problem seeking

AN
ARCHITECTURAL
PROGRAMMING
PRIMER

5TH EDITION

WILLIAM M. PEÑA
STEVEN A. PARSHALL

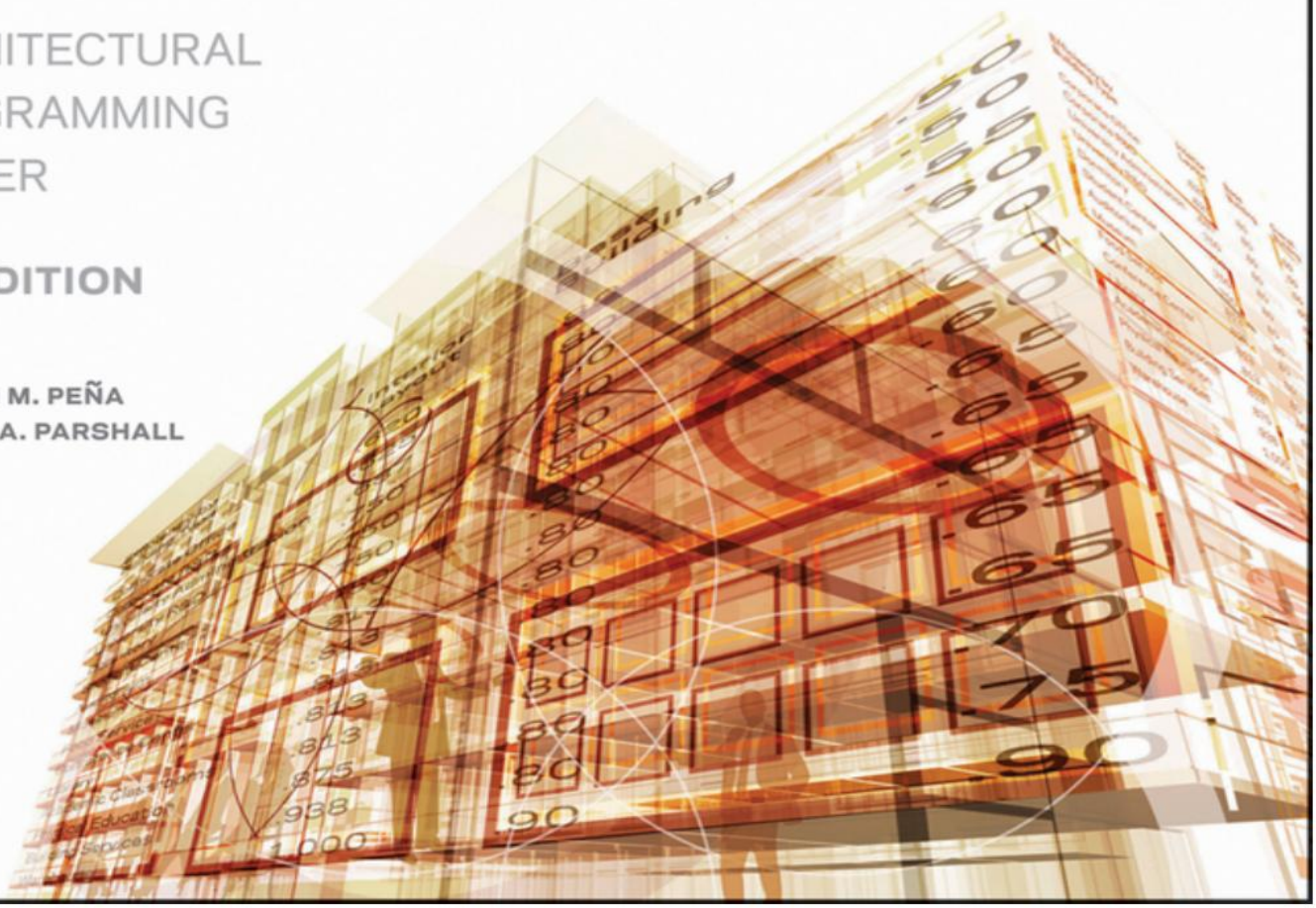


Table of Contents

Part Two: How to Use the Method

Fifth Edition

Problem Seeking

An Architectural Programming Primer

William M. Peña

Steven A. Parshall

John Wiley & Sons, Inc.

