Computers in architectural education

by JENS G. POHL
California Polytechnic State University
San Luis Obispo, California

THE “ART” OF BUILDING DESIGN

It is perhaps difficult to understand that even today the design of the built environment is best described as an “art” rather than a “science.” The designer or architect is an artist who relies heavily on creativity and intuition to solve technological problems.

The individual components, such as structure, cladding, mechanical and electrical services, furnishings, fittings and finishes, which are available for the design and construction of buildings have all been developed and tested in a scientific manner and may therefore be classified under the heading of technology. However, the occupants of buildings are people whose behavior, needs and expectations are governed by innumerable social and cultural interactions, many of which are not easily defined. The establishment and analysis of the interrelationships between the large number of technological components and an even larger number of behavioral patterns is the formidable task of the architect. Faced by the apparently insurmountable problem of analyzing (and perhaps optimizing) an almost incomprehensible number of largely undefined variables, it is no wonder that the architect has in the past and continues today to use intuition (or shall we call it creativity) to overcome the shortcomings of existing scientific methods. The fact is, that even today the range of operation research techniques available to the designer is entirely inadequate for solving, let alone optimizing, a complex building design system.

Perhaps the design systems could be simplified? In the past, this appealing approach has led to some horrendous failures. Simplification may be achieved by reducing the number of variables in the problem system. However, this cannot be accomplished successfully without full knowledge of the relative importance of not only the variables themselves, but also the interactions between the variables. Unfortunately, the state-of-the-art in the social and behavioral sciences cannot provide the architect with a comprehensive set of defined variables, ranked or unranked.

The alternative approach, which consists of subdividing the design system into a large number of often complex subsystems (e.g., structure, environmental control, user functions, etc.) to be solved more or less in isolation, has been more successful. For example, this piecemeal approach would allow the natural and artificial lighting design of a building to be completed independently of the design of the structural support system. There remains, however, the problem of determining the nature of the interface between each of the subsystem solutions.

It is this second approach which is in common use in architectural practice today. Scientifically based procedures have been developed for solving many of the subsystem problems which make up the building design system. All of these procedures are readily computerized into interactive and non-interactive software packages. Unfortunately, the integration of these separate solutions into one optimized design solution cannot be computerized prior to the development of considerably more sophisticated interface procedures. Existing operation research techniques appear to be entirely inadequate for the optimization of large building design systems.

COMPUTERS IN ARCHITECTURAL PRACTICE TODAY

The adoption of even the most elementary computer applications in architectural practice has been slow. Faced with the overwhelming complexity of the design system, the architect has tended to assume the role of a coordinator responsible for the integration of the various subsystem solutions mostly prepared by specialist consultants, such as structural engineers, electrical and mechanical engineers, building economists and construction engineers. This is particularly true for the design of large commercial buildings, where the final integrated design is very much a compromise solution based on fact, presumption and a considerable amount of intuition. The inability to rely on established numerical methods for the final and most important stage of the design process has made most architects inordinately suspicious of all mathematically based design procedures. It is therefore only natural that the same suspicion should have carried over to the “computer,” which has in the past been represented to the architect as a high speed calculator rather than a powerful data-processing and simulation tool.

To make matters worse, a number of early computer users in the contracting field experienced critical setbacks in their efforts to implement computer-based operations. At times, this has led to the complete rejection of computers as timesaving tools by individual contracting com-
TABLE I—Typical computer applications in the building industry

