh Concept Selection is a method of comparing concepts against criteria, similar to what we saw with a decision matrix. It is different in that it has

Table 1.7: Pugh Concept Selection matrix.

		Option 1 (Reference)	Option 2	Option 3	Option 4
Criteria 1	4	-	0	0	+1
Criteria 2	5	-	+1	-1	0
Criteria 3	2	-	-1	0	+1
Criteria 4	1	-	+1	+1	-1
Score		-	4	-4	5
Continue?		Combine	Yes	No	Combine

a simpler scoring method and it is an iterative process. The steps of Pugh Concept Selection are:

1. Select the comparison criteria, usually the engineering or marketing requirements.
2. Determine weights for the criterion.
3. Determine the concepts.
4. Select a baseline concept that is initially believed to be the best.
5. Compare all other concepts to the baseline, using the following scoring method: +1 better than, 0 equal to, -1 worse than.
6. Compute a weighted score for each concept, not including the baseline.
7. Examine each concept to determine if it should be retained, updated, or dropped. Synthesize the best elements of others into other concepts wherever possible.
8. Update the table and iterate until a superior concept emerges.

An example of a Pugh Concept Selection matrix is shown in Table 1.7.

1.3 Project Application: Concept Generation and Evaluation

The following advice is provided for teams in the concept generation and evaluation phase:

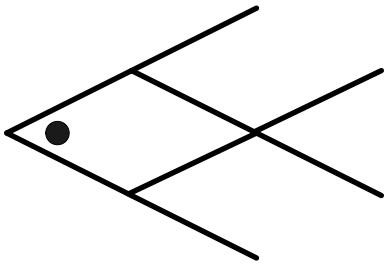
- Set aside time specifically for concept generation and evaluation and take it as a challenge to identify as many concepts as possible.

- Search externally, including literature reviews and patent searches.
- Search internally using brainstorming, brainwriting, or the Nominal Group Technique. Effective teams generate many concepts in a brainstorming session.
- Examine solutions for the entire design, for sub-functions of the design, and for individual components (such as integrated circuit selection). The techniques in this chapter can be combined with design methods presented in Chapters 5 and 6.
- Utilize SCAMPER, concept tables, and concepts fans as tools to facilitate and document concept generation.
- Critically and objectively evaluate concepts against common criteria.
- Clearly identify the concept(s) selected and the rationale for selection.

1.4 Summary and Further Reading

In the design process, it is important to creatively generate different concepts for a solution to a problem. This is followed by an evaluation of concepts to determine which are the most promising. This chapter identified barriers to creativity and provided strategies for enhancing creative ability. The concepts of vertical and lateral thinking were introduced, and their impact on the design process was explored. Methods of concept generation, including brainstorming, concept tables, and concepts fans were presented. Finally, methods for critically evaluating concepts (strength/weakness analysis, Analytical Hierarchy Process, and Pugh Concept selection) were presented.

There are many references that examine creativity and concept generation. Adams [Ada01] is a good reference for creativity and problem-solving with a technical bent. Alex Osborn was an advocate of creativity and developed two readable works that address the creative process, the need for creativity, and strategies for enhancing it [Osb48, Osb63]. Edward DeBono is another well-known authority in the field and has produced many works on lateral thinking and creativity [Deb67, Deb70]. Paul Sloane has published numerous books with lateral thinking puzzles [Slo91, Slo93, Slo94]. TRIZ [Alt99] is a more advanced and complex approach to concept generation, that centers around resolving tradeoffs in a problem. It is fairly complex and may be considered by more advanced teams.



1.5 Problems

1. Consider the nine dot puzzle shown in Figure 3 (b). Draw **three** connected straight lines that pass through all nine dots.

2. Consider the six sticks shown below. Rearrange the sticks to produce four equilateral triangles (the sticks cannot be broken).

3. Consider the fish shown below made of eight sticks and a coin for the eye. The objective is to make the fish face the other direction by moving only the coin and three sticks.

4. For each of the following lateral thinking puzzles, develop a plausible solution (from Paul Sloane's Lateral Thinking Puzzles [http://dspace.dial.pipex.com/sloane]):

   - A man walks into a bar and asks the barman for a glass of water. The barman pulls out a gun and points it at the man. The man says 'Thank you' and walks out.
   - A woman had two sons who were born on the same hour of the same day of the same year. But they were not twins. How could this be so?
   - Why is it better to have round manhole covers than square ones?
   - A man went to a party and drank some of the punch. He then left early. Everyone else at the party who drank the punch subsequently died of poisoning. Why did the man not die?

5. Legislation was passed to allow handguns in the cockpits of passenger airliners to prevent hijacking. Brainstorm to develop concepts that prevent anyone other than the pilot from using the handgun.

6. Imagine if scientists and engineers were able to develop a technology that would allow people to be transported from any place on earth

	Accuracy	Cost	Size	Availability
Accuracy	1	1/3	2	$\frac{1}{2}$
Cost	3	1	5	1
Size	1/2	1/5	1	2
Availability	2	1	$\frac{1}{2}$	1

to another instantaneously. Brainstorm to determine the potential impact this would have on society.

7. Student advising at many colleges and universities is seen as an area that can be improved. Brainstorm to develop ideas as to how student advising could be improved at your college or university.

8. In your own words, describe what a concept table and a concept fan are.

9. Consider the problem solved in Section 1.2.4. For this example assume that:

   - The following is the result of the paired comparison.
   - The parts costs are the following: resistors = \$0.05, bipolar transistors (BJTs) = \$0.10, op amps = \$0.35, and RTDs = \$0.25.
   - The parts have an in-stock availability of 99%, 90%, 85%, and 70% of the time for the resistors, BJTs, RTDs, and op amps respectively.
   - Everything else is the same as presented in Section 1.2.4.

   Compute the rankings of the design options using a weighted decision matrix of the type shown in Table 1.5.

10. **Project Application.** Utilize the methods in this chapter to generate concepts for your particular design problem. Critically evaluate the concepts generated using one or more of the techniques presented in the chapter that is appropriate for the problem. Section 1.3 provides guidance on how to conduct this process and document the results.

Chapter 2

System Design I: Functional Decomposition

> *At Sony, we assume all products of our competitors will have basically the same technology, price, performance, and features. Design is the one thing that differentiates one product from another in the marketplace.—Norio Ohgo, Chairman and CEO, Sony*

After the technical concept is selected, it is translated into a solution that satisfies the system requirements. The designer must put on paper, or the computer screen, a representation that is meaningful and clear; in other words, a useful abstraction of the system. Engineering designs are often complex, consisting of many systems and subsystems, thus this representation should facilitate the design process and effectively describe the system. In addition, it serves an important function in communicating the design to all members of the team. Imagine a scenario where each team member is responsible for designing part of a large system. Each person develops their part in isolation and several months later the team gets back together to integrate the pieces. Of course, the system won't work unless the team has collectively defined and communicated the functionality and interfaces for all subsystems in the design.

This chapter presents a well-known design technique—known as ***functional decomposition***—that is intuitive, flexible, and straightforward to apply. It is probably the most pervasive design technique used for engineering systems and is applicable to a wide variety of problems that extend well beyond electrical and computer engineering. In functional decomposition, systems are designed by determining the overall functionality and then

iteratively decomposing it into component subsystems, each with its own functionality.

The objective of this chapter is to present both basic design concepts and the functional decomposition design technique. A process for functional decomposition is provided and it is applied to examples in analog electronics, digital electronics, and software systems.

Learning Objectives

By the end of this chapter, the reader should:

- Understand the differences between bottom-up and top-down design.
- Know what functional decomposition is and how to apply it.
- Be able to apply functional decomposition to different problem domains.
- Understand the concepts of coupling and cohesion, and how they impact designs.

2.1 Bottom-Up and Top-Down Design

Two general approaches to synthesizing engineering designs are known as ***bottom-up*** and ***top-down***. In the case of bottom-up, the designer starts with basic components and synthesizes them to create the overall system. To use an analogy, consider the case of creating an automobile. In the bottom-up approach, you have pieces of the automobile, such as the tires, motor, axle, transmission, alternator, and they are brought together to create a car. The implication is that the final system depends upon the parts at hand. In other words, in the bottom-up approach, the parts and subsystems are given, and from them an artifact is created.

The top-down approach is analogous to the concept of divide-and-conquer. In top-down the designer has an overall vision of what the final system must do, and the problem is partitioned into components, or subsystems that work together to achieve the overall goal. Then each subsystem is successively refined and partitioned as necessary. In the case of the automobile, the overall objective is determined; the major subsystems are defined, such

as electrical, power drive-train, and the suspension; and then each subsystem is further refined into its component parts until the complete system is designed.

A debate that continues in the design community revolves around which is the better approach. It might appear that top-down is better, since it starts with the overall goal (requirements) and from that a solution is developed. Top-down is particularly valuable on large projects with many subsystems, where it is unlikely that bringing together pieces in an ad-hoc fashion will successfully solve the problem. A disadvantage of top-down design is that it tends to limit the solution space and innovation. Top-down design is inclined to follow a vertical thought process (Chapter 4) where the designer starts with a problem and successively refines the subsystems until a blueprint for solving the problem is defined. Furthermore, the designer cannot create a top-down design in a vacuum without bottom-up knowledge of existing technology and how the system can be realized.

