Title
How programming can become counterproductive: An analysis of approaches to programming

Permalink
https://escholarship.org/uc/item/4n05k3sd

Journal
Journal of Environmental Psychology, 12(1)

ISSN
02724944

Author
Mazumdar, Sanjoy

Publication Date
1992-03-01

DOI
10.1016/S0272-4944(05)80298-1

Peer reviewed

Journal of Environmental Psychology (1992) **12**, 65–91
© 1992 Academic Press Ltd

0272-4944/92/010065+27$03.00/0

Journal of ENVIRONMENTAL PSYCHOLOGY

HOW PROGRAMMING CAN BECOME COUNTERPRODUCTIVE: AN ANALYSIS OF APPROACHES TO PROGRAMMING

SANJOY MAZUMDAR

Program in Social Ecology, University of California, Irvine, CA 92717, U.S.A.

Abstract

I propose seeing environmental programming as varied and differentiated, and more specifically as having a diversity of approaches, which are best treated differently. I develop a typology of approaches to programming, based on the mode of data collection, analysis and presentation, and describe how their inherent assumptions, epistemological, ontological and methodological positions affect the efficacy of programming, often rendering it counterproductive to its original aim of assisting the designer in producing suitable physical environments. I discuss the implications of, and issues with, the different approaches to programming, and elaborate on the counterproductiveness of programming.

Introduction

Architectural and environmental programming were the names given to a set of activities introduced into the design process which involved the description of the needs/wants of the occupants through the systematic and comprehensive collection and analyses of data, the specification of goals, objectives and performance criteria, as well as the presentation of this information to the primary stakeholders—occupants, clients, designers (Preiser, 1975, 1976, 1978, p. 1; Wade, 1979; Palmer, 1981, p. 17). It was expected to assist occupants, clients and particularly designers in producing occupant suitable designs (Figs 1 and 2) (Richardson, 1969). Whereas in the past architects designed based on intuition, and whatever information they could obtain, with programming there was an effort at systematizing information collection through the introduction of scientific techniques and an emphasis on designs more responsive to occupants (Palmer, 1981).

These goals seemed laudable, and programming has been increasingly used in architecture since the 1960s (Davis, 1969), partly as a means to ensure a more useful and useable built environment. Indeed, to many, especially those who believe that architecture ought to cater to human behavioral needs (Lang *et al.*, 1974; Sanoff, 1977) and those who want goal-oriented architecture (Preiser, 1978, 1985; Brill, 1984), programming has become not only an important pre-design phase but almost an institution. This is an attempt at a systematic institutional analysis of programming. The question of interest is how effective is it in achieving its aims and how does programming become counterproductive?

I present a typology of approaches to programming along with an analysis of the assumptions and positions involved in each. The efficacy of the program can be affected by the format selected for presentation of program information. Even when one accepts the broader instrumentalist aims and assumptions of programming, analysis shows that inherent in the choice of an approach are stances, positions and assumption which affect the efficacy of the programming effort as this paper is intended to illustrate. There are ontological and epistomological differences between the approaches, as will be evident from the descriptions to follow. Viewing programming as varied allows us to highlight these assumptions and positions that underlie the choice of an approach, as well as the implications each has for the effectiveness of the programming effort (Zeisel, 1971). While in theory programs and programmers are supposed to ensure that the needs of all occupants are properly represented, this does not always happen, as I will show. I analyze issues related to the approach selected for the program, the advantages and counterproductive aspects of each. I conclude by elaborating on the implications of this study.

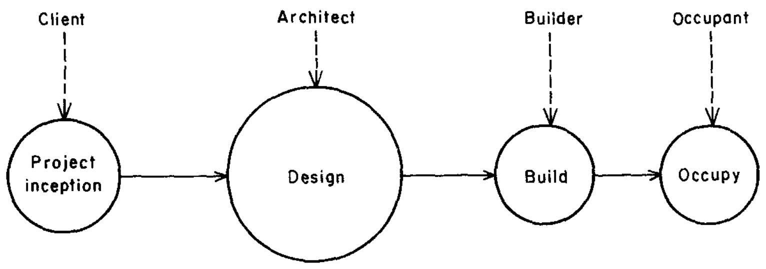

FIGURE 1. Pre-existing design process.

Analysis of the qualities and characteristics of the different approaches to programming can help us understand the strengths and weaknesses of each. It can assist in the selection of an approach, given a building project. It can also assist in devising new approaches to counteract problems with existing ones, by combining approaches for example. Thus, this critical analysis is intended to focus attention on the ultimate goal of creating more suitable and responsive environments.

The idea of developing a conceptually constructed typology of programming occurred while I was involved in developing a program ('X') for a department of a large organization.[1] The process was designed with sufficient care to ensure that this program contained almost all elements that would lead to the satisfaction of all concerned stakeholders. Program X[2] and the finished architectural product were satisfactory to most. Yet the act of doing program X, the research associated with it and further thought revealed that the approach to programming selected by our organization had some potential counterproductive aspects, and that several aspects and unresolved issues generic to programming can make it potentially counterproductive. The following sections deal with one set of these. They are based on reflections about the process of programming, the author's role as a programmer, on interviews carried out during and after the process was completed, a brief review of some writings on the subject, and a review of approximately 25 programs done by several organizations.[3]

Form of analysis

Before getting into the specifics of the different approaches it will be useful to describe some general aspects of the approach adopted here.

In examining and classifying the various approaches to programming I am concerned primarily with the manner in which program information is presented in the final program document to the stakeholders and also in the way the data is collected and analyzed. I am taking the view that programming is a step preceding design in the design process, in which information regarding the functions of the building and the wants of the occupants is collected and set forth as features the design ought to accommodate.

In analyzing each kind of program I shall look at several important aspects.

Philosophical position: Philosophical positions are involved in each of the approaches. How should people's wants related to building be catered to, and therefore how should program information be collected, analyzed and presented? On one end of the continuum emphasis is placed on society, a position I label 'culturalism' or 'societalism', with the individual seen as subordinate to society. Societalists emphasize and attend to social agreements, shared values, mores and norms of the group, social affinities and cleavages, socially negotiated arrangements, social aspirations and ideals, not found in the other approaches. At the other end is a position labeled 'individualism', where the emphasis is on

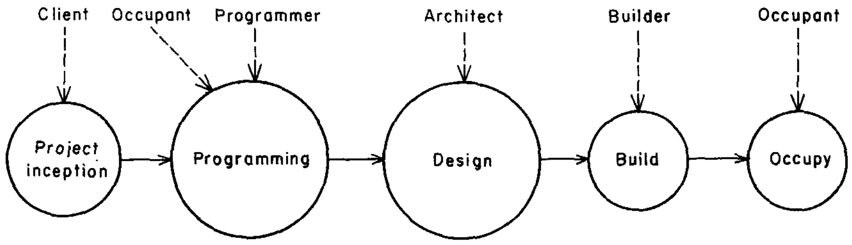

FIGURE 2. Modified design process including programming.

the individual, with little or no attention given to society. In this position individual preferences, desires and wants are described. The individual thus is seen as paramount. Individualism encourages the representation of individual idiosyncracies which the architect can cater to. Taking the philosophical position of individualism involves assuming that the individual occupant's wants are more important and ought to be represented as such in the program which may overlook other important aspects. While individual occupants are dealt with, group values and effects of social mediation of spatial wants are not addressed. Hence it is possible to err with regard to group or organisational suitability, since these are not dealt with by the program. In another position, 'aggregatism', individual preferences are subjugated to aggregated majority opinion. In instances individual responses are reduced to typicals or means mathematically computed from aggregates of individual opinions. This places greater emphasis on common features and through mathematical computation reduces some of the variability of individual preferences. Aggregationists feel this reduces some of the 'error' associated with individual responses. Catering to these would satisfy those whose profile most closely matches the typical. Yet most people do not think of themselves as typical, but rather prefer to think of themselves as somehow distinct from others and unique. Individualists would claim that this method takes away some of the characteristics and idiosyncracies that individualism is able to provide and cater to. Individualists assume that societal values appear in individual preferences, since an individual is a member of society and, responding to individual wants, will automatically cater to societal values. Culturalists would argue that unless the focus was on finding them wants expressed by individuals may not reflect societal values. Aggregationists may also believe that distilling commonalities of individual responses will capture societal values. Societalists, on the contrary, would argue that societal values and preferences are not equally distributed in the population and that majority opinion or mathematical computation of means from individual responses may not lead to an understanding of societal values. Thus it seems highly probable that the three philosophical positions and their methodologies can lead to very different kinds of information.

Neutrality / stance. Each approach can be seen as representing, aiding, to a greater or lesser degree, one set of stakeholders and therefore having a stance. Categorizing each approach as 'pro-architect', 'pro-occupant' or 'pro-programmer' in stance, even though this might cast the position a little starkly, helps in assessing the non-neutrality and implied position of each approach and brings attention to who a particular approach favors.

Representation. Representation of occupant wants is essential for the architect to understand them and design accordingly. What gets represented and what does not and the mode of representation can directly affect the effectiveness of the program.

Conversion. Conversion of social and behavioral information into clear spatial terms is necessary at some point for the programmatic requirements to be provided by the design. A good portion of the information collected in programming is social and behavioral in nature, including descriptions of activities, which can be presented as such in the program with the conversion left to the architect. Alternatively, the programmer can convert the information into spatial terms with specifications of spaces and their qualities and characteristics. The latter requires of the programmer some familiarity with the kinds of information architects need and the way they organize and use information (i.e. speak the architect's language) for it to be useful. Such information is likely to be more intelligible to architects than behavioral information. Who conducts this conversion with what skills and tools can affect the effectiveness of the program. As will be pointed out, some approaches leave this task in more capable hands than others.

Fidelity / discernability. The program can be expected to provide information that has a high level of fidelity or faithfulness and accuracy. As will be obvious later, the fidelity of the information provided by the different approaches is not equally high. Some actually obfuscate important information which can make the program counter-productive. Discernability is a factor by which we can judge whether the requirements of different individuals, aggregates or groups combined for programming purposes are identifiable in the program.

In-depth up-close knowledge and understanding. To what extent does the program provide the architect with good in-depth knowledge and understanding of the world and life of the occupants? To what extent can the architect remain distant from the world of the occupant? These relate to the sensitivity with which the architect understands the occupants' world and designs for it.

Verifiability. It is often necessary for the architect to find and verify specific pieces of information provided in the program, and to check for errors. Such verification is more easily accomplished with some approaches and impossible in others without re-programming. Checking design provisions against programmable requirements can assist in ascertaining the suitability of the design (Farbstein, 1978).

Skills. Certain skills, such as collecting, analyzing and presenting program information, are necessary for program-related tasks. Effectiveness of programs can be affected whether the programmer has the skills necessary to carry out those tasks. For example, does the programmer have the skills to collect and analyze individual, aggregate and group wants?

Appropriateness. Appropriate or suitable to the occupants was the reason for the introduction of programming. Should the building be appropriate in a coarse sense or fine? Does the approach selected for the program affect appropriateness?

Assumptions. Each approach has certain assumptions regarding the kind of information to be presented, how is it to be collected, and the role of the programmer. These affect decisions regarding the program. When these assumptions turn out to be invalid or wrong the programming effort can become counterproductive.

Approaches to programming

Conceptually, programming can have several different approaches based on the final form of the program and the mode of data collection and analysis. My intention is to consider what each selection entails, irrespective of whether these are different programs or in the same program. While some approaches may appear better than others, my attempt is not to 'advocate' any one approach.