<table>
<thead>
<tr>
<th>TYPE OF PROFESSIONAL SERVICE</th>
<th>AVAILABILITY OF COMPUTER PROGRAMS</th>
<th>MOST APPROPRIATE COMPUTER FACILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regional survey:</td>
<td>Abundance of programs dealing with the analysis of census and survey information.</td>
<td>Batch-processing</td>
</tr>
<tr>
<td>Land use planning:</td>
<td>Inventory programs capable of classifying land characteristics, mostly using overlay techniques.</td>
<td>Batch-processing with access to plotter.</td>
</tr>
<tr>
<td>Transportation planning:</td>
<td>Large scale mathematical simulation models which enable user to experience visual movement through designed space, also overlay and support facilities analysis and design programs.</td>
<td>Time-sharing with access to graphic display units for simulation programs. Batch-processing with access to plotter for inventory and overlay programs.</td>
</tr>
<tr>
<td>Infra-structure planning:</td>
<td>Mostly design programs dealing with district heating, cooling, water supply and waste disposal. Programs dealing with the integration of services are as yet scarce.</td>
<td>Time-sharing with access to graphic display unit desirable (Batch-processing with access to plotter for some applications).</td>
</tr>
<tr>
<td>Building development feasibility study:</td>
<td>Programs normally based on local building regulations and specific building types, such as residential and commercial.</td>
<td>Time-sharing</td>
</tr>
<tr>
<td>Project time scheduling:</td>
<td>Numerous programs available to analyze and update PERT networks.</td>
<td>Time-sharing (Batch-processing for large projects).</td>
</tr>
<tr>
<td>Site analysis:</td>
<td>Contour analysis and cut and fill calculations</td>
<td>Batch-processing with access to plotter</td>
</tr>
<tr>
<td>Design brief development:</td>
<td>Report generating programs with the ability to establish correlations, contradictions, and omissions.</td>
<td>Batch-processing.</td>
</tr>
<tr>
<td>Establishment of activity and space relationships:</td>
<td>Programs capable of forming activity and space interaction matrices</td>
<td>Batch-processing or time-sharing if input and output not large</td>
</tr>
<tr>
<td>Comparative space groupings:</td>
<td>Two and three-dimensional layout optimization programs.</td>
<td>Time-sharing (Batch-processing if user interaction not required).</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Comparative heating-cooling, ventilation, lighting, acoustic and structural analyses:</td>
<td>Increased availability of programs which establish environmental and structural design guidelines.</td>
<td>Time-sharing.</td>
</tr>
<tr>
<td>Preliminary cost analysis:</td>
<td>Programs in which the accuracy achieved is related to the degree of detail of the input.</td>
<td>Time-sharing.</td>
</tr>
<tr>
<td>Preparation of design drawings:</td>
<td>Small number of recently developed programs allow graphic display unit to be used as sketch-pad. Programs are often linked to data bases incorporating fittings and standard drafting symbols. Abundance of perspective programs with and without ability to delete hidden lines.</td>
<td>Time-sharing with access to graphic display unit for design process. Batch-processing with access to plotter for presentation drawings.</td>
</tr>
<tr>
<td>Structural design:</td>
<td>Abundance of comprehensive and specific programs mostly directed toward structural analysis rather than design.</td>
<td>(Batch-processing for very large programs). Time-sharing.</td>
</tr>
<tr>
<td>Environmental design (acoustics, lighting, heating-cooling and ventilation):</td>
<td>Mostly batch-processing programs, although the number of interactive time-sharing programs is rapidly increasing.</td>
<td>Time-sharing.</td>
</tr>
<tr>
<td>Design and integration of mechanical and electrical services:</td>
<td>Many design programs, but very few deal with the relationships between two or more support services.</td>
<td>Time-sharing (Batch-processing with access to plotter for layout drawings).</td>
</tr>
<tr>
<td>Detailed cost estimate based on design drawings:</td>
<td>Programs mostly owned by large design offices, construction firms and cost consultants.</td>
<td>Time-sharing.</td>
</tr>
<tr>
<td>Preparation of specifications:</td>
<td>Complete specification service provided by specialist firms.</td>
<td>Batch-processing with access to high speed printer.</td>
</tr>
<tr>
<td>Preparation of working drawings:</td>
<td>Few programs available to date, due to the requirement of comprehensive data bases incorporating materials, fittings, doors, windows and standard construction details.</td>
<td>Time-sharing (preferably with access to digitizer to transfer dimensions and quantities directly from drawings to computer).</td>
</tr>
<tr>
<td>Development of bidding strategies:</td>
<td>Small number of interactive construction management games, which enable the user to develop bidding and scheduling strategies on the basis of simulated conditions.</td>
<td>Time-sharing.</td>
</tr>
<tr>
<td>Project scheduling:</td>
<td>Large number of CPM and precedence diagram programs available. These programs are capable of generating calendar time schedules and allocating manpower and equipment resources.</td>
<td>Batch-processing (time-sharing may be economical for small construction projects).</td>
</tr>
<tr>
<td>Equipment records and costs:</td>
<td>Resource leveling and allocation programs require large computers and are normally based on CPM network analysis.</td>
<td>Batch-processing.</td>
</tr>
<tr>
<td>Accounting, payroll and record keeping:</td>
<td>Many programs available, which are based on time card input and prepare payroll checks, financial statement and productivity reports.</td>
<td>Batch-processing.</td>
</tr>
<tr>
<td>Development of progress cost reports:</td>
<td>Programs normally record direct and indirect costs in various categories and prepare progress claims at monthly intervals during the construction phase of any size building project.</td>
<td>Time-sharing and batch-processing.</td>
</tr>
<tr>
<td>Interior planning and space utilization reports:</td>
<td>Typical programs allow building plans to be transferred from drawings to computer, to form the basis of new layout designs and their comparative evaluation (e.g., department stores).</td>
<td>Batch-processing in conjunction with digitizer to transfer existing layout from drawing to computer.</td>
</tr>
</tbody>
</table>
panies. In most cases these teething problems can be traced to the existence of poor communication channels between the construction expert and the computer specialist. Typically, the computer specialist would embark upon the development of a program, blissfully unaware of the fundamental characteristics of the building industry. His only source of information would have been a construction expert who neither appreciated the type of information required for the production of a meaningful program, nor had the ability to foresee the interpretation likely to be placed on his information by the computer specialist. Inevitably, the end result was a computer program which did not save time and yet required the contractor to adopt a most costly office procedure. These unfortunate circumstances forced some early pioneers of computer applications in the building industry to become staunch supporters of traditional practices.