This book is printed on acid-free paper.♾

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.

Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Peña, William.

Problem seeking : an architectural programming primer / William M. Peña, Steven A. Parshall.—5th ed.

p. cm.

Includes bibliographical references and index.

ISBN 978-1-118-08414-4 (cloth); ISBN 978-1-118-13361-3 (ebk); ISBN 978-1-118-13362-0 (ebk); ISBN 978-1-118-15287-4 (ebk); ISBN 978-1-118-15292-8 (ebk); ISBN 978-1-118-15293-5 (ebk)

1. Architecture—Data processing.I. Parshall, Steven, 1951–-II. Title.III. Title: Architectural programming primer.

NA2728.P462012

720.285'536—dc23

2011020611

Foreword

The fifth edition of *Problem Seeking: An Architectural Programming Primer* is written for clients, architects, and students. The broad range of principles and techniques presented in this book has evolved over 50 years of architectural practice. In 1969, William Peña wrote the first edition of the book, and it was used in 1973 by the National Council of Architectural Registration Boards as a basis for the predesign section of the professional exam.

In 1994, Hellmuth, Obata + Kassabaum (HOK) acquired CRSS Architects, which had evolved from the original firm of Caudill, Rowlett and Scott (CRS). Many of the principles and techniques presented in this book can be attributed to Bill Caudill, one of the founders of CRS, and an AIA Gold Medalist. HOK's practice was founded on the same principle as CRS—both firms viewed design as problem solving.

William Peña (“Willie”) dedicated his professional career to the definition, development, and pioneering of architectural programming. He became the champion, teacher, and mentor to countless professionals who followed his path as specialists in the analysis of architectural problems.

In the end, *Problem Seeking* is not the product of one person, but the theoretical and practical contribution of many professionals who worked at CRS and now HOK. Assisting Willie with the publication of the book have been several co-authors, including: John Focke, FAIA (first and second editions), William Caudill, FAIA (second edition), Kevin Kelly, FAIA (third edition), and Steven Parshall, FAIA (third, fourth, and fifth editions).

While the method has adapted to new considerations and techniques with each edition of the book, the principles outlined in the first part of the book, “The Primer,” have withstood the test of time. As we look forward, the role of programmer as analyst and information manager will increase significantly as the profession adopts Building Information Modeling (BIM) to meet client expectations for more sustainable and integrated design solutions.

HOK is proud to continue the tradition of involving and interacting with clients in architectural programming as the first step of the design process.

Bill Hellmuth

President, HOK, Inc.

Preface

This book is the fifth edition of *Problem Seeking: An Architectural Programming Primer*. The first edition, in 1969, was based on 20 years of prior research and practice in architectural programming. The subsequent editions evolved over the next 40 years, reflecting changes in communication techniques and expanded scope of applications, although the original theory remained intact. This edition, then, has the advantage of some 60 years of professional application experience—**indicating a practice-tested validity.**

This is a two-part book. **Part One** is a primer on programming. It is written to help you understand **one programming method,** whether you are an architect, a student, or a client getting ready to start a building project. **Part Two** explains **how to apply the method;** it comprises a collection of definitions, examples, considerations, activities, and techniques that expand on the principles explained in the primer.

What is new in this edition?

Published in 2001, the fourth edition of the book has been read, primarily, by architectural practitioners and, secondly, by students as a college course text book.

In the fifth edition we have **simplified Part One: The Primer.** In **Part Two, regarding methods,** we take up new topics that have emerged in the profession since writing the fourth edition. It addresses the role of programming when considering **sustainability**in the design process, and explains how **technology** has enabled new techniques for project delivery, team communication, and information management.

While the Problem Seeking® process has withstood the test of time as a powerful problem analysis method, the content and technology of architectural practice have evolved over the past decade.

Today, **sustainability** has become a major consideration in architectural projects throughout the world. In 1998, the U.S. Green Building Council (USGBC) established the Leadership in Energy and Environmental Design (LEED) standards and system for rating green buildings.