The categories I have devised have two primary dimensions. One dimension takes into consideration the orientation of the program (Farbstein, 1978). Broadly, the program can be space oriented or person oriented. Information for the program can be collected and presented from the point of view of the spaces required or that of occupants and their wants (Proshansky *et al.*, 1970).[4] The second dimension, whether space or people oriented, is based on the program's analytical unit, which can be individual, aggregate or group. Together these yield six possible approaches to programming as shown (Fig. 3). Individual is used to indicate unitary space or person. Aggregate refers to the clumping together of units sometimes on the basis of certain common characteristics. It is essentially a mathematically computed picture. Clump refers to aggregates not very meaningful to the occupants. Aggregate spaces refers to spaces aggregated or clumped together so that they could be dealt with together. Aggregated persons refers to a number or set of persons aggregated together, often based on certain common characteristics, so that they could be dealt with together. Group refers to an organization or combination of units meaningful to one or more sets of occupants or other stakeholders. Grouped persons refers to a social unit of interacting individuals with shared, negotiated values and preferences; they need not share common characteristics other than group membership. Grouped spaces refers to the grouping together of spaces, not necessarily with common characteristics, which serve a common architectural purpose or theme to be utilized in the design.

These are analytical categories, and could also be seen as positions on a continuum. The distinctions among the categories depicted are not very rigid as they may seem, since in practice it is possible for a programmer to produce different programs using different approaches or to combine several approaches in a single program.

A description of each approach can be concisely presented in a chart (Fig. 4) which enables comparison along several dimensions.

The dimensions described are objectives, characteristics, assumptions, advantages and disadvantages/counterproductive aspects. Each approach can be said to have a primary objective of providing a certain kind of information, such as whether to present the requirements of the people or of the spaces. This is described under the section titled objectives. The section on characteristics describes

	ANALYTICAL UNITS		
	INDIVIDUAL	AGGREGATE	GROUP
SPACES	Individual Spaces	Aggregated (clumped) Spaces	Grouped Spaces
PERSONS	Individual persons	Aggregated (clumped) persons	Grouped persons

FIGURE 3. Approaches to programming.

	Individual Spaces approach	Aggregated Spaces approach
Objective	To describe requirements of each individual space	To describe requirements of spaces aggregated in some way
Characteristics	Unit of analysis and description—an individual unit of space Each space given name or number for identification Lists and describes: • architectural, physical and environmental qualities • relationship to other spaces • preferred location • activities supported/housed by that space • who occupies space • number of occupants Emphasis on physical aspects. Lack of emphasis on human and social aspects.	Unit of analysis and description is an aggregate of spaces Each aggregate given a name of number for identification Lists and describes: • architectural, physical and environmental qualities for each aggregate • relationship to other aggregates • preferred location • activities to be supported/housed by this aggregate • the occupants of the aggregated spaces • the number of aggregates and the number in each aggregate Emphasis of physical aspects. Lack of emphasis of human and social aspects.
Position	'Pro-architect' in stance	'Pro-programmer' in stance
Assumptions	Philosophical position of 'individualism' Representation of occupant wants through reified individual spaces Conversion of occupant wants into spatial terms can be accomplished by programmer Fidelity in representation and discernability of occupant wants high Servicing occupant wants enabled by this approach Intelligibility of information to the architect high In-depth knowledge about the occupants provided Verification of occupant wants and checking of design provisions possible Skills required by this approach available with the professionals Responsibility for tasks and errors are clear	Philosophical position of 'aggregatism' Representation of occupant wants possible through depicting requirements of aggregated spaces Conversion of social and behavioral data into spatial terms can be done by the programmer Fidelity of information and discernability of occupant wants high Servicing of occupants wants will be done through this approach Intelligibility of information will be high In-depth knowledge about the occupants provided by this approach Verification of occupant wants and checking design provisions can be done in this approach Skills for data collection, analysis and depiction will be with the programmer Responsibility of the programmer will not exceed his/her skills
Advantages	Architect's interests and concerns attended to Architect not required to make conversion of occupant wants into spatial requirements Architect not required to attend to human and social aspects not within his/her training Architect's responsibility reduced to dealing purely with design and meeting programmatic requirements Checking of design provisions can be done and major blunders avoided	Architect's interests and concerns attended to Architect not required to convert occupant wants into spatial requirements Architect does not have to carry out human and social analysis Architect's responsibility is design of programmatic requirements Programmer can reduce redundancy and repetition by aggregating spaces as he/she sees fit Checking of design provisions possible when units in aggregate are identical
Disadvantages/ Counter productive aspects	Philosophical position of 'individualism' implies neglect of others Representation of occupant wants difficult when several occupants in space, and for large jobs with numerous spaces Conversion of occupant wants into spatial terms not easily possible Fidelity and discernability jeopardized by emphasis on spaces Servicing not easy as requirements need to be reformulated for trades In-depth knowledge of occupants not provided Verification not possible without reprogramming Skills for data collection and analysis generally available to programmers, but not for conversion Responsibility of programmer high	Philosophical position of aggregation neglects others and has own problems Representing occupant wants and requirements of individual spaces not directly possible Conversely not easy due to scarcity of tools Fidelity not as high as some approximation included Discernability not possible Servicing not easier Intelligibility of representational mode, aggregation, low Verification not poossible, checking of individual spaces not possible Skills of data collection, analysis, and representation available but not for conversion Responsibility of programmer rather high

(continued)

	Grouped Spaces approach	Individual Persons approach
Objective	To describe requirements of spaces grouped in a way meaningful and useful to the architect	To describe and present information on and spatial requirements of each individual occupant
Characteristics	Unit of analysis and description is a group of spaces Each group is given a name or number for identification Lists and describes: • architectural, physical and environmental qualities for each group • relationship to other group • preferred location • activities to be supported by the group of spaces • the occupants • the number of occupants Emphasis on physical aspects Lack of emphasis on human and social aspects	Unit of data collection and writeup is the individual person Individual person given priority Individual wants, needs, preferences and idiosyncracies described Lists and describes: • each occupant's characteristics, activities, roles • time, duration, and frequency of the activities • spatial preferences for the above
Position	'Pro-architect' in stance	'Pro-occupant' in stance
Assumptions	Philosophical position of 'grouping' spaces will lead to more suitable design Representation of occupant wants through depicting requirements of groups of spaces Conversion of social and behavioral information into spatial can be achieved Fidelity and discernability high Servicing of occupant wants enabled by this approach Intelligibility of grouped spaces information high In-depth knowledge about the occupants will be provided by this approach Verification of program information and checking design provisions against requirements possible Skills required by this approach will be available to the programmer Responsibility of the programmer not beyond his/her skills and activities	Philosophical position of 'individualism', individual occupant most important Representation of individual wants can lead to appropriate design Conversion of social and behavioral information into spatial terms possible Fidelity of the information will be high and individual wants discernible Servicing of occupants can be done Intelligibility of the information will be high In-depth knowledge of the occupants will be provided by this approach Verification of program information can be done as can checking of design provisions Programmers will have skills required for data collection, analysis and depiction Responsibility of the professionals will not exceed capabilities
Advantages	Architect's interests and concerns well attended to Architect not required to do conversion Architect not required to do social and behavioral analysis Architect's responsibility only servicing programmatic requirements Programmer compelled to consider architectonics of design Intelligibility of information likely to be high	The focus is on individual persons Individual wants depicted Individual occupant wants can be serviced Architect compelled to become familiar with individual wants Discernability of individual wants high Verification and checking possible Programmers have the skills necessary Responsibility of programmer within his/her skills
Disadvantages/ Counter productive aspects	Philosophical position of grouping spaces may not lead to more suitable design Representation of occupant's wants not directly possible Conversion made problematic by the dearth of techniques Fidelity may not be very high due to problems with conversion Discernability is poor Servicing of occupant wants not directly possible In-depth knowledge of occupants not provided, encourages distance Verification not possible Skills for grouping are not common among programmers Responsibility of programmer extremely high	Philosophical position of individualism leaves out social considerations Representation of individual wants tedious with large numbers of occupants Conversely confounded by lack of tools Fidelity may not be high depending on method Servicing of individual wants possible but may need additional processing Intelligibility of the behavioral and social information can be low for architect In-depth knowledge on occupant provided but is often neglected Responsibility of architect includes conversion

(continued)

	Aggregated Persons approach	Grouped Persons approach
Objective	To present information on and spatial wants of occupants aggregated together based on common characteristics	To present information on the nature, functioning and spatial wants of groups of occupants
Characteristics	Units of analysis is the aggregate Emphasis on aggregated preferences and common requirements of aggregates Lists and describes: • typical characteristics of persons aggregated ∘ activities of each aggregate, their time, duration, frequency • manifest spatial requirements and preferences	Unit of analysis is a group of person Emphasis on the group and its preferences Lists and describes: • the shared values, interests, understandings, and way of life of the group • common negotiated arrangements and agreements • social cleavages and boundaries • space related preferences
Position	'Pro-programmer' in stance	'Pro-occupant' in stance
Assumptions	Philosophical position of 'aggregatism', aggregated wants seen as important Representation of occupant wants best done through aggregation Conversion of social and behavioral data into spatial requirements possible Fidelity of information and discernability high Servicing of occupant wants enabled by aggregating occupant wants Information provided intelligible, aggregating will not obfuscate In-depth knowledge about the occupants provided Verification of program information possible and design provisions can be checked Programmers have the skills to present aggregated information and architects able to make conversion and design Responsibility of the professionals within their skills and knowledge	Philosophical position of 'societalism' Representation of occupant wants best done through study of groups Conversion of this information into spatial requirements possible Group information will have high fidelity and discernability Servicing better through depiction of group social information Information will not obfuscate and will be negligible In-depth knowledge about occupants provided Verification of program information and checking design provisions possible Programmers have skills for cultural analysis Responsibility of professionals involved will not exceed skills and abilities
Advantages	Focus on occupants, not on spaces When number of occupants large: • reduces tedium and work • introduces statistical techniques Responsibility of programmer within skills usually	Emphasis on people Group developed values, mores, ways of relating to environment described Programmer compelled to understand social, cultural aspects In-depth up-close knowledge of social unit provided Architect introduced to world of the occupant Designs can be based on aspects meaningful to the group Designs can be supportive of social/cultural values Effects of imposition of external values can be considered Works well for existing social units Verification possible
Disadvantages/ Counter productive aspects	Philosophical position of 'aggregatism' neglects viable alternatives Representation of majority preferences and ways may be misrepresentative Conversion made difficult by lack of tools Fidelity problematic as approximations built in Discernability impossible Servicing aimed at 'typical' which may be non-existent Information categories not intelligible to occupants or architect In-depth, up-close knowledge not provided only generalized picture Verification not possible without re-programming, checking not possible Skills in sampling, and quantitative analysis not always available	Philosophical position of 'societalism' ignores individual and aggregate aspects Representation of group values, individual preferences not depicted Conversion problematic due to unavailability of tools Discernability of individual wants not possible Servicing dependent on proper conversion Intelligibility of information may be problematic to architects unless jargon-free Checking of design provisions not possible Skills of cultural study not common among programmers Responsibility of programmer very high

FIGURE 4. Comparison of the six approaches to programming.

special aspects of the approach, including the unit of analysis and description and the aspects of program information emphasized or focused on. The assumptions section describes the underlying assumptions of each approach. The advantages and strengths of each approach are described in the advantages section. The section on disadvantages and counterproductive aspects describes how the assumptions may not be correct, and how the programming effort can become unproductive or counterproductive. Disadvantages that may not be counterproductive are also described. Finally, I provide examples wherever possible.

(1) The Individual Spaces approach.

The objective in the Individual Spaces approach is to describe the requirements for each of the spaces to be provided.

The unit of analysis and description is an individual space—such as a room. Each space is given a name or a number for identification. The programmer investigates and described or lists the size of this space, relationship to other spaces, preferred location, the architectural, physical and environmental qualities required, what activities occur in that space, who occupies it, and the number of occupants.

Space is a primary element designers create and manipulate in their designs. And since in this approach the programmer attempts to collect, analyze and present information through spaces, i.e. in a form, and speak in a language, intelligible to designers, this approach favors the architect and is 'pro-architect' in stance.