Fortunately, today the question is no longer whether architects should make use of computers, but rather, how can they best capitalize on existing computer capabilities?

The typical computer applications listed in Table I are all related to the solution of subsystems within the overall building design systems. Applying computers to isolated tasks, in this manner, is not necessarily cost effective. The real benefits are obtained when these separate tasks are linked together by a common data base. Unfortunately, many obstacles stand in the way of the development of central data bases in the architectural profession. The flow of information in the building industry is complicated by the relatively large number of decision makers involved. The architect is only one principal in a large open system incorporating the building owner, structural engineer, material supplier, fabricator, mechanical and electrical engineer, landscape architect, contractor and, in most countries outside the U.S., the quantity surveyor or building economist. Each of these decision makers speaks a slightly different language requiring a certain amount of translation. Simplification of the open system shown in Figure 1 by the addition of a central computerized data base (Figure 2), presupposes the existence of a high degree of standardization. It is only during very recent years that trends have become apparent, which suggest that the building industry is at last undergoing the painfully slow process of preparing and adopting uniform standards, codes, regulations, and documentation and information classification systems.

**ARCHITECT—COMPUTER COMMUNICATION**

Architects communicate most efficiently with scale drawings and sketches, and therefore require access to computer systems incorporating graphic devices, such as plotters, digitizers, light pens and cathode ray tubes. In addition, a high degree of interaction is desirable for most architectural computer applications. While interactive time-sharing systems are now widely available to the building industry, computer graphics systems are thinly spread and relatively expensive. As far as the educational sector is concerned, few schools of architecture have access to even the most elementary graphics facilities. In the minority of cases where a university can claim a graphics capability, more than likely the utilization of this capability is severely limited by the absence of adequate interface facilities between the graphics system and the other batch-processing and interactive time-sharing computer facilities available on campus.