Bottom-up has the advantage of lending itself to creativity. It allows the designer to take different technologies and from them create something new, allowing more "*what if?*" questions to be asked. Bottom-up design is applicable when there are constraints on the components that can be used. This is a realistic scenario. Consider the case of variant design, where the goal is to improve the performance of an existing, or legacy, system. For example, automobile manufacturers might have to redesign their models to meet new emissions, mileage, or safety standards. If you are not starting with a new design and must utilize existing systems, it requires bottom-up thinking. In reality, most problems require a combination of bottom-up and top-down thinking, and the designer must alternate between them.

In summary, it is most effective to work between bottom-up and top-down. A completely top-down approach is not feasible because the designer must have an understanding of the bottom level technology for the components of the design hierarchy to be realistic. Likewise, completely bottom-up by itself is generally not feasible, particularly as the system complexity grows.

2.2 Functional Decomposition

Functional decomposition is a recursive process that iteratively describes the functionality of all system components. It is analogous to the mathematical concept of a function, for example, $y = f(x)$. In this function there is an input, x, an output, y, and a transformation between the input and out-

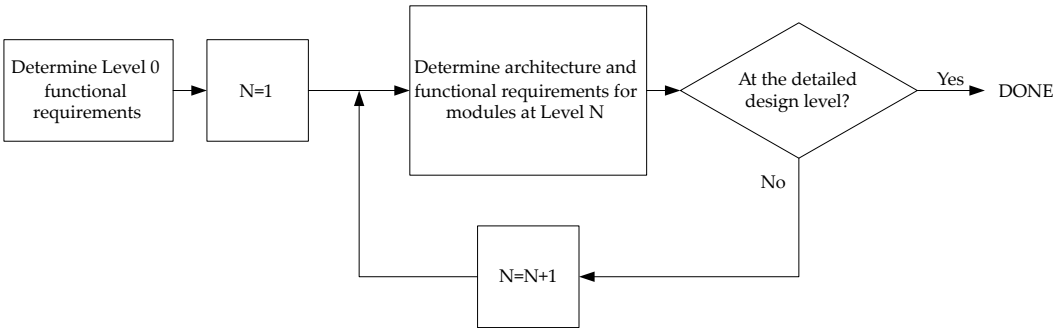


Figure 2.1: A process for developing designs using functional decomposition.

put, f. This is easily extended to the case of multiple inputs and outputs where the inputs and outputs are vectors, $\vec{y} = f(\vec{x})$. In functional decomposition, the same items are defined as in the mathematical analogy—the inputs, the outputs, and the transformation between the inputs and outputs (the functionality). Those three items constitute what is known as the ***functional specification*** or ***functional requirement***—the requirement that a functional module should meet. A ***module*** is a block, or subsystem, that performs a function. Functional decomposition has a strong top-down flavor, due to the fact that the highest level functionality is defined and then further refined into sub-functions, each with its own inputs, outputs, and functionality. The process is repeated until some base level functionality is reached where the modules can be actualized with physical components.

A process for applying functional decomposition is illustrated in Figure 2.1. It starts with a definition of the highest level (Level 0) of system functionality (the functional requirement for the system). This is followed by definition of the next level of the hierarchy that is needed to achieve the design objective. The Level 1 design is typically referred to as the main ***design architecture*** of the system. In this context, architecture means the organization and interconnections between modules. Care must be taken at each design level to ensure that it satisfies the requirements of the higher level. The process is repeated for successive levels of the design and stops when the ***detailed design*** level is reached. Detailed design is where the problem can be decomposed no further and the identification of elements such as circuit components, logic gates, or software code takes place. The number of levels in the design hierarchy depends upon the complexity of the problem.

2.3 Guidance

The following guidance is provided before examining applications of the functional decomposition technique:

- *It is an iterative process.* During the first pass, it is not possible to know all of the detailed interfaces between components and the exact functionality of each block. In fact, some details are not known until the implementation level is reached, so the designer needs to iterate, work between top-down and bottom-up, and adjust the design as necessary.
- *Set aside sufficient time to develop the design.* This is a corollary to the previous point. The iterative nature means that it takes time to examine different solutions and to refine the details into a working solution.
- *Pair together items of similar complexity.* Modules at each level should have similar complexity and granularity.
- *A good design will have the interfaces and functionality of modules well-defined.* It is fairly easy to piece together some blocks into an apparently reasonable design. However, the functional requirements should be clearly defined and the technical feasibility understood. If not, the design will fall apart when it comes to the implementation stage. Consider the following advice of a well-known architectural designer:

  > *The details are not the details. They make the design.— Charles Eames*

- *Look for innovations.* Top-down designs tend to follow a vertical thinking process, where the designer proceeds linearly from problem to solution. Try to incorporate lateral thinking strategies from Chapter 4 and examine alternative architectures and technologies for the solution.
- *Don't take functional requirements to the absurd level.* Common elements, such as analog multipliers or digital logic gates, do not require explicit functional specifications. Doing so may become cumbersome and add little to the design. However, it depends upon the level at which you are working. If the goal is to design an analog multiplier chip, it is entirely appropriate to develop the functional requirement for the multiplier.

- *Combine functional decomposition with other methods of describing system behavior.* There is no single method or unifying theory for developing designs. Functional decomposition alone cannot describe all system behaviors. It may be supplemented by other tools such as flowcharts (logical behavior), state diagrams (stimulus-response), or data flow diagrams. In the digital stopwatch example presented later in this chapter, the behavior is defined using state diagrams. Other methods for describing system behavior are addressed in Chapter 6.

- *Find similar design architectures.* Determine if there exist similar designs and how they operate. Realize that this creates a bias towards existing solutions.

- *Use existing technology.* Many designers take the attitude that they are going to develop the entire design themselves, the sentiment being to ignore technology that they did not develop. Furthermore, engineering education predisposes us to design at a fundamental level. Both factors lead to time spent re-inventing the wheel. If existing technology is available that meets both the engineering and cost requirements, then use it.

- *Keep it simple.* Do not add complexity that is not needed.

  > *A designer knows that he has achieved perfection not when there is nothing left to add, but when there is nothing left to take away.—Antoine de St-Exupery*

- *Communicate the results.* It is important to describe the theory of operation (the *why*) as well as the implementation (the *what*). The *what* in the completed design is usually quite clear from the implementation, but documenting the description of operation and design decisions helps later when the system must upgraded. Designs can also become very complex, so consider how much information can be effectively communicated on a single page. If the information is too complex to show reasonably on a page or two, then it probably is too detailed and another level in the hierarchy should be added.

2.4 Application: Electronics Design

We now examine the application of functional decomposition in different problem domains. In the domain of analog electronics, the inputs and out-

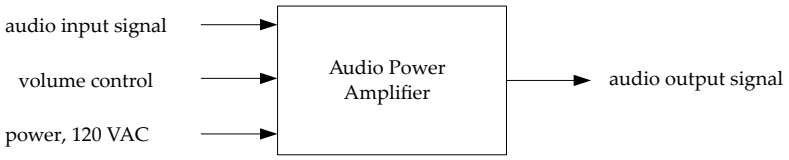


Figure 2.2: Level 0 audio power amplifier functionality.

puts of modules are voltage and current signals. Typical transformations applied to the inputs are alterations in amplitude, power, phase, frequency, and spectral characteristics. Consider the design of an audio power amplifier that has the following engineering requirements.

The system must

- Accept an audio input signal source with a maximum input voltage of 0.5V peak.
- Have adjustable volume control between zero and the maximum volume level.
- Deliver a maximum of 50W to an 8Ω speaker.
- Be powered by a standard 120V 60Hz AC outlet.

Level 0

The Level 0 functionality for the amplifier is shown in Figure 2.2, which is fairly simple—the inputs are an audio signal, volume control, and wall outlet power, and the output is an amplified audio signal.

The system should be described in as much detail as possible for each level via the functional requirement. The Level 0 functional requirement for this design is as follows.

Not all values can be known on the first pass through the design as was indicated in the guidelines. Underlined items represent values that need to be determined or refined as the design proceeds. In this case, the peak value of the audio output voltage is determined from the system requirements on power gain. Knowing that the maximum power is given by $P_{max} = V_{peak}^2/R$, allows the maximum output voltage to be computed as $V_{peak} = \sqrt{8\Omega * 50W} = 20V$.

Table 2.1: A concept table for generating ideas for a personal computing system. The potential solution is identified by the combination of circled elements.

Module	Audio Power Amplifier
Inputs	• Audio input signal: 0.5V peak. • Power: 120 volts AC rms, 60Hz. • User volume control: variable control.
Outputs	• Audio output signal: ?V peak value.
Functionality	Amplify the input signal to produce a 50W maximum output signal. The amplification should have variable user control. The output volume should be variable between no volume and a maximum volume level.

