DOCUMENT RESUME

ED 037 930 EF 004 014

AUTHOR	Pena, William M.; Focke, John W.
TITLE	Problem Seeking. New Directions in Architectural Programming.
INSTITUTION	Caudill, Rowlett and Scott, Houston, Tex. Architects.
PUB DATE	69
NOTE	44p.
AVAILABLE FROM	Caudill, Rowlett & Scott, Architects, CRS Building, 1111 West Loop South, Houston, Texas 77027
EDRS PRICE	EDRS Price MF-$0.25 HC Not Available from EDRS.
DESCRIPTORS	Architects, *Architectural Programing, Communications, Data Collection, Data Processing, *Decision Making, Interdisciplinary Approach, *Methods, *Techniques

ABSTRACT

The rationale, principles, and methods of pre-design architectural programing are explained for those responsible for overall policy decision-making in the area of facility planning. This programing process provides an orderly framework that aids the architect in defining a client's total problem. A general background is given on data collection, team composition, communications, and various approaches to programing. Also discussed are communication between architect and client, techniques for processing information, and the future of programing. Diagrams and charts graphically illustrate each major topic. (TC)

Problem

Seeking

Problem Seeking: New directions in architectural programming

by
William M. Peña, AIA,
General Partner and Director of Programming
John W. Focke
Associate, Programming Department

Caudill Rowlett Scott
Architects, Planners, Engineers

This study is directed to business and facilities planning officials on the staffs of institutions, corporations and various public bodies. In addition, it is written for those who are charged, in an even more primary sense, with overall policy decision making in the matter of planning facilities.

In this day of changing building user requirements, inflated construction costs and swiftly evolving trends in education, health care, housing, leisure and business, it is crucial for those charged with planning these facilities to base their decisions on the true needs of their respective institutions. What kinds of facilities are needed? How much building will the budget buy? Conversely, what are the funds needed to meet requirements? What are the basic goals that must be met by the facility? What are the time limitations and priorities? Should a project be phased? How can new developments unknown at time of planning be accommodated in the overall definition of the problem? It is important to find answers to such questions before an organization thinking of expanding its facilities over-commits itself to a particular course of action. A course correction can be very costly.

Readers will gain from this report a valuable insight into how answers to these questions are best obtained. Its title was chosen deliberately: it is only by first *seeking out* the problem and defining it that a valid *solution* can be developed. Ways in which the architect-programmer can best work with a client to develop the data needed to define the problem, as well as techniques of evaluating the growing quantities and complexity of such data, are presented in the report's various chapters.

As traditional architectural services expand to include such pre-design programming, the client's role becomes more and more vital in reaching programming decisions on which sound design solutions can be based. The techniques of "problem seeking" have evolved over a long period of architectural practice. As presented in this report, they are not the product of one man, but the accumulated efforts of many members of the firm.

CRS is happy to present this report to its clients and friends.

William M. Peña

John W. Focke

November 1969

Numbers in parenthesis refer to drawings unless otherwise noted.

Programming is a specialized and often misunderstood term. It is "*a statement of an architectural problem* and the requirements to be met in offering a solution". While the term is used with other descriptive adjectives such as *computer* programming, *educational* programming, *functional* programming, etc., in this report programming is used to refer only to *architectural* programming.

Why programming? The client has a project with many unidentified sub-problems. The architect must define the client's total problem.

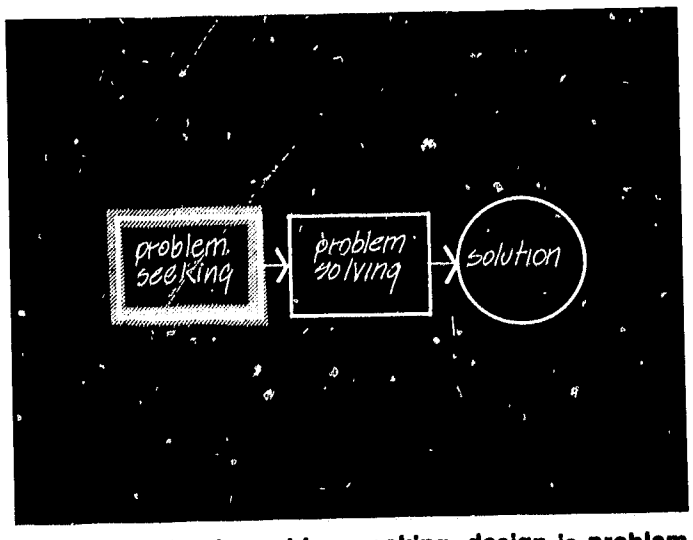
Design is problem solving; programming is problem seeking. The end result of the programming process is a statement of the total problem; such a statement is the element that joins programming and design. The "total problem" then serves to point up constitutent problems, in terms of four considerations, those of function, form, economy and time. The aim of programming is to provide a sound basis for effective design. The Statement of the Problem represents the essense and the uniqueness of the project. Furthermore, it suggests the solution to the problem by defining the main issues and giving direction to the designer. (1)

An Orderly Framework

The steps of the programming process are not inflexibly strict and the information is not scrupulously accurate. Yet, the steps form an orderly framework for the classification and documentation of information. In addition, they are communicable so as to make possible coordination of individual efforts within the client-architect team. The process is essentially an analytical one, but it does leave room for intuitive insight based on knowledge and experience. Thus, analysis through the classification and cross-linking of steps and considerations (function, form, economy, time) provides a framework for dialog and henceforth a better understanding between client and architect/planner

Who is "the Team?"

Successful projects result from the efforts of creative clients and talented architects. This is just as true during the programming process as in the design phase. Indeed, it is safe to say that *the major responsibility for creative thinking rests on the client in the programming phase and on the architect during the design phase.* Nevertheless, through



1 Programming is problem seeking, design is problem solving.

interaction on the team, the architect can help the client realize new program relationships by testing programmatic concepts and spawning alternatives. If, as occasionally happens, the client contribution lacks innovation, the architect still has opportunities for creativity, first in the Statement of the Problem, and later on through design.

Communication Within the Team

One of the prime prerequisites for effective programming is the way in which the client-architect team is organized. The team should be headed by two responsible group leaders who are able to coordinate the individual efforts of their group members, who can make decisions or cause them to be made, and who can establish and maintain channels within and between the groups.

The team concept requires a high degree of communication. It is the analytical approach, (described on page 11), with its explicit steps and considerations, which provides the format for productive dialog.

There are many techniques that can be used to involve people and to provide effective communication. Among these are: the "squatters" technique, the user's conference, gaming techniques, interviews and questionnaires.

Whether dealing with small groups or large ones, information must be presented and documented graphically if it is to be discussed and evaluated. Such visual aids as analysis cards and brown sheets (described on page 29) can prompt quick appraisal and understanding.

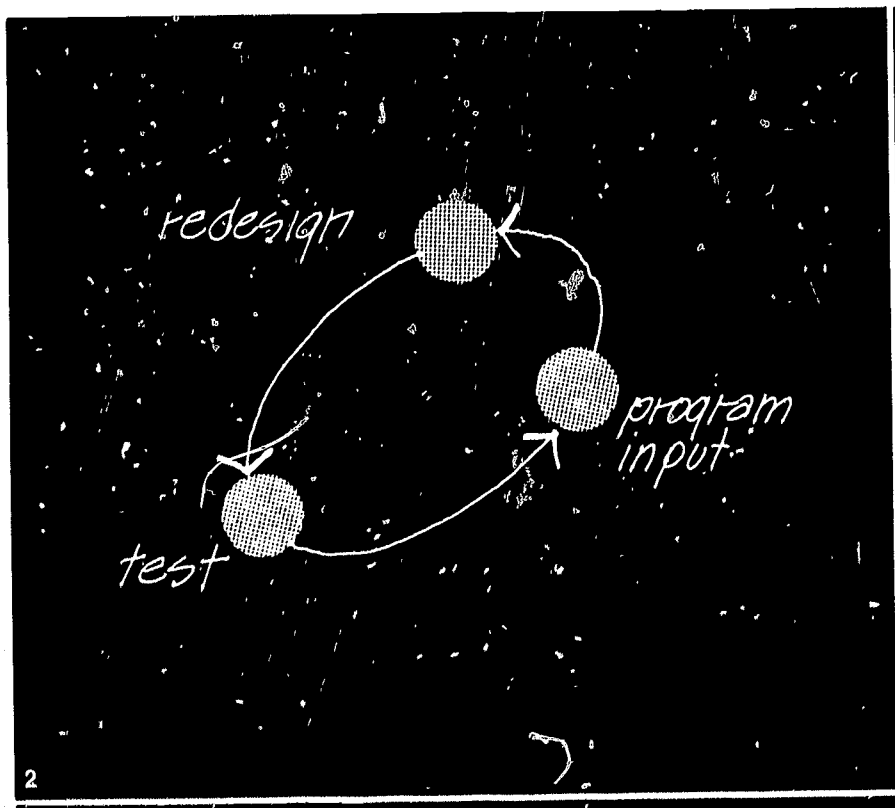
The Importance of Decision Making During Programming

How effective decision making is in the programming phase will determine the amount of trial and error in the design and subsequent phases of a project. This does not deny alternatives in design; it means merely that decisions in programming will provide a sounder base of requirements to be met. The less relevant the information, the fewer the decisions in programming, the more elusive the problem and the more chance for error in the solution.

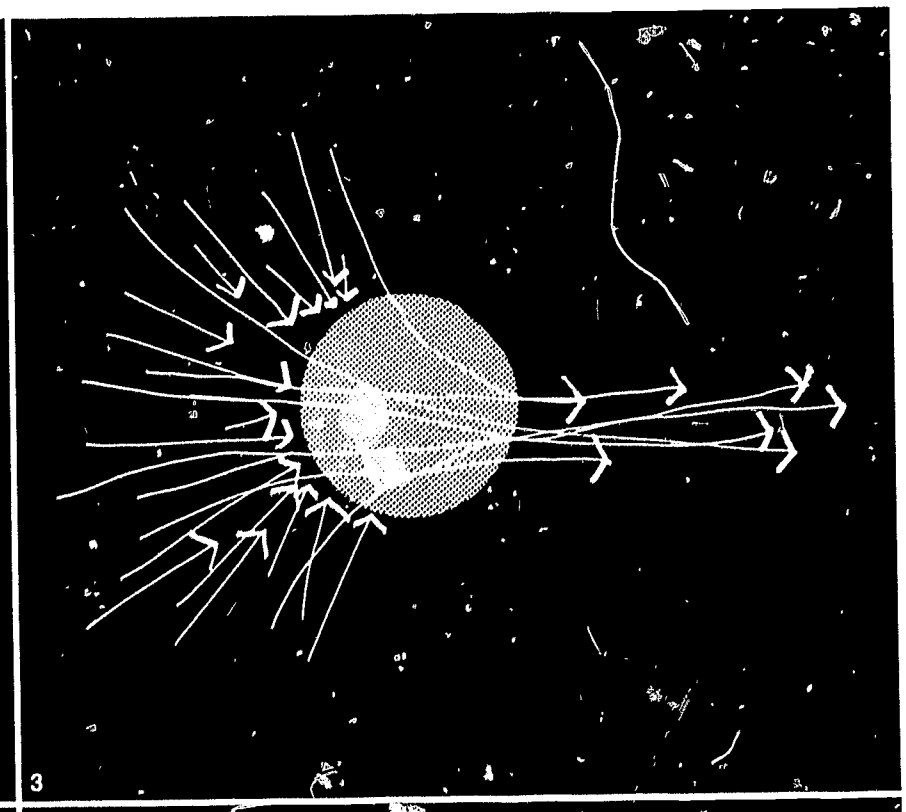
How Much Information is Enough?

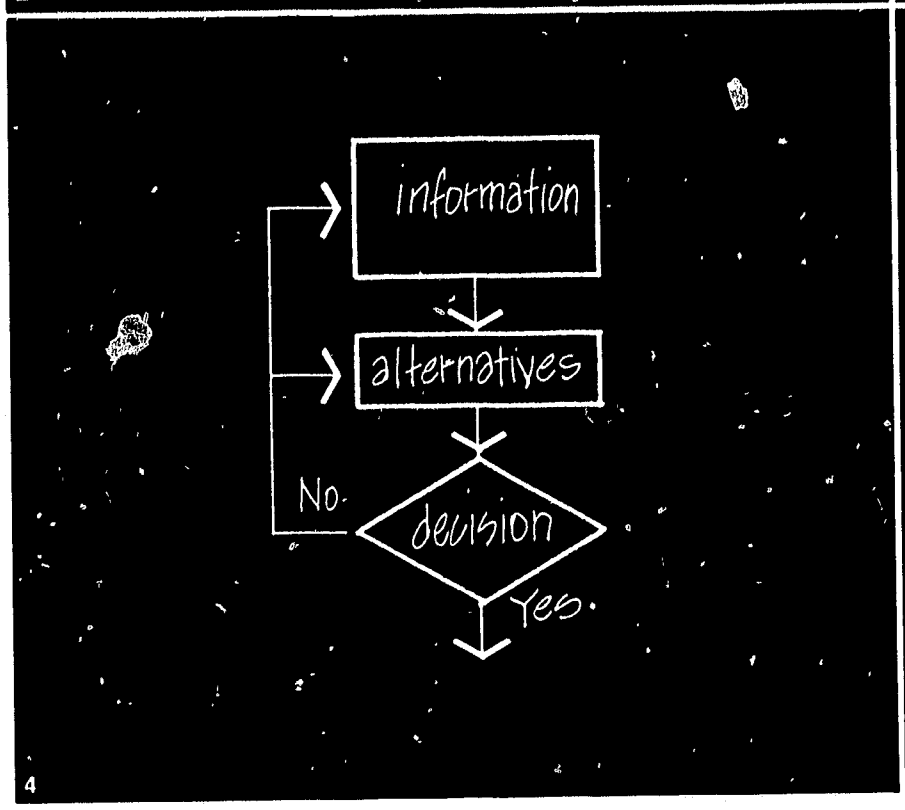
If a client approaches the architect with very little information, the architect may have to respond by programming through

2 Programming through design, testing, and redesign is inefficient.
3 Discrimination between major ideas and details is necessary to avoid confusion in problem solving.
4 The conglomerate-client requires organized precedures for communication, processing information and decision making.
5 The client is involved in the process.