Perhaps the architect’s need for “interactive” computer systems requires further explanation. The design process involves long decision chains coupled with the progressive evaluation of information sets, in which each decision stage may affect both the type of information to be considered and the evaluation procedure. In this situation, there is a critical need for direct interaction between architect and machine. Firstly, the architect cannot afford to pause in the decision sequence while cards are punched and a program is processed in batch mode. Secondly, the architect must be able to change the sequence of the decision chain without modification of the computer program. Ideally, the computer might become a partner in the design process, with the ability to respond intelligently to the proposals made by the architect. In the absence of computers with some degree of artificial intelligence, an interactive time-sharing system appears to provide the best vehicle for satisfying the rapid response requirement.

**SHOULD THE ARCHITECT “PROGRAM” THE MACHINE?**

Architectural educators have been asking themselves this question for some years now. Their task would be made so much easier if there existed a set of standard computer programs used on a day-to-day basis by the building industry. Unfortunately, the very existence of such standard programs would severely limit the application potential of computers. Developments in design methods over the past few years have demonstrated the strong influence of computers in shaping the methodologies of the 1980’s. At the same time, present and
future developments in computer technology are likely to provide the building industry with tools so powerful that they will have a profound influence on the scope and effectiveness of the services offered by the practicing professional. It is essential that the architect should have direct input in this development process and actively participate in the formulation and coding of programs, which will embody the most suitable planning, design and construction methodologies. The prospective building owner of the 1980's is likely to expect not one or two but eight to ten alternative design solutions for a major building complex, each alternative to be based on a detailed analysis of a large volume of accurate information. To achieve this standard of service, the architect must have access to comprehensive data bases and fast retrieval systems. There is no doubt that the computer will constitute the most inexpensive means of providing both.

It would be a grave error to look upon the computer as no more than a high-speed desk calculator or super slide rule. One of the most inspiring capabilities of the computer is its ability to "simulate" entire systems and perform long sequences of operations. The effect of a single computer program in influencing the end product will therefore be much greater than has been the case with the application of less powerful design and planning tools in the past. To ensure that the control of the various planning and design processes continues to be exercised by the designer, the latter will need to be actively involved in the development of computer programs. In short, the practicing architect will require not only a general knowledge of computer systems, capabilities and limitations, but also a working knowledge of at least one computer language. Disregard of this requirement will eventually lead to a serious loss of influence and credibility of the architect. Unfortunately, some minor symptoms of a lack of professional foresight and responsibility in this area are already today a fact of life. There is a sizable number of computer programs in use today, particularly in educational institutions, which are of unknown origin, which have been subjected to virtually no testing and which are based on unexplained and sometimes erroneous theories. To make matters worse, the untrained user tends to be extremely
From the collection of the Computer History Museum (www.computerhistory.org)

The need to train undergraduate architecture students in the computer field was first recognized by schools of architecture in the middle 1960’s. In those early years, most architectural educators with little or no knowledge of computers found themselves at a complete loss to develop and implement in-house computer courses. Under these circumstances, it was found to be most convenient to place this responsibility into the eager hands of mathematics, science and computer science departments. In fact, it was most common at that time as it is still today in some schools of architecture, for computer course material to be considered an adjunct to if not synonymous with mathematical topics, such as calculus.

The Boston Computer Conference of 1964, the first formal gathering of computer enthusiasts in architecture, was attended primarily by a group of young architectural educators from the greater Boston-Cambridge university complex. These pioneers were looked upon by their senior colleagues, and indeed by the profession at large, as misguided theoreticians with a warped view of architecture. Quickly the term “computer freak” was coined in academic circles, as a form of resistance to the possible insurgence of this new electronic monster into the inner precinct of architecture, namely the design process. Naturally, this atmosphere was not conducive to the systematic evaluation of potential computer applications in schools of architecture, nor did it lead to the careful structuring of effective computer courses.

A decade later, with most of the early paranoia dissipated, it is high time that architectural educators should re-examine the structure and content of existing computer courses. Which of the following instruction levels is most appropriate for architecture students?