This approach makes the following assumptions. It assumes that representation of occupant wants can be done through those of spaces, that is by reifying spaces and attributing wants/needs to them. Conversion of occupant wants into spatial terms can be done by a programmer or occupant and that these stakeholders are able to think spatially in order to do this. In the list of spaces and their requirements occupant wants associated with each space will be discernible. Servicing of occupant wants can be properly accomplished by listing spaces and their qualities and characteristics.

The advantages of this approach are the following. The architect's interests and concerns are attended to and program information presented in a manner the architect can easily understand. The architect is not required to convert occupant wants into spaces. Human and social aspects, not within the training of an architect and therefore difficult, are not mentioned and do not have to be dealt with by the architect, and he/she can deal purely with physical factors and design. The architects' responsibility and work are reduced and made easier in this approach, since the architect simply has to meet only the physical requirements stated. This program can be used as a checklist to check whether requirements as stated in the program have been provided in the design prior to construction and major blunders avoided.

There are disadvantages that can make this approach counterproductive. Representing occupant wants through those of spaces may not be easily accomplished as occupant wants are not directly depicted. It is easier when there is one-to-one correspondence between an occupant and a space, and when the program specifies how the space is to perform. When the Individual Spaces program is unable to represent wants of occupants (individuals, aggregates or groups) there may be obfuscation and the effort may become counterproductive. Listing individual space requirements for large jobs with numerous spaces can be quite tedious.[5] Conversion of occupant wants into spatial terms expected of the programmer is not easily possible due to the paucity of tools, as will be explained later. Discerning occupant wants is difficult since occupant wants are not represented as such in the program. This is exacerbated when spaces have multiple occupants. It then becomes extremely difficult for the designer to cater to individual wants. In such instances the program becomes counterproductive. Servicing requirements may not be easy in an Individual Spaces approach, even with listing of requirements by spaces. Specific environmental services are often associated with particular trades, such as plumbing, heating, air conditioning, electrical, carpentry and metal working. In practice, program documents are generally dealt with by architects and are rarely turned over to engineers or to the trades. Architects in the U.S.A. find it easier, and are sometimes required to produce drawings for specific trades, as opposed to indicating all services in one set of drawings. An Individual Spaces program has to be untangled for servicing by differing trades (MIT, 1982). Individual Spaces programs done by architects often list services by trades. In-depth knowledge and understanding of the lives of the occupants is not required of the architect in this approach. This approach does not provide a good understanding of the world of the occupant. While programmers can be expected to obtain in-depth knowledge this does not always happen. Their task is seen as completing blanks in a list or form and programmers often resort to simply asking occupants their space-related wants, mostly through a questionnaire (Appendices 1 and 2) (see also Sterling *et al.*, 1988, pp. 288–289). The architect can simply provide programmatic requirements. Verification of occupant wants is rather difficult as they are not represented in the program. If there is an error in representation it will be difficult for the architect to check, rectify the error or change the program, as that will require reprogramming. Making changes in the program is often further hampered by various unanticipated uses of programs.[6] 'Performance Specifications' (Brill, 1971, 1972) was probably a response to the inability to obtain appropriately designed buildings through the standard Individual Spaces approach, due to its inability to accurately and completely represent occupant wants and therefore assuring appropriately designed buildings. The programmer is expected to have the skills necessary to collect and analyze behavioral information. But converting this information into spatial terms is usually not within the skills of the programmer. The responsibility of the architect is primarily to provide the requirements as stated in the program. In contrast, the programmer's responsibility is rather high, including not only collecting and analyzing social and behavioral information but also converting them into spatial terms and presenting them in the program. Mismatches between program requirements and design provisions is the responsibility of the architect. But mismatches in occupant wants and their representation in the program is the responsibility of the programmer.

Examples are Kantrowitz, Min & Associates (1991), Welch + Epp Associates (1986, 1987), Caucus Partnership (1987*a*), Research Facilities Design (1985) and Stone, Marraccini & Patterson (1986*b*).

(2) The Aggregated Spaces approach

The objective is to present the requirements of the required spaces aggregated together using common characteristics, such as size, or commonality of architectural qualities and requirements.

Aggregates of spaces are the units of analysis and individual spaces lose importance. Each set of aggregated spaces is given a name, e.g. Office Type A, or number for identification. It lists and describes the architectural, physical and environmental qualities of each aggregate, the size, preferred location, the occupants, their numbers and their activities.

Convenience of the programmer and, to a lesser extent, that of the architect is prioritized by this kind of program. This approach displays a 'pro-programmer' stance.

Several assumptions are made in this approach, some similar to the Individual Spaces approach. Representation of occupant wants can be appropriately accomplished by describing the requirements of aggregated spaces. Conversion of occupant wants into spatial terms can be done by a programmer or occupant who will be able to think spatially in order to do this. Occupant wants will be discernible on the list of spatial requirements. Occupant wants associated with each aggregate will be similar, if not identical (i.e. all 50 officers will have nearly identical wants catered to by 50 Type A officers). Servicing of occupant wants can be appropriately done through a program depicting requirements of aggregated spaces. The abstractions, categories and aggregations, e.g. 'Office Type A', devised by the programmer will be intelligible and useful to the architect and occupants. It is also assumed that provision of the nearly identical aggregated spaces will be seen as identical by the occupants, i.e. geographical and social space will be seen as equivalent, flat and level.

This approach has several advantages. The architect does not have to translate occupant wants into spatial terms. By following this approach the programmer can simplify or reduce a large amount of the work and tedium associated with redundancy and repetition of the Individual Spaces approach. For example, instead of doing offices of a certain type 50 different times, the programmer could name it 'Manager's Office' or 'Faculty Office', 'number required = 50' and be done simply by doing one. In so far as requirements are identical it allows checking of design provisions with those stated in the program.

Several disadvantages can make the effort counterproductive. The philosophical position is that spaces with common features can be aggregated, with emphasis on majority values. Representation of occupant wants through those of aggregated spaces is difficult for several reasons. In representing reified spatial requirements occupant wants do not get depicted directly. Representation of individual spaces, unless they are identical, is not possible since spaces are being aggregated. Conversion of social and behavioral information into spatial terms is not easy as tools for conversion are scarce. Aggregation leads to homogenization. Discernability of individual occupant wants associated with each set of aggregated spaces becomes even more difficult than in the Individual Spaces approach as the program for 50 Type A offices may not display the wants of the individual occupants. As a result the architect can cater to the programmatic requirements for the Type A office but not to those of individual occupants. If, as is likely, the programmer believes that occupant wants are not identical then approximations and errors would be built into the program from the outset. Fidelity of program information can become an issue. The programmer may avoid repetition by aggregating 50 offices having commonalities along several dimensions. If not identical along these dimensions the programmer has to make judgment calls regarding which descriptor—mean, mode or other—to use. Fidelity is low as these are approximations in representation of occupant wants, especially if variance and deviance from the selected descriptor are large. Consequently a design based on such a program would not be able to achieve its purpose, and the programming would be counterproductive. Servicing may not be easier with a program listing requirements by aggregated spaces unless requirements are listed by trade. Even if the program did not specifically call for identicalness, such aggregation may often lead to the assumption that provision of 50 Type A offices will satisfy the 50 occupants and to the provision of identical spaces. This implies the improbable that the 50 Type A office occupants have identical wants and goes against conventional knowledge regarding people's need for personalization, differentiation (Sommer, 1969).[7] Fine tuned appropriateness is not possible. Also identically sized and shaped spaces are frequently not seen as identical by the occupants as geographical and social spaces are not seen as equivalent, flat and level, and characteristics such as geographical location, view from the space, slight differences in shape or size make some of these 'identical' spaces more desirable than others. In one office building, view from the space became a premium in otherwise very similar offices. Intelligibility of the representational mode, aggregation, is likely to be low to the architect and occupant. Aggregating spaces is useful to the architect if it leads to meaningful design solutions. This is often not the case. If the architect is not required to congregate physically in one area the 50 Type A offices clumped together in the program and they can be located anywhere and in any configuration, then the aggregation is not meaningful or useful to the architect. Aggregation is an abstraction devised by the programmer for ease in programming, and may similarly not be meaningful or useful to occupants. Also, while aggregation is based on a single or few common characteristics, commonality in other aspects may be assumed. In-depth knowledge and understanding of the occupant's world is extremely low as the program depicts only aggregated information. This distancing of the designer from the end occupant is likely to lead to less, rather than more, appropriate design. Verification of individual wants would be very difficult as they would not be featured. This automatically places a heavier burden on the programmer to ensure that the spaces called for will be conducive to the activities of the occupants and that a 'user-needs gap' will not exist. Skills such as finding commonalities, computing means and deviations and variances are fairly common with programmers trained in the social sciences but may not be so common among those trained in architecture. Since in this approach the programmer has to colect and analyze information, convert them into spatial terms and aggregate them, the programmer's responsibility is very high.

Examples are Caucus Partnership (1987*a*, 1988), CRSS (1989, 1990), UCI/OPP (1987), NBBJ (1990), Zimmer, Gunsul, Frasca (1989*a*,*b*), MIT (1973, 1982), Farbstein & Associates (1991), Donaudy (1989), MBT Associates (1988) and Ryan (1989).

(3) The Grouped Spaces approach

The purpose of this approach is to provide information on spaces grouped in a way comprehensible, meaningful and useful to the architect in designing. The major distinction with the Aggregated Spaces approach is that aggregations are usually purely mathematical and do not have any implications for design, whereas groupings usually do.

In this approach the unit of analysis is a group of spaces. Spaces could be grouped on the basis of a variety of criteria, such as proximal relationships on common functions, for example 'master spaces and servant spaces', commonality of structural bay and services. The grouping can be congruent with the social grouping of the occupants, which is more likely to result in a better fit. Generally there is no conscious attempt to follow the groups used by the occupants' social organization.

This approach is most helpful to architects, and so in stance is 'pro-architect'.

The assumptions are the following. The philosophical position is that grouping spaces and thus speaking the architect's language will engage the architect into producing a more suitable building. Representation of occupant wants can be done through those of spaces. A programmer will be able to understand occupant wants, convert them into space requirements without much slippage and loss. Occupant wants associated with each space will be discernible and understandable. Servicing occupant wants will be enabled by a Grouped Spaces approach. The form selected for the grouping will be intelligible primarily to the architect and that this will enable a better fit of occupant wants with the design than simply aggregating them. It is expected that the programmer will have the skills for data collection, analysis, conversion, representation, and for abstracting them into groups comprehensible and useful to the architect.

Advantages of this approach include the following. The approach compels the programmer, in the process of grouping spaces, to consider the architectonics of design, and thus to attend to some of the architect's considerations. The groupings should be intelligible to the architect since it is based on architectural considerations. It presents materials in a manner architects can understand and relate to. The architect's work is greatly reduced in this approach. The architect's responsibility is limited to the design of the grouped spaces and does not include conversion, grouping or social analysis. Grouping spaces is likely to lead to architecturally well-organized spaces.

There are several possible disadvantages. The philosophical position of grouping spaces in the program is likely to lead to buildings that work out to be less, rather than more, appropriate, as the categories for grouping are preconceived and imposed as opposed to being contextually developed. While it is possible that speaking the architect's language will encourage the architect to take greater interest in the program and be more responsive, it is not clear that this is more likely. Providing the architect more processed information which can be readily incorporated in his/her work encourages the architect to take such information as 'given' to be accepted rather than understood, questioned and challenged. Representation of occupant wants is not direct but through reified spatial requirements, as with the other space-oriented approaches. Conversion has to be carried out by a programmer, but tools are scarce. Discernability of occupant wants is not high but requirements of individual spaces in the group is better than in the Aggregated Spaces approach. In-depth up-close knowledge of the occupants' social world is not presented. The architect is not persuaded to understand the world of the occupants and can remain unfamiliar and distant from it. This approach actually encourages such distance by presenting the architect with more processed and organized information. Verification of occupant wants is extremely difficult as these are not presented. Skills for deriving meaningful abstract groupings or spaces requires the programmer to understand architectural design. While programmers can be expected to have this capability many do not. Responsibility of the programmer in this approach is extremely high, proceeding far into the design by organizing the spaces.