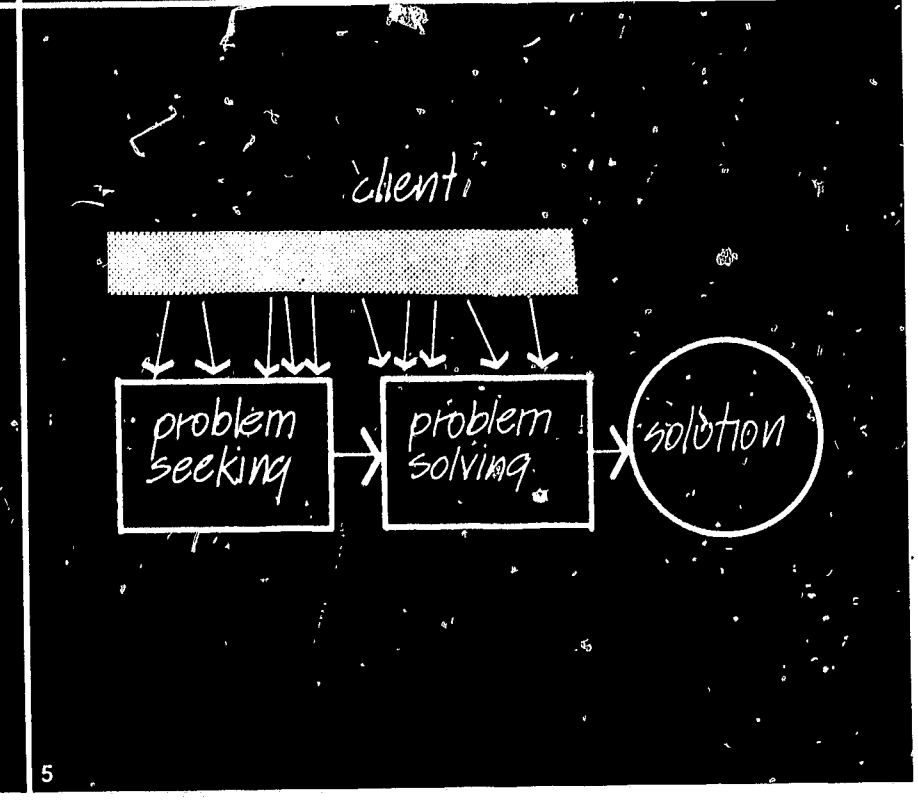


2

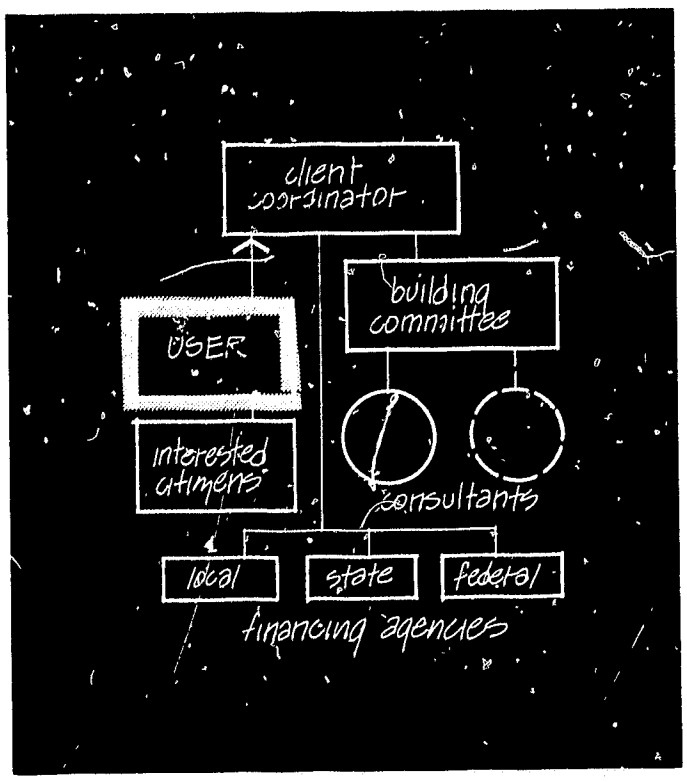

3



4



5



6 The client group includes many participants.

design. He could produce sketch after sketch and plan after plan trying to satisfy undefined requirements. Programming through design can involve misuse of talent and, indeed, risks creating a "solution" to the wrong problem. (2)

On the other hand, a client may present the architect with too much information but involving mostly irrelevant details. The risk here is that the architect's solution will be based on details rather than on major ideas. In this case, the architect must plough through an abundance of information and discriminate between major ideas and details. (3)

The analytical procedure used by CRS provides the framework for decision making. Within it the architect can help the client identify and make those decisions that need to be made prior to design. Within it, the architect can suggest alternatives and other information to bring about decisions. There are times when the architect must evaluate the gain and risks in order to *stimulate* a decision. Yet, note the emphasis on client decisions; the architect merely participates and at most, recommends. (4)

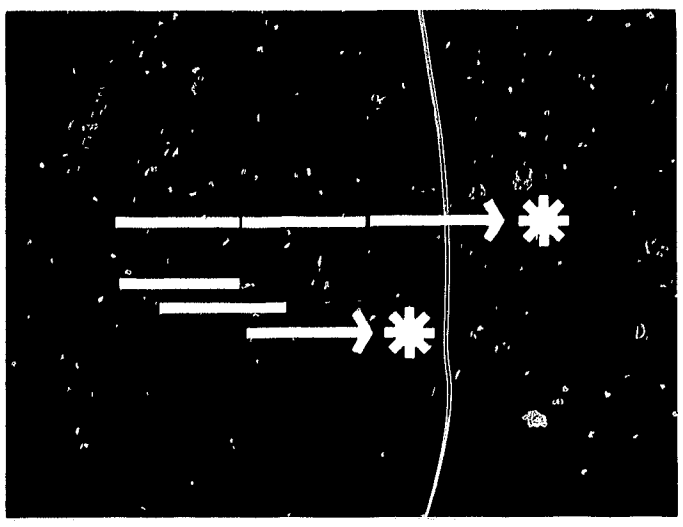
The new sophisticated client wants to know how his project will be processed and when he will be involved. He wants to remove the mystique usually associated with the programming and design of his project. (5)

The need for organized-programming becomes even more important with the conglomerate client involving the client-owner, the client-user, governmental agencies, special interest groups and many others. The conglomerate client requires an organized framework for dialog within which numerous opinions, facts and ideas must be classified and analyzed for decision making. The more people that are involved, the greater the need for effective communication. (6)

Not only are projects becoming more complex and the client more sophisticated, but the time for project delivery is becoming more compressed. This requires more refined and efficient processes. (7)

The Two Types of Concepts

Two terms need to be understood and added to the glossary of architectural practice: "Programmatic concepts" and "design concepts." Programmatic concepts refer to ideas



7 Greater efficiency in procedures is necessary with compressed time schedules.

intended mainly as solutions to the client's own management problems so far as they concern function and organization. Design concepts, on the other hand, refer to ideas intended as physical solutions to architectural problems.

Programmatic concepts and design concepts are so closely related that one is mistaken for the other. Design concepts are the physical response to programmatic concepts. For example, *open planning* is a physical response to *integration* of activities. In practice the confusion is compounded because most architects and some clients tend to think more easily in physical terms.

Programmatic concepts must be stated abstractly so as not to inhibit design alternatives unnecessarily. For example, the programmatic concept of *decentralization* may find a design response in either *compactness* (vertical or horizontal) or *dispersion* (varying degrees).

Wants vs. Needs

Wants must be distinguished from real needs. A client may have a clear understanding of his functional requirements, but neglect to relate them to his financial position.

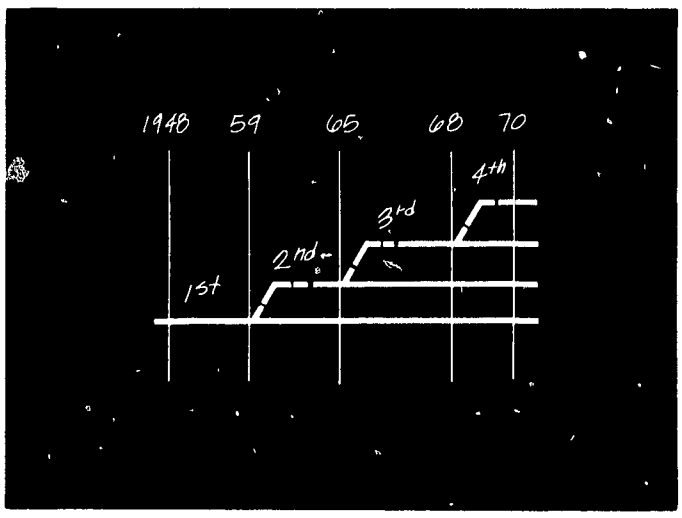
A wants vs. needs situation occurs whenever the client defines his problem in terms of architectural solutions rather than functional requirements. The architect's job then is to determine those assumptions on which the client based his solutions and to evaluate these.

Four Levels

Problem seeking procedures are applied, with due modification, through the first four levels of sophistication in programming. (8)

The first level consists largely of the traditional architectural service in which the program contribution by the client is sufficient to enable the architect to state the problem.

The second level occurs when input by the client is insufficient and the architect must provide management consultant service (in whatever discipline the project falls, such as education, health, business, etc.): he recommends if, and what, the client should build, and finally, states the problem.



8 Problem seeking has developed through three levels of sophistication and is beginning a fourth level involving urban problems.

These management-type non-building services could be reinforced with computer applications.

The third level includes complex programming services often involved in multi-phase construction projects, buildings using system construction, projects entailing negotiable user requirements, concurrent scheduling and mathematical analytical procedures. In virtually all such cases, the computer is an indispensable tool to aid in decision making.

With the inevitable evolution of programming, we look forward to *a fourth level.* CRS is already developing such techniques, which will help clients deal more effectively with the vastly complex problems of urban planning. For example, the computer has already begun to show its usefulness in mass data processing, educational facilities inventory, traffic flow simulation, economic model simulation and planning cycle simulation.

As fourth level programming evolves, sociologists, mathematicians, computer experts and communications specialists will become more involved as members of problem seeking teams.

Fourth level programming will provide the basis for a continuing service on a cyclical basis. For the first time, programming will be keyed to political, financial and other important cyclical public functions.

CRS Approach to Programming

CRS approaches programming in a way that is at once cooperative, analytical and creative.

The Cooperative Approach

Since so many people play a role in reaching the architectural solution to a client's problem (at the programming stage, there are the owner, the users, the architect, special consultants, governmental agencies, etc.), successful programming can come about only with overall cooperation among all participants.

Programming requires the joint effort of two groups—the client group and the architectural group. Together they form the programming team. Each group assigns to one person complete authority to make decisions or to cause decisions to be made; this simplifies communication between the two groups. Collaboration between the two group leaders can then result in productive work. (9)

The client group is primarily responsible for information concerning the client's functional and organizational requirements, and the financial objectives.

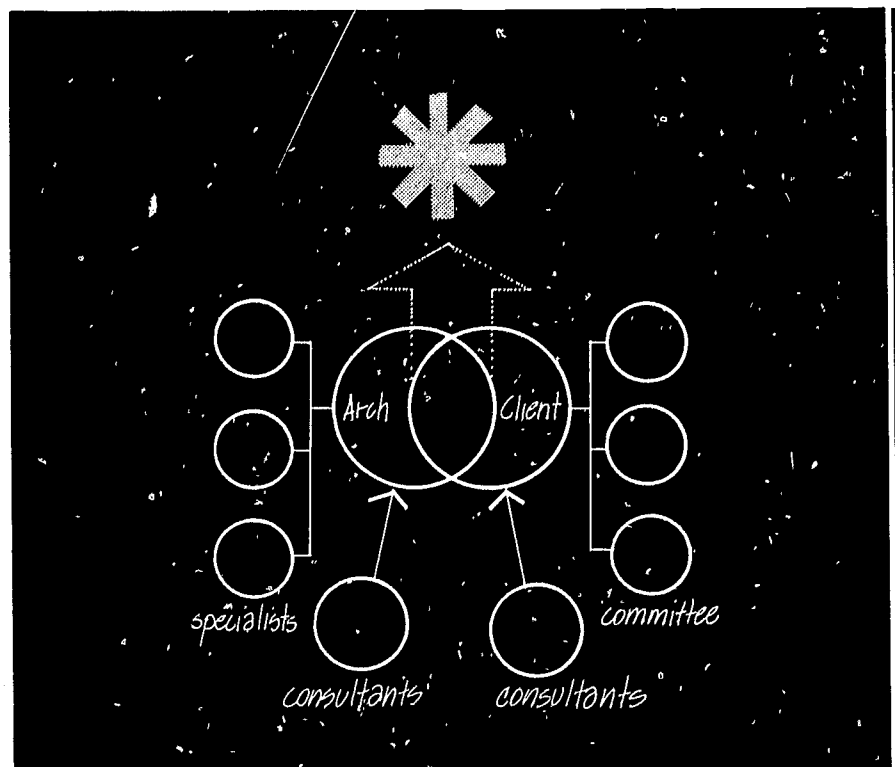
It generally includes the group leader (coordinator), the building committee, the users, interested citizens, and special consultants (health, educational, economic, equipment, etc.). The client group's work may precede the work of the architect group or the two groups may work concurrently.

The architect group is responsible for analyzing the functional program and its space requirements, the site analysis and the cost estimate analysis. The group includes the group leader (project manager), specialists (programming, design, building type, cost estimators, etc.), and special consultants (engineering, interiors, etc.)

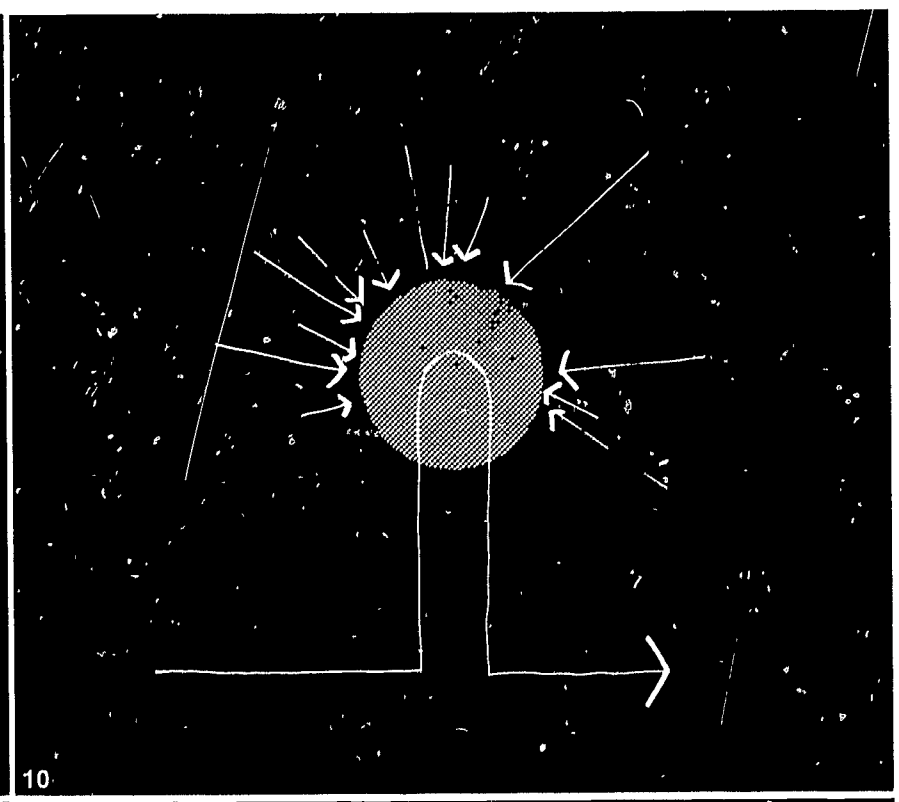
The two groups form one team and as such analyze the facts and identify programmatic concepts for the project facilities. Together they must balance space requirements and the budget.