Level 1

The Level 1 diagram, or system architecture, is shown in Figure 2.3. This architecture is common in amplifier design and is but one possible solution. It contains three cascaded amplifier stages and a DC supply that powers the three stages. The first amplifier stage, the *buffer amplifier*, provides a high resistance buffer that minimizes loading effects with the source. Buffer amplifiers have extremely high input resistance and a unity signal gain. The *high gain amplifier* increases the amplitude of the signal, but provides little in terms of the output current necessary to drive the speakers. The last stage in the cascade is the *power output stage*, which provides the current needed to drive the speakers, but has no voltage amplification.

The functional requirements for the Level 1 subsystems are now detailed, starting with the buffer amplifier.

Where did the ± 25V DC value for the DC input power come from? The system must produce a ± 20V AC output signal to satisfy the Level 0 requirement, so supply values that exceed that are required to power the electronics. How about the values for the input and output resistance? They are educated guesses, based on knowledge of what is achievable with the technology (bottom-up knowledge). The exact resistance requirements are refined later based upon the overall design, taking into account the input

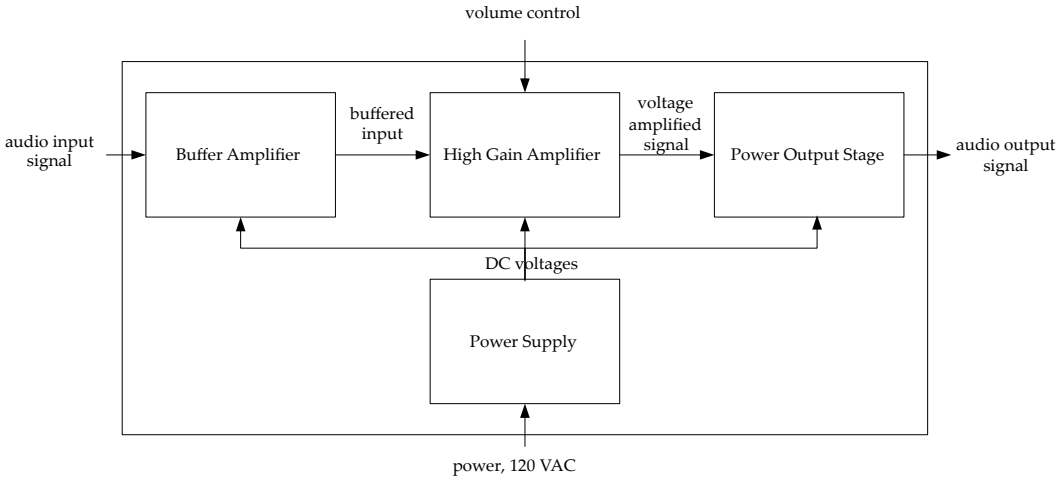


Figure 2.3: Level 1 audio amplifier design.

Module	Buffer Amplifier
Inputs	• Audio input signal: 0.5V peak. • Power: ± 25V DC.
Outputs	• Audio signal: 0.5V peak.
Functionality	Buffer the input signal and provide unity voltage gain. It should have an input resistance >1MΩ and an output resistance <100Ω.

Module	High Gain Amplifier
Inputs	• Audio input signal: 0.5V peak. • User volume control: variable control. • Power: ± 25V DC
Outputs	• Audio signal: 20V peak.
Functionality	Provide an adjustable voltage gain, between 1 and 40. It should have an input resistance >100kΩ and an output resistance <100Ω.

Module	Power Output Stage
Inputs	• Audio input signal: 20V peak. • Power: ± 25V DC.
Outputs	• Audio signal: 20V peak at up to 2.5A.
Functionality	Provide unity voltage gain with output current as required by a resistive load of up to 2.5A. It should have an input resistance >1MΩ and output resistance <1Ω.

and output resistances for all stages.

Now consider the functional requirements for the high gain amplifier.

The gain of 40 is determined from the overall system power and gain requirements (the maximum input voltage of 0.5V must be able to be amplified to 20V), while the resistances are again educated guesses.

Now consider the power output stage.

For the power output stage, it is clear that 20V peak needs to be delivered, but how was the requirement on current determined? The current needed to drive the speaker is determined from Ohm's Law as $I = V/R = 10V/8\Omega = 2.5A$.

The last module to examine at this level is the power supply.

It is clear that the power supply needs to deliver ± 25V DC, while the 3.0A current capability was selected to supply the 2.5A needed for the

Module	Power Supply
Inputs	• 120 Volts AC rms.
Outputs	• Power: ± 25V DC with up to 3.0 A of current with a regulation of <1%.
Functionality	Convert AC wall outlet voltage to positive and negative DC output voltages, and provide enough current to drive all amplifiers.

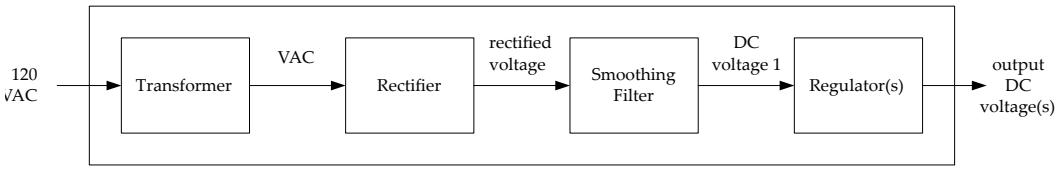


Figure 2.4: Level 2 design of the power supply.

peak output power requirement plus the current needed to power the other amplifier stages.

Finally, it is necessary to determine if the values of the input and output resistances selected for the stages are realistic. For cascaded amplifier stages, the overall voltage gain is given by the product of gains multiplied by the voltage divider losses between stages [Sed04]. In this case the overall gain is

So the resistance values selected in the functional requirements satisfy the overall system requirements. If not, it would be necessary to go back and refine them.

Level 2

At this point, the three amplifier stages are ready for detailed component level design, while the power supply needs another level of refinement as shown in Figure 2.4. The functional requirement for each of the elements in the power supply would be developed similarly. Functional decomposition stops at this point—all levels of the hierarchy are defined and the next step is the detailed design, where the actual circuit components are determined.

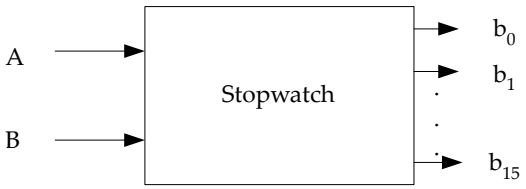


Figure 2.5: Level 0 digital stopwatch functionality.

2.5 Application: Digital Design

Functional decomposition is widely applied to the design of digital systems, where it is known as *entity-architecture* design. The inputs and outputs refer to the entity, and the architecture describes the functionality. The application of functional decomposition to digital systems is demonstrated in the following example. Consider the design of a simple digital stopwatch that keeps track of seconds and has the following engineering requirements.

The system must

- Have no more than two control buttons.
- Implement Run, Stop, and Reset functions.
- Output a 16-bit binary number that represents seconds elapsed.

Level 0

The Level 0 diagram and functional requirements are shown below.

Level 1

The Level 1 architecture in Figure 2.6 contains three modules: a seconds counter, a clock divider, and a finite state machine (FSM). The stopwatch counts seconds, thus the seconds counter module counts the seconds and outputs a 16-bit number representing the number of seconds elapsed. The clock divider generates a 1Hz signal that triggers the seconds counter. The FSM responds to the button press stimuli and produces the appropriate control signals for the seconds counter. The system clock is included to clock both the FSM and the clock divider.

The functionality of the Level 1 modules is described as follows, starting with the finite state machine.

Module	Stopwatch
Inputs	• A: Reset button signal. When the button is pushed it resets the counter to zero. • B: Run/stop toggle signal. When the button is pushed it toggles between run and stop modes.
Outputs	• b_{15}-b_0: 16-bit binary number that represents the number of seconds elapsed.
Functionality	The stopwatch counts the number of seconds after B is pushed when the system is in the Reset or Stop mode. When in Run mode and B is pushed, the stopwatch stops counting. A reset button push (A) will reset the output value of the counter to zero only when in Stop mode.