1. A general descriptive course dealing with a variety of typical computer applications.
2. A descriptive course which deals not only with typical computer applications, but also provides the architecture student with a working knowledge of a number of existing computer programs.
3. A description-implementation course which also incorporates elementary computer programming.
4. A complete computer programming course incorporating also detailed application information.
5. A rigorous computer science course.

In a general descriptive course (i.e., course (1)) the student learns little or nothing about computer systems, but gains only a general knowledge of existing computer applications in the profession. This type of course tends to be ineffective and unconvincing, while providing the student with little information and no experience for hands-on operation at a later date. Although the student comes into direct contact with computers in course (2), he is forced to use programs which he does not fully understand. This is a very dangerous situation and could easily lead to the acceptance of computer results without adequate control over the accuracy of the program. Course (3) allows the student to develop simple programs and understand existing programs. Not only does this type of course emphasize the importance of checking programs for logic and syntax, but it also forces the student to define the problem in a definitive manner. This course meets most closely the special requirements of architecture students.

Course (4) is more suitable as a follow-up elective course, for the interested architecture student who wishes to gain more in-depth computer knowledge for possible specialization after graduation. A rigorous computer science course (i.e., course (5)) is not suitable for architecture students at any undergraduate level, since it would unnecessarily make programmers of the students.

WHO SHOULD TEACH?

Many computer applications in architecture are so strongly interwoven in the design process, that the computer is not just a tool, but an integral part of the design methodology. In this respect, computer applications often form the very basis of the development of new design methods. From this point of view alone, it is most important that the computer course should be offered in-house rather than by an outside school or department. Moreover, computer service courses are normally offered to students in a wide range of majors, with the result that programming examples tend to have little direct relevance to any particular major. Experience has shown that architecture students, in particular, find it difficult to separate the programming aspects in advanced mathematical examples, since they are unfamiliar with both the mathematics and the computer programming. Finally, there is the important consideration of faculty participation in an in-house computer course. Not only does the in-house course provide an opportunity for faculty training, but it also encourages the integration of computer applications in concurrent and subsequent non-computer courses.

A CASE STUDY

The computer course presently offered in the School of Architecture and Environmental Design, California Polytechnic State University, San Luis Obispo, has been developed to meet the specialized requirements of undergraduate students in architecture and related majors. The School of Architecture and Environmental Design is the largest school of its kind in the United States with a total enrollment of around 1500 students. While the architectural engineering, city and regional planning, landscape archi-
Figure 3—Computer course structure in the School of Architecture and Environmental Design of the California Polytechnic State University, San Luis Obispo

Architecture and construction engineering, the architecture program carries by far the largest portion of the enrollment. All students are required to take a two credit unit computer course during their sophomore year.

Approximately 120 students are enrolled in the computer course each quarter. To avoid unnecessary duplication and yet retain a strong tutorial emphasis, the course is divided into a one hour lecture and a three hour tutorial sequence each week. The lecture is attended by all students concurrently, with the subject matter shared among participating faculty for presentation. In this manner, each instructor is encouraged to develop his or her interest area into a comprehensive lecture series.

The tutorial sessions are attended by approximately 24 students and are intended to extend the lecture material, as well as provide assignment assistance on a group and individual basis. The course is designed to introduce students with no previous knowledge of computers and computer applications in architecture to the capabilities and limitations of computer systems and their utilization...
in environmental design, with the following specific objectives:

- To provide students with a general understanding of computer hardware and the peripheral units which are normally combined in batch-processing, interactive time-sharing, remote job entry and graphical computer systems.
- To explore the existing capabilities and limitations of computers in relation to existing and potential architectural applications.
- To familiarize students with existing computer applications in architecture and, in particular, the range of computer programs available to students in the school.
- To acquaint students with the esoteric vocabulary which has been developed as a byproduct of the growth of computer applications in all fields.
- To provide students with a working knowledge of one computer language (e.g., FORTRAN IV, BASIC, etc.), to enable them to code relatively simple computer programs and understand listings of existing programs.
- To introduce students to procedures for defining problems and structuring information prior to computerization.
- To familiarize students with computer program documentation and checking procedures.