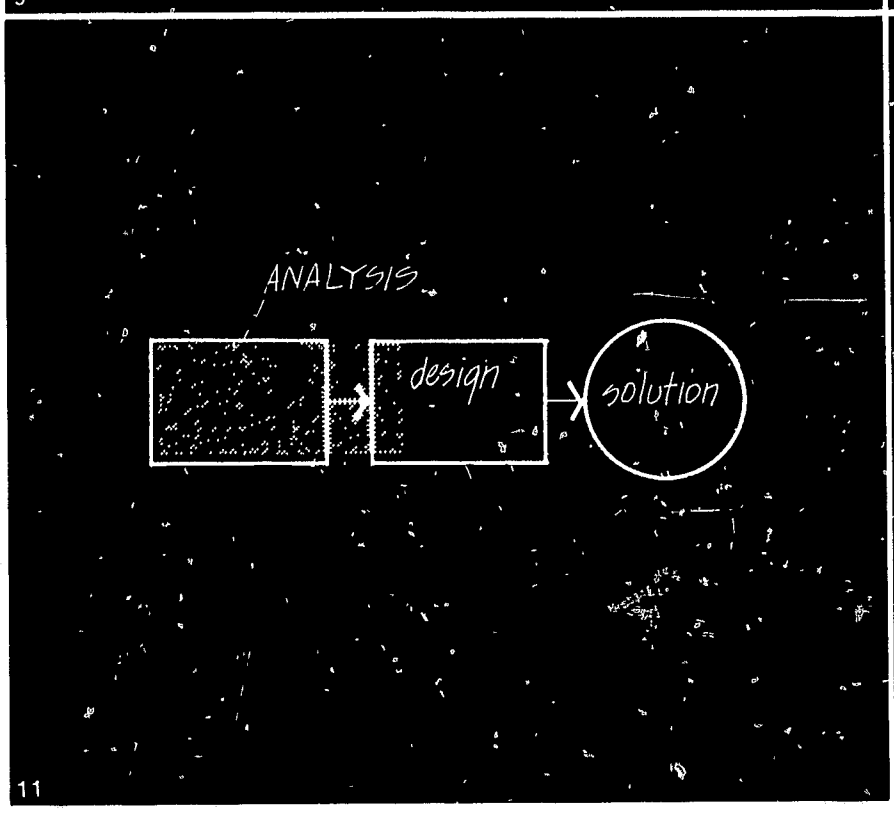
The team requires a high degree of communication and organized programming provides the framework. Every program item must be *graphically* documented for quick

9 The client group and the architect group form *One Team.*
10 The squatters technique brings the *Total Team* together.
11 Problem seeking is the analytical stage before problem solving.
12 Programming is a two phase process which provides the appropriate information for the two phase design process.

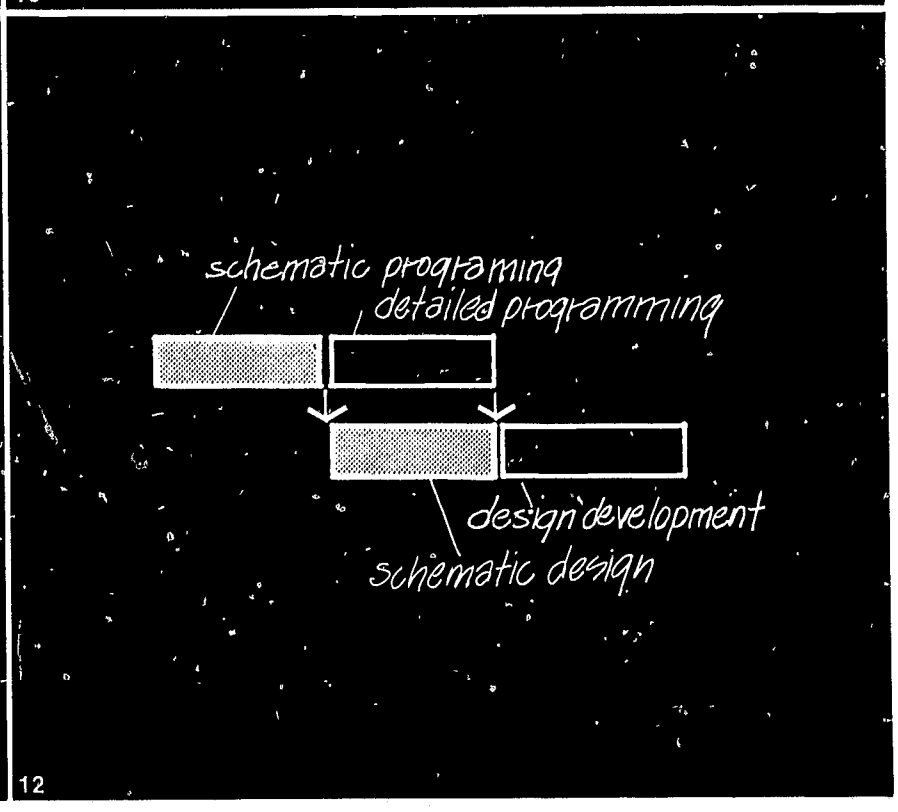


9

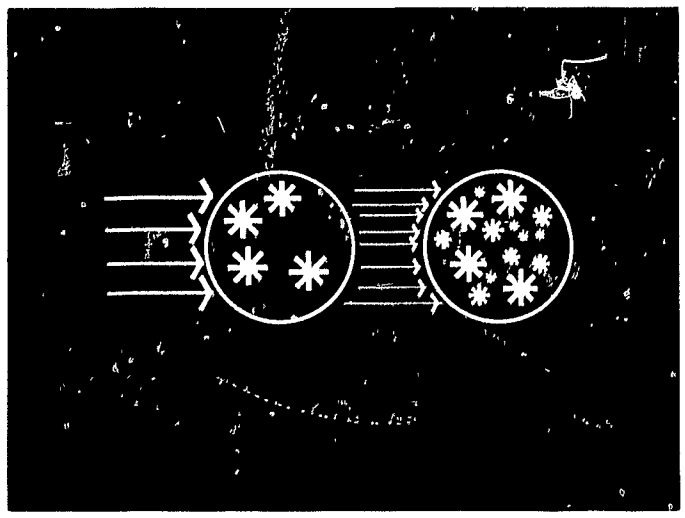

10



11



12



13 Details must not obscure the major ideas which will influence schematic design.

reference. Final graphic documentation is the responsibility of the architect—not only as evidence of having mastered the facts but also to provide feedback to the client, (in general it is best to use the client's terminology).

One technique used by CRS is the "squatters". The squatters technique is an on-the-site, intense work session which involves the total client-architect team. It can last from 1-5 days, and requires thorough pre-planning, cooperation and effective communication. The programming squatters brings the total team together so it can focus on identifying the problem in all its aspects. The relatively short and intense time period requires that the process be organized in a way to bring the proper team members together for various levels of analysis and decision making. (The same squatters technique is also used later in the *problem solving* or design session in which the client is *again a part of the team.)* (10)

The Analytical Approach

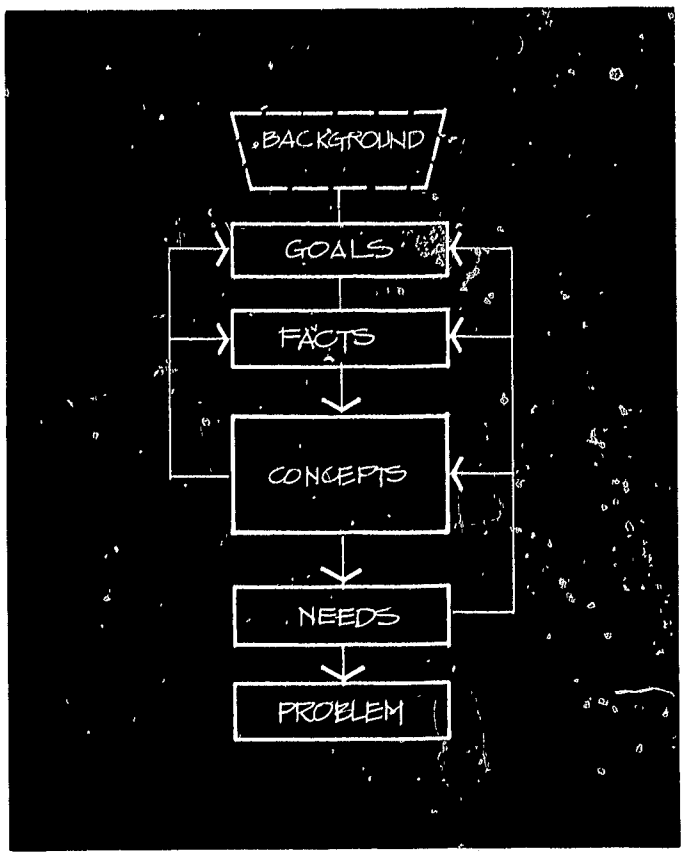
A name commonly used for programming is "architectural analysis". Indeed, problem seeking requires an objective, analytical attitude in dealing with the realities of the problem. During the pre-problem solving stage, all pertinent information must be brought to light without regard for any individual bias by team members. CRS discourages preconceptions at this stage, even though it does not preclude intuitive insight, if intuition is taken to mean a conclusion reached without conscious and thorough analysis. For there are many times during the analytical process when intuition comes into play, as when

(1) Discriminating between facts which will lead to form giving ideas.
(2) Establishing aims no matter how much logic is used.
(3) Understanding the client's functional relationships, particularly when the client is not able to verbalize them satisfactorily. (11)

Such intuition is based on experience and knowledge.

The Creative Approach

There is creativity within the programming process just as there is analysis within the design process. The objective of the creative approach to programming at CRS is to find new combinations of ideas.



14 Problem seeking involves five steps with feedback and evaluation loops at each step. Inexperience in the building type requires that an information background be established through research.

But this does not relieve the client of the major responsibility to be creative, for he is the one responsible for the programmatic concepts.

Programming as a Two-Phase Process

At CRS, programming is divided into two phases, in the same way as design. These are *schematic* programming and *detail* programming. The reason is that the client-architect team should seek to organize program information according to a priority, in order to determine its relevance for the two phases of design. During the schematic design phase, a designer seeks to solve the major overall problems first. During the design development phase he proceeds progressively to the more detailed problems. The programming process should provide the appropriate information at each design phase. (12)

The *first phase* of programming deals with large amounts of information, which must be analyzed to separate relevant facts from the less important details, the basic concepts from the small features. (13)

The *second phase* must then provide the information needed to develop and refine the design. This includes such areas as furniture, equipment and specific utilities. By then the details will not obscure the major comprehensive statements that emerged in the first phase.

The Need for Background Research

At CRS the programmer is required to develop a background understanding of the area of the client's problem, through a survey of similar projects, library research, etc. *He must seek information, not solutions.*

With enough background information, the number, kinds and point of greatest use of consultants can be determined.

The Five Steps

Separating problem seeking from problem solving is essential to avoid confusion as a project moves from conception to completion. The research for a definition of the problem calls for a step-by-step analytical procedure. (14)

CRS uses five such steps in problem seeking, as follows:

1 *Establish Goals*
2 *Collect, Organize and Analyze Facts*
3 *Uncover and Test Programmatic Concepts*
4 *Determine the Real Needs*
5 *State the Problem*

Sequence of the steps may vary, but the steps themselves form an orderly framework for classifying and documenting information that comes from many sources.

In practice, the first three steps may be concurrent. They can cause a re-evaluation of the client's goals and a "recycling" to confirm earlier facts and concepts. Step 4 is taken after evaluating the first three to determine space requirements, performance criteria, and project budget. An imbalance here would cause another recycle analysis by the client-architect team to review and adjust the first four steps. The fifth step is taken after re-evaluating the previous steps; it shapes a statement in such a way as to spell out the total problem uniqueness and basic essence.

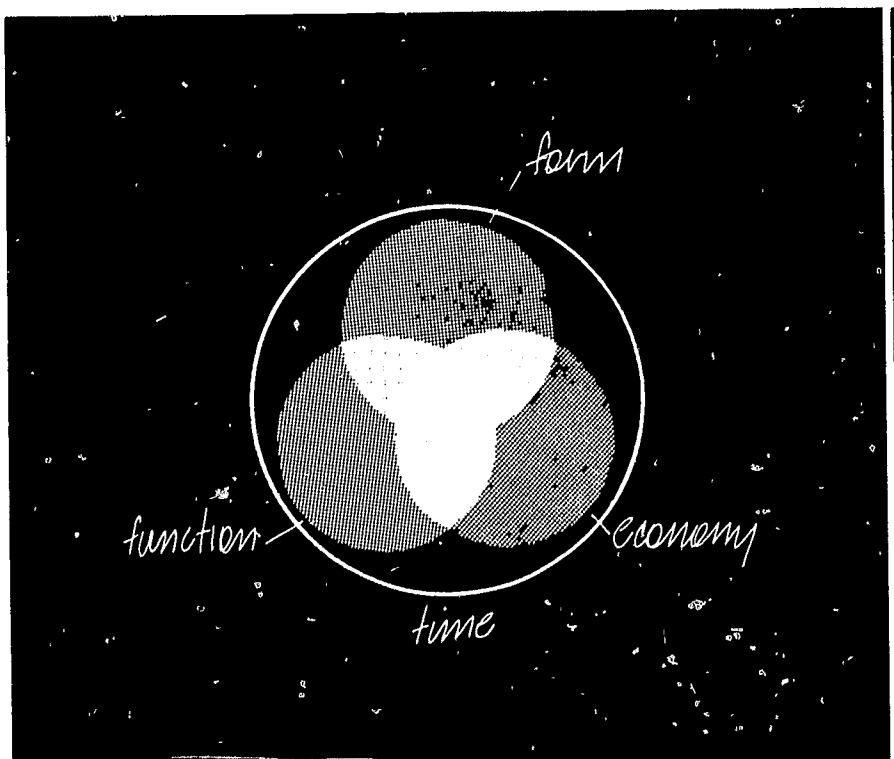
The Four Basic Considerations

If design of the facility is to solve problems of function, form, economy and time, then programming must treat these as basic considerations by which to classify information. (15)

The first of these, *function,* deals with the functional implications of the client's aims, methods to be used to meet them, and numbers and types of people. It deals with social and functional organization. Contributions to the client could be by management consultants, behavioral scientists, and architects with intuitive insight into social values.