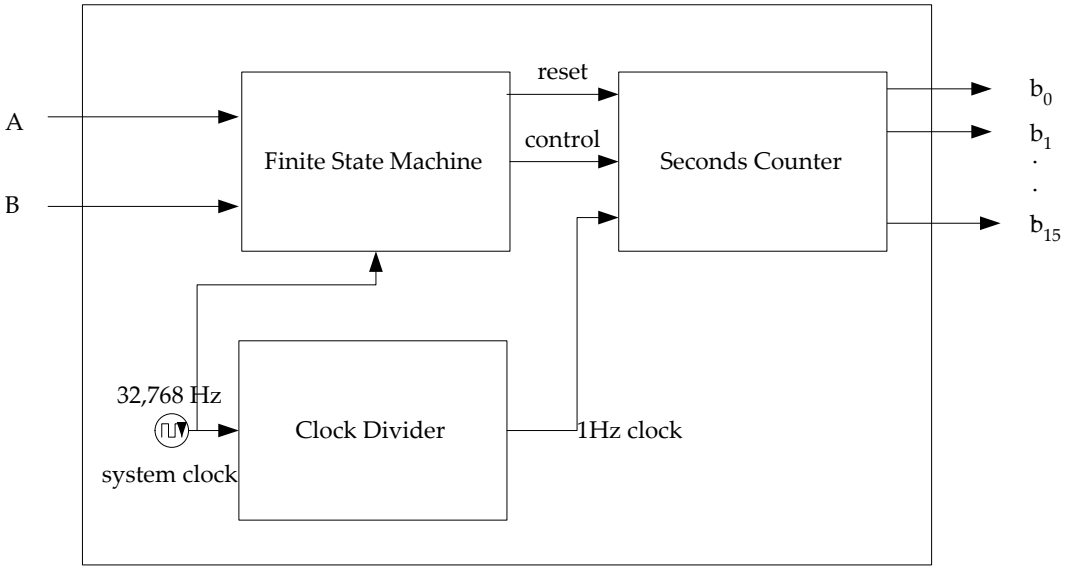
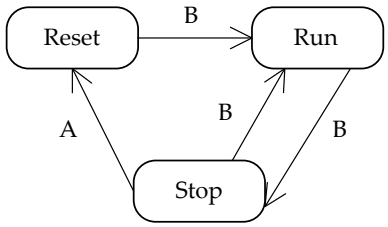


Figure 2.6: Level 1 design for the digital stopwatch..

Module	Finite State Machine
Inputs	• A: Signal to reset the counter. • B: Signal to toggle the stopwatch between run and stop modes. • Clock: 1Hz clock signal.
Outputs	• Reset: Signal to reset the counter to zero. • Control: Signal that enables or disables the counter.
Functionality	

The functionality of the finite state machine is described with a tool that is probably familiar to the reader, the state diagram. State diagrams are covered in more detail in Chapter 6. The state diagram describes stimulus-response behavior, and shows how the system transitions between states based upon logic signals from the button presses.

Next, consider the clock divider.

The value of 32,768 Hz was selected for the system clock for several reasons. It is a power of 2 that is easily divisible by digital circuitry to produce a 1Hz output signal. It is also well above the clock rate needed for

Module	Clock Divider
Inputs	• System clock: 32,768Hz.
Outputs	• Internal clock: 1Hz clock for seconds elapsed.
Functionality	Divide the system clock by 32,768 to produce a 1Hz clock.

Module	Seconds Counter
Inputs	• Reset: Reset the counter to zero. • Control: Enable/disable the counter. • Clock: Increment the counter.
Outputs	• b_{15}-b_0: 16-bit binary representation of number of seconds elapsed.
Functionality	Count the seconds when enabled and resets to zero when reset signal enabled.

detecting button presses and there is a wide selection of crystals that can meet this requirement.

Finally, consider the seconds counter.

The system decomposition would end here, assuming that the design is to be implemented using off-the-shelf chips. The next step would be to determine components at the detailed design level. However, if it were an integrated circuit design, the description would continue until the transistor level is reached.

2.6 Application: Software Design

Software also lends itself to functional decomposition since virtually all computing languages provide the capability to call functions, subroutines, or modules. Functional software design simplifies program development by eliminating the need to create redundant code via the use of functions that are called repeatedly.

Structure charts are specialized block diagrams for visualizing functional software designs. The modules used in a structure chart are shown in Figure 2.7. The larger arrows indicate connections to other modules, while the smaller arrows represent data and control information passed between modules. Five basic modules are utilized:

1. *Input modules.* Receive information.

2. *Output modules.* Return information.

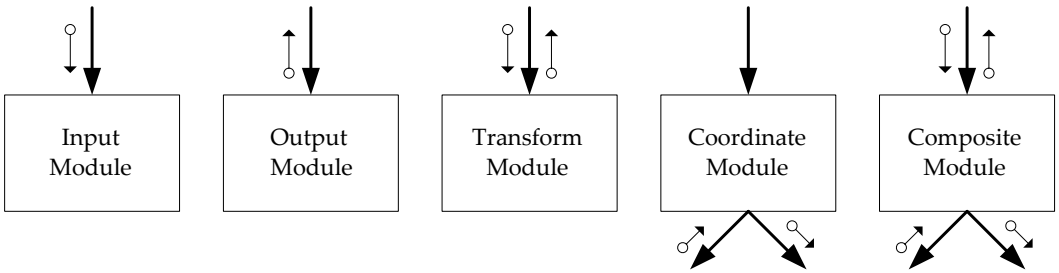


Figure 2.7: Module types for functional software design. The larger arrows indicate connections between modules and the smaller arrows represent data and control.

3. *Transform modules.* Receive information, change it, and return the changed information.

4. *Coordination modules.* Coordinate or synchronize activities between modules.

5. *Composite modules.* Any possible combination of the other four.

This approach to software design, also known as *structured design*, was formalized in the 1970's by IBM researchers [Ste99].

The following example demonstrates the application of functional decomposition to a software design with the following requirements.

The system must

- Accept an ASCII file of integer numbers as input.
- Sort the numbers into ascending order and save the sorted numbers to disk.
- Compute the mean of the numbers.
- Display the mean on the screen.

This is a fairly simple task that could easily be done in a single function, but doing so would not allow components of the design to be easily reused, tested, or troubleshot. The engineering requirements themselves provide some guidance in terms of how to arrange the functionality of the modules (*form follows function*). The architecture in Figure 2.8 contains a main module that calls three sub-modules. In this design main is a coordinating module that controls the processing and calling of the other modules, a

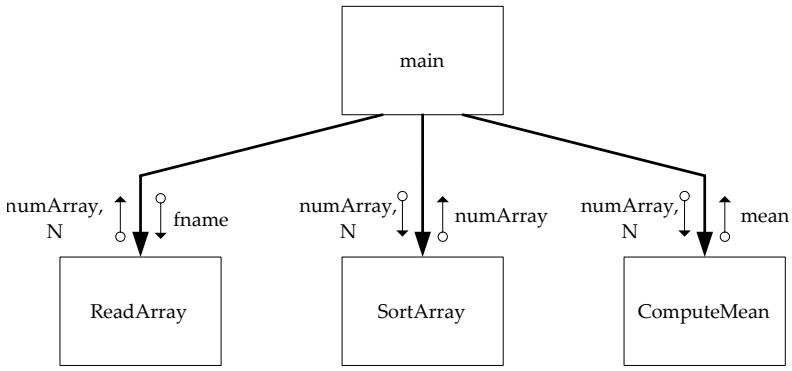


Figure 2.8: Structure chart design of sorting and mean computation program.

common scenario. It was also decided that all user interaction would take place within main. The order of the processing is not described by structure charts. In our program, main calls ReadArray, SortArray, and ComputeMean in sequential order. main passes the filename (fname) to ReadArray, which reads in the array and the number of elements in it, and returns this information to main. The choice of passing in the filename was deliberate; the user could have been prompted for the filename in ReadArray, but doing so might limit future reuse of the function since you may not always want to do so when reading an array of data. SortArray is then called, which accepts the array of numbers and the number of elements in the array, and returns the sorted values in the same array. Finally, ComputeMean is executed, which accepts the sorted array and the number of elements, computes the mean value, and returns it to main.

The functional requirements for each module in the structure chart are detailed in Table 2.2. The structure chart provides a visual relationship between modules in the design, but also has some disadvantages. It is difficult to visualize designs as the complexity of the software increases. This can be addressed by expanding sublevels in the design as necessary in different diagrams. Structure charts also lack a temporal aspect that indicates the calling order. Most software systems have many layers in the hierarchy and highly complex calling patterns. In this example, main calls three modules in a well-defined order, but if there was another level in the hierarchy, there is no reason why it could not be called by a module at any other level. That leads to some of the unique problems associated with software design. Functional design works well for small-to-moderately complex software, but

Table 2.2: Functional design requirements for the number sort program.

tends to fall short when applied to large scale software systems. As such, it has given way to the object-oriented design approach.

2.7 Application: Thermometer Design

The final example includes both analog and digital modules where the objective is to design a thermometer that meets the following engineering requirements.

The system must

- Measure temperature between 0 and 200°C.
- Have an accuracy of 0.4% of full scale.
- Display the temperature digitally, including one digit beyond the decimal point.
- Be powered by a standard 120V 60Hz AC outlet.
- Use an RTD (thermal resistive device) that has an accuracy of 0.55°C over the range. The resistance of the RTD varies linearly with temperature from 100Ω at 0°C to 178Ω at 200°C. (Note: this requirement does not meet the abstractness property identified in Chapter 3, since it identifies part of the solution. This requirement is given to provide guidance in this example.)