Examples are Welch + Epp Associates (1986), Sterling, Wilford & Associates (1988), CRSS (1990, 1991), Caucus Partnership (1988) and Farbstein & Associates (1991).

(4) The Individual Persons approach

The objective of this approach is to present wants of each individual occupant.

The unit for data collection and write-up is the individual person. Each individual occupant is of paramount importance, and the individual's wants, needs, preferences and idiosyncracies are included.

This approach has the interests of the individual occupant in mind, favors the occupant and so in stance is 'pro-occupant'. It would probably be followed by occupants if they were to do the program.

Several assumptions are made in this approach. The philosophical position of 'individualism' is taken and individual occupant's wants seen as paramount. The building, it is felt, ought to be designed for the wants of the individual occupant. Representation of individual wants will enable the design of an appropriate building. Individual idiosyncratic wants are to be depicted. Conversion of social and behavioral data presented in the program into spatial terms can be carried out by the architect. Discernability of occupant wants will be possible, it is assumed. Servicing of occupant wants can be done using this approach. Information presented by this approach will be intelligible to the architect. In-depth up-close knowledge of the occupants will be provided by this approach.

Advantages of this approach are the following. The focus is on people—the occupants. Detailed information on individuals can be sought and provided. Display and consideration of the individual's spatial and architectural preferences compels the architect to become familiar with individual occupants which increases the likelihood of their needs being serviced. The architect can then choose to categorize the wants in a manner best suited to his/her individual approach to design. In small projects it is possible for the architect to cater to individual wants and idiosyncracies. Discernability of individual wants is high. Verification of individual wants is possible. There is also the potential of using the program as a checklist to ensure that programmatic requirements are furnished by the design. Programmers usually have the skills to do this kind of program.

Disadvantages and counterproductive aspects include the following. The philosophical position of individualism leaves out consideration of other positions and may not be the most appropriate in instances when addressing group wants is important. Representation of individual wants for a large number of occupants can become quite tedious and may not enable more appropriate designs. It is expected that large projects be allocated more time so that programmers and designers can do the necessary task for each individual, but in practice this rarely occurs. As time is reduced, the programmer has to find ways to reduce the work. The individual then is not treated as paramount. In practice, architects rarely organize large numbers of occupants into categories and sub-categories so that a larger number of occupants can be more appropriately serviced. Often the tendency is to reduce this number to a more manageable set through very rough estimation or 'eye-balling', or development of a 'typical', 'prototypical', 'user profile', or standards (De Chiara & Callender, 1980) rather than use proper analysis. Such eye-balling may lead to errors which then can get concretized in the design. Conversion of behavioral and social information into spatial terms is left to the architect but is difficult due to the dearth of tools. While this approach has high discernability, fidelity is not necessarily high. Individual wants are often learned through self-administered questionnaires, which may not provide options for individuals to really express their preferences. Individuals can provide information on aspects they have thought about, and have readily available. They are unable to provide information they do not have. Servicing individual occupant wants is furthered by this approach, but the information produced may need additional processing for servicing. For example, the architect needs to know not only individual occupant wants but also how to service them through spaces. Intelligibility of the information can be low and the behavioral information may be confusing to the architect. For example, reconciling different preferences for ambient temperature or lighting may be difficult for the architect. In-depth knowledge and understanding of the group is often neglected. Responsibility of the programmer is only for collecting, analyzing and representing behavioral information. The architect's responsibility is expanded to include conversion into spatial terms.

Examples are Caucus Partnership (1987*a*,*b*).

(5) The Aggregated Persons approach

The objective is to present spatial information and wants of occupants aggregated together on the basis of some common characteristics. Examples are social status or rank (associate professors), similarities in activities or jobs (secretaries, computer operators, carpenters), gender (women), color or race (Black, White, Asian) and similarity in space needs (assembly line workers).

Aggregates of persons are used as the units of analysis. These programs describe the space-related wants of a certain aggregate, for example secretaries or assembly line workers, and simply list the number of each even though these persons may be involved in different tasks in different places. The primary distinction with the Aggregated Spaces approach is that the former uses people characteristics while the latter used similarity in spatial characteristics as the basis for aggregation. Thus associate professors may have different spatial requirements while Type A offices may have a variety of occupants. This approach would most commonly be used by people not very familiar with the people the program is for.

Since the aggregates are devised by the programmer for convenience in programming, it favours programmers and is 'pro-programmer' in stance.

Several assumptions are made by this approach, some similar to the Individual Persons approach. The philosophical position is one of 'aggregatism', with the view that aggregate wants are primary and that the building ought to be designed for aggregate wants. Representation of aggregated wants and majority opinions are seen as important. Aggregate statistics such as mean, average or mode are seen as the best way to represent preferences of large numbers of occupants. Conversion of aggregated social and behavioral information into spatial terms can be done by the architect. It is assumed that individual and group wants will be discernible in the program. Aggregation of individual wants will not obfuscate or hamper the understanding of program information or the production of an appropriate design. Servicing occupant wants will be enabled by this approach. Aggregations devised by the programmer will be intelligible to the architect and the occupants. It will provide an in-depth up-close picture of the occupants' social world. The programmer will have the skills to collect and analyze aggregate data. Skills in sampling and quantitative analysis are required for data collection and analysis.

The following are the advantages of this approach. There is a focus on occupants and their requirements rather than on spaces. When the number of people is large it introduces the possibility of using statistical techniques to compute the mean, typical or unit. Aggregation reduces the programmer's work particularly in large projects.

The disadvantages of this approach are the following. Taking the position of aggregatism implies neglect of viable alternate positions. Designing buildings for aggregate wants means that the building will not be appropriate for those whose wants do not match the aggregated wants. Representation of aggregated wants and majority opinions may result in problems and lead to unsuitable buildings, as the discussion below will illustrate. Conversion of information to space-related specifications is difficult due to the lack of tools. Discernability of individual wants is not possible, unless they are identical, as individual wants are not depicted. It is unlikely that many individuals will be identical or match the typical even along a small number of criteria. Fidelity is a problem. When it is known that all persons are not identical the programmer may use mathematically computed approximations, such as typical, mean, average or mode. Aggregated statistics are approximations and may not be adequately representative when the population is diverse. Reliance on approximations implies some misrepresentations of individual wants. When deviations and variances from the selected descriptor are large errors built in become rather large and the program can become obfuscating and misleading, and this approach becomes unsatisfactory and counterproductive. In servicing occupant needs, aggregating people on the basis of common characteristics implies that design fit and appropriateness can only be aimed at some approximate aggregated version of the occupants, not at individuals. While this approach may prevent glaring errors in fit, it is unlikely to enable close and excellent fit, especially with diverse populations. With great diversity, majority opinions may not be appropriate to use in design. Further, these aggregation categories can direct attention away from social alliances and cleavages that may have much more to do with acceptance or rejection of the design by the occupants. Intelligibility of the information is also of concern. An aggregate is an abstraction by a programmer for convenience in programming and may not have any relation to either the way the occupants see themselves organized or would like to have the information organized. They are generally not important or meaningful, and at times unintelligible, to the occupants or to the architect—two major participants in the process. Aggregating can follow functional categories or structural aspects of an organization, but this tends not to be the case. Since the basis for the aggregating is external to the group or organized unit it often fails to capture the values of the group. Instead the emphasis is frequently on generalized biological, ergonomic or other 'needs'. Categories designed for convenience in programming get reified and treated as legitimate socially constructed ones, as if the world is organized the way the program information is presented, and designs done accordingly. Architects design based on program documents rather than treat them as semi-processed information to be further analyzed and reformulated for a more suitable design. Inasmuch as designers are intentionally or unintentionally misled this approach can be obfuscating and therefore counterproductive. Aggregations created can obfuscate information and be misleading. In-depth up-close knowledge is not provided by this approach on either individuals or groups. Group wants are not addressed. Verifying program information is difficult without repeating at least the analysis. Direct checking of design provisions is not possible as physical qualities required are not depicted. Programmers with social science training often have the required skills, but those with only architectural training often do not. Whereas the architect has increased responsibility, the programmer's responsibility, while substantial, is less than in the space-oriented approaches.

Examples are Stone and Luchetti (1985), Caucus Partnership (1987*a*), Welch + Epp (1987), Kantrowitz & Associates (1978, 1990), Steinfeld (1975) and MIT (1973).

(6) The Grouped Persons approach

The objective of this approach is to describe the nature, functioning and spatial wants of groups of occupants.

The unit of analysis is a group of persons. A group consists of a collection of people, possibly with different characteristics, having shared values, interests, understandings, way of life, who interact or operate together and may have a common purpose, aim or goal, such as producing a common product. These and space-related preferences of the group are described. Examples of Grouped Persons are a work group, divisional group, division, small organization or other common purpose group. Since groupings are derived from the way the occupants see themselves organized, the programmer has to learn about and understand the group, and explicate on its preferences, values and wants. The distinction with the Aggregate Persons approach is that the basis for grouping in the former is expected to be congruent with the divisions meaningful to occupants, while the latter is not necessarily congruent.

While study of inter-individual relationships such as communication patterns may give some ideas of groups, it may give a misleading picture as it may not display information about, and even ignore, social cleavages or boundaries. Aggregated individual values are not likely to provide information on group values—ideal or negotiated. It is for this reason that the Grouped Persons approach is expected to yield different information from the other approaches.

Since it uses the occupants' form of grouping, in stance this approach favors occupant groups and is 'pro-occupant'.

In this approach the assumptions are as follows. The philosophical position of 'societalism' taken gives the group and its values paramount importance. Representing group values and preferences is seen as primary, preferable to those of individuals. Conversion of group wants into spatial requirements can be accomplished by the architect. Servicing of occupant wants and design of a suitable building will be furthered by a Grouped Persons program. The information presented in the program will be intelligible to the architect. It is assumed that in-depth-up-close knowledge about the social world of the occupant will be provided by this approach. Skills for cultural study and analysis can be expected of programmers, i.e. programmers will have the skills for data collection and analysis, including the ability to understand, apprehend and depict group values, even if time available is relatively short. Also, the programmer will have the skills to objectively evaluate the functioning of groups and recommend appropriate interventions.

Among the advantages of this approach are the following. The focus is on occupants. It provides in-depth up-close knowledge and understanding of the groups as a social unit. Group developed values, mores and ways of relating to the physical environment are described. It compels the programmer to understand social issues, divisions and boundaries considered important by the groups and the effects of imposition of artificial and external ones. It introduces the architect to the world of occupant, albeit in a second-hand way. Designs based on understanding of these values would most likely be meaningful to the group. Servicing groups wants is enabled by this program. When survival and continued existence of the social unit is desired, it is important for the design to cater to and be supportive of the values of the social unit, Hence, for existing social units this approach can work well.

Counterproductiveness occurs when the assumptions are incorrect. The philosophical position of societalism implies subordination of individual preferences to those of the group. Individuals are considered primarily as members of the group. Representation of group values in the program means overlooking individual preferences. Conversion of social and behavioral information into spatial requirements is difficult. Servicing is dependent on proper conversion, and individual wants may get neglected. The program information will be intelligible to the architect when the presentation is jargon-free and simple; when not, the information may not be intelligible and may create difficulties. Many programs provide facts and descriptions about the people but few provide good information for culturally appropriate design. If this approach does not provide an in-depth up-close picture of the occupants' social world it will not be possible for the architect to design for group values. Most methods for cultural study are fairly time consuming and require intensive involvement and labor on the part of the researcher. In practice, most programmers are not trained in cultural analysis and often lack the time or the interest in in-depth naturalistic field study. When trained programmers and adequate time is unavailable a Grouped Persons approach may produce misleading results. Responsibilities of the programmer require evaluating the status of the group and recommending architectural interventions. This is rather complicated. Learning, understanding and depicting group values is difficult, as pointed out. To further evaluate the status of the group and pro-actively recommend architectural interventions involves big conceptual leaps which are difficult to accomplish, as techniques for actually accomplishing this are scarce.