Form, the second consideration, is used by CRS to evoke questions regarding the physical and psychological environment to be provided, the quality of construction and the conditions of the site. The physical environment involves physical needs such as illumination, heating, ventilating, air-conditioning and acoustics. The psychological environment raises values which might affect user behavior; the architect must inject these intuitively until such time as analytical means are developed.

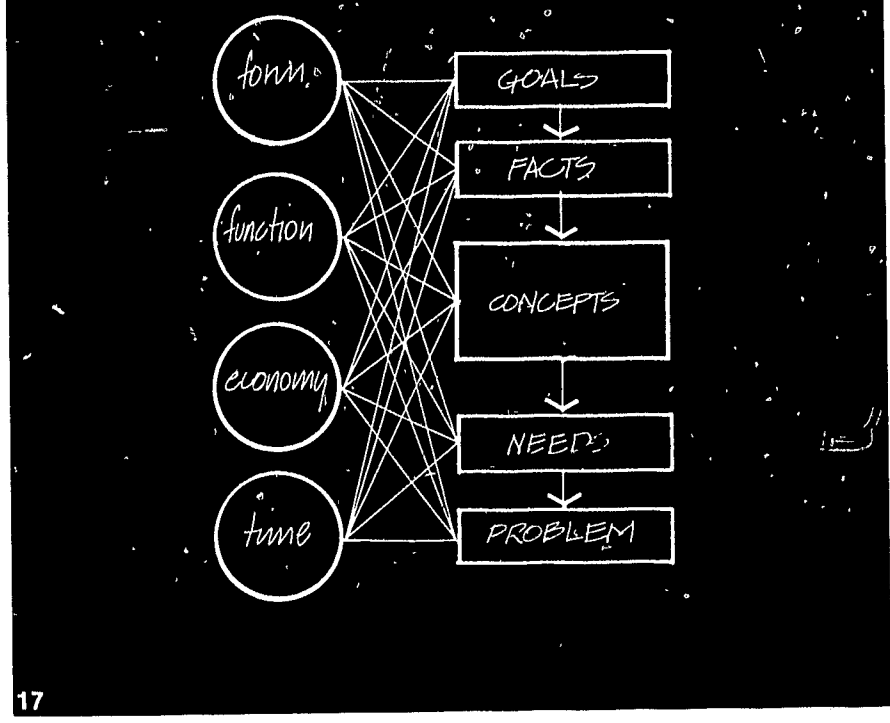
15 The whole problem consists of the considerations of form, function, economy and time.
16 The information index lists the kinds of information classifiable under the 4 categories.
17 Each step involves all four considerations.
18 The dots shown in the matrix indicate a value judgement of the emphasis in a theoretical program.



15

CATEGORIES	INFORMATION INDEX
FORM	site
	environment
	quality
FUNCTION	aims
	methods
	people
ECONOMY	initial budget
	operating budget
	long term budget
TIME	past
	present
	future

16



17

	form	function	economy	time
GOALS	●	●	●	●
FACTS	●	●	●	●
CONCEPTS	●	●	●	●
NEEDS	●	●	●	●
PROBLEM	●	●	●	●

18

The third consideration, eco*nomy*, emphasizes the need for early cost control and brings up for consideration by the programming team the initial budget, the operating cost and the long term cost which may be affected by initial quality of construction. (16)

Consideration four, *time*, brings out the factors of change and growth, which affect function, form and economy.

Combining Steps and Considerations

At CRS, form, function, economy and time are the basic considerations—the co*ntent* of the programming process. Later, they will serve as criteria for evaluating the completed programming package as well as the design solution. But these considerations are not in themselves a process: one is not considered ahead of the others. All four are considered simultaneously at each of the five steps in the analytical procedure. CRS uses them as key words in seeking information, and as general categories or classifications for organizing information at every step of the programming procedure. (17)

In summary, then, form, function, economy and time, within the framework of the process, provide:

1 A format for collecting programming information.
2 Classifications for organizing such information.
3 Criteria for evaluating the results of programming and design.

A Matrix Checklist

A useful way of coordinating the steps and considerations is to establish a simple matrix. Such a matrix serves as a tool in generating the necessary information at the programming phase. It can be used not only as a checklist for missing information but also as a device to record the emphasis or amount of information regarding form, function, economy and time—at each step. This emphasis could also indicate priority of considerations. CRS considers form, function, economy and time simultaneously but within the content of the project each might be given its own priority. Thus, each project would have a different "profile" in the matrix. (18)

This chapter describes in detail the content of the five programming steps first introduced in the last chapter, and it shows how these steps are treated in terms of function, form, economy and cost.

Step 1 Establish Goals

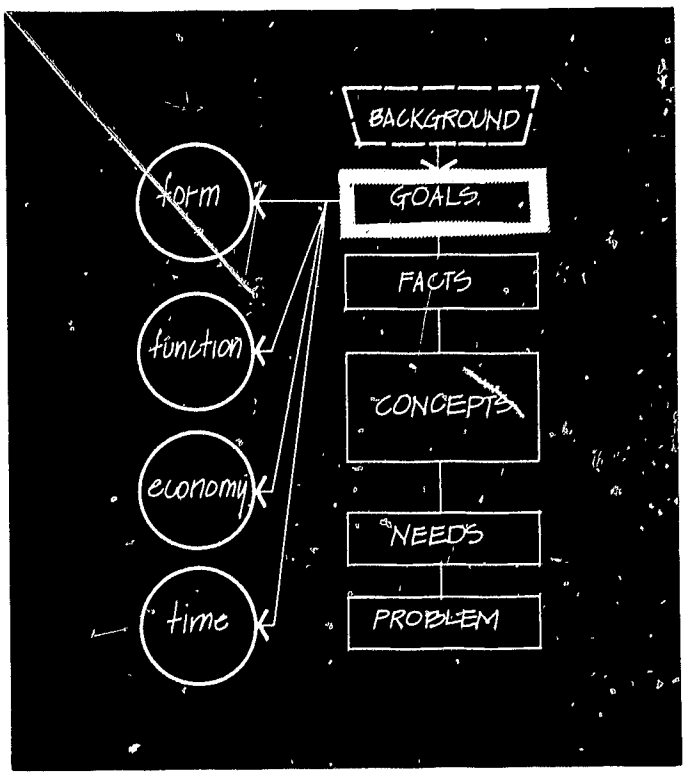
The client usually finds it easier to express his goals for the project at the very beginning while he has the total project in mind and before his thinking becomes involved with details. When goals are established at this stage, they will provide a direction for programming. The gathering of facts can thus be related to the goals, and the tests for programmatic concepts will determine whether the goals are being followed or not. (19)

A client will often place special emphasis on one or more considerations (function, form, economy, time) through his statements of goals. CRS encourages clients to state their goals in terms of all four considerations. For example, if the client tends to think only of functional goals the programmer should be ready to explore his goals for form, economy and time.

The client-owner may establish general overall goals. The client-user usually establishes more specific goals. Any conflict between them must be clarified and reconciled. The subsequent steps depend notably on clear-cut, coordinated goals. A more refined approach to the establishment of proper goals involves the ranking of values. This counts most in cases where the client-user may have a different set or hierarchy of values than the client-owner. Behavioral scientists are beginning to provide this additional perspective as well as more effective information gathering techniques; these should lead to more precise and comprehensive programming.

Goals may lead the client directly to decisions on management policies. These policies, treated as a part of goals, provide readily useful information for CRS programmers.

Form: Goals concerned with form may be expressed in terms of site utilization and the fate of existing trees and structures. They may refer to the general character or effectiveness of the physical and psychological environment of the project



19 The *Client's Goals* provides the basis for the subsequent development of the program.

location. A goal stated in terms of quality is always useful since it influences the decision making process and must ultimately be tested in reaching an equilibrium between quality, space and cost.

Function: Functional goals are expressed in terms of how the facility will affect the activities or processes to be housed. They may deal with personal and social implications. It is always useful at this stage to state goals by asking such questions as—Why is the new facility needed? What is the purpose in building? What is to be accomplished?

Economy: Most clients have a limit to their available funds. An economy goal establishes this limit. Even in cases where the client wants the architect to estimate the total cost, the client still has a limit in mind. Without exposure at the 'goals' step, the limit cannot be evaluated and subsequent "recycling" of steps may result in drastic changes.

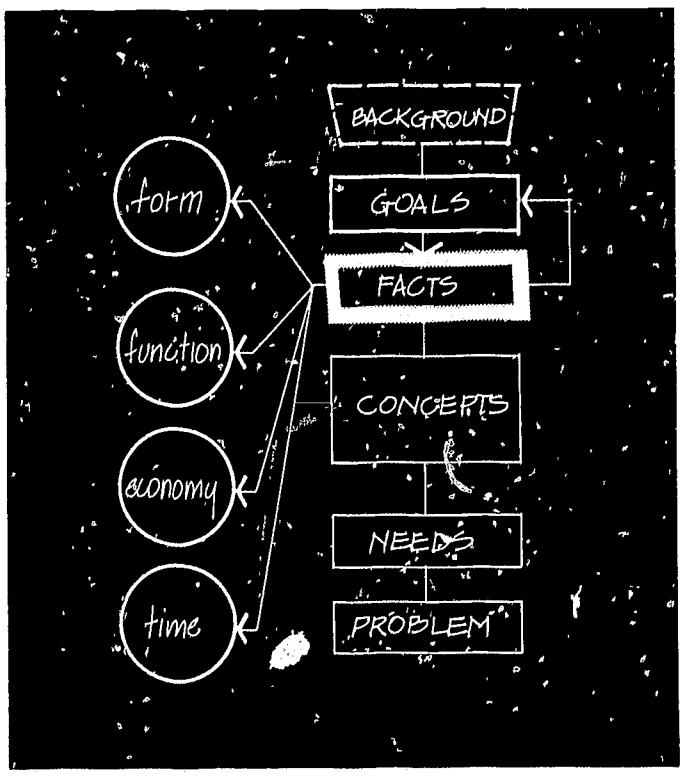
Time: Time goals may be stated in terms of anticipated change and growth as well as of expected occupancy. Time goals should be expected to show whether or not the project schedule is realistic.

Step 2 Collect, Organize and Analyze Facts

Facts by themselves will tell us nothing. They have to be organized and analyzed before they will reveal their importance. The classification of facts under form, function, economy and time is a useful way of organizing and analyzing the information. (20)

Goals determine what kinds of information will be meaningful. Yet, the programmer still has to discriminate between immediately useful facts and details which will be useful at a later phase. The details must not be allowed to distort or confuse important data in the immediate programming phase. That is why CRS uses two-phase programming, briefly described in the previous chapter.

Checklists are often developed for the collection and documentation of information for each building or planning project type. These checklists cover the following categories:



20 The *Classification of Information* under form, function, economy and time is a useful way of organizing and analyzing the information.

Form: Data is needed on the site—its physical characteristics, climate conditions, legal aspects, coincident planning by other agencies; the availability of materials and the make-up of the local construction industry; information on building codes as they might affect the form of the building.

Function: Statistical data is needed on the numbers of people to be housed and their activities. Included should be space generating parameters—area per person, per activity—group sizes, kinds of groups and utilization requirements.

Economy: Data is required on budget limitations, local cost indices, building cost per square foot, operating costs and long-term-costs where applicable, method of financing, the economic influence of other agencies.

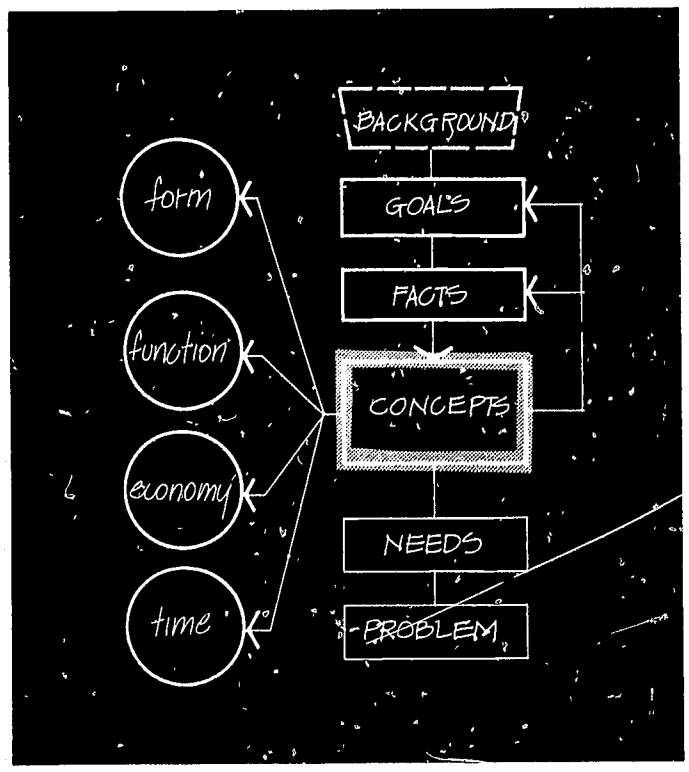
Time: The programmer needs to know the project schedule, phasing and growth, price escalation, anticipated changes and projections.

Step 3 Uncover and Test Concepts

While Step 2 in general deals with facts and Step 3 with concepts, it is difficult to separate the two: The grouping of facts stems from ideas, while ideas or concepts of functional organization stem from facts. For purposes of analysis, facts concern quantitative information; concepts qualitative information. (21)

Discovering concepts is probably one of the most elusive and difficult phases of programming. This is partly due to the nature of concepts: they demand that architect and client think abstractly. But concepts are also elusive and difficult to handle simply because architects and clients frequently do not agree on a common definition of concept.

"Programmatic concepts" is a term used to describe methods of implementing the goals (Step 1). Most concepts are organizational when they implement the client's functional goals. This heavy emphasis on function is a direct result of the client's participation on the team. It is here that the client can display his most creative thinking.