Emphasis is placed on the development of user experience by the student and the integration of this newly gained knowledge in concurrent and subsequent courses. While it is not intended for the architecture student to become a computer expert, follow-up elective courses are offered both within and outside the school for those students who wish to extend their knowledge in this field (Figure 3).

**Course content**

Computer languages, by and large, have been developed for scientists to facilitate the solution of mathematical problems involving complex and repetitious calculations requiring little input and output capability. Most architectural problems, on the other hand, require only a modest degree of computing power, but depend on the transfer of large quantities of data (i.e., numeric and alphabetic) to and from the machine. In addition, the architect has in the past and will continue to rely largely on graphical means of communication. To date no computer language specifically designed for the solution of architectural problems has been developed.

If we follow the commonly accepted classification of computer languages into machine-oriented, procedure-oriented and problem-oriented, then it is without doubt the procedure-oriented language which most closely fulfills the solution needs of an architectural problem. Of the half-dozen or so major procedure-oriented languages available today, FORTRAN IV was selected to form the basis of the course. The principal reason for this choice being the availability of FORTRAN on almost every substantial computer system, today. From time to time, BASIC has come under serious consideration, not because of its simplicity, but because it is often the only language available on low cost mini-computer and programmable calculator systems.

The course, as it is taught now, incorporates ten lectures structured in the following manner:

**FIRST WEEK:** *Introduction:* What are computers and how do they work? Computer languages and computer systems.

**SECOND WEEK:** *Fortran IV:* Statement types; integer and real constants and variables; operators; functions and expressions; ARITHMETIC statements; STOP and END statements; PRINT statement (Watfor compiler).

**THIRD WEEK:** *Input and Output Statements:* Example No. 1 batch-processing program (Watfor compiler); READ, WRITE and FORMAT statements; time-sharing terminals as input and output units; Example No. 2 time-sharing program.

**FOURTH WEEK:** *Transfer Statements:* Unconditional GO TO statements; conditional IF statements; COMMENT statement (time-sharing and batch-processing); Example No. 3 time-sharing program.

**FIFTH WEEK:** *Graphics:* Computer graphics in architecture; large graphics systems, minicomputers and programmable calculators.

**SIXTH WEEK:** *Graphics (Cont.):* Hewlett-Packard 9830 graphics system. *Site Planning:* Computer application in site analysis and mapping; application of the GRID (batch-processing) program.

**SEVENTH WEEK:** *Site Planning (Cont.):* Application of the GRID (batch-processing) program.

**EIGHTH WEEK:** *Repetition:* The DO statement—acting as a counter—as acting as a number generator—controlled by program user; Example No. 4 time-sharing program.

**NINTH WEEK:** *Arrays:* What are arrays? Comparison of arrays and simple variables; DIMENSION statements, arrays in input and output statements; Example No. 5 time-sharing program.

**TENTH WEEK:** *Data Files:* Data file input statements; data file entry under the EDIT subsystem; data storage and retrieval; Example No. 6 time-sharing program.

These lecture topics are strongly reinforced by tutorials, during which the subject matter of the previous lecture is extended by suitable programming examples, typical executions of existing programs, films, demonstrations of in-house computer facilities and visits to the University Computer Center.

In addition, students are required to submit one assignment each week. All of the assignments require access to the University computer facilities with emphasis on the
interactive time-sharing system. Assignment topics cover a wide range of architectural subjects and include programming exercises (Figure 4), as well as the execution of a variety of existing programs (Figure 5).

Support material

At the time of implementation of the course a review of available publications revealed a singular lack of any comprehensive computer textbook suitable for undergraduate architecture students. Existing texts could generally be divided into three categories.

- DESCRIPTIVE texts written for the practicing architect, which describe areas of architectural utilization of computers, but do not address themselves to problem definition and programming.
- USER-ORIENTED texts written for computer users in a number of disciplines, such as engineering, social science and psychology, but not architecture.
- SPECIALIST computer science texts.

