

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.

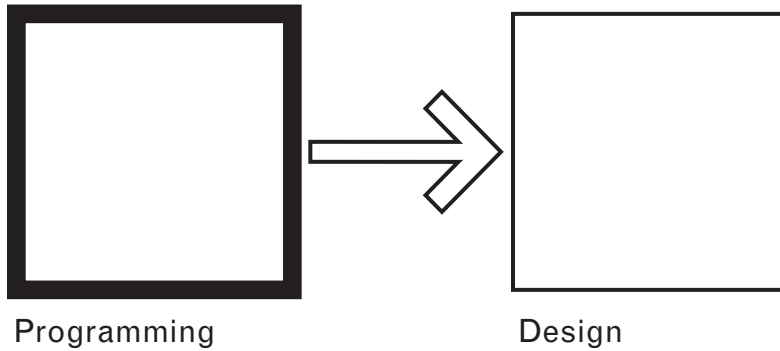
The five-step process can be applied to most any discipline—banking, engineering, or education—but when applied specifically to architecture, it has its proper content that is an architectural product: a room, a building, or a town. The principle of this process is that a product will have a much better 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.

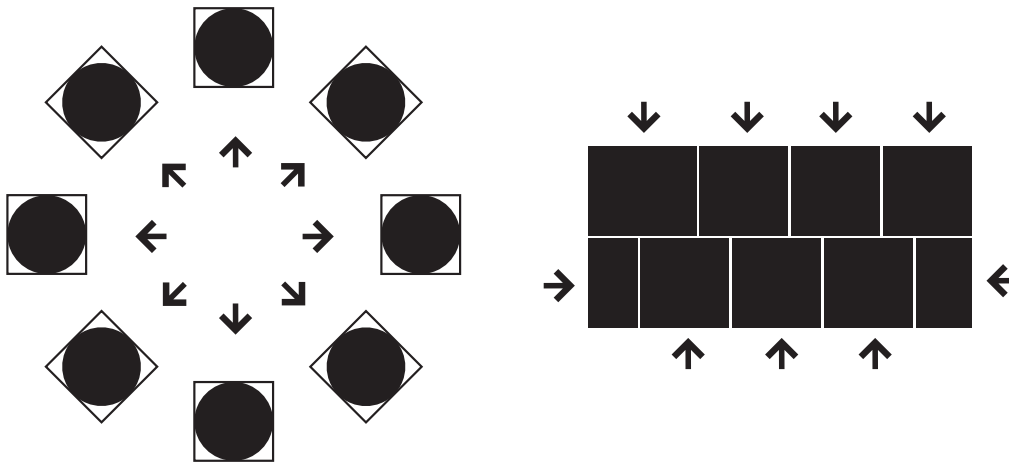
Qualifications of programmers and designers are different. Programmers and designers are separate specialists because the problems of each are very complex and require two different mental capabilities: one for analysis, another for synthesis.

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

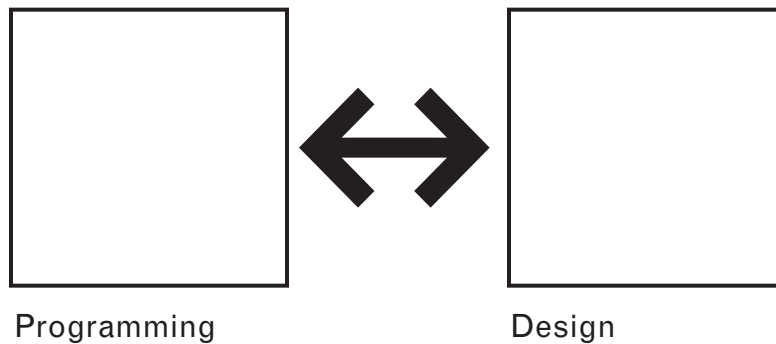
a creative effort. It is. But the creative effort includes similar stages: Analysis becomes preparation or exposure, and synthesis becomes illumination or insight. The total design process is, indeed, a creative process.

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

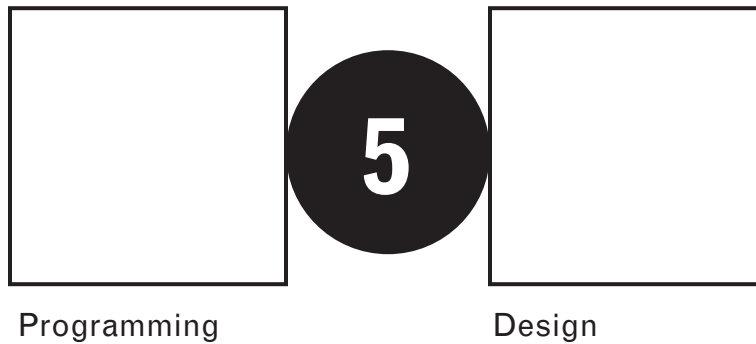
what you will, they can be serious deterrents in the planning of a successful building if done at the wrong time—before programming or during the programming process. Before the whole problem is defined, solutions can only be partial and premature. A designer who can't wait for a complete, carefully 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.