Petronis *et al.* (1978) is a good example. Other examples which could be classified as Grouped Persons approach would be Brawn & Associates (1976) and Preiser (1985).

Concluding Discussion

Programming: monolithic?

Programming has been treated largely in an undifferentiated monolithic manner in the literature. Early writers (Studer, 1966*a*,*b*; Agostini, 1969; Peña, 1969; Richardson, 1969; Seaton, 1969; Peña *et al.*, 1987), enthused about having found a way to address the 'user-needs gap', did not make distinctions while later writers continued the undifferentiated treatment. In contrast, by categorizing programming based on its orientation, i.e. whether space or people oriented, and unit of analysis—individual, aggregate or group—I have shown that each approach has inherent assumptions and problems that can affect design. I have emphasized that it is advantageous to view programming as varied, differentiated, diverse and multipronged, with each approach having particular strengths and weaknesses, as this allows us to look for and accommodate some of the variety and distinctions in people—built environment relationships as pointed out by Stokols and Shumaker (1981).[8] I have further proposed that both spaces—independently, not as a derivative of people categories—and people can be categorized. When seen in conjunction, Stokols and Shumaker's (1981) characterization of settings adds another dimension and layer of complexity by alerting us to the possibility that while spaces can be so categorized, each space itself can accommodate individuals, aggregates or groups of occupants. It is important then to carefully consider the unit of analysis, the assumptive level differences and possible counterproductivities of each approach. These affect programming as I have described. All these factors make it imperative to view programming as varied.

An examination of approximately 25 programs indicates that the Aggregated Spaces, Aggregated Persons and Individual Spaces approaches are quite commonly followed but the Grouped Spaces, Individual Persons and Grouped Persons approaches are less common. Most neglected is the socio-cultural information of the Grouped Persons approach. It is possible in practice to mix the categories I have described and provide information from more than one approach and philosophical position in one program. Combination approaches, such as Individual Spaces and Aggregated Spaces approaches, are also used (RFD, 1985; SMP, 1986*a*,*b*; Zimmer & Associates, 1989*a*,*b*). This trend is likely to reduce the potential counterproductivities associated with singular approaches. It is conceivable, though not common, that a single program will provide information along all six categories—that would make an excellent program. But choice of any one involves assumptions and issues I described. While many programs these days combine approaches it is still necessary to consider the approaches I have described as many are still single approach programs and rarely are all approaches included. This paper is an attempt to make the selection of approaches more systematic.

Alternate views of programming

Here I have accepted the instrumentalist aims of programming, that of serving human purposes and wants related to people's 'immediate space' and not the larger global environment, in developing a typology of programming and describing the counterproductiveness of each. Programming can be viewed and studied from a variety of other angles not considered here. Programming can be a way of surfacing and resolving differences and conflicts regarding space. It can be a form of negotiation, and exchange, a *quid pro quo*, a way of arriving at negotiated agreements, building consensus and continuing group identity. Programming can be a form of occupant education about the social organization and about the building (Caucus Partnership, 1987*a*,*b*, 1988). It can be a political device or strategy (Silverstein & Jacobson, 1978), a symbolic act (Zeisel, 1973) or even a ritual. Alternatively, programming can be seen as a design activity (Balchen, 1973; Hack, 1976) wherein certain design decisions are arrived at and some programs incorporate design options and solutions (Welch + Epp Associates, 1987; CRSS, 1991). Conversely, design can be seen as a way of programming,[9] or at least the initial design sketches as working toward a program by making stakeholders more aware of design possibilities, and of incorporating programming information into design. Viewing design as a part of programming would mean that there may not be a final product to classify. Yet a philosophical position may still be involved in the way data is collected and analyzed. Programming can be seen as an integral part of the design process without a distinct product.[10] This happens when the architect and the programmer is one and the same. The more programming becomes an integral part of the design process the better. But architects are often not equipped to carry out programming research, and programmers often lack the skills of design. When this happens the program document becomes a way of communicating program requirements to the architect. To some the program document becomes a way of ensuring that the architect follows the requirements set out in the program, and to hold the architect (sometimes legally) accountable for lapses (Becker, 1959; Farbstein, 1978).[11] The program document then is an important instrument of communication and also of compliance (Farbstein, 1978). Programming can be seen as a co-operative endeavor between programmers and architects which is possible and desirable. It is possible to conceive programming as a process, even a continuing, cyclical or iterative one, in which information is prioritized such that initial effort is through broad strokes to resolve the major problems and through successive attempts at fine tuning to ensure fit of minor or less important 'details'.[12]

It is possible to conceive of other kinds of programming based on different criteria, such as the expected achievements of the program (Brill, 1971; Farbstein, 1984; Hershberger, 1985). Most of these could, by their approach or final form, be classified using my categories, which are based on the approach to information collection, analysis and presentation. I describe two of these briefly. One, commonly referred to as 'user-needs programming', strongly advocates a 'pro-occupant' stance. Yet this stance may be deeply affected by the form the program is presented in. For example, if the Aggregated Spaces approach is selected the nature of the information presented and the manner of presentation may prevent it from being strongly pro-occupant. The second, 'building program', conjures up an image of a list of sets of rooms, names and corresponding areas. Although it appears to be strongly 'pro-architect' in stance, in reality it is not, since it does not provide the architect with good information (Gutman, 1969).

Other aspects of programming, such as its process, methodology and uses, could be studied as well.

There are alternate assumptions regarding architecture, and people–environment relationships I could have taken. Alternate views of architecture, other than serving human wants, are also possible. Architecture can be seen as primarily an object of art, an engineering accomplishment, a way of technologically outfitting spaces, as having qualities that move the spirit (Stokols, 1988; Mazumdar, 1992) or as referents of the culture. In attempting to produce more suitable buildings primary attention has been paid to the 'functional' aspects: whether the building functions better. That is, a building is seen primarily, and almost solely, as an envelope and container of occupants, activities and objects. But architects see buildings serving multiple functions (Anderson, 1987) and consider other criteria, such as fit with the surrounding built and natural environment and the site, esthetic appeal, functioning in terms of materials of construction and energy consumption. A building's ecological functioning is becoming increasingly important. These criteria are just as valid as 'functionality'. Thus, values are associated with such judgments (Mazumdar, 1985).

Alternate positions regarding people–environment relationships involve questions such as is a person's environment only his/her room or should it also include common spaces and spaces occupied by others? Should only immediate proximate environments be addressed or should larger environments be considered? Should the environment be seen as distinct from people (as programming has) or should it be seen as one bound up integral socio-physical whole where one affects the other, with the spaces taking on the 'personality' of the people and the people taking on the mood of the space?[13]

I have not elaborated on the effects of costs and budgets. Decisions are made simply on the basis of overt costs. For example, cost is frequently presented as a reason for selecting the Aggregated Persons approach over the Individual Persons approach. The rationale is that a building designed to closely fit individual, particularly idiosyncratic, wants will become unsuitable when the population or their wants change (see note 2), whereas the Aggregated Persons approach, which does not seek a close fit with individual wants, is likely to remain somewhat suitable. Some approaches, such as the Individual Spaces and Individual Persons approaches, can be quite expensive, while the Grouped Persons approach can be time consuming and possibly expensive. These constitute calculation of overt costs. Yet covert costs of not following certain approaches, such as the Grouped Persons approach, are not even considered,[14] and the costs of not aligning a building closely with individual wants and of incorporating a certain amount of misfit are neither calculated nor considered. A building designed for individual fit may not become completely unsuitable if individual wants change, but only in servicing those idiosyncratic wants. An average fit can be considered a misfit whose degree of fit depends on the variance. For example, a majority of right-handers may lead the Aggregated Persons program to call for right-handed writing top chairs. For left handers, no matter how many, this chair would be a misfit, and inappropriateness built in from the outset. Yet the costs of doing so are not estimated. If cost has to be the determining factor both kinds of costs ought to be considered.

Assumptions underlying programming need to be elaborated and tested occasionally for their efficacy, which can become problematic as I have described. Once these assumptions are elaborated the task of developing appropriate ontological and epistemological stances and methodologies can be taken up. These have important implications for programming.

Approaches to programming and their implications

A question of concern to programmers is what kind of program will best serve the architect's need for information and encourage the architect to produce a more appropriate building? And how well is programming able to provide the necessary information and decisions for more appropriate designs?

Each approach involves philosophical positions and underlying assumptions, advantages, disadvantages and quirks that can deeply affect the effectiveness of an approach. Appropriateness of a particular approach, and therefore choice, depends on the context.

Choice of the Individual Persons approach implies taking the philosophical position of individualism with the belief that catering to individual values, appropriateness and satisfaction are most important. Information obtained by the Individual Persons approach is likely to display idiolectal variations (Thomas, 1979, p. 104) and individual idiosyncracies. It is not very useful in understanding a group's cultural system or shared cultural values. The Grouped Persons approach, on the other hand, places most emphasis on the group and on understanding and portraying group values, wants, shared understandings and meanings. Group values are seen as paramount, to the neglect of individual ones, and it is assumed that designs suitable to group values will be appropriate for individuals as they are members of the group. An Aggregated Persons approach assumes that aggregated statistics such as means or averages are the best ways to understand preferences of occupants. Emphasis is placed on majority opinions and preferences. Hence there is a choice of whether one believes the individual drives the group or the group drives the individual (Kuper, 1972; Michelson, 1987). A number of individuals, and their individual values and preferences, do not necessarily indicate group values and socially mediated and negotiated arrangements.

Conversely, appropriateness of a particular approach, and therefore choice, will vary based on characteristics of the approach and the project at hand. If individual appropriateness and satisfaction are most important, the Individual Persons approach will be the most suitable. However, if group existence and cohesion are important then the Grouped Persons approach may be the most appropriate. When occupants are not known, or are too large in number, the Aggregated Persons approach could be useful.

There are implications for environmental design research. One of the primary ways environmental design research has affected the field of architectural design is through programming. What affects programming can affect both architectural design and environmental design research. The appropriateness of a building depends on the kind of research carried out, the context and the programming approach selected. Providing good valid information in the program through environmental design research can increase the likelihood of the architectural design being more appropriate to the occupants. Perhaps environmental design research ought to be viewed from the perspective that there are six or nine vantage points or approaches.[15] Given the differences in approaches, methodology and relationships to the physical environment that have been pointed out (see also Stokols & Shumaker, 1981) it is important for environmental design researchers to not only make these distinctions and state which aspects they are focusing on but also to seek the interrelationships between them. For example, for a study dealing with individual opinions it will be important to know that the focus is on the individual, but also to know the idiolectal and idiosyncratic, what aspects are typical and generalizable to the population, and what aspects are cultural.

Quality of the information is of utmost concern, but reliance on methodological purity is not necessarily the best way. Methodologically, greater attention needs to be devoted to ways in which individual, aggregate and group relationships can be learned. Whose opinions or preferences are obtained is an important question (Farbstein, 1976). In an Individual Persons approach hopefully every occupant's preferences are obtained, as in a census. In some approaches, Aggregated and Grouped Persons in particular, data collection involves occupant sampling, biases in which can lead to erroneous results. In one example observed volunteers comprised the sample of occupants from whom information was obtained.

Some feel that designers ought to be held accountable for their designs. Accountability considerations raise questions of accountable to whom, how many, and along how many criteria must the building satisfy the occupants (Davis, 1975, p. 18)? Should 100% of the occupants have to be satisfied on 100% of the criteria? Given the various other constraints on design this cannot normally be expected. In some approaches, such as the Aggregated Spaces and Aggregated Persons approaches, which have approximations built in, this is not possible. So what should these numbers be? Are some criteria to get priority or are all criteria to be weighted equally and their scores simply totalled in figuring out the accountability factor? Further, are all occupants to be considered equal for satisfaction purposes or do some, such as those with social position or power, get preference or priority? For example, in one organization facility managers attempting to make facilities more 'democratic' programmed, reduced and equalized space allocations for several categories of 'technical' and 'administrative' level people, but managers, the more powerful group, were in some instances given more space and environmental elements. These issues, which require the appropriateness scale to be adjusted, are often not addressed in programming.