21 Programmatic concepts are a means of implementing the client's goals. The classification of concepts into the four categories of form, function, economy and time is a means to analyzing their implications.

The architect must stimulate the client to make decisions in terms of his functional relationships, as well as his organizational structure. The testing of concepts provides a means of stimulating the client's decision-making.

The programming architect must be creative in the sense of finding combinations and alternatives to cause the client to participate and react with the required decision. The programmer must provide the analyses to bring out concepts and to stimulate the decision. But it is the *client* who makes the decision.

Programmatic and Design Concepts

It is not always easy to understand the difference between *programmatic concepts* and *design concepts.* Most often, concepts are thought of only as a form of design solution. This misconception is reinforced by the fact that design concepts respond at the design stage to programmatic concepts raised during programming, and therefore become so closely related that it is difficult to know which came first.

Programmatic concepts are abstract, and are expressed in terms of organizational structure, relationships and other functional requirements.

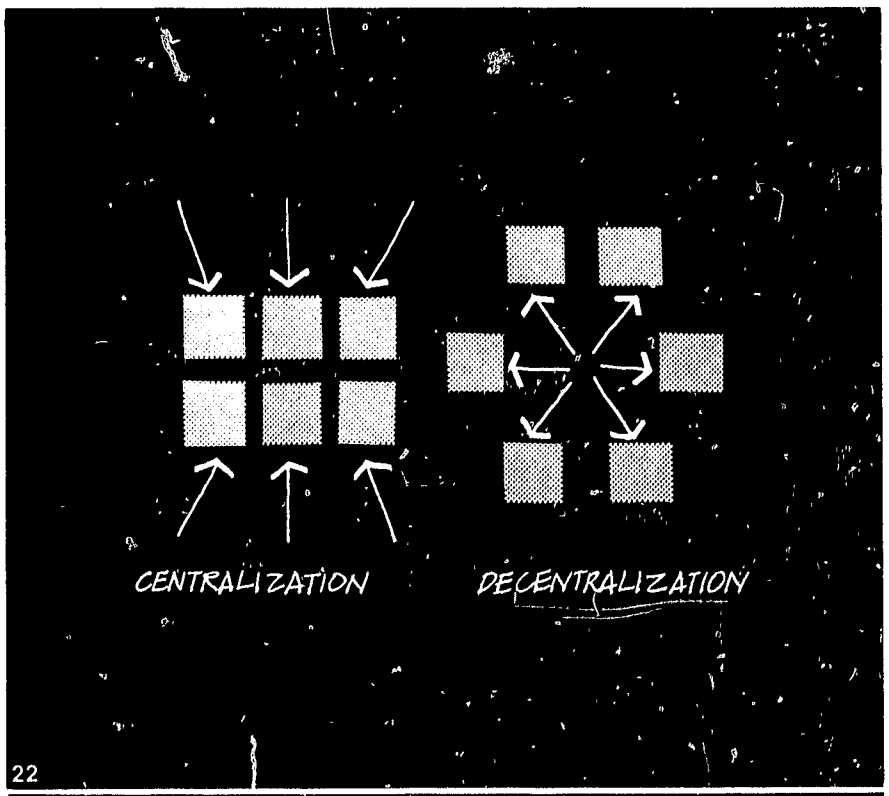
'Evocative' Words

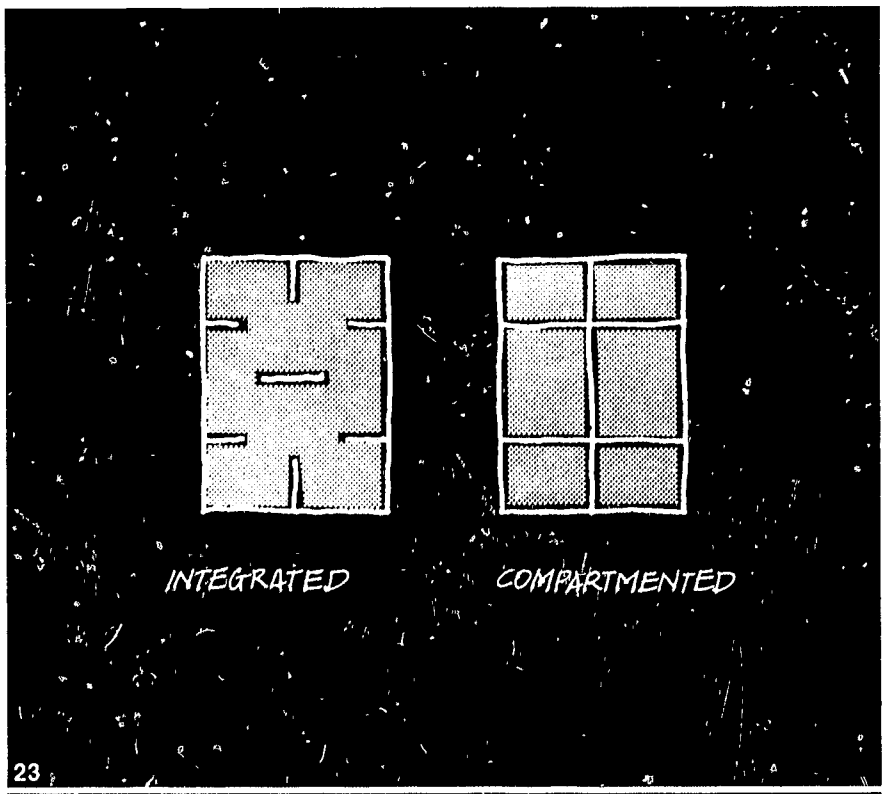
Concepts are brought out and tested through the use of what CRS calls "evocative words." These words trigger useful information; they may be found in the subcategories (page 23) of the basic considerations (form, function, economy and time), or they may be identified with recurring concepts. (Recurring concepts are explained on page 22.) Examples of evocative words are "site," "quality," "people," "priority."

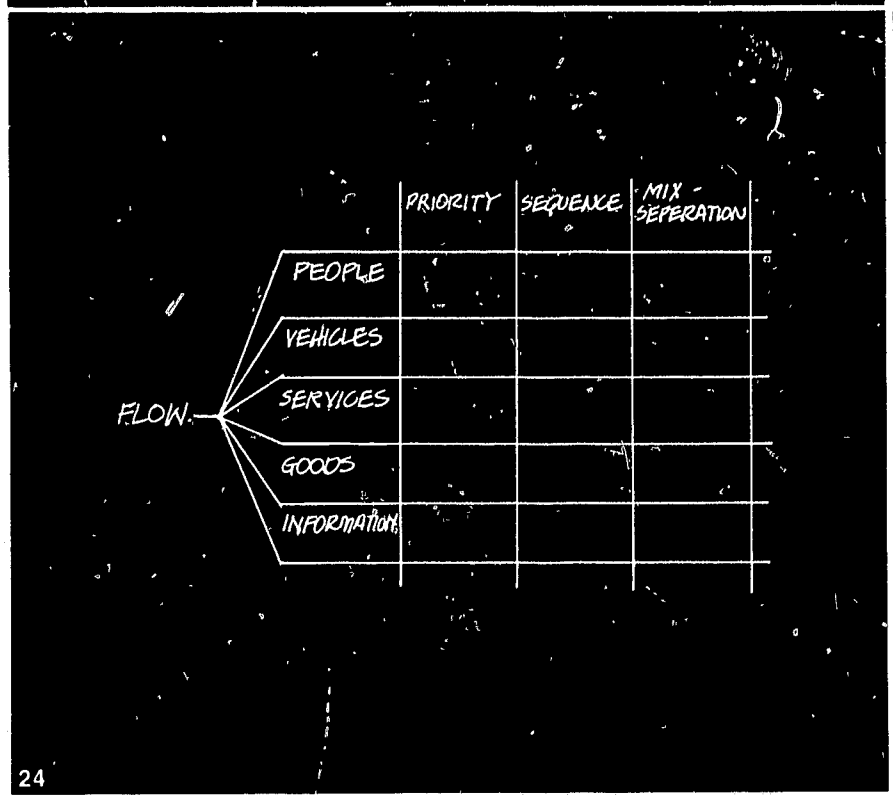
The architect-programmer must be alert to concepts and record them as they emerge in discussions with the client, be they expressed however briefly.

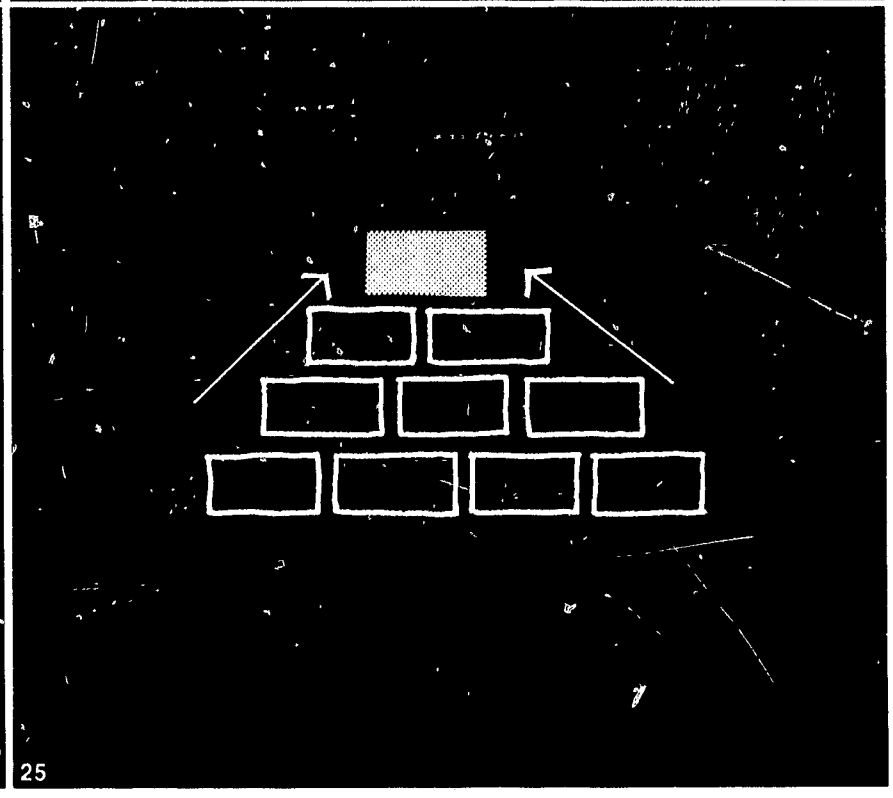
How Concepts are Classified

Concepts are classified under form, function, economy, and time simply as a way of analyzing their implications. This is often a matter for interpretation—a particular concept could easily be listed under two or more classifications.


CENTRALIZATION
DECENTRALIZATION
22


INTEGRATED
COMPARTMENTED
23


PRIORITY
SEQUENCE
MIX - SEPERATION
PEOPLE
VEHICLES
SERVICES
GOODS
INFORMATION
FLOW
24


25

The following classification is based on CRS experience in testing these recurring concepts:

Form: While concepts dealing with function, economy and time can be stated abstractly, it is more difficult to do so with form concepts, but the client should be encouraged to do so, in terms of the site, the physical and psychological environment, and quality.

When the client cannot express a form concept except in physical terms, it is a premature design solution. In this case the programmer must look behind this solution to bring out the reasons that led up to it.

Firm but preconceived opinions by the client can yield useful information and should be deliberately exposed during programming. These can be discussed and analyzed, and can lead to a better understanding of the requirements. Ignoring this can lead to difficulties later in the project.

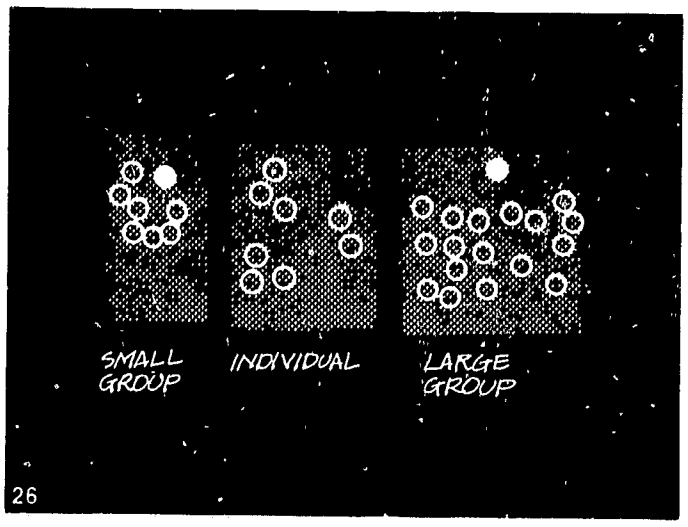
Function: The programmer can raise and test a number of recurring concepts with the client as a way to give him an avenue for expressing his functional needs.