Module name	main
Module type	Coordination
Input arguments	None
Output arguments	None
Description	The main function calls ReadArray() to read the input file from disk, SortArray() to sort the array, and ComputeMean() to determine the mean value of elements in the array. User interaction requires the user to enter the filename, and the mean value is displayed on the screen.
Modules invoked	ReadArray, SortArray, and ComputeMean

Module name	ReadArray()
Module type	Input and output
Input arguments	• fname[]: character array with filename to read from.
Output Arguments	• numArray[]: integer array with elements read from file. • N: number of elements in numArray[].
Description	Read data from input data file and store elements in array numArray[]. The number of elements read is placed in N.
Modules invoked	None

Module name	SortArray()
Module type	Transformation
Input arguments	• numArray[]: integer array of numbers. • N: number of elements in numArray[].
Output Arguments	• numArray[]: sorted array of integer numbers.
Description	Sort elements in array using a shell sort algorithm. Saves the the sorted array to disk.
Modules invoked	None

Module name	ComputeMean()
Module type	Input and output
Input arguments	• numArray[]: integer array of numbers. • N: number of elements in numArray[].
Output arguments	• mean: mean value of the elements in the array.
Description	Computes the mean value of the integer elements in the array.
Modules invoked	None

Figure 2.9: Level 0 digital thermometer functionality.

Module name	Digital Thjermometer
Inputs	• Ambient temperature: 0-200°C. • Power: 120V AC power.
Outputs	• Digital temperature display: A four digit display, including one digit beyond the decimal point.
Functionality	Displays temperature on digital readout with an accuracy of 0.4% of full scale.

Level 0

The overall goal is to convert a sensed temperature to a digital temperature reading. The Level 0 description is as follows.

Level 1

The Level 1 architecture selected is shown in Figure 2.10. The temperature conversion unit converts the temperature to an analog voltage using the RTD that is sampled by the analog-to-digital converter. The N-bit binary output from the converter is translated into binary-coded decimal (BCD). BCD is a 4-bit representation of the digits between 0 and 9. Since there are four display digits, there are four separate binary encoded outputs from the BCD conversion unit. Common 7-segment LEDs are used for the display. However, they do not directly accept BCD and instead have seven input lines, each of which is individually switched to control the display segments. The requirements did not specifically address cost or size constraints, nor clearly define the environment, so there are many possible solutions. For example, an analog-to-digital current converter could be used, integrated circuit temperature sensing packages could be considered, and microcontroller-based

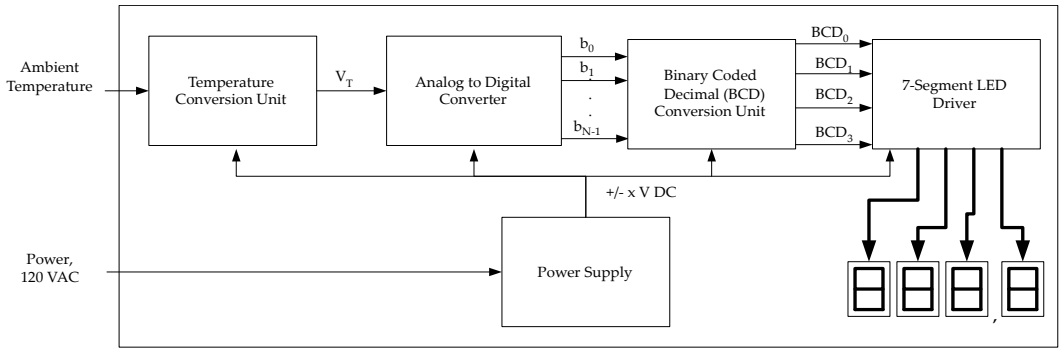


Figure 2.10: Level 1 design of the digital thermometer.

Module	Temperature Conversion Unit
Inputs	• Ambient temperature: 0-200°C. • Power: ?V DC (to power the electronics).
Outputs	• V_T: temperature proportional voltage. $V_T = \alpha T$, and ranges from ? to ?V.
Functionality	Produces an output voltage that is linearly proportional to temperature. It must achieve an accuracy of ?%.

solutions are feasible as well.

From a system design perspective, an error budget is needed to identify the maximum error that each subsystem may introduce, while still achieving the overall accuracy. In this case, error is introduced in the temperature conversion unit and A/D converter, but not in the remaining digital components. The overall accuracy that the system must achieve is 0.4%, and that translates into 0.8°C of allowable error for the 200°C range. Let's now examine the modules.

The functionality of the Level 1 modules is described as follows, starting with the temperature conversion unit.

There are several unknowns at this point. The voltage necessary to power the electronics is not known, but a reasonable assumption could be made. The output voltage range and the accuracy are unknown. It is known that the RTD will introduce up to 0.55°C of error and that the electronics

Module	A/D Converter
Inputs	• V_T: voltage proportional to temperature that ranges from ? to ?V. • Power: ?V DC.
Outputs	• b_{N-1} -b_0: ?-bit binary representation of V_T.
Functionality	Converts analog input to binary digital output.

themselves will introduce additional error (the exact amount is unknown at this point). An educated guess is made that the maximum error allowed for the temperature unit is 0.6°C. This means that the electronics themselves would be required to introduce no more than 0.05°C of error due to the 0.55°C of error introduced by the RTD.

Now consider the analog to digital converter.

The A/D converter is not likely to be something that is designed due to the availability of low cost, off-the-shelf solutions. The requirements drive the converter selection. There are two unknowns—the number of bits and the range of the input voltage. The number of bits affects the accuracy, since the greater the number of bits, the better the accuracy. The number of bits needed for the converter is calculated from the maximum allowable error that the A/D can introduce (0.2°C), the number of discrete intervals, and the temperate range as

$$Maxerror = \frac{range}{numberofintervals} = \frac{200°C}{2^N} \leq 0.2°C \Rightarrow N \geq 9.97bits$$

So the A/D converter needs to have at least 10 bits. How is the voltage range selected? It is typically fixed for a particular integrated circuit solution, but the temperature conversion subsystem output should be matched to the voltage range so that all bits are effectively utilized, otherwise, error is introduced.

Now, consider the BCD conversion unit.

The objective of the BCD conversion unit is fairly simple, although the component level design of the circuitry to accomplish the conversion is not.

This leads to the last module, the 7-segment LED driver, whose functionality is described as follows.

Module	BCD Conversion Unit
Inputs	• 10-bit binary number (b_9-b_0): Represents the range 0.0-200.0°C. • Power: ?V DC.
Outputs	• BCD_0: 4-bit BCD representation of tenths digit (after decimal). • BCD_1: 4-bit BCD representation of one's digit. • BCD_2: 4-bit BCD representation of ten's digit. • BCD_3: 4-bit BCD representation of hundred's digit.
Functionality	Converts the 10-bit binary number to BCD representation of temperature. Must refresh the displays twice a second.

Module	7-Segment LED Driver
Inputs	• BCD_0: 4-bit BCD representation of tenths digit (after decimal). • BCD_1: 4-bit BCD representation of one's digit. • BCD_2: 4-bit BCD representation of ten's digit. • BCD_3: 4-bit BCD representation of hundred's digit. • Power: ?V DC.
Outputs	• Four 7-segment driver lines.
Functionality second.	Converts the BCD for each digit into outputs that turn on LEDs in 7-segment package to display the temperature.

Module	Power supply
Inputs	• 120 Volts AC rms.
Outputs	• ± ?V DC with up to ?mA of current. • Regulation of ?%.
Functionality	Convert AC wall outlet voltage to positive and negative DC output voltages, with enough current to drive all circuit subsystems.

For completeness, the functional requirements of the power supply are supplied. They are similar to the power supply requirements utilized in the audio amplifier design in Section 2.4.

At this point, the requirements for the major subsystems are completed and ready for design at the component level. Illustration of the complete design would require a fair amount of detail, and while it is not presented here, some of the issues involved are discussed. First, there are a variety of electronic circuits (inverting op amps, single BJT configurations, and current mirrors, etc.—see Example 4.1 in Chapter 4) that could be utilized as a current source to drive the RTD in the temperature conversion subsystem. A midrange resolution A/D converter is needed, and its particular input voltage range drives the output voltage requirements for the temperature conversion module. The BCD conversion circuitry could be implemented using combinational digital logic (tedious due to the number of discrete gates), or a more efficient, but slower, sequential logic design. Finally, the 7-segment display converters could be designed using combinational logic that maps the BCD inputs into outputs to activate the appropriate display segments.

2.8 Coupling and Cohesion

The concepts of coupling and cohesion are examined before concluding this chapter. They originated to describe software designs [Ste99], but are applicable to electrical and computer systems. To understand their importance, consider the relationship between the number of modules in a system and the number of connections between them. For our purposes, a connection

between two modules may consist of any number of signals without regard to their direction. Thus, a system consisting of two modules has, at most, one connection. If the number of modules is increased to three, the number of possible connections increases to three, a system with four modules has six possible connections, and five modules increases the number of possible connections to ten. The point is that the maximum number of potential connections increases rapidly with the number of modules in the system. The relationship between the maximum possible connections and number of modules (n) is given by

$$Connections_{max} = \frac{n(n-1)}{2}$$

Modules are coupled if they depend upon each other in some way to operate properly. ***Coupling*** is the extent to which modules or subsystems are connected [Jal97]. Although there is no agreed upon mathematical definition of coupling, it seems obvious that increasing the exchange of control and data between two modules leads to a higher degree of coupling. When systems are highly coupled, it is difficult to change one module without impacting the other. Consider the extreme case where all modules in a system are connected to each other—an error in one module has the potential to impact every other module in the system. Errors in a module are propagated to others to a degree that is related to the amount of coupling. From this point of view, it is good to minimize coupling. Yet coupling cannot be eliminated, since the point of functional decomposition is to break a design into components that work together to produce a higher level behavior.