In addition to updates in content, sustainability practices encourage an integrated design approach that is highly participatory among all the stakeholders in the design, construction, and operation of buildings. Bill Caudill first introduced this type of collaboration in his book, *Architecture by Team*, in 1971. The principles regarding the user on the team, effective group action, and participatory process are embedded in Problem Seeking® as well. The USGBC encourages the organization of a work session at the outset of the project, during which the key stakeholders establish project goals and determine the level of sustainability to be achieved in design and construction. Once again, these predesign sustainability activities are easily incorporated into the programming process as outlined in *Problem Seeking*.

Two of the emerging trends in architectural practice that are enabled by technology involve Building Information Modeling (BIM) and Integrated Project Delivery (IPD).

BIM is the process of generating and managing building data during the design process, including the program of requirements. Typically, it uses three-dimensional, real-time, parametric modeling software to increase productivity in building design and construction. The process produces the building information model, which encompasses building geometry, spatial relationships,

geographic information, and quantities and properties of building components. The BIM process begins with capturing the program of requirements for each phase of the design process.

IPD is a project delivery method that integrates people, systems, business structures, and practices into a process that collaboratively harnesses the expertise and knowledge of stakeholders to optimize project results, increase value to the owner, reduce waste, and maximize efficiency throughout the phases of project delivery.

The fifth edition explains how the role of the programmer may expand to encompass the program of requirements for the life cycle of a building. This involves an extended information management role for the programmer. Information management has been a cornerstone of the Information Index, organizing information, a phased process, data clog, and processing and discarding information. While principles are fundamental to the programming process, the techniques and tools that a programmer uses today take advantage of current digital technology and software to capture and manage information.

Not only has technology improved the programmer's ability to manage information, it is allowing new forms of interaction and collaboration among the project team. Advanced collaboration technologies are proving a viable alternative to the traditional on-site squatters' technique. Now programmers can facilitate virtual squatting with clients and project team members located worldwide without ever leaving their place of work.

William M. Peña, FAIA Founder, Caudill, Rowlett and Scott, Inc.

Steven A. Parshall, FAIA Senior Vice President, HOK Inc.

Acknowledgments

HOK Team

Editor:	Melinda Parshall
Project Manager:	Lauren Gibbs
Special Contributors:	Erik Andersen, Robin Ellerthorpe, William Hellmuth, Frank Kutilek, Eberhard Laepple
Graphics & Photography:	Gerald Callo, HOK Visual Communications
Cover Graphics:	Jay Dacon, HOK Visual Communications

We are grateful to those programmers, past and present, who have contributed to this book—some much more than others—but all contributing more than they realize.

Part One

Problem Seeking

An Architectural Programming Primer

Overview

The Primer

Good buildings don't just happen. They are planned to look good and perform well. They come about when good architects and good clients join in thoughtful, cooperative effort. Programming the requirements of a proposed building is the architect's first task, often the most important.

There are a few underlying principles that apply to programming—whether the most complex hospital or a simple house. This book concerns these principles.

Programming concerns five steps:

1. Establish **Goals.**
2. Collect and analyze **Facts.**
3. Uncover and test **Concepts.**
4. Determine **Needs.**
5. State the **Problem.**

The approach is at once simple and comprehensive—simple enough for the process to be repeatable for different building types, and comprehensive enough to cover the wide range of factors that influence the design of buildings.

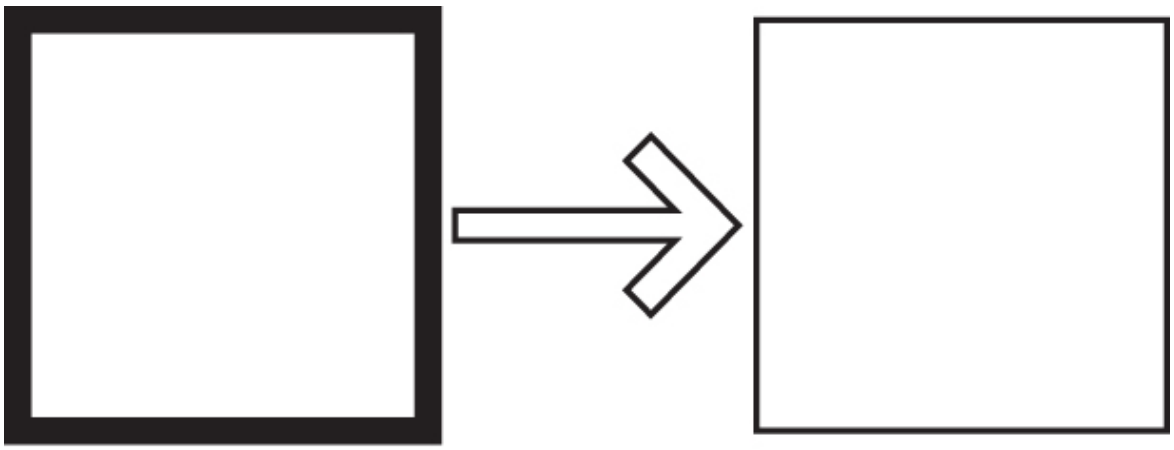
The five-step process can be applied to most any discipline—banking, engineering, or education—but when applied specifically to architecture, it has its proper content that is an architectural product: a room, a building, or a town. The principle of this process is that a product will have a much better chance of being successful if, during the design, four major considerations are regarded simultaneously.