"Recurring" Concepts

A recurring concept is one which the architect has learned does not only appear at just one project or one type of institution, but appears as a potential aspect of any project or institution. Concepts like people, flow, flexibility, priority of needs, centralization, integration, belong to this class. A few of these recurring concepts are described here in more detail:

"Centralization vs. Decentralization": This concept deals with centralization or decentralization of activities, services or personnel. It can influence the program in terms of organizational structures, functional relationships, and overall space affinities. The programmatic concept should not be confused with the design concept of *compactness vs. dispersion:* the programmatic concept can have several alternatives of compactness or dispersion. (22)

"Integration vs. Compartmentalization": The programmer must find out from the client if activities should be integrated or compartmented. A group of closely related functions would indicate integration; the need for some degree or kinds



26

of privacy (acoustical or visual), would imply compartmentalization. Here too, there is a difference between the programmatic concept of *integration* and the design concept of the *open plan.* (23)

"Flow": This concept concerns the flow of people, vehicles, goods, services and information in terms of priority, sequence and degree of mix or separation. This concept expands on affinities and relationships but excludes a table of organization. Connections between corresponding units can be coded numerically, and abstract flow diagrams can be drawn up and manipulated to minimize conflicts in circulation. This can be a computer or a manual function. (24)

"Priority": An important evocative word is "priority". Priority has to do with *people, spaces, things.* It has to do with priority of functions and needs, such as relative position, size, social value and others. (25)

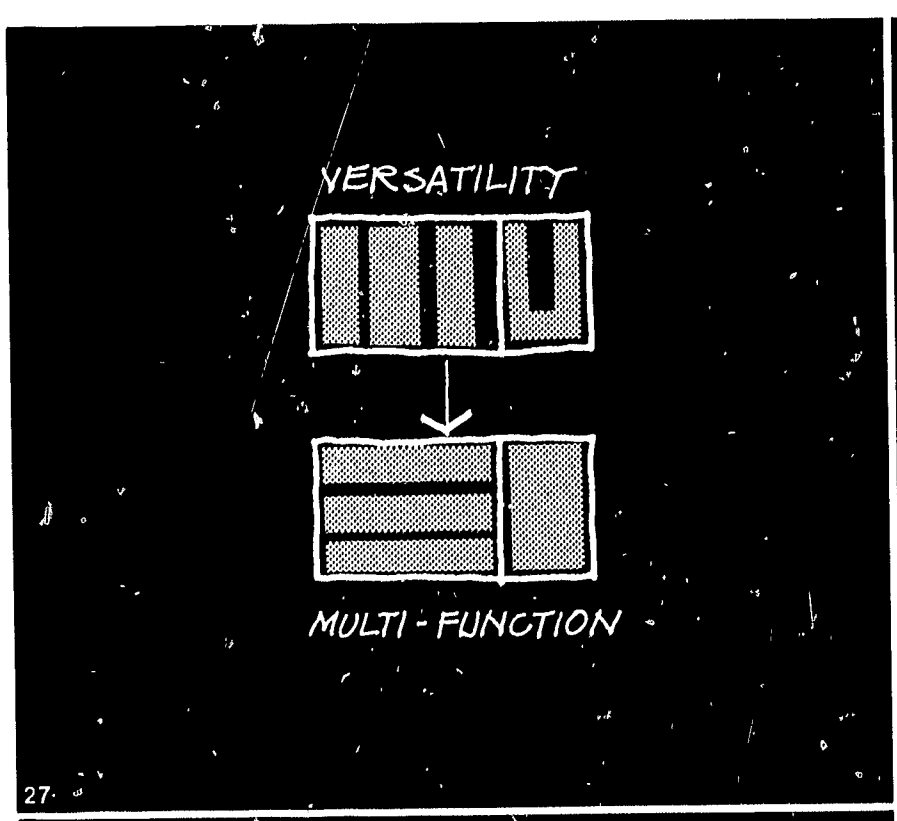
"People" is another evocative word which can generate concepts derived from the physical, social and psychological characteristics of people—as individuals, in small groups, and in large groups. In this area there is no substitution for expertise, and the behavioral scientist has a major contribution to make. (26)

Economy: Concepts under this heading are particularly useful in the recycle and analysis if there has been a lack of balance between budget, space requirements and quality.

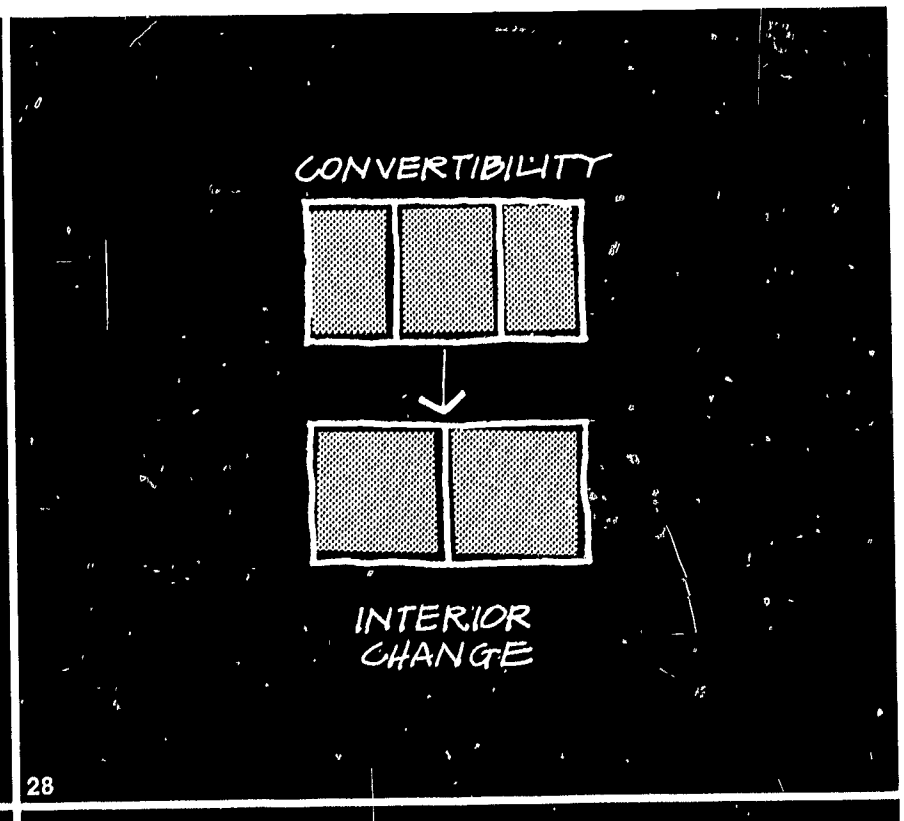
"Versatility" is the first concept to be tested as a means of achieving such a balance; it could, however, result in reduced efficiency for each of the several functions to be combined. (27)

Other evocative words which can generate economy concepts are: *phasing, optimization, efficiency and cost/effectiveness.*

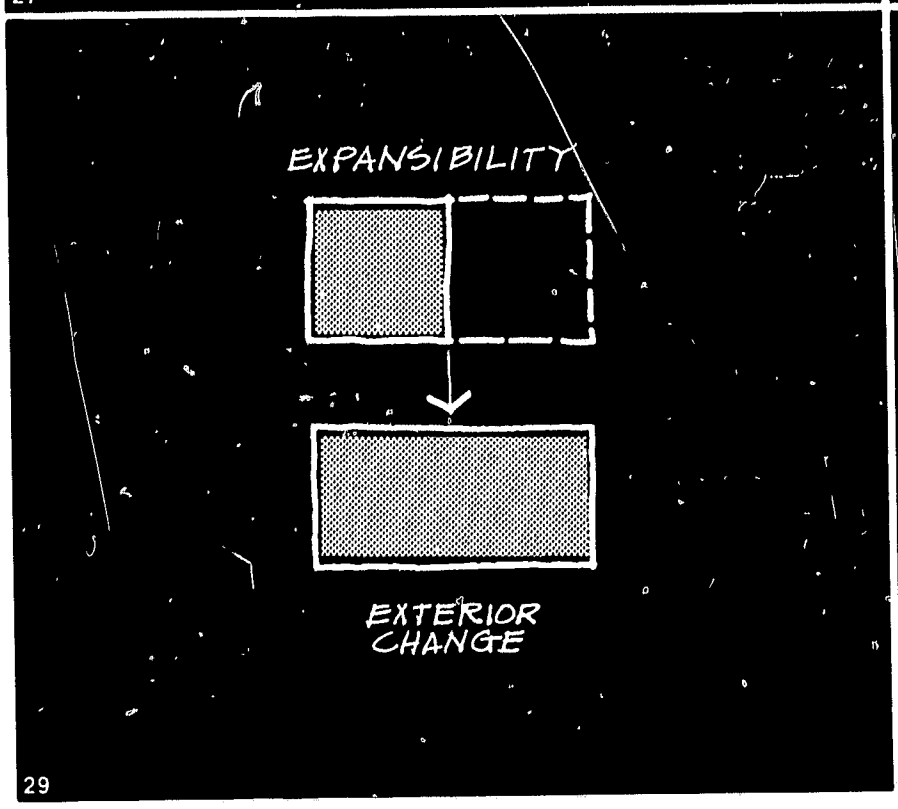
Time: The following concepts can occur:
"Convertibility" is a concept that allows for anticipated change in functional requirements. The team must establish the degree of convertibility as: (1) immediate, (2) weekend or (3) long range. (28)


VERSATILITY
MULTI - FUNCTION

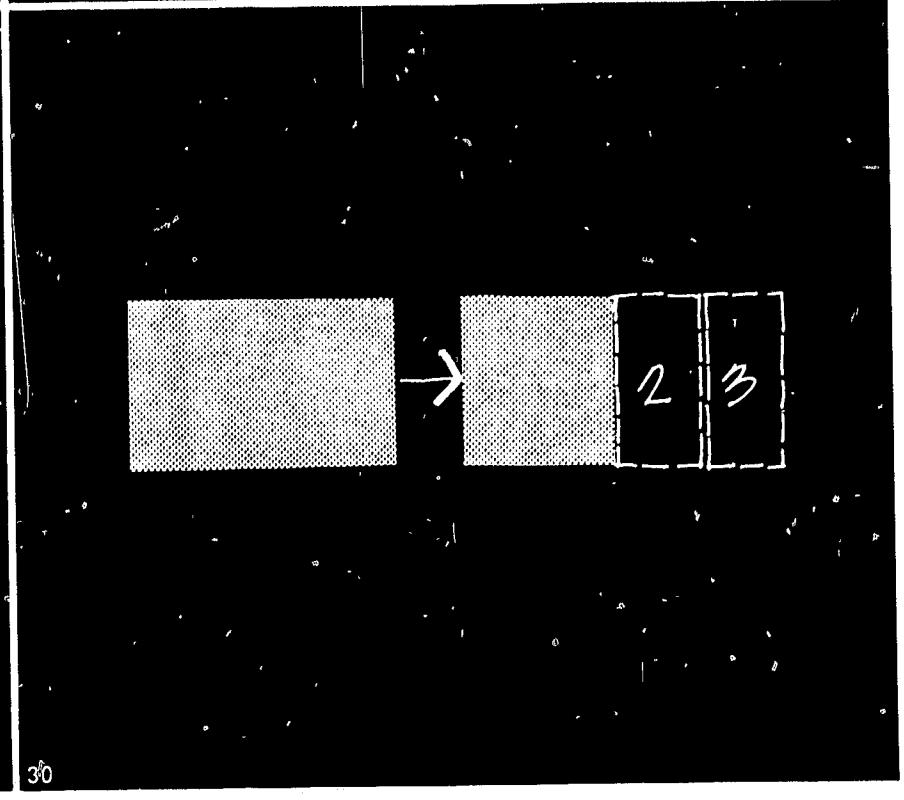
27


CONVERTIBILITY
INTERIOR
CHANGE

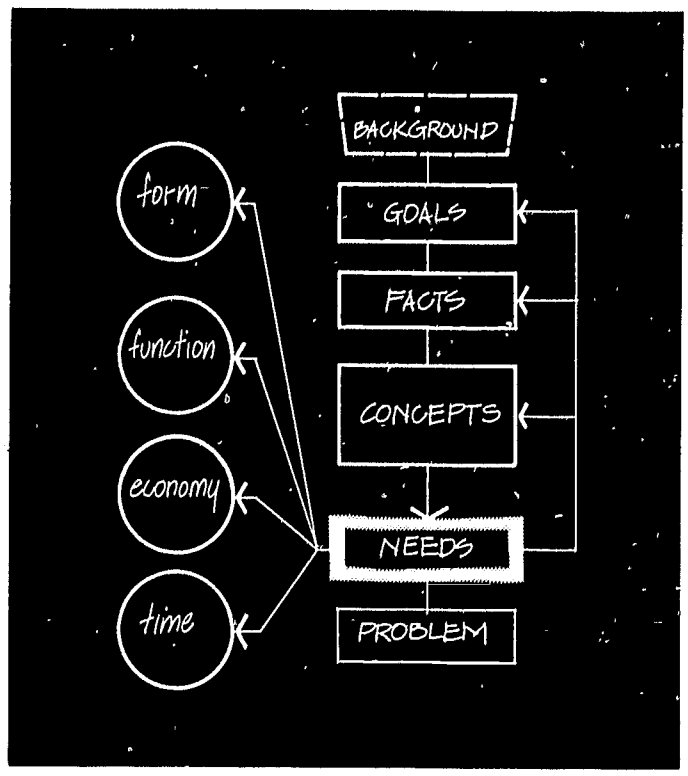
28


EXPANSIBILITY
EXTERIOR
CHANGE

29


2
3

30



31 Needs emphasize the *Balance* of quality level, space requirements and budget.

"Expansibility": anticipated growth triggers the programmatic concept of "expansibility". (29)

"Phasing" is a time-economy concept useful in efforts to attain functional and/or economic feasibility of the project (30)

The recurring concepts stem from experience CRS has accumulated with many building types.

Step 4 Determine Needs

Determination of needs is fourth of the five analytical steps and seeks to establish quantitative needs of the client in terms of *space requirements, a budget* (as predicted for the *time* of construction) and quality (cost per square foot). (31)

The proposed space requirements and the expected level of quality must be tested against the proposed budget at this stage of programming.

If a balance cannot be achieved between space, quality, budget and time, at least one of these four elements must be negotiable. Thus, if agreement is reached on quality, budget and time, the adjustment must be made in the amount of space. A serious imbalance might require cycling to re-evaluate goals, facts and concepts.

Form: The proposed quality of construction is expressed in quantitative terms as costs per square foot. At this stage this figure is based on experience and/or background survey and analysis. Both the physical and psychological environment are factors in quality of construction and, in turn, in the cost per square foot. Furthermore, site conditions will affect the form of the building and influence the construction budget.

Function: The client's functional needs (as determined by facts and concepts) have a direct bearing on space requirements, which are generated by people and activities. Allowance must be made for a reasonable building efficiency as expressed by the relationship of net to gross areas.

Economy: The cost estimate analysis must be as comprehensive and realistic as possible, with no doubt as to what comprises the total budget required. The building cost

Cost Estimate Analysis	Blank Junior College	
Date: 25 October 67	Revised	
A. Building Cost	315,500 S.F. @ $20.00/S.F.	$6,310,000
B. Fixed Equipment	(8% of A)	504,800
C. Site Development	(20% of A)	$1,262,000
D. Total Construction	(A + B + C)	$8,076,800
E. Site Acquisition		300,000
F. Movable Equipment	(20% of A)	$1,262,000
G. Professional Fees	(6% of D)	484,600
H. Contingencies	(10% of D)	807,600
J. Administrative Cost	(1% of A)	63,100
K. Total Budget	(D + E thru J)	$10,994,100

32 The cost estimate analysis must account for all budget items including predicted escalation of building cost to the mid-construction date.

at this stage is usually based on a cost per square foot. Other cost items such as site development, fixed and movable equipment are based on percentages of building cost unless more refined figures are available. Most often these cost parameters are based on published indices and on experience tempered by the local situation. (32)

Time: In determining the cost per square foot, a realistic escalation factor must be included to cover the time lag between programming and mid-construction.

Phasing of construction may be considered as an alternative:

a When the initial budget is limited.
b When the funds are available over a period of time.
c When the functional needs are expected to grow.