To overcome this problem, a textbook incorporating both a description of computer hardware, computer systems, implementation alternatives and elementary FORTRAN IV programming was prepared and published by faculty instructing in this area.4

During the early implementation period of the course, it was recognized that the type of computer program useful to the building designer could not be developed by the architect during the design process. Therefore, to facilitate the integration of computer applications into all phases of the architecture degree program, the school immediately embarked upon the establishment of an architectural program library. With an abundance of batch-processing computer programs commercially available from other educational institutions and industry, the limited need for this type of non-interactive program was soon satisfied.5

Unfortunately, the same situation did not apply to interactive time-sharing. Even today, there is a singular lack of commercially available architectural time-sharing programs, and the few programs which are offered for sale by computer service bureaus are well outside the financial means of the entire California State University and Colleges System, let alone an individual school. Faced with this situation the School of Architecture and Environmental Design undertook the time-consuming task of software development. Over a period of two years faculty and senior students were able to develop more than 20 major interactive time-sharing programs, ranging from the design of environmental control systems (i.e., natural and artificial lighting, acoustics and heat transfer) to structural analysis, computer-aided design and construction management games. Each program is carefully documented (i.e., program name, language, author, disclaimer, program description, limitations, operating instructions, input and output, theoretical basis) and, after a period of testing, groups of programs are formalized into printed manuals.6 These manuals are available to students at production costs through the University bookstore. The school has now obtained approval from the University to market these computer programs commercially, and it is hoped that in the near future sales profits will in themselves sustain the entire program development operation.

Computer facilities

The school presently has access to the following computer systems:

1. BATCH-PROCESSING (LOCAL): IBM 360/40 computer system with 180 K (approx.) core storage, 1100 lines per minute printer, 1000 cards per minute reader, 300 cards per minute punch, three disc drives, controller and tape drive.
2. TIME-SHARING (BATCH OR RJE): IBM 360/20 computer system with 8 K core storage, 315 lines per minute printer, 120 cards per minute reader and 90 cards per minute punch, which serves principally as

---

**ASSIGNMENT NO. 4**

Develop a TIME-SHARING computer program capable of estimating construction costs based on the name, quantity and unit cost (i.e., rate) of each material nominated by the user.

Program to calculate individual material costs and provide a total cost estimate for the building.

(List program to a maximum of 20 materials and 20 characters per material name.)

Build a safeguard into the program so that unit costs (i.e., rates) cannot exceed $100.

---

**ASSIGNMENT NO. 5**

Use the TIME-SHARING program HEAT1 to determine the number of air-changes required to maintain a temperature of 85°F inside of building space which has two (2) surfaces (i.e., east and south walls) exposed to direct solar radiation.

<table>
<thead>
<tr>
<th>SURFACE</th>
<th>LENGTH</th>
<th>WIDTH</th>
<th>AREA</th>
<th>CONSTRUCTION (OF WALL)</th>
<th>##AIR, TEMP.</th>
</tr>
</thead>
<tbody>
<tr>
<td>east wall</td>
<td>30 ft</td>
<td>12 ft</td>
<td>yes 80 ft²</td>
<td>0 ft² (free choice)</td>
<td>15°C/°F</td>
</tr>
<tr>
<td>south wall</td>
<td>20 ft</td>
<td>13 ft</td>
<td>no</td>
<td>10 ft² (free choice)</td>
<td>12°C/°F</td>
</tr>
</tbody>
</table>

*(Design your own system of construction)*

**Air temperature in shade is 80°F**

How many air-changes are required to maintain a temperature of 75°F inside the same building space?

---

Figure 4—Typical computer programming assignment

Figure 5—Typical assignment requiring the execution of an existing computer program
CONNECT TIME LIMIT FOR 7 PORTS

57%  69%  51%  61%  70%  80%  43%  90% (Port utilization) *1

University

School of Architecture & Environmental Design

*1 7 Ports available an average of 13hrs. per day for 7 days a week.
*2 Each time period is equal to 9 effective instructional weeks.
*3 Percentage of total university time-sharing usage.