These considerations (or design determinants) indicate the types of information needed to define a comprehensive architectural problem:

Function Form Economy Time

Architectural programming, therefore, involves an organized method of inquiry—a five-step process interacting with four considerations.

The Search



Programming is a process. What kind? *Webster's* spells it out specifically: “A process leading to the statement of an architectural problem and the requirements to be met in offering a solution.”

This process, derived from the definition and referred to as the five-step process, is basic. The word “basic” is used advisedly. Since the advent of systematic programming six decades ago, different degrees of sophistication have evolved. But the procedures presented here remain basic to all.

Back to the definition.

Note “statement of an architectural problem.” This implies problem solving. Although usually identified with scientific methods, problem solving is a creative effort. There are many different problem-solving methods, but only those few that emphasize goals and concepts (ends and means) can be applied to architectural design problems.

Almost all problem-solving methods include a step for problem definition—stating the problem. But most of the methods lead to confusing duality—finding out what the problem is and trying to solve it at the same time. You can't solve a problem unless you know what it is.

What, then, is the main idea behind programming? It's the search for sufficient information to clarify, to understand, and to state the problem.

If programming is problem seeking, then design is problem solving.

These are two distinct processes, requiring different attitudes, even different capabilities. Problem solving is a valid approach to design when, indeed, the design solution responds to the client's design problem. Only after a thorough search for pertinent information can the client's design problem be stated: “Seek and you shall define!”

Programmers and Designers

Who does what? Do designers program? They can, but it takes highly trained architects who are specialized in asking the right questions at the right time, who can separate wants from needs, and who have the skills to sort things out. Programmers must be objective (to a degree) and analytical, at ease with abstract ideas, and

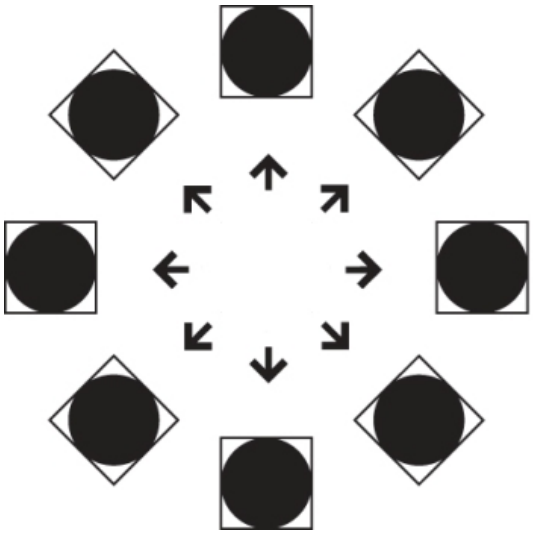
able to evaluate information and identify important factors while postponing irrelevant material. Designers can't always do this. Designers generally are subjective, intuitive, and facile with physical concepts.

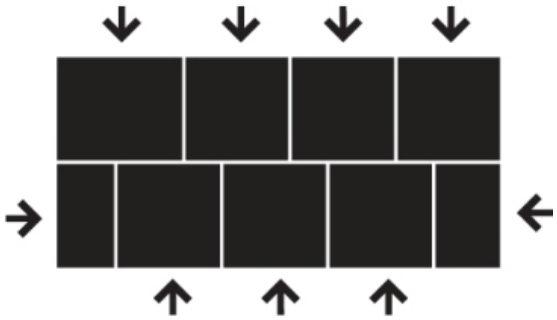
Qualifications of programmers and designers are different. Programmers and designers are separate specialists because the problems of each are very complex and require two different mental capabilities: one for analysis, another for synthesis.