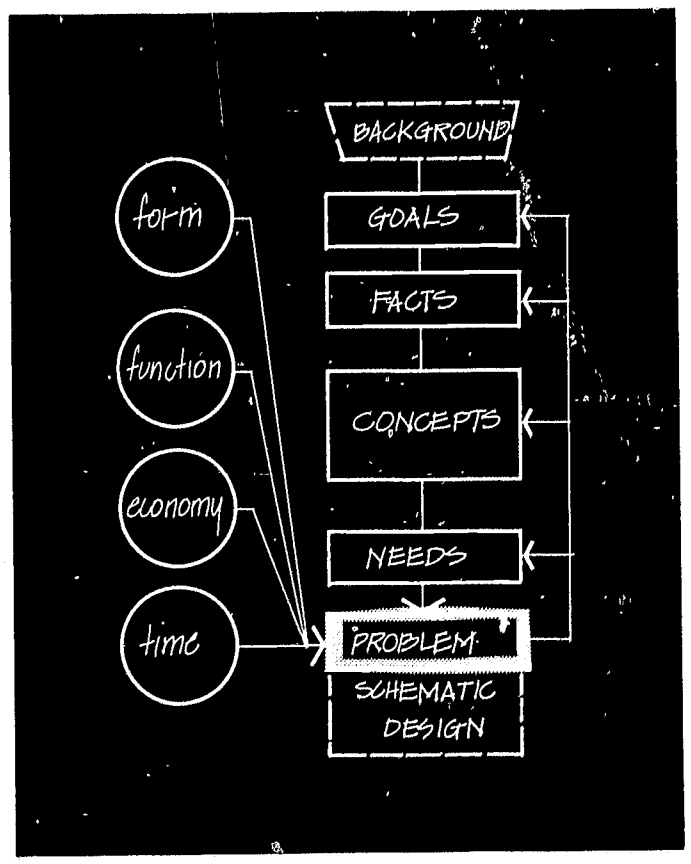
Step 5 State the Problem

Once the first four steps in programming have been completed and evaluated, the programmer and the designer begin together to formulate a series of very succinct statements. These constitute the Statement of the Problem. (33)

This Statement of the Problem is the link between *problem definition* and *problem solving,* between *programming* and *design.* The problem must be stated in *qualitative* terms in a way that will bring out the *essence* and *uniqueness* of the project at hand. There should be no less than four such statements—dealing with *form, function, economy* and *time.* The statements should not exceed ten, except for very complex projects. Too many statements may be a sign that details are being used as premises for design, which runs counter to CRS programming practice. (34)

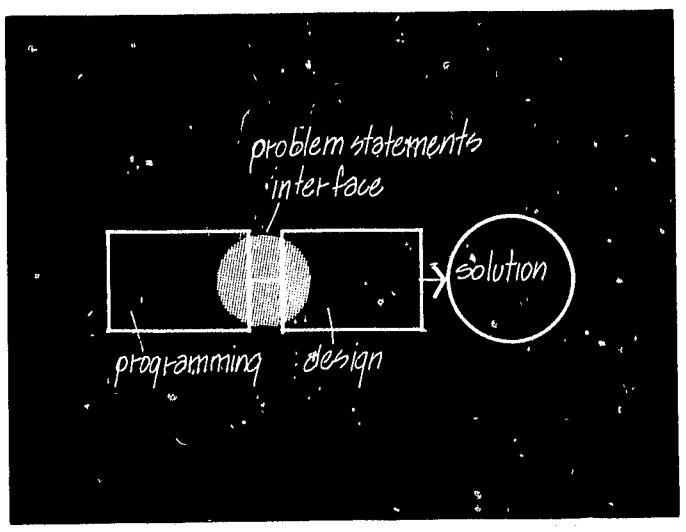
Meanwhile, supporting data contains all the information gathered on goals, facts, concepts and needs. All this is recorded (or documented). It is available—in qualitative and quantitative terms.

The statements must deal with the *unique,* not the *universal* aspects of the problem. Further, they should be made in terms of performance, so as not to close the door to different expressions in architectural form. In some cases the



33 The statement of the problem is the final step in programming and the first step in schematic design.

34 The statement of the problem in qualitative terms is the link between *Programming* and *Design*.



statement of the problem may even be developed to the point of a simplified performance specification, that points up the functional requirements to be met by the solution.

Any of the preceding four steps—goals, facts, concepts, needs—can be a source of such a programming statement, so long as it is an important physical form giver.

An effective method for developing each statement is first to state an important condition and then to suggest a direction by means of the statement.

Form: Form connected statements may describe the particular environmental influences and site conditions and then establish a performance requirement to respond to these, for example: "The design should take into account the dramatic view across the lake to the Sugarloaf Range", or "Since the structure will occupy its own block, it should be handsome on all sides".

Function: The first four steps may have uncovered unique performance requirements that will influence the functional aspects of the solution. These can be expressed in a statement, such as "To some people the intermission is as important as the performance; therefore the building should foster the thrill of the individual being in a large group".

Economy: To give direction to the designer who may have different projects underway, with different budgets, a program statement must establish an attitude toward the budget of the project at hand. For example, "Every detail, as well as every major planning concept, must be developed with the million dollar budget in mind".

Time: Statements must be made concerning the implications of change and growth on long range performance, e.g. "The campus must grow. There should be visual unity at each stage of development".

The object of programming is problem definition. The problem needs to be defined in terms of the major issues that affect the total problem. The Statement of the Problem is really a short series of categorized statements which the designer will later use to evaluate the solution.

NORTHERN BRANCH - SUBURBAN CAMPUS
SPACE CALCULATIONS FOR PLANNING

DESCRIPTION OF FACILITIES

	TOTAL NET AREA @ 1,000 FTE	TOTAL NET AREA @ 2,000 FTE	TOTAL NET AREA @ 3,000 FTE
	44,952 sf	82,194 sf	114,985 sf
	6,252	10,989	15,980

INSTRUCTIONAL FACILITIES

LECTURE/CLASSROOM/ SEMINAR FACILITIES

1,200 SF
100 SS @ 12 SF
LECTURE ROOM

60 SS @ 165 SF =
960 SF EACH
CLASSROOMS

32 SS @ 16 SF =
512 SF EACH
CLASSROOMS

12 SS @ 20 SF =
240 SF EACH
SEMINAR

300 SF 225 SF 259 SF
MISC SUPPORT

LEARNING LAB FACILITIES

30 SS @ 25 SF =
750 SF EACH
LEARNING LABS

75 SF 40 SF
MISC SUPPORT

BUSINESS LAB FACILITIES

2,535 SF
BUSINESS LABS

1,180 SF 1,240 SF

105 SF 70 SF 105 SF
SUPPORT

11355 SF
OCCUPATIONAL

SCIENCE LAB FACILITIES

6,350 SF
SCIENCE LABS

7400 SF 5385 SF

1265 SF 1,480 SF 1,080 SF
SUPPORT

FACULTY OFFICES

400 SF

CHAIRMAN SECRETARIAL/CLERICAL

FACULTY WORK AREA

5,550 SF 5,670 SF
CUBICLES @ 70 SF

300 SF 150
CLERICAL
& WORK AR

ADMINISTRATIVE FUNCTI

ADMINISTRATION

2,460 SF 510 SF 440 SF
& SUPPORT STAFFS

In the final count, a successful statement of a problem for a project is no better than the ease with which the client and architect communicate their thoughts to each other. Each may have his own specialized terminology, which may become a serious language barrier. The architect with background information or experience in the client's building type must make an effort also to learn the client's language. CRS has found that use of simple terms works best and that, generally, the most widely understood ideas are the strongest ideas. (35)

Analysis Cards

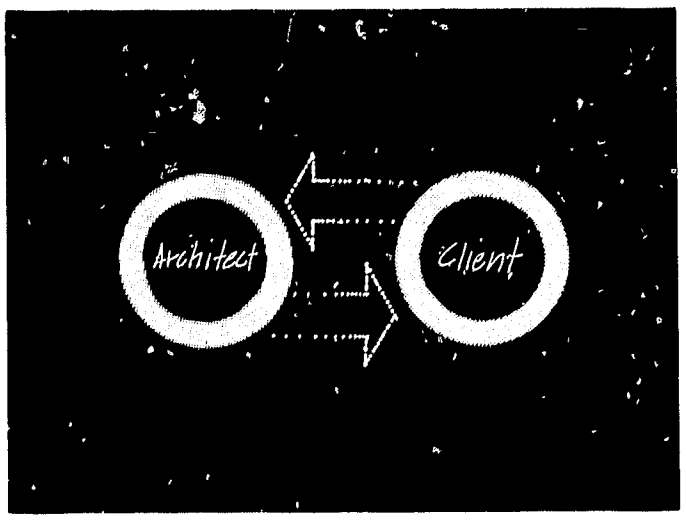
Yet words do not always come across easiest, and for a client-architect team in action the typewritten page does not communicate as easily as a diagram. A visual image is more easily retained than a word image. In programming, CRS seeks to diagram every idea, every fact, if the diagram will lead to understanding. The written text is a last resort, and even then is best when reduced to a succinct statement. *A good idea or an important fact cannot be discussed and evaluated unless it can be expressed clearly to all concerned.*

CRS uses 5" x 7" analysis cards as visual aids for feedback to the client; each card contains only one thought, one piece of information, or one concept. Each graphic image is drawn as if writing a telegram. The guide is: "Think about the essence of the message and put it to paper economically, using very few elements, using color for emphasis". The organized collection of cards is displayed on a wall for discussion with the client, ready for reference in arriving at the statement of the problem and later during the schematic design process. A visual check of the display should show whether or not a particular piece of information was classified and recorded. This should be done on analysis cards. Undocumented information is not likely to be considered and evaluated.

Informality is important with this communication medium. It is a working technique. It does not require meticulously drawn graphics which tend to look as final documents.

Visual Aids

An opaque projector is used to present the analysis cards to the client's committee. The advantage of using an opaque



35 The client and the architect must agree on a common vocabulary.

projector is that analysis cards can be produced up to the last minute before presentation. When more time is available between production and presentation or when duplicate sets of the cards are required for sub-committee discussion, the cards can be reproduced in slide form. On various occasions two or even three projectors have been used simultaneously to compare visually two or three ideas or bits of information.

Brown Sheets

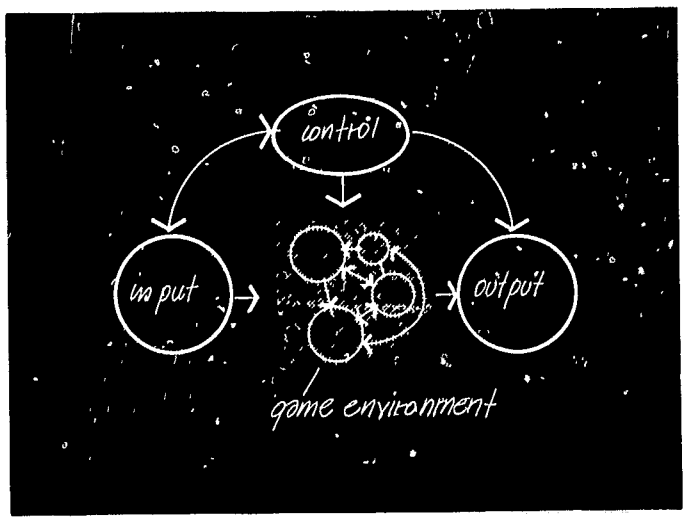
"Brown sheets" are a graphic representation of space requirements drawn to a 1/16" scale on brown wrapping paper. The relative sizes and number of spaces on the brown sheets help the client group and the architect group grasp their significance quickly and more easily. Discussion can be focused on specific space requirements within the context of the total space program which, by looking at the wall display, can be seen as a whole.

The brown sheets are most useful in communication with the client group during the fourth step of programming, determination of needs. The brown pape is used not only for its large size but for its informal quality which designates the sheets as a working tool. The worksheets are used to make additions and deletions of space until agreement can be reached on a definite space program for which funds are available and which the client must approve. As the sheets are being patched and corrected by interaction with the client group, the sheets become the official documentation of the space program. Later the corrected and approved brown paper can be reduced photographically to legal size sheets for table use by the design team.

Programming "Squatters"

Squatters are intensive work sessions where the client-architect team comes together, usually at the project site, and focuses on goals, facts, concepts and needs.

The technique is useful in taking one or more of these steps in the process, or all of them in one session. The session time is predetermined and pre-planned—commonly three to five days, very rarely two weeks.



36 Gaming techniques can expose information not otherwise possible.

Concentrated work is made easier as the architect group is removed from other projects back at the office. Similarly CRS recommends that the client group be scheduled and available for participation.

Users' Conference
The users' conference may be a part of the squatters session or occur separately when the client-owner is not the user of the facility. Pre-planning is required for the conference since the required kinds of information need to be defined and a format for recording the information established.

Large brown sheets can be used during the work session to record the user information. A tape recorder can also be used but lacks the visual and cumulative characteristics of the sheets.

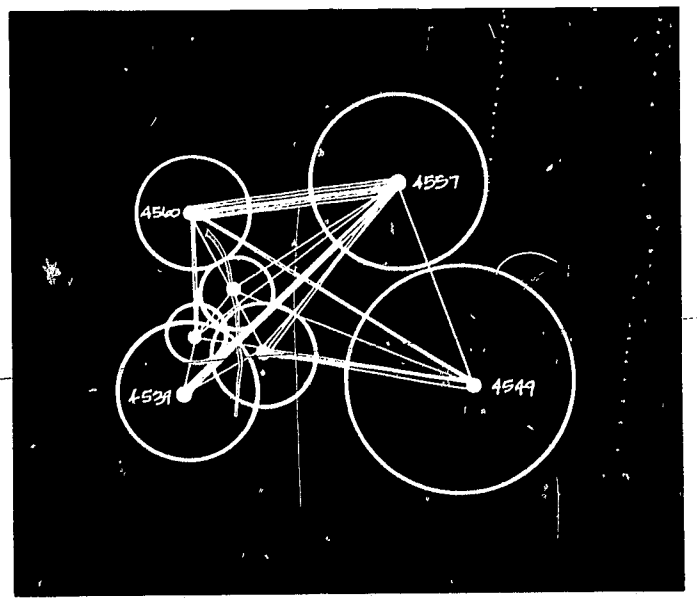
Questionnaires
Questionnaires are especially useful as tools to collect information and opinion, where the user group is so large as to rule out a work session. A concise format helps, and the amount and kinds of data are best limited to a manageable volume. Questionnaires must be carefully structured so as not to predirect responses.

Interviews
The questionnaire often serves as a basis for further clarification through interviews. The success of interviews depend on the programmer's ability to ask pertinent questions but on an organized basis. The framework of the five programming steps and four considerations provide him with an organized source of questions as well as a format for recording the answers. To encourage spontaneity, the questioner usually uses a recorder rather than visibly taking notes. In this sense interviews differ from user conferences, where the need for visual and cumulative recording outweighs the advantages of the tape recorder.