Several researchable questions remain. Should time be considered an important factor for architectural responsiveness and for validity of program information? We have as yet not seen programming as a process requiring explication. Neither, for programming, have we seen occupant groups as cultures.

Counterproductiveness of programming

Social and behavioral scientists have long criticized architects for carelessness in satisfying occupants, and for not having the skills and tools to acquire the necessary information. In effect, the claim is that architectural design has, in many instances, been counterproductive.

It was thought that architecture could be improved through the introduction of a 'programming' stage (done using primarily social science techniques) into the design process (Figs 1 and 2). Evaluation of the productiveness and efficacy of new interventions, such as programming, is necessary on occasion. Of the question 'How good is programming?', this was an attempt to answer only a part through the question 'Can programming become counterproductive?'.

A general assumption behind the introduction of programming into the design process was that it will provide information which will assist architects design better (by some set of considerations) buildings than was possible through the extant form of design. It was expected to provide a clear understanding of the occupants' requirements, faithfully and accurately provide needed information, and perhaps deal effectively with issues relating to policy, conflict and choice where necessary, and not obfuscate or confuse the designer. If programming is unable to do these effectively or is unable to offer some 'value added' then it can be considered counterproductive.

I have suggested that inherent counterproductivities in programming make it imperative to direct adequate attention to the selection of appropriate approaches and examination of their assumptions. The programming effort can become counter-productive when assumptions made turn out to be incorrect; when the information contained in the program is obfuscating, such as when space-oriented programs do not properly represent occupant wants; when choice of method and methodological problems prevent proper information from being captured, such as when group values are not represented but ought to be; and when approximations are built in or verification is prevented.

There are several potential disadvantages in each of the approaches which can make the programming efforts counterproductive. These are summed up below.

Each approach has a philosophical position which takes a different view of the social world, such as individualism's focus on individuals, to the neglect of information from other positions that may have aided the design of a more appropriate building. Even though there is an effort to provide information aimed at suitable design, the information provided is incomplete and misleading, and therefore counterproductive. In the Grouped Persons approach it is assumed that even though individual values are likely to differ to some degree from those of the social unit catering to group values will be satisfactory to the individuals in the group. This is likely to be acceptable when all individuals are aspiring to be good group members. For those who do not see themselves as part of the group and do not share the same values and aspirations this approach will not represent their individual values and the design is also likely not to be supportive. When the position is made explicit it is possible to verify if the program presents information to enable a design suitable to that position. For example, aggregationists would like to verify that a majority of occupants will be satisfactorily serviced and that small changes in population or wants will be accommodated, and societalists would like to know if cultural information is presented and understandable.

It appears from this analysis that programs do not convey occupant wants in a neutral way. I have emphasized that in each of the several approaches to programming the programmer could consciously or unconsciously be an advocate for a stakeholder (Farbstein, 1976; Hershberger, 1985). This may be surprising to those who see programming as neutral but not to those who see programming as playing an advocacy role and empowering powerless occupants so that their wants would be addressed (Zeisel, 1971). Space-oriented approaches focus on identification of spaces and their 'needs'. Since spaces do not have wants people do, all space-oriented programs tend to be written for the convenience of the professional involved. Since space is a primary element designers create and manipulate in their designs, and since the programmer attempts to collect, analyze and present information in a form, and speak in a language intelligible to architects, the space-oriented programs mostly tend to be 'pro-architect' in stance. While it is possible to base space-oriented programs on a deep understanding of the values and preferences of occupants regarding space, activities and aspirations, this rarely happens when the primary task is seen as identifying and listing spaces.[16] Thus it is quite possible to produce a program and yet not fulfil the original aim of programming—that of providing information which will assist in making designs more appropriate and responsive of occupants. In space-oriented programs there is a tendency to think of spaces primarily as serving specific, singular activities, rather than see architecture as a means for aiding human and social aspirations. Assumptions of mono-functionalism and designs logically based on catering to singular activities are more commonly held in the West (Potash, 1985), and so there is a slant towards the Western approach.

People-oriented programs, by keeping the focus on occupants, can be more humanistic and occupant oriented and offer possibilities of fulfilling the aims of programming. Yet in people-oriented programs, too, a programmer has to make several choices and assumptions which involve taking positions and selecting methods of information collection and presentation which can affect the effectiveness of the effort, as I have pointed out.

The most 'pro-architect' stance is displayed by the Grouped Spaces approach as it not only converts occupant wants into spatial terms but groups them together. The Individual Spaces approach is also 'pro-architect' instance. The Aggregated Spaces approach and the Aggregated Persons approach are both 'pro-programmer' in stance since aggregation is done primarily for the convenience of the programmer, although to a lesser extent convenience of the architect is considered in the Aggregated Spaces approach and that of the occupant in the Aggregated Persons approach. A 'pro-occupant' stance is taken by the Individual Persons and Grouped Persons approaches a priority is given to representing occupant wants.

In representing occupant wants each approach gives an incomplete picture by taking a somewhat different view of the social world and representing a different picture. People-oriented approaches provide social and behavioral information related to space but do not make any attempt to provide exact spatial requirements and therefore do not provide specific directives to the architect, Space-oriented approaches, on the other hand, represent little social and behavioral information. As a result the architect cannot get a sense of the social and behavioral requirements but only of architectural ones. Combination approaches attempt to provide glimpses of both.

The programming effort can become counterproductive when program information cannot be utilized in design. Conversion of social and behavioral data into architectural requirements enables utilization. Various methods are available for collecting and analyzing useful data (White, 1972; Michelson, 1975; Sanoff, 1977; Sommer & Sommer, 1980; Zeisel, 1981; Bechtel *et al.*, 1987). But the assumption that the architect or programmer will be able to convert social and behavioral information into spatial terms is confounded by the dearth of appropriate tools and techniques to convert activity, behavioral and equipment preference data into specific spaces and environmental requirements.[17] As a result it remains a creative act. If conversion cannot be made the program's effort is quite likely to become counterproductive. Activity and equipment mapping in which a programmer can gather information on current and future activities of all occupants, descriptions of spatial wants and list of equipment (fixed and movable) the occupants want (e.g. CRSS, 1991) are the few tools that can help in conversion by giving some idea of the sizes, shapes and areas of the spaces to be designed, if one takes an instrumental view. The ability to think spatially is required. Who should do the conversion of translation is an important question which can affect the efficacy of the program. With their experience in studying and design buildings, architects may be in a more advantageous position than others to carry out such conversion. The space-oriented approaches leave the conversion to the programmer while the people-oriented approaches entrust it to the architect, who may be more capable.

Fidelity of the information conveyed is of utmost concern. Yet not all approaches have the same level of fidelity.[18] The aggregate approaches incorporate approximations, as pointed out. And when there are several sets in the aggregate the aggregated values may give a misleading picture. Individual Persons approach can have high fidelity provided the data is properly collected. In the Grouped Spaces approach the information is organized and presented in a manner showing careful thought to the architectural organization architects are not likely to check if the program has high fidelity. Inaccurate information can make the program counter-productive.

In order for the architect to address occupant wants and design appropriate spaces, discernability of occupant wants is important, especially if different occupants occupy a space (Wells, 1967). Without discernability, the architect will not be able to get a picture of occupant wants, and can only attempt to meet requirements as stated in the program. In the space-oriented approaches requirements of the different spaces have to be discernible, but the aggregated approaches may not permit such discernability. If not discernible the architect may reasonably assume they are identical, and design identical spaces. Person-oriented programs, too, do not allow such discernability, could be obfuscating information and making the program counterproductive.

Enabling proper servicing of occupant wants should be the focus of programming efforts. As explained earlier, the form of the program makes servicing more difficult in some approaches, the person-oriented approaches for example. The Individual and Grouped Spaces approaches are easier to service directly. But in the Grouped Spaces approach the grouping is often the result of *a priori* theorizing imposed on the problem at hand than a development from an analysis of the context. Such importation of ideas and theories, and their imposition, often leads to neglect of the situational context and actually leads to less responsiveness rather than more. In the space-oriented approaches the architect can only service the requirements described in the program, as occupant wants are not directly depicted.

Intelligibility of the information for the architect varies from one approach to another. A good program should be comprehensible to the architect. The aggregated approaches can obfuscate information important in design, such as subsets with different wants which ought to be treated separately. Often aggregation is based on commonalities along a few criteria and other criteria are not mentioned. It is thus not clear whether the units in the aggregate ought to be treated as identical, or having differences and whether they need to be supplied with identical spaces. With a Grouped Persons program the architect may not be able to understand the social mores, cleavages and complexities of group dynamics if these are not presented simply and clearly.

In-depth up-close knowledge of the occupants' social world is necessary for the architect to produce a suitable design. This kind of knowledge is provided only by the Grouped Persons approach. The other approaches do not require the architect to get intimately involved in the lives of the occupants and feel as they do. Some approaches, such as the Aggregated Persons and the space-oriented approaches, actually encourage lack of such knowledge and distance. The Individual persons approach can provide in-depth knowledge about in-depth knowledge about individuals, but often does not.

Verification of program information is important to ensure that the program is still valid. Verification is not possible without reprogramming in the space-oriented approaches. Verification of individual wants is possible on site with the Individual Persons program by asking the necessary questions of the individuals. This is not possible with the Grouped Spaces approach, as individual wants are not even represented. Verifying societal values and aggregate wants is much more time consuming and difficult to accomplish quickly. Checking design provisions against programmatic requirements gives a sense of the extent to which the design satisfies them. Such checking is not possible in the person-oriented approaches since they often do not list spatial requirements.

It is expected that programmers and architects, as hired professionals, will have the necessary skills to do the job. Programmers with social science training usually have the skills for data collection and analysis for individual and aggregated approaches. Those with skills only in architecture often lack skills in social and behavioral data collection and analysis. Some programs require skills not only in programming and social or behavioraly sciences but also in architectural design, as in the Grouped Spaces program. Those lacking design skills also have difficulty with conversion to spatial requirements and developing meaningful groups of spaces. Most programmers do not have the skills for cultural analysis required in the Grouped Persons approach; they also often lack the inclination, interest or time to conduct cultural analysis. It is also assumed that a programmer will be skilled in collecting, analyzing and depicting social and behavioral data, and the architect will be skilled in design. But not all programmers are highly skilled in social analysis and not all architects in design. There are no specialised degrees of specific systematic educational requirements to become a programmer.

The responsibility of the programmer is very high in some approaches. For example, in the Grouped Spaces approach the programmer has to take the additional responsibility of developing categories or groups of spaces useful to the architect, for which some familiarity with the ways architects design is necessary. In practice, the Grouped Spaces approach is taken often by architect-programmers to get further along the process of design once program information has been obtained and analyzed. Also this approach requires the programmer to take complete responsibility for the conversion of occupants' wants into spaces and spatial characteristics, since verification of the conversion is difficult. When the programmer or the architect has responsibility for tasks for which he or she lacks the skills and training, the chances for errors increase greatly. Slips and errors effectively render the programming effort counterproductive. For example, an occupant may want an office with a smaller private portion to it. Mentioning an area of 100 sq. ft. or asking for an L-shaped office may still not accomplish a 'smaller private portion'.