It may well be that one person can manage both analysis and synthesis. If so, he or she must be of two minds and use them alternately. However, for clarity, these different qualifications will be represented by different people—programmers and designers.

Photo courtesy of HOK

Analysis and Synthesis





The total design process includes two stages: analysis and synthesis. In analysis, the parts of a design problem are separated and identified. In synthesis, the parts are put together to form a coherent design solution. The difference between programming and design is the difference between analysis and synthesis.

Programming *Is* analysis. Design *Is* synthesis.

You may not perceive the design process in terms of analysis and synthesis. You may even question problem solving as an approach. You may think of the design process as a creative effort. It is. But the creative effort includes similar stages: Analysis becomes preparation or exposure, and synthesis becomes illumination or insight. The total design process is, indeed, a creative process.

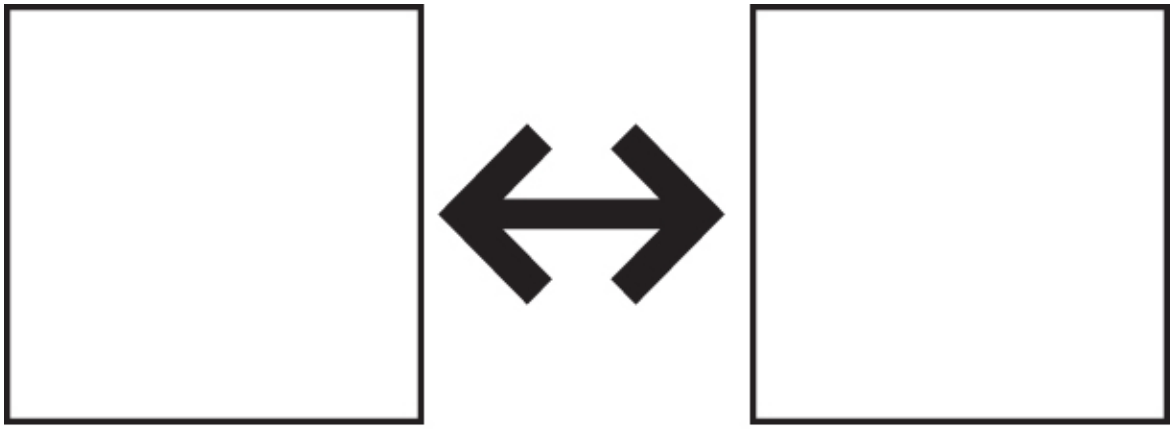
Does programming inhibit creativity? Definitely not! Programming establishes the considerations, the limits, and the possibilities of the design problem. (We prefer “considerations” to “constraints” to avoid being petulant.) Creativity thrives when the limits of a problem are known.

> *Sometimes I think we arrive at a solution before we know what the problem is. We say: “My next design will*

be Round!” without logic or analysis.

—William Peña

The Separation



Programming precedes design just as analysis precedes synthesis. The separation of the two is imperative and prevents trial-and-error design alternatives. Separation is central to an understanding of a rational architectural process, which leads to good buildings and satisfied clients.

The problem-seeking method described in this book requires a **distinct separation of programming and design.**