Gaming Techniques
Gaming is a communication medium and information gathering technique to assist decision making. The technique stimulates interaction between people by creating an imaginary environment, a scenario in which each person acts out the role of key people or groups. (36)

This interaction gaming is simply a means of bringing people together to seek to identify and clarify a problem. It is an aid to the collection and documenting of information to influence later (real) decision making.

CRS is currently applying gaming techniques to the programming of complex institutional projects.



37 An Origin/Destination overlay can be used to identify major affinities between activity areas.

Huge quantities of data, complicated functional interrelationships, fluctuating economic conditions and changing social values services to compound the task of the programmer in understanding complex problems.

The computer has become an important aid in analysis and decision making, and has inspired a number of analytical tools designed to help the programming process.

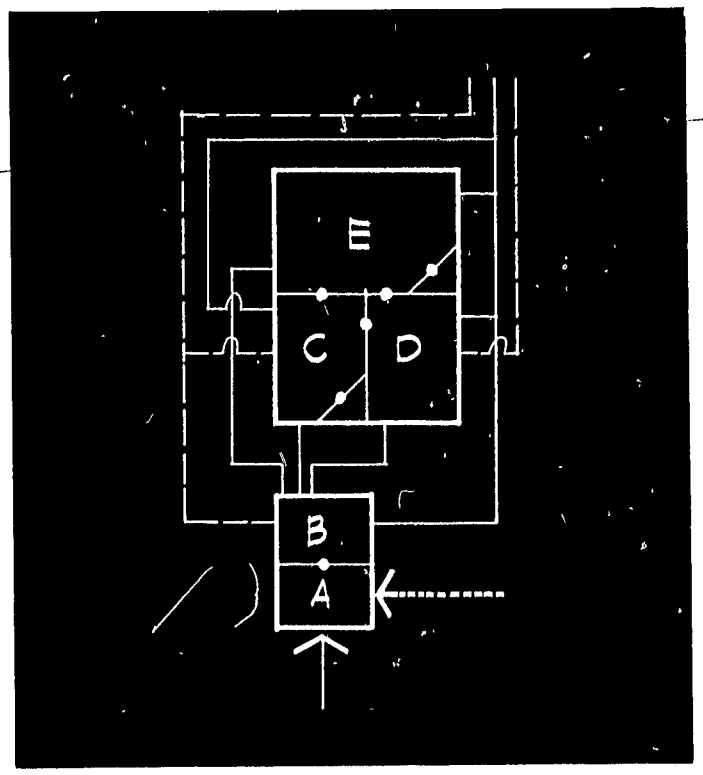
Evaluating Questionnaires
The questionnaire is designed to collect and quantify programming data from the user where the user is a very large group. It includes information about people, space and activities.

The complex aspect of using the questionnaire is that of analyzing the resulting data. This can be a computer function, since the computer is able to process large quantities of coded data and manipulate these according to given instructions. The output is in the form of tables, charts, and diagrams, which are notably helpful in several of the programming steps.

Computer Generated Space Requirements
A major task of programming is the detailed generation of space requirements based on current and projected occupancy, functional requirements, utilization factors and programmatic concepts. Working from this input data to the space requirements is a tedious manual task that entails repetitive mathematical calculations. Yet such calculations are a simple computer function, which moreover allows for a continuous updating of parameters. The computer has enabled the programmer to generate detailed and accurate user requirements from critical variables provided by the client.

What is more, the computer, due to its great speed, allows the programmer to explore the effects on the program of alternatives, both in terms of user needs and space requirements.

Analysis by Affinity Diagram
"Affinity" is a term given to the desirability of placing an activity close to (or separate from) another. It is a



38 The flow diagram describes the important interconnections that must be realized in the solution.

useful programming technique for grouping and locating activities. A number of tabular and graphic display techniques has been developed to analyze affinities, and the computer has been useful due to the large quantity of data involved. (37)

The affinities of various activity groups can be expressed in terms of computer generated circle diagrams or circulation frequency/load diagrams.

Analysis of these will aid the programmer, and later the planner or designer, in locating new facilities so as to relate to existing facilitites, or in organizing complex relationships between numerous activities.

Flow Analysis Diagrams

Very complex problems of function and organization, such as those for medical facilities, require of the programmer, a comprehensive understanding of organizational structure, affinities and circulation requirements. (38)

The concept of flow has produced a precise method of coding and diagramming affinities and circulation, in an abstract way, as programming information.

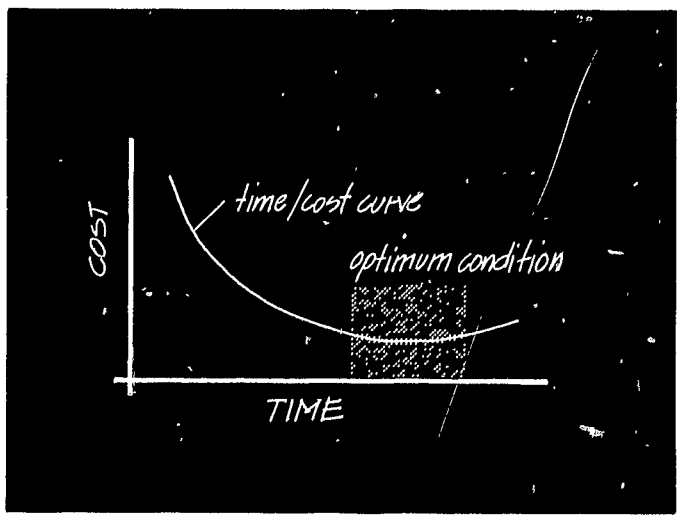
In flow, interconnections between functional spaces, after being given a code, are recorded in an interaction matrix and each circulation type is given a priority. An abstract diagram is then constructed from the matrix describing connection affinities of spaces for each circulation type. An overlay of diagrams is then made and circulation conflicts are resolved by manipulating the diagrams while maintaining the affinity codes.

This technique can be applied when the user has very precise affinity and circulation requirements.

Simulation by Mathematical Model

Mathematical model simulation allows the programmer to evaluate the consequences of manipulating the variables in a programming situation. (39)

A complex situation can be described in the form of a set of mathematical equations that approximate a real situation.



39 Model simulation is a computer aided method of analyzing *Economic Feasibility* or *Time/Cost Optimization.*

The sum of these equations constitutes a mathematical model, and describes the behavior of a system as a whole. Numerical values for the variables within the model are introduced and then processed to determine the consequences. Many sets of numerical data may be introduced in short order to compare the results of real life alternatives.

CRS has used computer aided model simulation, as in the economic analysis of an income-producing building in terms of the particular effects of occupancy type and mix, basic design assumptions and predicted market conditions.

This technique has introduced a new tool into the problem seeking process and promises to give new direction to solving urban problems.

Programming as a Non-Building Service

The intent of CRS programming is to seek out, and understand the *real* problems of a client. In complex projects, and in situations where there is a conglomerate client group (client-user, client-owner), programming becomes an intricate process and requires a high level of experience and organized expertise. Several developments now underway are an indication of changes that are beginning to have their effect on programming.

The technique of concurrent scheduling (conveniently termed "fast track" scheduling) compresses the project schedule, by overlapping activities that traditionally come one after the other. This scheduling method calls for a *"multi-phase"* form of programming that is integrated with the overlapping planning, design, production and construction activities. Fast track scheduling requires at least three concurrent phases of activities leading to a) site/foundation construction, b) building shell construction, and c) interior systems installation.

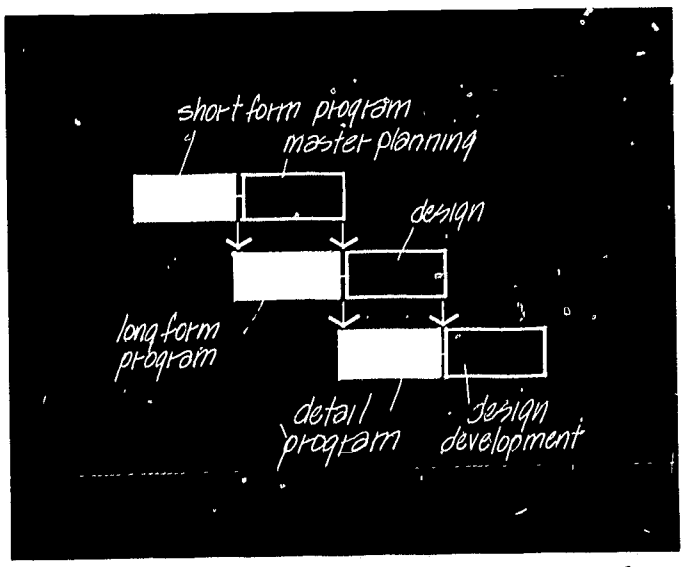
Programming procedures respond to fast-track phasing by means of *short-form* programming (to precede master planning); *long form* programming (to precede schematic design); and *detail* programming (to precede design of interior systems). (40)

Building project scheduling will always affect programming procedures, and flexibility within the framework of programming serves to respond to variations in project scheduling.

More and more often, programming becomes a distinct and identifiable service rendered as a separate phase prior to design. It is a unique non-building service in problem identifying, problem structuring, process management, and information generation and communication.

Building Systems and "Negotiable Programming"

The expanding trend to system building affects the entire building project delivery process. In programming terms, a resolve to use building systems is a goals-oriented decision which is tested at the first (goals formulation) step in programming and, if verified, will affect program content.



40 Concurrent scheduling requires three phases of programming.

The use of system building makes possible a more general, flexible form of programming conveniently referred to as "negotiable programming". Negotiable programming presupposes that the building system has been developed from user requirements and performance criteria, and that it will produce the kind of flexibility that will make net space requirements "negotiable" within a fixed gross area. The aim is to make the end product a building with the flexibility to change as the user requirements change. (41)

Through recourse to system building every program requirement remains negotiable throughout the design and building process, and because of inherent flexibility the functional organization of the interior remains always negotiable.

Multi-Disciplined Specialists

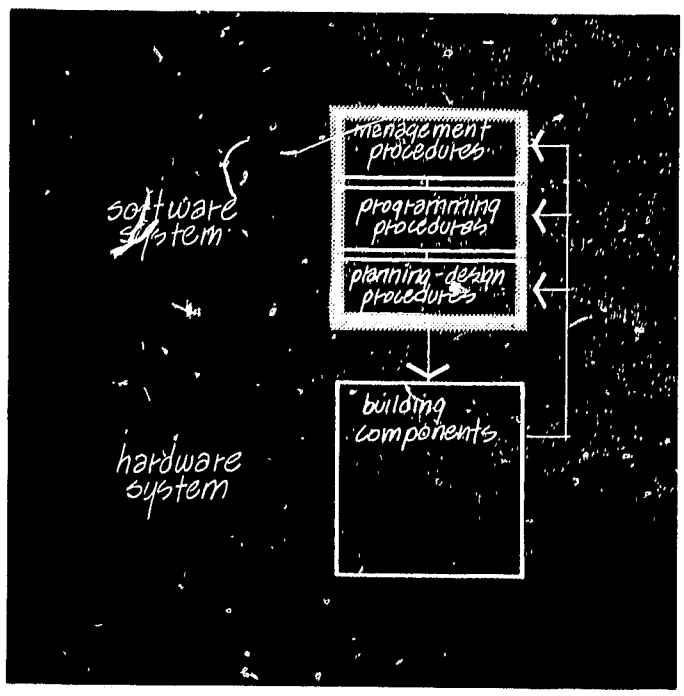
The third level programming process (page 8) will require new multi-discipline specialists: the architect/computer specialist, the programmer/systems analyst, the programmer/building type specialist, the architect/systems managers, the architect/economist.

Software: Where The Action Is

The pressures of complex problems have led to the development of computer programs (software) to serve the needs of architectural programming, planning, design, production and management.

The development and maintenance of such software has become a major source of concern for the well-prepared professional who understands both architectural and planning problems and the abilities and limitations of the computer. The life of a software system is very short, computer technology is changing rapidly and the scope of client problems is expanding.

A software system as a rule has applications beyond a particular project. The subsequent applications become an additional service that the architect can provide. Hand in hand with the evolution of software systems goes an expansion of the architect's professional service into the non-building design fields. It allows firms such as CRS to



41 The total systems approach links the software system and the hardware system.

undertake management services, information coding, analysis, storage and retrieval, model simulation, cost analysis and other services.

All these are at the basis of the emerging science of programming as it evolves in the years ahead.

This report has hopefully served the reader in providing a look at the present and future of programming—a discipline that is at the heart of every successful undertaking in planning and architectural design.

Investigation Series

1 Some Thoughts Concerning Beauty, by William W. Caudill, Thomas A. Bullock—June 1960*
2 Air Conditioning of Schools, by William W. Caudill, William M. Peña, Joe B. Thomas—June 1960*
3 Decentralized School vs. Centralized School, by C. Herbert Paseur—July 1960*
4 On-the-Spot Design of a School for Religion, by Franklin D. Lawyer—September 1960*
5 The Team Approach to Planning a College Science Building, by David B. Yarbrough—September 1960*
6 Two Trips in Sixty, by William C. Caudill, William M. Peña —February 1961*
7 Investigating the Feasibility and the Cost of Fallout Protection for a New Schoolhouse, by Corvair-Fort Worth and the CRS Team—July 1961*
8 Shells and the Educating Process, by William W. Caudill —July 1963*
9 The Primitive Quality in CRS Architecture, by John M. Rowlett—April 1964
10 Quality Profiles, by William W. Caudill, Franklin D. Lawyer, Charles E. Lawrence—June 1964*
11 In Education the Most Important Number is One, by William W. Caudill—December 1964, August 1967
12 The Practice of the Caudill Rowlett Scott Firm, by the CRS Team—November 1965; revised July 1968
13 The Bridge, by Alfred Paul Bay, MD; Matthew Dumont, MD; William W. Caudill—January 1968
14 Probes. A Search for Uniqueness of the Community College, by the CRS Team—January 1967
15 Pima County Junior College: An Approach to Campus Planning, by the CRS Team—September 1967
16 Educational Park, *A Case Study,* by the CRS Team— July 1968
17 The Urban Community College 68/69, by Bob Reed— November 1969
18 Problem Seeking, by William M. Peña and John W. Focke—November 1969

*Out of print

Additional copies of this and other investigations will be sent, where in print, upon request to Caudill Rowlett Scott, 3636 Richmond Avenue, Houston, Texas 77027

CRS

Investigation No. 18

Caudill Rowlett Scott
Houston, New York, Hartford

Design: Chermayeff & Geismar Associates