There are two ways to reduce coupling—minimize the number of connections between modules and maximize cohesion within modules. ***Cohesion*** refers to how focused a module is—highly cohesive systems do one or a few things very well. Stevens *et al.* [Ste99] defined six types of cohesion from the weakest to strongest as: coincidental, logical, temporal, communicational, sequential, and functional. More information on this can be found in the original work, but the conclusion is that modules with high functional cohesion are the most desirable. So it is best to design modules with a single well-defined functional objective consistent with the philosophy of functional decomposition. This leads to the important design principle that it is desirable to maximize cohesion, while minimizing coupling.

Coupling and cohesion impact the later stages of testing and system integration. If a particular module is highly cohesive, then it should be possible to test it independently of the other modules to verify its operability. This does not mean that it will necessarily operate properly when integrated

into the overall system, but the probability that it will is higher if provided with proper inputs from connected modules. Contrast this to the case of a low cohesion system. In that case, it will likely be difficult to test the individual modules without first integrating them.

To develop a better understanding, consider the amplifier design in Figure 2.3 (Section 2.4) with three cascaded amplifier stages. Each stage is highly cohesive, performing a singular function of signal amplification. Each of these stages could easily operate as a stand-alone module independent of the complete system. How about coupling? In terms of the number of connections, it is fairly low as each amplifier stage has an input and output voltage signal. The most coupled module in the system is the power supply, and not surprisingly, its failure leads to a complete system failure. Coupling in this case can also be viewed in terms of the resistance matching between input and output of the cascaded stages, producing the voltage divider effect in equation (1). For voltage amplifiers, the goal is to have high input resistance and low output resistance, which minimize both voltage losses and coupling. The stages are not completely uncoupled, because the input resistances, although large, are not infinite, and the output resistances are not zero. The modules in the power supply unit in Figure 2.4 (rectifier, smoothing filter, and regulator) have a much higher degree of coupling. In fact, it is difficult to develop a clear functional decomposition of the power supply module because the elements in the smoothing filter also serve as part of the rectifier circuit (refer to a basic electronics textbook [Sed04] for more information).

As another example, consider a software design where two options are under consideration: one large function with 1000 lines of code, versus 15 cohesive functions, each with an average of 100 lines of code. Both perform the same function, but which runs faster? Most likely the first, as it would be highly integrated and would not suffer from overhead needed with multiple functions. Which is easier to upgrade and debug a year from now? That is clearly the second case. Although loosely coupled and highly cohesive designs may facilitate better design and testing, they may not be best in terms of performance.

2.9 Project Application: The Functional Design

The following is a format for documenting and presenting functional designs.

Design Level 0

- Present a single module block diagram with inputs and outputs identified.
- Present the functional requirements: inputs, outputs, and functionality.

Design Level 1

- Present the Level 1 diagram (system architecture) with all modules and interconnections shown.
- Describe the theory of operation. This should explain how the modules work together to achieve the functional objectives.
- Present the functional requirements for each module at this level.

Design Level N (for N>1)

- Repeat the process from Design Level 1 for as many levels as necessary.

Design Alternatives

- Describe the different alternatives that were considered, the tradeoffs, and the rationale for the choices made. This should be based upon concept evaluation methods communicated in Chapter 4.

2.10 Summary and Further Reading

This chapter presented the functional decomposition design technique, where every level of the design is decomposed into sub-modules, each of which is the domain of the next lower level. The inputs, outputs, and functionality must be determined for a given module. Applying the process in Figure 2.1 and following the guidelines in Section 2.3 should aid in the application of functional decomposition. Functional decomposition is applicable to a wide variety of systems, and in this chapter designs of analog electronics, digital electronics, and software were examined.

Nigel Cross presents a good overview of the functional decomposition method with application to mechanical systems [Cro00], but with less focus on the description of the functional requirements than presented here. The work by Stevens *et al.* [Ste99] is interesting reading that gives an understanding of the evolution of structured design. It delves into the concepts

of coupling and cohesion. Coupling and cohesion are also addressed well in the book by Jalote [Jal97]. An in-depth treatment of structured systems design is found in The Practical Guide to Structured Systems Design [Pag88]. This guide also integrates data flow diagrams with functional techniques. Finally, the thermometer design example was inspired by Stadtmiller's book [Sta01] on electronics design.

2.11 Problems

1. Describe the differences between *bottom-up* and *top-down* design.

2. Develop a functional design for an audio graphic equalizer. A graphic equalizer decomposes an audio signal into component frequencies bands, allows the user to apply amplification to each individual band, and recombines the component signals. The design can employ either analog or digital processing. Be sure to clearly identify the design levels, functional requirements, and theory of operation for the different levels in the architecture.

   The system must

   - Accept an audio input signal source, with a source resistance of 1000Ω and a maximum input voltage of 1V peak-to-peak.
   - Have an adjustable volume control.
   - Deliver a maximum of 40W to an 8Ω speaker.
   - Have four frequency bands into which the audio is decomposed (you select the frequency ranges).
   - Operate from standard wall outlet power, 120V rms.

3. Develop a functional design for a system that measures and displays the speed of a bicycle. Be sure to clearly identify the design levels, functional requirements, and theory of operation for each level.
   The system must

   - Measure instantaneous velocities between zero and 75 miles per hour with an accuracy of 1% of full scale.
   - Display the velocity digitally and include one digit beyond the decimal point.
   - Operate with bicycle tires that have 19, 24, 26, and 27 inch diameters.

4. Draw a structure chart for the following C++ program:

```
void IncBy5(int \&a, int \&b);
int Multiply(int a, int b);
void Print(int a, int b);
```

```
main() {
    int x=y=z=0;
    IncBy5(x,y);
    z=Mult(x,y);
    Print(x,z);
}

void IncBy5(int \&a, int \&b) {
    a+=5;
    b+=5;
    Print(a,b);
}

int Multiply(int a, int b) {
    return (a*b);
}

void Print(int a, int b) {
    cout  << a << ``, `` << b;
}
```

5. Develop a functional design for software that meets the following requirements.
   The system must

   - Read an array of floating point numbers from an ASCII file on disk.
   - Compute the average, median, and standard deviation of the numbers.
   - Store the average, median, and standard deviation values on disk.

   The design should have multiple modules and include the following elements: (a) a structure chart, and (b) a functional description of each module.

6. Describe in your own words what is meant by coupling in design. Describe the advantages of both loosely and tightly coupled designs.

7. **Project Application.** Develop a functional design for your project. Follow the presentation guidelines in Section 5.9 for communicating the results of the design.

Appendix A Glossary

Term	Definition
acceptance test	An acceptance test verifies that the system meets the ***Requirements Specification*** and stipulates the conditions under which the customer will accept the system (Chapter 7).
activity on node	A form of a ***network diagram*** used in a project plan. In the Activity on Node (AON) form, activities are represented by nodes and the dependencies by arrows (Chapter 10).
activity	An activity is a combination of a ***task*** and its associated ***deliverables*** that is part of a project plan (Chapter 10).
activity view	The activity view is part of the ***Unified Modeling Language***. It is characterized by an activity diagram; its ***intention*** is to describe the sequencing of processes required to complete a task (Chapter 6).
Analytical Hierarchy Process (AHP)	A decision-making process that combines both quantitative and qualitative inputs. It is characterized by weighted criteria against which the decision is made, a numeric ranking of alternatives, and computation of a numerical score for each alternative (Appendix B and Chapters 2 and 4).
artifact	System, component, or process that is the end-result of a design (Chapter 2).
automated script test	An automated script test is a sequence of commands given to a unit under test. For example, a test may consist of a sequence of inputs that are provided to the unit, where the outputs for each input are then verified against pre-specified values (Chapter 7).
baseline requirements	The original set of requirements that are developed for a system (Chapter 3).
black box test	A test that is performed without any knowledge of internal workings of the unit under test (Chapter 7).