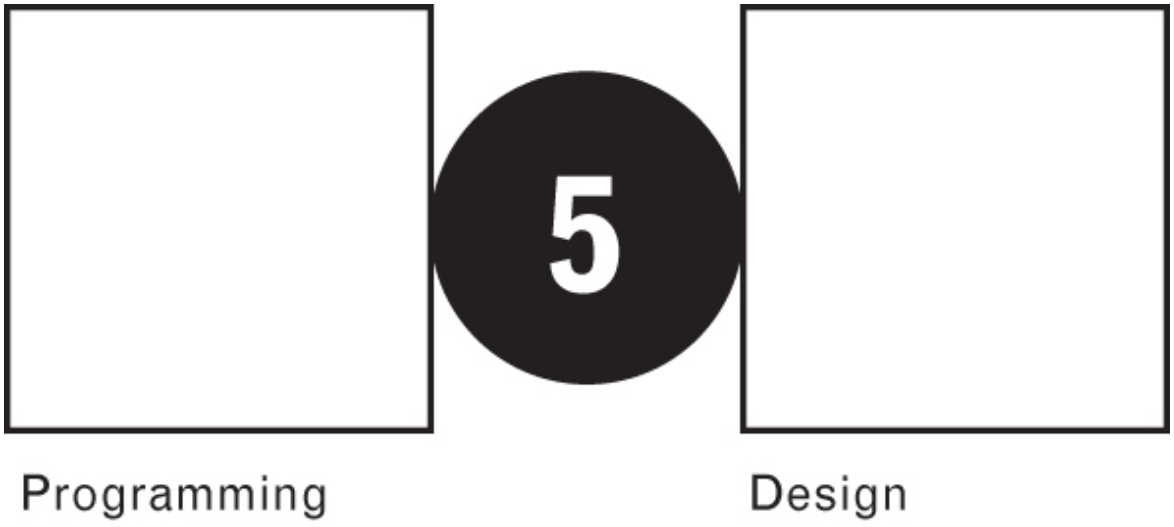
Most designers love to draw, to make “thumbnail sketches,” as they used to call these drawings. Today, the jargon favors “conceptual sketches” and “schematics.” Call them what you will, they can be serious deterrents in the planning of a successful building if done at the wrong time—before programming or during the programming process. Before the whole problem is defined, solutions can only be partial and premature. A designer who can't wait for a complete, carefully prepared program is like

the tailor who doesn't bother to measure a customer before starting to cut the cloth.

Experienced, creative designers withhold judgment and resist preconceived solutions and the pressure to synthesize until all the information is in. They refuse to make sketches until they know the client's problem. They believe in thorough analysis before synthesis. They know that programming is the prelude to good design—although it does not guarantee it.

> *Corita Kent, artist and educator, wrote, "Rule Eight: Don't try to create and analyze at the same time. They are two different processes."*
>
> —Today You Need a Rule Book, 1973

The Interface



The product of programming is a statement of the problem. Stating the problem is the last step of the five-step process in problem seeking (programming); it is also the first step in problem solving (design). **The problem statement, then, is the interface between programming and design.** It's the baton in a relay race. It's the handoff from programmer to designer. In

any case, the problem statement is one of the most important documents in the chain that comprises the total project delivery system.

While many theorists extol the virtues of the problem statement, few practitioners stop to formulate a statement, to verbalize it. This programming method requires that you actually write out a clear statement of the problem. Since this statement is the first step in design, as well as the last step in programming, its composition must be the joint effort of the designer and the programmer.

Process

Five Steps

The competent programmer always keeps in mind the steps in programming: **(1) Establish Goals, (2) Collect and Analyze Facts, (3) Uncover and Test Concepts, (4) Determine Needs,** and **(5) State the Problem.** The first three steps are primarily the search for pertinent information. The fourth is a feasibility test. The last step is distilling what has been found.

Curiously enough, the steps are alternately qualitative and quantitative. Goals, con-cepts, and the problem statement are essentially qualitative. Facts and needs are essentially quantitative.

Programming is based on a combination of interviews and work sessions. Interviews are used for asking questions and collecting data, particularly during the first three steps. Work sessions are used to verify information and to stimulate client decisions—particularly during the fourth step.

Briefly, the five steps pose these questions:

1. Goals: *What* does the client want to achieve, and *why*?

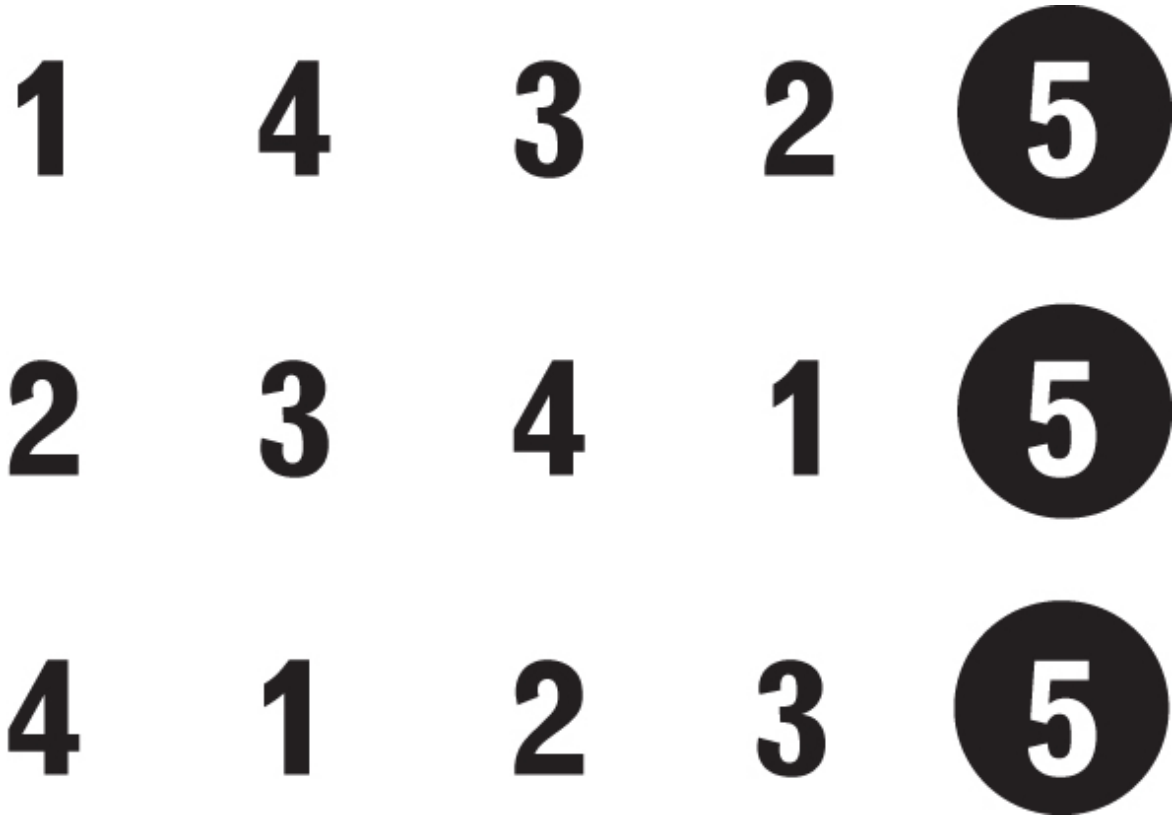
2. Facts: *What* do we know? *What* is given?

3. Concepts: *How* does the client want to achieve the goals?

4. Needs: *How* much money and space? What level of quality?

5. Problem: *What* are the significant conditions affecting the design of the building? What are the general directions the design should take?

Procedure



The five steps, then, are not inflexibly strict. They usually have no consistent sequence; nor is the information scrupulously accurate. For example, a 10,000-student university, a 300-bed hospital, and a 25-student classroom are only nominal rather than actual sizes. Information sources are not always reliable, and predictive capabilities may be limited.

The steps and the information, then, do not have the rigor or the accuracy of a math-ematical problem.