Here I focused on the form of the product of programming and its assumptions. Clearly, further studies of the product and process of programming (how it gets done) (Evans, 1969; Hack, 1976; White, 1991) need to be carried out before we are able to determine in toto the usefulness of the addition of programming into the design process. This cannot be done through post-occupancy evaluations (Friedman *et al.*, 1978; Welch+Epp, 1988) alone, as that amounts to comparing 'apples and oranges', as often it is difficult to tell if design inappropriateness is due to the design, the program, or the occupants. While programming is seen as a process by many, building is seen primarily as a product and not as a process. Typically programs are implemented and forgotten. But programs can be used as the datum for subsequent building and program evaluations. New tools, combining individual and social system perspectives, need to be devised to test the efficacy of programming. Making distinctions between the quality of the program and how well it represents the different viewpoints, individual, aggregate and group will help. Meanwhile, it is hoped that this paper will spur some thought and perhaps even some debate on how programs are being counterproductive—since, as I have described, the potential clearly exists.[19]

Acknowledgements

Sincere thanks to Daniel Stokols, Gary Evans, David Canter and the 'blind' reviewers for their help, encouragement, and comments in the rewriting of this paper and to the people and organizations who shared their programs with me, especially NBBJ (Deepak Dandekar) and MIT for permission to reproduce materials.

Notes

(1) The views expressed here are those of the author, who was a member of the team and the organization, and not necessarily those of the latter. For confidentiality reasons this program is referred to simply as 'Program X'.

(2) Although information was collected from the point of view of individual persons and groups, the program information was presented in the manner of an Aggregated Spaces program. This was done partly as a response to organizational policy where not too much time could be invested in doing an Individual Persons program. The organization also felt that since the occupants were likely to change, it probably was not appropriate to cater the design too closely to those present at the time of the program.

(3) Request for programs were sent to programmers in U.S.A., Germany, U.K., Australia, New Zealand. At the time of writing about 25 responses were received, all from the U.S.A. Some of the programs too multiple approaches and are featured as examples in more than one approach. While it is possible to analyze and criticize the work of architects as these are in the public domain unfortunately we cannot do the same with programmers as their work is not in the public domain and is often proprietary.

(4) Space oriented programs could involve data gathering and analysis of occupants and their wants. But, the final form of the program has certain implications as I shall point out, see also note 2.

(5) Programming for a large facility with numerous spaces can be quite tedious. Examples are the program for the 80,000 sq. ft. Social Science II building at U.C.I. (NBBJ, 1990), and M.I.T. A.E.C.S. (1982). Scale is an important issue. The aggregated approaches are more suitable for larger projects as I have mentioned. Nevertheless there are problems with them, and when we select an approach we should do so conscious of the limitation of that approach, or try to devise new approaches to mitigate their negative effects.

(6) The material on 'unanticipated uses of programming' forms the subject of another paper.

(7) Architects have often been blamed by social scientists for providing similar and identical spaces and not allowing differentiation.

(8) While developed independently, my categories closely match those offered by Stokols and Shumaker (1981).

(9) I am thankful to the 'blind' reviewer who suggested this point of view.

(10) Some, such as Farbstein (1984), have seen programming and design as distinct yet circularly iterative activities, not as distinct linearly phased activities with one beginning where another ended (Moleski, 1974).

(11) According to Professor E. T. White (telephone conversation 4 June 1991) programming in America is seen more as a legal and managerial tool, while briefing in England is a way of getting to know the client and occupants. Min Kantrowitz (telephone conversation 3 April 1991) also mentioned that the program has sometimes been used as a legal contract requiring signatures from all parties.

(12) I am grateful to Professor Daniel Stokols for debating this aspect with me and pointing out that it is possible to conceive of programming as constituting progressively finer stages where individual idiosyncratic

features are progressively incorporated. Whether 'details' are achieved through such refinement or whether they 'determine' the success of a program or building is open to debate. Post-occupancy evaluations often tend to bring out what many architects consider to be 'minor details' and therefore unimportant. In any event, I like to make a distinction between the process, along with its concomitant temporal aspects, and the product of programming. This paper deails much more with the product of programming rather than its process. 'Process of programming' can be the subject of another paper. But even if such individual idiosyncratic wants are brought in at a later stage, they should be part of the program document given to the architect.

(13) I am thankful to the 'blind' reviewer who suggested this point of view.

(14) 'The Costs of Not Knowing' was the wonderful theme of a conference (see Wineman *et al.*, 1986).

(15) The idea that environmental design research perhaps affords opportunities of a number of approaches or vantage points needs to be explicated in another essay. Meanwhile see Stokols and Shumaker (1981) in conjunction with this piece.

(16) The task becomes one of filling out blanks in standardized forms and it becomes difficult to retain the focus on occupants and appropriate design, as I found out.

(17) The paucity of tools and methods can be seen as 'constraints on programming'. Exposition of the subject, is not possible here due to space limitations, but can from the subjects of another paper.

(18) One could argue that all programs ought to represent occupant wants with a high degree of fidelity. Yet, while some program display information which allows for verification and checks of fidelity, not all do. Presented information in an architecturally 'digested' way increases the likelihood of the architect taking the information and analysis for granted and designing. In general, space oriented approaches put the focus on meeting the spatial requirements called for whether they would satisfy the occupants or not. Also, in general, people oriented programs put the focus on meeting occupant wants rather than spatial requirements alone. It is in this sense, I argue, that the programmer's responsibility is greater. Also, it is not unreasonable to expect that architecture will cater to occupants' values, wants, desires and aspirations, but experience shows that this does not always happen.

(19) Only a few authors have written about their experience with programming (Presier, 1976, 1985; Farbstein, 1984). I hope this paper will encourage others to publish their experiences and problems with programming so that more thought can be devoted to these issues.

References

Agostini, E. J. (1969). The value of facilities programming to the client. *Building Research*, **6**(2), 29–32.

Anderson, S, (1987). The fiction of function, *Assemblage*, **2**, 19–31.

Balchen, B. (1973). Where programming is the design. *AIA Journal*, **59**(4), 39–48.

Bechtel, R. B., Marians, R. W. & Michelson, W. (Eds) (1987). *Methods in environmental and behavioral research.* New York, NY: Van Nostrand Reinhold.

Becker, N. (1959). Space analysis in architecture. *AIA Journal*, **31**(4), 40–47.

Brawn, G. & Associates (1976). *Functional Program: Lane County Adult Corrections.* Vancouver, BC: Brawn & Associates.

Brill, M. E. (1971). Evaluating buildings on a performance basis. In C. Curnett *et al.*, Eds., *Architecture for Human Behavior*, Philadelphia, PA: A.I.A. Chapter.

Brill, M. E. (1972). Techniques for developing performance specifications for buildings. In B. E. Foster, Ed., *Performance Concept in Buildings, Vol. 1: Invited Papers.* Proceedings of a Symposium Sponsored by RILEM, ASTM & CIB. Philadelphia, PA: National Bureau of Standards Special Publications, **361**, pp. 171–180.

Brill, M. E. (1984). *Using Office Design to Increase Productivity.* Buffalo, NY: Workplace Design & Productivity.

Caucus Partnership (1987*a*). *Data Collection Report on International Bank Library Move.* Buffalo, NY: The Caucus Partnership.

Caucus Partnership (1987*b*). *Space Program, International Bank.* Buffalo, NY: The Caucus Partnership.

Caucus Partnership (1988). *Revised Space Program, Computer Services, International Bank,.* Buffalo, NY: The Caucus Partnership.

CRSS (1989). *Land Phasing Plan Program, Texas.* Irvine, CA: CRSS.

CRSS (1990). *Telephone Company, Interior Design Program.* Irvine, CA, CRSS.

CRSS (1991). *Fine Arts Addition and Gallery/Museum.* Irvine, CA, CRSS.

Davis, G. (1969). The independent building program consultant. *Building Research*, **6**(2), 16–21.

Davis, T. A. (1975). Formulating habitability criteria from research information. In W. Preiser, Ed., *Programming for Habitability.* Urbana Champaign, IL: University of Illinois, Department of Architecture, pp. 18–21.

De Chiara J. & Callender, J. H. (Eds) (1980). *Time-saver Standards for Building Types.* New York, NY: McGraw-Hill.

Donaudy, T. (1989). *Project Program: Village of Patchogue Recreation Center.* Tallahassee, FL: Florida A & M University, School of Architecture Project.

Evans, B. (1969). Architectural programming practices. *Building Research*, **6**(2), 12–15.

Farbstein, J. (1976). Assumptions in environmental programming. In P. Suedfeld & J. A. Russell, Eds., *The Behavioral Basis of Design.* Stroudsburg, PA: Dowden Hutchinson & Ross, pp. 21–26.

Farbstein, J. (1978). A juvenile services center program. In W. F. E. Preiser, Ed., *Facility Programming.* Stroudsburg, PA: Dowden, Hutchinson & Ross.

Farbstein, J. (1984). Using the program, application to design, occupancy and evaluations. In D. Duerk & D. Campbell, Eds., *EDRA 15: The Challenge of Diversity.* San Luis Obispo, CA: EDRA, pp. 240 ff.

Farbstein, Jay & Associates with Patrick Sullivan Associates (1991). *Architectural Program and Concept Design for Model Residential Facility.* San Jose, CA: Jay Farbstein & Associates.

Friedman, A., Zube, E. & Zimring, C. (Eds) (1978). *Environmental Design Evaluation.* New York, NY: Plenum Press.

Gutman, R. (1969). The sociological implication of programming practices. *Building Research*, **6**(2), 26–27.

Hack, G. (1976). *Environmental programming: creating responsive settings.* Cambridge, MA, MIT PhD Dissertation.

Hershberger, R. G. (1985). Values: a theoretical foundation for architectural programming. In W. F. E., Preiser, Ed., *Programming the Built Environment.* New York, NY: Van Nostrand Reinhold.

Kantrowitz, Min & Associates (1978)..*A Program for Housing for Low Income Elderly in North Barelas.* Albuquerque, NM: City of Albuquerque & MKA.

Kantrowitz, Min & Associates (1990). *Therapeutic Recreation Center Study, Phase I.* Albuquerque, NM: Albuquerque Parks and Recreation Department & MKA.

Kantrowitz, Min & Associates (1991). *Therapeutic Recreation Center Study, Phase II.* Albuquerque, NM: Albuquerque Parks and Recreation Department & MKA.

Kuper, H. (1972). The language of sites in the politics of space. *American Anthropologist*, **72**, 411–427.

Lang, J., Burnette, C., Moleski, W. & Vachon, D. (Eds) (1974). *Designing for Human Behaviour: Architecture and the Behavioral Sciences,* Stroudsburg, PA: Dowden, Hutchinson & Ross.

Mazumdar, S. (1985). Architecture—an artifact of culture? *Reflections*, **3**(1), 36–49.

Mazumdar, S. (1992). The romance of architecture—an optimistic view. In J. Bassin, Ed., *An Optimistic View of Architecture.* New York, NY: NIAE.

MBT Associates (1988). *Updated Detailed Project Program: PS1 Renovation.* San Francisco, CA: MBT Associates.

Michelson, W. (Ed.) (1975). *Behavioural Research Methods in Environmental Design.* Stroudsberg, PA: Dowden, Hutchinson & Ross.

Michelson, W. (1987). Groups, aggregates and the environment. In E. Zube & G. T. Moore, Eds., *Advances in Environment and Behavior Research.* New York, NY: Plenum, pp. 161–186.

MIT Planning Office (1973). *Program: Undergraduate Housing, West Campus.* Cambridge, MA: MIT Planning Office.

MIT AECS (1982). *Program: Undergraduate Teaching Laboratories, Department of Chemistry.* Cambridge, MA: MIT/AECS.

Moleski, W. (1974). Behavioral analysis and environmental programming for offices. In J. Lang, C. Burnette, W. Moleski & D. Vachon, Eds., *Designing for Human Behavior: Architecture and the Behavioral Sciences.* Stroudsberg, PA: Dowden Hutchinson & Ross.

NBBJ Group (1990). *University of California, Irvine, Social Sciences Unit-2: Detailed Project Program.* San Francisco, CA: NBBJ

Palmer, M. (1981). *The Architect's Guide to Facility Programming.* Washington, DC: AIA.

Peña, W. (1969). Organization for programming. *Building Research*, **6**(2), 8–11.