Figure 6—Time-sharing computer usage by the School of Architecture and Environmental Design at the California Polytechnic State University, San Luis Obispo
3. TIME-SHARING (INTERACTIVE): Central time-sharing network supported by a CDC 3170 dual processor computer installation located at California State University, Northridge. Of 96 ports distributed among the 19 campuses of the California State University and Colleges System, seven (7) ports are available on this campus.

4. GRAPHICS: Until recently the university leased an IBM 2250 graphics terminal connected to the local IBM 360/40 computer system. This has now been replaced by a mini-computer graphics system consisting of a PDP 11/35 processor, two display tubes and a 36 in. CAL-COMP drum plotter. A second CAL-COMP plotter is presently supported by an IBM 1130 computer system at a nearby community college, under a limited time exchange and verbal agreement. In addition, the school owns a Hewlett-Packard 9830A programmable calculator which supports a digitizer, plotter, cassette tape unit and printer.

The utilization of these facilities by the school has increased steadily during the past three years. However, by far the most dramatic increase in usage has been in the interactive time-sharing field, where the school now uses over 40 percent of the entire time-sharing capability available to the University (Figure 6). This extraordinarily high usage level has been achieved despite the saturation of all computer systems on campus and the presence of strong computer science and engineering departments.

PROJECTIONS FOR THE YEAR 1980!

By far the most critical problem facing the continued development of computer applications in the School of Architecture and Environmental Design at the California Polytechnic State University, San Luis Obispo is the complete inadequacy of the existing computer facilities available on and off campus. Despite the frequent breakdown of teletype and CRT terminals, the university normally achieves a time-sharing port utilization in excess of 85 percent. At the same time, job turnaround times on the RJE and local batch-processing systems are in the order of 10 to 15 hours and often reach 30 hours toward the end of each academic quarter.

Conservative estimates based on a detailed analysis of courses offered by the school indicate that by 1980, architecture students will require approximately 1,000 time-sharing connect hours per week (i.e., 40,000 hours per year), with access to at least 58 ports (i.e., based on 17 hours of time-sharing availability per day). These estimates presuppose that the time-sharing system will have a graphics capability, supported by suitable input-output devices at each fifth port.

In addition, by 1980, the availability of large data bases will have become an integral component of architectural computer applications. The full utilization of such data bases will require greatly improved system interfaces, to facilitate the transfer of data from one computer system to another. For example, a user will expect to be able to access a regional land-use data base on an interactive time-sharing system, transfer a sizable parcel of information to a larger computer system and process this information using a data analysis program of the capability of SYMAP, in batch mode.

It would appear that university computer center managements, in general, are not fully aware of the implications of an almost explosive increase in computer usage by non-computer science students. There is no doubt that during the next decade computer science departments, which now hold the privileged position of principal computer user on most campuses, will relinquish this position to users of library programs, such as students majoring in social and behavioral science, business administration, agricultural science, engineering, medicine, biological sciences and architecture. This change in user distribution will necessitate university computer centers to broaden the scope of their services by providing increased assistance to non-computer science users. Moreover, some attention will need to be given to the streamlining of job entry procedures. For example, in the California State University and Colleges System students wishing to access library programs under the RJE batch system are normally required to submit at least five control cards with each job (e.g., the SYMAP program may require up to 20 control cards). Needless to say, the architecture student learns absolutely nothing about environmental design by struggling with the interpretation, punching, assembly and debugging of half-a-dozen unnecessary control cards.

From a more radical point of view, consideration might be given to the complete reorganization of computer facilities in universities. At the present time, university computer centers carefully guard their right to centralize computer facilities, in particular processors. It may be difficult to continue to uphold this policy in the 1980’s. Processors are neither the most expensive components of a computer system, nor do they normally require a great deal of maintenance. Peripheral devices, such as line printers and plotters, on the other hand, account for a sizable proportion of both capital and maintenance costs of a computer system. It may be argued that centralization of these units in a computer center, accessed by processors distributed around the campus, could contribute greatly to a more flexible and user responsive computer service.

REFERENCES