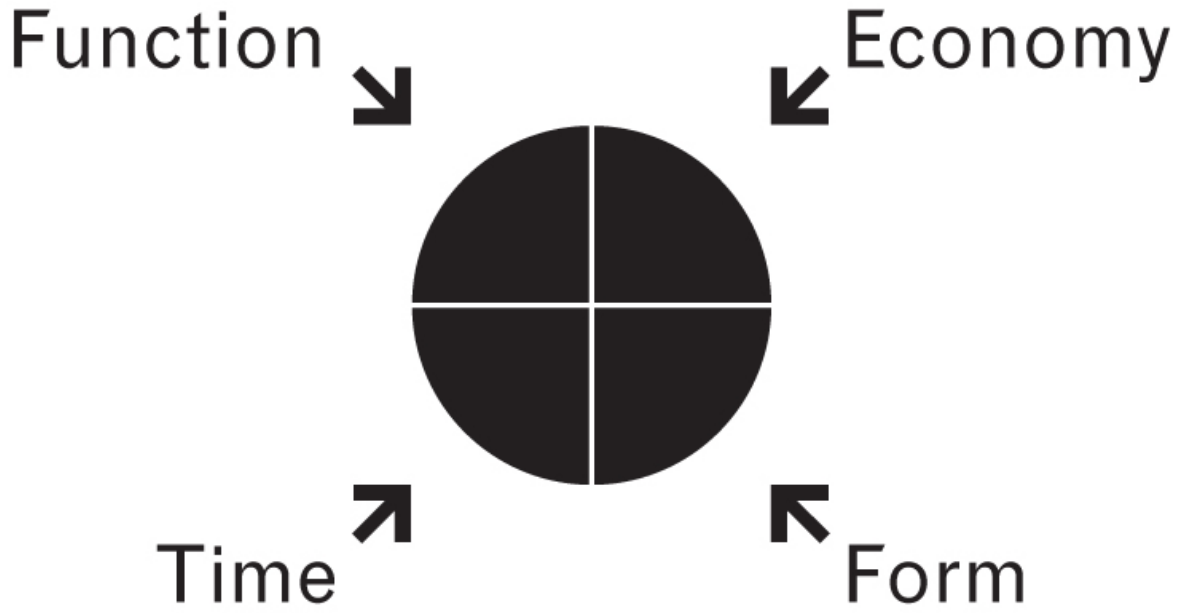
Programming, therefore, is a heuristic process and not an algorithm. As such, even good programming cannot guarantee finding the right problem, but it can reduce the amount of guesswork. The method is just as good as the judgment of the people involved.

Working through the steps in numerical sequence is preferable; theoretically, this is the logical order. But, in actual practice, **steps may be taken in a different order or at the same time**—all but the last step. It is frequently necessary, for example, to start with a given list of spaces and a budget (fourth step) before asking about Goals, Facts, and Concepts (first, second, and third steps). It usually is necessary to work on the first four steps simultaneously, cross-checking among them for the integrity, usefulness, relevance, and congruence of information.

The fifth step is taken only after marshalling all the previous information, extracting, abstracting, and getting to the very essence of the problem.

Considerations

The Whole Problem



It's important to search for and find the whole problem. To accomplish this, the prob-lem must be identified in terms of **Function, Form, Economy,** and **Time.** Classifying information accordingly simplifies the problem while maintaining a compre-hensive approach. A wide range of factors makes up the whole problem, but all can be classified in the four areas that serve later as design considerations.

Too little information leads to a partial statement of the problem and a premature and partial design solution. The appropriate amount of information is broad enough in scope to pertain to the whole design problem, but not so broad as to pertain to some universal problem. As the Spanish proverb states: “He who grasps too much, squeezes little.” Grasp only what you can manage and what will be useful to the designer.

As a professor might say, “Before you answer individual questions, be sure to look at the whole examination.” Designers should look at the whole problem before starting to solve any of its parts. How can a designer who does not have a clear understanding of the whole problem come up with a comprehensive solution?

Four Considerations

Function	1 People 2 Activities 3 Relationships
Form	4 Site 5 Environment 6 Quality
Economy	7 Initial budget 8 Operating costs 9 Life-cycle costs
Time	10 Past 11 Present 12 Future

Take a closer look at Function, Form, Economy, and Time. **There are three key words to each consideration:**

Function implies “what's going to happen in the building.” It concerns activities, relationship of spaces, and people—their number and characteristics. Key words are: (1) people, (2) activities, and (3) relationships.

Form relates to the site, the physical environment (psychological, too), and the quality of space and

construction. Form is what you will see and feel. It's “what is there now” and “what will be there.” Key words are (4) site, (5) environment, and (6) quality.

Economy concerns the initial budget and quality of construction, but also may include consideration of operating and life-cycle costs. Key words are: (7) initial budget, (8) operating costs, and (9) life-cycle costs.

Time has three classifications—past, present, and future—which deal with the influ-ences of history, the inevitability of changes from the present, and projections into the future. Key words are: (10) past, (11) present, and (12) future.

Framework

	1	2	3	4	5
Function					
Form					
Economy					
Time					

Use the four considerations to guide you at each step during programming. By estab-lishing a systematic set of relationships between the steps in problem seeking and these considerations, between process and content, a