Peña, W., Parshall, S. & Kelly, K. (1987). *Problem Seeking: An Architectural Programming Primer.* Washington, DC: AIA Press.

Petronis, J. P., Kline, L. S. & Pugh, R. (1978). Programming across cultures: a cultural food preparation center for Cochiti Indian pueblo. In W. F. E. Presier, Ed., *Facility Programming.* Stroudsburg, PA: Dowden, Hutchinson & Ross.

Potash, B. (1985). Kitchens are for sleeping: anthropology and the training of architects. Williamsburg, VA: College of William & Mary, Anthropology, Studies in Third World Societies, Publication 31, pp. 143–161.

Preiser, W. F. (Ed.) (1975). *Programming for Habitability: Symposium Proceedings.* Urbana-Champaign, IL: University of llinois, Department of Architecture, Monograph.

Presier, W. F. *et al.* (1976). User-oriented programming of facilities: workshop summary. In P. Suedfeld, J. A. Russel, L. M. Ward, F. Szigeti & G. Davis, Eds., *The Behavioral Basis of Design: Book 2.* Stroudsburg, PA: Dowden, Hutchinson & Ross.

Preiser, W. F. (Ed.) (1978). *Facility Programming: Methods and Appplications.* Stroudsberg, PA: Dowden, Hutchinson & Ross.

Preiser, W. F. (Ed.) (1985). *Programming the Built Environment.* New York, NY: Van Nostrand Reinhold.

Proshansky, H. M., Ittelson, W. H. & Rivlin, L. D. (Eds) (1970). *Environmental Psychology: Man and His Physical Setting.* New York, NY: Holt Reinhart & Winston.

Research Facilities Design (RFD) (1985). *Detailed Project Program: Physical Sciences Unit 1.* San Diego, CA: RFD.

Richardson, S. (1969). The value of a program to the architect. *Building Research*, **6**(2), 40–42.

Ryan, T. J. (1989). *A Community High School Facility for Northern Leon County.* Tallahassee, FL: Florida & A & M University, Department of Architecture, B. Arch. Thesis.

Sanoff, H. (1977). *Methods of Architectural Programming.* Stroudsburg, PA: Dowden, Hutchinson & Ross.

Seaton, R. (1969). Research for building programming. *Building Research*, **6**(2), 36–39.

Silverstein, M. & Jacobson, M. (1978). Restructuring the hidden program: toward an architecture of social change. In W. F. Presier, Ed., *Facility Programming: Methods and Applications.* Stroudsburg, PA: Dowden, Hutchinson & Ross.

Sommer, R. (1969). *Personal Space: The Behavioral Basis of Design.* Englewood Cliffs, NJ: Prentice Hall.

Sommer, R. & Sommer, B. (1980). *A Practical Guide to Behavioral Research Tools and Techniques.* New York, NY: Oxford University Press.

Steinfield, E. (1975). *Barrier Free Design for the Elderly and the Disabled:* III: Programmed Workbook. Syracuse, NY: Syracuse University.

Stirling, J., Wilford, M. & Associates & IBI Group/Paul Zajfen (1988). *Detailed Project Program: Science Library, UCI.* Irvine, CA: UCI. 321 pp.

Stokols, D. (1988). Instrumental and spiritual views of people–environmental relations. Paper presented at Symposium on The Role of Psychological Science in Promoting Environmental Quality, Annual Conference of the Eastern Psychological Association, Buffalo, NY, 22 April.

Stokols, D. & Shumaker, S. A. (1981). People in places: a transactional view of settings. In J. H. Harvey, Ed., *Cognition, Social Behaviour, and the Environment.* Hillsdale, NJ: L. Erlbaum, pp. 441–488.

Stone, Marraccini & Patterson (SMP) (1986*a*). *Detailed Project Program: Steinhaus Hall (School of Biological Sciences) Renovation.* San Francisco, CA: SMP.

Stone, Marraccini & Patterson (SMP) (1986*b*). *Detailed Project Program: Biological Science Unit 2.* San Francisco, CA: SMP.

Stone, P. J. & Luchetti, R. (1985). Your office is where you are. *Harvard Business Review*, **63**(21), 102–117.

Studer, R, (1966*a*). On environmental programming. *Arena, The Architectural Association Journal*, **81**(902), 290–296.

Studer, R. (1966*b*). Architectural programming, environmental design and human behavior. *Journal of Social Issues*, **4**, 127–136.

Thomas, D. H. (1979). *Archaeology.* New York, NY: Holt Rinehart & Winston.

University of California, Irvine, Office of Physical Planning (1987). *Campus Child Care Center: Detailed Project Program.* Irvine, CA: UCI/OPP, Nov.

Wade, J. W. (1979). Architectural programming. In J. C. Snyder & A. J. Catanese, Eds., *Introduction to Architecture.* New York, NY: McGraw-Hill, pp. 191–207.

Welch+Epp Associates (1986). *Architectural Program for the South Boston Police Station* for City of Boston Public Facilities Department. Boston, MA: Welch+Epp Associates.

Welch+Epp Associates with Prellwitz/Chilinski Architects (1987). *Architectural Program for Yarmouth Teen Mothers Residence* for Executive Office of Communities and Development and Yarmouth Housing Authority. Boston, MA: Welch+Epp Associates.

Welch+Epp Associates (1988). *Post Occupancy Evaluation: Quincy Mental Health Center.* Boston, MA. Welch+Epp Associates.

Wells, B. (1967). Individual differences in environmental response. *Arena: The Architectural Association Journal*, **82**, 167–171.

White, E. T. (1972). *Introduction to Architectural Programming.* Tucson, AZ: Architectural Media, Ltd.

White, E. T. (1991). *Design Briefing in England.* Tucson, AZ: Architectural Media, Ltd.

Wineman, J., Barnes, R. & Zimring, C. (Eds) (1986). *The Costs of Not Knowing: Proceedings of EDRA 17.* Washington, DC: Environmental Design Research Association (EDRA).

Zeisel, J. (1971). Fundamental values in planning with then non-paying client. In C. Burnette *et al.*, Eds., *Architecture for Human Behavior.* Philadelphia, PA: AIA, pp. 23–30.

Zeisel, J. (1973). Symbolic meaning of space and the physical dimension of social relations; a case study. In J. Walton & D. Carns, Eds., *Cities in Change.* Boston, MA: Allyn & Bacon.

Zeisel, J. (1981). *Inquiry by Design.* Monterey, CA: Brooks/Cole Publishing Company.

Zimmer, Gunsul, Frasca Partnership & Earl Walls Associates (1989*a*). *Detailed Project Program. Engineering Unit 2: Vol. 1: Final Document.* Irvine, CA: UCI.

Zimmer, Gunsul, Frasca Partnership & Earl Walls Associates (1989*b*), *Detailed Project Program: Engineering Unit 2: Vol. 2: Room Data Sheets and Program Drawings, Final Document.* Irvine, CA: UCI.

Manuscript received 3 *August* 1990
Revised manuscript received 30 *August* 1991

Appendix 1 Room Data Sheet

Source: NBBJ (1990) (reprinted with permission)

The NBBJ Group Room Data Sheet

Project	Department
Job Number	Room Name
Date	Room Numbers
Prepared by	

Space Function

Key Relationships
Key Operational Assumptions

Sequence of Procedures

Hours of Use: Average Occupancy		Length of Procedures	
Hours of Use: Maximum Occupancy		Length of Clean-up	

Equipment

Description Make/Model	Size	Fixed/ mobile	Power req'mt	Emerg. power	Ground req'mt	Vent req'mt	Light req'mt	Stab volt	Other
1									
2									
3									
4									
5									
6									

Attach Manufacturer's specifications if possible
Use additional sheets to expand list

Furnishings/ Casework

Desks	Size	Quantity	Counter (length)	Size	Quantity	Other	Size	Quantity
Chairs			Files					
Tables			Carrels					
Multile seating			Chalkboard					
Typewriter			Tackboard					
CRT			Screen					
Shelving (length)			Other					

Finishes and Critical Dimension

Floor Material	Comments	Wall Material	Comments	Ceiling Material	Comments
Finish		Finish		Finish	
Access		Wainscot		Access	
Base		Handrail		Height	
Dimensions		Bumper		☐ IV Track	

The NBBJ Group

Openings	Door	Comments	Relites	Comments	Windows
	Size		Size		☐ Required
	Finish		Glazing		☐ Not required
	Lock		☐ Operable		☐ Not desired
	Rating		☐ Fixed		☐/☐ Fixed/Operable
	☐ Manual		☐ Draperies		
	☐ Auto				☐ Light control
	☐ Closer				☐ Draperies
	☐ X-ray prot.				☐ Blinds

Electrical	Power	Comments	Lighting	Comments	Communication
	☐ 120 V ☐ 208 V		☐ Fluorescent		☐ Telephone
	☐ 1ϕ ☐ 3ϕ		☐ Incandescent		☐Intercom
	☐ Emergency		☐ Emergency		☐ Paging
	☐ Strip recep.		☐ Task		☐ Music
	☐ Floor recep.		Control		☐ Data process.
	☐ Ceiling recep.		Level (fc)		☐ CCTV camera
	☐ Grounding		Other		☐ CCTV monitor
	☐ Expl. proof		Exam		☐ Clock
	☐ Waterproof		Fixed		☐ Emergency power
	☐ Uninteruptible Power Supply		Moveable		☐ Telemetry
	☐ Clean (Filtered) Power				☐ Physiological Monitor
	☐ Dedicated Circuit				☐ Nurse Call
	Other				☐ Dictation

Mechanical	Ventilation	Comments	Air Conditioning	Comments	Pressure
	☐ Mechanical		Min. Temperature		☐ Positive
	☐ Natural		Max. Temperature		☐ Negative
	☐ Exhaust		Min. Relative Humidity		☐ Neutral
			Max. Relative Humidity		Control
			Acoustical control		☐ Individual
			Filtering		☐ Zone

Plumbing	Piped Service	Quantity	Fixtures	Comments	Fixtures		
	☐ Oxygen		☐ Sink		☐ WEC Law		
	☐ Nitrogen		Type		☐ Floor Drain		
	☐ Air		Material		☐ EWC		
	☐ Vacuum		Sink Control				
	☐ Nitrous Oxide		☐ Wrist		Accessories	Size	Quantity
	☐ Carbon dioxide		☐ Foot				
	Other		☐ Electronic		☐ Soap Dish		
	☐ Hot Cold Water				☐ Shelf		
	☐ Vacuum Cleaner		☐ WC		☐ Mirror		
	☐ Steam		☐ BP Washer		☐ Towel Bar		
	☐ Natural Gas		☐ Urinal		☐ Grab Bar		
	☐ Distilled Water		☐ Lavatory		☐ Hook		
	☐ Deionized Water		☐ Tub		☐ Waste Recep.		
			☐ Size		☐ Toilet Paper		
	Location		☐ Showerstall		☐ San. Napkins		
	☐ Wall		☐ Service Sink		☐ Seat Cover Disp.		
	☐ Ceiling		☐ Flushing Rim		☐ Bench		
	☐ Column		☐ Acid Waste		☐ Shower Curtain		
			☐ Hose		☐ Other		
			☐ Spray				
			☐ Dist. Water				
			☐ Plaster Trap				

Appendix 2 Room Data Sheet

(Source: MIT (1982) (reprinted with permission))

GROUP:
NAME OF SPACE:
NUMBER REQUIRED:
TOTAL AREA:
FACILITY NUMBER:

AREA PER SPACE

PURPOSE

OCCUPANT

LOCATION

FLOORS

WALLS

CEILINGS

CEILING HEIGHT

DOORS

WINDOWS

ACOUSTICS

HVAC

FACILITY NUMBER:

ELECTRICAL

PLUMBING

PIPED SERVICES

SAFETY REQUIREMENTS

COMMUNICATIONS

SPECIAL REQUIREMENTS

FURNISHINGS/EQUIPMENT

FIXED

MOVABLE