Term	Definition
bottom-up design	An approach to system design where the designer starts with basic components and synthesizes them to achieve the design objectives. This is contrasted to ***top-down*** design (Chapter 5).
Bohrbug	Bohrbugs are reliable ***bugs***, in which the error is always in the same place. This is analogous to the electrons in the Bohr atomic model which assume a definite orbit (Chapter 7).
brainstorming	A freeform approach to concept generation that is often done in groups. This process employs five basic rules: 1) no criticism of ideas, 2) wild ideas are encouraged, 3) quantity is stressed over quality, 4) build upon the ideas of others, and 5) all ideas are recorded (Chapter 4).
Brainwriting	A variation of ***brainstorming*** where a group of people systematically generate ideas and write them down. Ideas are then passed to other team members who must build upon them.
break-even point	The break-even point is the point where the number of units sold is such that there is no profit or loss. It is determined from the total costs and revenue (Chapter 10).
bug	A problem or error in a system that causes it to operate incorrectly (Chapter 7).
cardinality ratio	The cardinality ratio describes the multiplicity of the entities in a relationship. It is applied to ***entity relationship diagrams*** and Unified Modeling Language ***static view diagrams*** (Chapter 6).
class	Classes are used in object-oriented system design. A class defines the attributes and methods (functions) of an ***object*** (Chapter 6).
cohesion	Refers to how focused a module is—highly cohesive systems do one or a few things very well. Also see ***coupling*** (Chapter 5).
component design specification	See ***subsystem design specification*** (Chapter 3).
concept fan	A graphical tree representation of design decisions and potential solutions to a problem. Also see ***concept table*** (Chapters 1 and 4).
concept generation	A phase in the ***design process*** where many potential solutions to solve the problem are identified (Chapter 1).
concept table	A tool for generating concepts to solve a problem. It allows systematic examination of different combinations, arrangements, and substitutions of different elements for a system. Also see ***concept fan*** (Chapter 4).

Term	Definition
conditional rule-based ethics	An ethics system in which there are certain conditions under which an individual can break a rule. This is generally because it is believed that the moral good of the situation outweighs the rule. Also see ***rule-based ethics*** (Chapter 11).
constraint	A special type of requirement that encapsulates a design decision imposed by the environment or a stakeholder. Constraints often violate the abstractness property of engineering requirements (Chapter 3).
controllability	A principle that applies to testing. Controllability is the ability to set any node of the system to a prescribed value (Chapter 7).
copyright	Copyrights protect published works such as books, articles, music, and software. A copyright means that others cannot distribute copyrighted material without permission of the owner (Chapter 11).
coupling	Modules are coupled if they depend upon each other in some way to operate properly. Coupling is the extent to which modules or subsystems are connected. See also ***cohesion*** (Chapter 5).
creative design	A formal categorization of design projects. Creative designs represent new and innovative designs (Chapter 2).
critical path	The path with the longest duration in a project plan. It represents the minimum time required to complete the project (Chapter 10).
cross-functional team	Cross-functional teams are those that are composed of people from different organizational functions, such as engineering, marketing, and manufacturing. Also see ***multi-disciplinary team*** (Chapter 9).
data dictionary	A dictionary of data contained in a ***data flow diagram***. It contains specific information on the data flows and is defined using a formal language (Chapter 6).
data flow diagram	The ***intention*** of a data flow diagram (DFD) is to model the processing and flow of data inside a system (Chapter 6).
decision matrix	A matrix that is used to evaluate and rank concepts. It integrates both the user-needs and the technical merits of different concepts (Chapter 4).
derating	A decrease in the maximum amount of power that can be dissipated by a device. The amount of derating is based upon operating conditions, notably increases in temperature (Chapter 8).
deliverable	Deliverables are entities that are delivered to the project based upon completion of ***tasks.*** Also see ***activity*** (Chapter 10).

Term	Definition
descriptive design process	Describes typical activities involved in realizing designs with less emphasis on exact sequencing than a ***prescriptive design process*** (Chapter 1).
design architecture	The main (Level 1) organization and interconnection of modules in a system (Chapter 5).
design phase	Phase in the ***design process*** where the technical solution is developed, ultimately producing a detailed system design. Upon its completion, all major systems and subsystems are identified and described using an appropriate model (Chapter 1).
design process	The steps required to take an idea from concept to realization of the final system. It is a problem-solving methodology that aims to develop a system that best meets the customer's need within given constraints (Chapter 1).
design space	The space, or collection, of all possible solutions to a design problem (Chapter 2).
detailed design	A phase in the technical design where the problem can be decomposed no further and the identification of elements such as circuit components, logic gates, or software code takes place (Chapter 5).
engineering requirement	A requirement of the system that applies to the technical aspects of the design. An engineering requirement should be abstract, unambiguous, verifiable, traceable, and realistic (Chapter 3).
entity relationship diagram (ERD)	An ERD is used to model database systems. The ***intention*** of an ERD is to catalog a set of related objects (entities), their attributes, and the relationships between them (Chapter 6).
entity relationship matrix	A matrix that is used to identify relationships between entities in a database system (Chapter 6).
ethics	Philosophy that studies ***morality***, the nature of good and bad, and choices to be made (Chapter 11).
event	An event is an occurrence at a specific time and place that needs to be remembered and taken into consideration in the system design (Chapter 6).
event table	A table that is used to store information about ***events*** in the system. It includes information regarding the event trigger, the source of the event, and process triggered by the event (Chapter 6).
failure function	The failure function, $F(t)$, is a mathematical function that provides the probability that a device has failed at time t (Chapter 8).

Term	Definition
failure rate	The failure rate, $\lambda(t)$, for a device is the expected number of device failures that will occur per unit time (Chapter 8).
fixed costs	Fixed costs are those that are constant regardless of the number of units produced and cannot be directly charged to a process or activity (Chapter 10).
float	The amount of ***slippage*** that an activity in a project plan can experience without it becoming part of a new ***critical path*** (Chapter 10).
flowchart	A modeling diagram whose intention is to visually describe a process or algorithm, including its steps and control (Chapter 6).
functional decomposition	A design technique in which a system is designed by determining its overall functionality and then iteratively decomposing it into component subsystems, each with its own functionality (Chapter 5).
functional requirement	A ***subsystem design specification*** that describes the inputs, outputs, and functionality of a system or component (Chapters 3 and 5).
Gantt chart	Gantt charts are a bar graph representation of a project plan where the activities are shown on a timeline (Chapter 10).
Heisenbugs	Heisenbugs are ***bugs*** that are not always reproducible with the same input. This is analogous to the Heisenberg Uncertainty Principle, in which the position of an electron is uncertain (Chapter 7).
high-performance team	A team that significantly outperforms all similar teams. Part of the Katzenbach and Smith team model (Chapter 9).
integration test	An integration test is performed after the units of a system have been constructed and tested. The integration test verifies the operation of the integrated system behavior (Chapter 7).
intention	The intention of a model is the target behavior that it aims to describe (Chapter 6).
interaction view	The interaction view is part of the ***Unified Modeling Language***. Its ***intention*** is to show the interaction between objects. It is characterized by collaboration and sequence diagrams (Chapter 6).
key attribute	An attribute for an entity in a database system that uniquely identifies an instance of the entity (Chapter 6).

Term	Definition
lateral thinking	A thought process that attempts to identify creative solutions to a problem. It is not concerned with developing the solution for the problem, or right or wrong solutions. It encourages jumping around between ideas. It is contrasted to ***vertical thinking*** (Chapter 4).
liable	Required to pay monetary damages according to law (Chapter 11).
marketing requirement (specifications)	A statement that describe the needs of the customer or end-user of a system. They are typically stated in language that the customer would use (Chapters 2 and 3).
maintenance phase	Phase in the ***design process*** where the system is maintained, upgraded to add new functionality, or design problems are corrected (Chapter 1).
matrix test	A matrix test is a test that is suited to cases where the inputs submitted are structurally the same and differ only in their values (Chapter 7).
mean time to failure	The mean time to failure (MTTF) is a mathematical quantity which answers the question, "*On average how long does it take for a device to fail?*" (Chapter 8).
module	A block, or subsystem, in a design that performs a function (Chapter 5).
morals	The ***principles*** of right and wrong and the decisions that derive from those principles (Chapter 11).
multi-disciplinary team	In general, a multi-disciplinary team is one in which the members have complementary skills and the team may have representation from multiple technical disciplines. Also see ***cross-functional team*** (Chapter 9).
negligence	Failure to exercise caution, which in the case of design could be in not following reasonable standards and rules that apply to the situation (Chapter 11).
network diagram	A network diagram is a directed graph representation of the activities and dependencies between them for a project (Chapter 10).
Nominal Group Technique (NGT)	A formal approach to brainstorming and meeting facilitation. In NGT, each team member silently generates ideas that are reported out in a round-robin fashion so that all members have an opportunity to present their ideas. Concepts are selected by a multi-voting scheme with each member casting a predetermined number of votes for the ideas. The ideas are then ranked and discussed (Chapters 4 and 9).

Term	Definition
non-disclosure agreement	An agreement that prevents the signer from disseminating information about a company's products, services, and trade secrets (Chapter 11).
object	Objects represent both data (attributes) and the methods (functions) that can act upon data. An object represents a particular instance of a ***class***, which defines the attributes and methods (Chapter 6).
object type	Characteristic of a model used in design. The object type is capable of encapsulating the actual components used to construct the system (Chapter 6).
objective tree	A hierarchical tree representation of the customer's needs. The branches of the tree are organized based upon functional similarity of the needs (Chapter 2).
observability	This principle applies to testing. Observability is the ability to observe any node of a system (Chapter 7).
over-specificity	This refers to applying targets for ***engineering requirements*** that go beyond what is necessary for the system. Over-specificity limits the size of the ***design space*** (Chapter 3).
pairwise comparison	A method of systematically comparing all customer needs against each other. A comparison matrix is used for the comparison and the output is a scoring of each of the needs (Appendix B, Chapter 2, and Chapter 4).
parallel system	A system that contains multiple modules performing the same function where a single module would suffice. The overall system functions correctly when any one of the submodules is functioning (Chapter 8).
patent	A patent is a legal device for protecting a design or invention. If a patent is held for a technology, others cannot use it without permission of the owner (Chapter 11).
path-complete coverage	Path-complete coverage is where every possible ***processing path*** is tested (Chapter 7).
performance requirement	A particular type of ***engineering requirement*** that specifies performance related measures (Chapter 3).
physical view	The physical view is part of the ***Unified Modeling Language***. Its ***intention*** is to demonstrate the physical components of a system and how the logical views map to them. It is characterized by a component and deployment diagram (Chapter 6).

Term	Definition
potential team	A team where the sum effort of the team equals that of the individuals working in isolation. Part of the Katzenbach and Smith team model (Chapter 9).
prescriptive design process	An exact process, or systematic recipe, for realizing a system. Prescriptive design processes are often algorithmic in nature and expressed using flowcharts with decision logic (Chapter 1).
principle	Fundamental rules or beliefs that govern behavior, such as the Golden Rule (Chapter 11).
problem identification	The first phase in the design process where the problem is identified, the customer needs identified, and the project feasibility determined (Chapter 1).
processing path	A processing path is a sequence of consecutive instructions or states encountered while performing a computation. They are used to develop test cases (Chapter 7).
prototyping and construction phase	Phase in the ***design process*** in which different elements of the system are constructed and tested. The objective is to model some aspect of the system, demonstrating functionality to be employed in the final realization (Chapter 1).
pseudo-team	An under-performing team where the sum effort of the team is below that of the individuals working in isolation. Part of the Katzenbach and Smith team model (Chapter 9).
Pugh Concept Selection	A technique for comparing design concepts to the user needs. It is an iterative process where concepts are scored relative to the needs. Each concept is combined, improved, or removed from consideration in each iteration of the process (Chapter 4).
real team	A team where the sum effort of the team exceeds that of the individuals working in isolation. Part of the Katzenbach and Smith team model (Chapter 9).
redundancy	A design has redundancy if it contains multiple modules performing the same function where a single module would suffice. Redundancy is used to increase ***reliability*** (Chapter 8).
reliability	Reliability, $R(t)$, is the probability that a device is functioning properly (has not failed) at time t (Chapter 8).
research phase	Phase in the ***design process*** where research on the basic engineering and scientific principles, related technologies, and existing solutions for the problem are explored (Chapter 1).

Term	Definition
Requirements Specification	A collection of engineering and marketing requirements that a system must satisfy in order for it to meet the needs of the customer or end-user. Alternate terms that are used for the Requirements Specification are the *Product Design Specification* and the *Systems Requirements Specification* (Chapter 1 and 3).
reverse-engineering	Process where a device or process is taken apart to understand how it works (Chapter 11).
routine design	A formal categorization of design projects. They represent the design of artifacts for which theory and practice are well-developed (Chapter 2).
rule-based ethics	Rule-based ethics are based upon a set of rules that can be applied to make decisions. In the strictest form, they are considered to be absolute in terms of governing behavior (Chapter 11).
satisfice	Satisfice means that a solution may meet the design requirements, but not be the optimal solution (Chapter 11).
series system	A system in which the failure of a single component (or subsystem) leads to failure of the overall system (Chapter 8).
situational ethics	Situational ethics are where decisions are made based on whether they produce the highest good for the person (Chapter 11).
slippage	Refers to an activity in a project plan taking longer than its planned time to complete. See also ***critical path*** and ***float*** (Chapter 10).
standards	A standard or established way of doing things. Standards ensure that products work together, from home plumbing fixtures to the modules in a modern computer. They ensure the health and safety of products (Chapter 3).
state	The state of a system represents the net effect of all the previous inputs to the system. Since the state characterizes the history of previous inputs, it is often synonymous with the word memory (Chapter 6).
state diagram (machine)	Diagram used to describe systems with memory. It consists of states and transitions between states (Chapter 6).
static view	The static view is part of the ***Unified Modeling Language***. The ***intention*** of the static view is to show the classes in a system and their relationships. The static view is characterized by a class diagram (Chapter 6).
step-by-step test	A step-by-step test case is a prescription for generating a test and checking the results. It is most effective when the test consists of a complex sequence of steps (Chapter 7).

Term	Definition
strengths and weakness analysis	A technique for the evaluation of potential solutions to a design problem where the strengths and weaknesses are identified (Chapter 4).
structure charts	Specialized block diagrams for visualizing functional software designs. They employ input, output, transform, coordinate, and composite modules (Chapter 5).
strict liability	A form of ***liability*** that focuses only on the product itself—if the product contains a defect that caused harm, the manufacturer is liable (Chapter 11).
stub	A stub is a device that is used to simulate a subcomponent of a system during testing. Stubs simulate inputs or monitor outputs from the unit under test (Chapter 7).
subsystem design specification	Engineering requirements for subsystems that are constituents of a larger, more complex system (Chapter 3).
system integration	Phase in the ***design process*** where all of the subsystems are brought together to produce a complete working system (Chapter 1).
task	Tasks are actions that accomplish a job as part of a project plan. Also see ***activity*** and ***deliverable*** (Chapter 10).
Team Process Guidelines	Guidelines developed by a team that govern their behavior and identify expectations for performance (Chapter 9).
technical specification	A list of the technical details for a given system, such as operating voltages, processor architecture, and types of memory. The technical specification is fundamentally different from a requirement in that it indicates what was achieved in the end versus what a system needs to achieve from the outset. (Chapter 3).
test coverage	Test coverage is the extent to which the test cases cover all possible ***processing paths*** (Chapter 7).
test phase	Phase in the design process where the system is tested to demonstrate that it meets the requirements (Chapters 1 and 7).
testable	A design is testable when a failure of a component or subsystem can be quickly located. A testable design is easier to debug, manufacture, and service in the field (Chapter 7).
top-down design	An approach to design in which the designer has an overall vision of what the final system must do, and the problem is partitioned into components, or subsystems that work together to achieve the overall goal. Then each subsystem is successively refined and partitioned as necessary. This is contrasted to ***bottom-up*** design (Chapter 5).

Term	Definition
tort	The basis for which a lawsuit is brought forth (Chapter 11).
trade secret	An approach to protecting intellectual property where the information is held secretly, without ***patent*** protection, so that a competitor cannot access it (Chapter 11).
under-specificity	This refers to a state of the ***Requirements Specification***. When it is under-specified, requirements do not meet the needs of the user and/or embody all of the requirements needed to implement the system (Chapter 3).
Unified Modeling Language (UML)	A modeling language that captures the best practices of object-oriented system design. It encompasses six different system views that can be used to model electrical and computer systems (Chapter 6).
unit test	A unit test is a test of the functionality of a system module in isolation. It establishes that a subsystem performs a single unit of functionality to some specification (Chapter 7).
use-case view	The use-case view is part of the ***Unified Modeling Language***. Its ***intention*** is to capture the overall behavior of the system from the user's point of view and to describe cases in which the system will be used (Chapter 6).
utilitarian ethics	In utilitarian ethics, decisions are made based upon the decision that brings about the highest good for all, relative to all other decisions (Chapter 11).
validation	The process of determining whether the requirements meet the needs of the user (Chapter 3).
value	A value is something that a person or group believes to be valuable or worthwhile. Also see ***principles*** and ***morals*** (Chapter 11).
variable costs	Variable costs vary depending upon the process or items being produced, and fluctuate directly with the number of units produced (Chapter 10).
variant design	A formal categorization of design projects. They represent the design of existing systems, where the intent is to improve performance or add features (Chapter 2).
verifiable	Refers to a property of an engineering requirement. It means that there should be a way to measure or demonstrate that the requirement is met in the final system realization (Chapter 3).
vertical thinking	A linear, or sequential, thought process that proceeds logically towards the solution of a problem. It seeks to eliminate incorrect solutions. It is contrasted to ***lateral thinking*** (Chapter 4).

Term	Definition
whistleblower	A person who goes outside of their company or organization to report an ethical or safety problem (Chapter 11).
white box test	White box tests are those that are conducted with knowledge of the internal working of the unit under test (Chapter 7).
work breakdown structure	The work breakdown structure (WBS) is a hierarchical breakdown of the tasks and deliverables that need to be completed in order to accomplish a project (Chapter 10).
working group	A group of individuals working in isolation, who come together occasionally to share information. Part of the Katzenbach and Smith team model (Chapter 9).