Designer-Client Relationships In Architectural And Software Design

by

Yehuda E. Kalay

Department of Architecture
University of California, Berkeley

&

Carlo Séquin

Department of Electrical Engineering & Computer Science
University of California, Berkeley
Berkeley, California

Yehuda E. Kalay

is a professor of Architecture at the University of California, Berkeley. He joined Berkeley in 1992 after spending 10 years at the State University of New York at Buffalo. His current research focuses on developing software that can assist architects to predicts and evaluate their design decisions.

Carlo Séquin

has been a professor of Computer Science at the University of California, Berkeley since 1977. He first worked in digital systems design and on CAD tool for VLSI. Now he works in computer graphics, solid modeling and on CAD tools for mechanical engineers and architects.

An upper-level undergraduate architectural design studio and a graduate computer science CAD course were paired to study client-designer interactions. The dual nature of these courses led to two sets of products: building designs compatible with the specifications of the clients, and prototype CAD tool to assist architects in the conceptual design phases. First, the computer scientists acted as clients to the architects, who designed a

building for the computer science department. Once the computer science students had become familiar, through observation, with the architectural design process, they began developing tools for the architects' use. In that reversed-role, the architects became the clients of the computer scientists. For both parties this interaction provided an opportunity to experience the social aspects of the design process, in particular, the designer-client relationships, which most often are absent in traditional educational settings. This paper describes the objectives of this integrated pair of courses, the methods and processes used, and some of the results.

INTRODUCTION

Simulating the process of designing buildings in an architectural design studio is often isolated from 'real world' concerns which might distract students from their primary effort of creating architectural objects. While such abstraction helps to focus the energy of the students on the synthesis and composition of buildings, it ignores the social aspects of the architectural design process itself, its designer-client relationships, and its legal and economic aspects. Often it conveys to students the wrong impression that architects have the sole responsibility for shaping the building, when in fact clients, statutory authorities, and neighborhood action groups have as much (or more) influence on the product as the architects themselves [Cuff, 1991]. It also ignores the role of architects as educators of their clients, wrom end-users during the tool development process itself.

In an effort to address some of these concerns in the architectural design studio, as well as in the software design courses of the computer science department, and to study their impact on the learning process and on the quality of the products, we have offered architecture and computer science students the opportunity to participate in a new, unique, multi-disciplinary, collaborative design studio. Students from the Computer Science Division in the Electrical Engineering and Computer Science (EECS) department acted as *clients* to the architects who designed a building for their use. At the same time, the computer scientists designed and developed CAD software for use by the architects. This experiment provided both groups of students with an opportunity to experience both the roles of designers and clients, who share the responsibility for the emerging products (in our case óbuildings and software).

Our experiment differed from several other recent efforts that address the issues of communication and collaboration among the various partners in the architectural design process. While Khedro et al [1993] have focused on facilitating the sharing of design information among the participants in the design process, and while Fruchter et

al [1995], and Chen et al [1994] have focused on cross-educating students from different disciplines about the methods and practices of the other disciplines, our experiment focused on simulating designer-client relationships. It mimicked, to some extent, certain real world architect-client design collaborations, as reported by Novitski [1994].

We chose to focus on designers-clients collaboration because, after the talent, knowledge and experience of the designers themselves, it is the most critical aspect of designing both buildings and software [Larson, 1983]. All professions depend, to a smaller or a larger degree, on their clients. But architects and computer scientists, more than physicians, engineers and lawyers, have developed a special relationship with their clients: buildings and software affect how clients will live and do businessósomething clients know more about than the architects and computer scientists. Thus, clients determine much of the products' function (see, for example, the collection of articles on the importance of client input, in the form of specifications, in designing software, in Communications of the ACM special issue on Requirements Gathering, 38(5), May 1995).

Our experiment was based in one of the Department of Architecture's upper-level undergraduate design studios, taught concurrently with a graduate-level course of the Division of Computer Science. The aims of these two courses were to study architectural and software design processes, the methods and the tools that can support them from the architects' and the computer-scientists' points of view, and the designer-client relationships that shape the processes and the products of both disciplines.

The choice of partnership between the departments of Architecture and Computer Science followed from the computer-based nature of the studio, as much as from the nature of the experiment itself. The Computer Science Division at U.C. Berkeley has recently moved into its newly constructed building, Soda Hall. The CS students have been discovering the good and the bad features of their new space; thus they have been 'sensitized' to architectural issues. The process of design and construction of Soda Hall has also prompted the development of a rather detailed, furnished computer model of the building and an interactive walkthrough program that allowed the clients to explore their building before it was constructed [Funkhouser & Sequin, 1993]. Several CS faculty and students were actively involved with the Building Program Committee, where they had acquired first-hand experience with the benefits and shortcomings of architect-client discussions. Their frustration with the fact that not everything turned out as they had intended, provided strong motivation for an attempt to build some computer-based

tools that would help the building process proceed more smoothly and reduce the number of issues that are lost in the process.

OBJECTIVES

The unifying task for the two courses was the redesign of a building for the Division of Computer Science, using the well-established program of their existing building-Soda Hall. We picked a different site than the actual one, which offered new constraints and opportunities for a totally newly shaped building. The existing building was used as one of several realistic case studies, giving the students an opportunity to visualize possible implementations of the specifications in the program, as well as a concrete focal point for their discussions. The explicit objectives of these two concurrent courses have been stated as follows:

Education: Cross-educate professionals from different design fields about the needs, views, and concerns of their clients, as well as the methods and practices used by another design-oriented discipline, and reveal to them different, alternative thought processes.

Collaboration: Exercise and learn about collaboration between different participants in the design process, so each can be made aware of the needs and expectations of the other, and of the implications that decisions made by one group have on the other. In this case, the collaboration involved designers and users of their products: architects and occupants, as well as software developers and CAD tool users. Such collaboration has been shown to increase both the quality of the product (the building and the software) itself, as well as the satisfaction of its users, who become partners to the decision making process, hence have an 'ownership' stake in the product [Shibley & Schneekloth, 1988].

Communication: Make use of the emerging communication technologies that are destined to impact the profession of architecture as well as other design professions. Specifically, the Internet's World Wide Web ('the Web'), an asynchronous means of multi-media communication that provides easy access to information from different hardware platforms.

Record: Maintain a record of the decisions made during the design process, the reasons for making them, and the results of these decisions, thereby providing a 'log' of the design and an audit trail of the decisions that affected it. The Log includes text, sketches, and scanned photographs. It provides a unified basis for decision making at design

time, and will constitute a ‘user manual’ for operating the product and maintaining it throughout its lifecycle.

Building Model: Create a consistent solid model of the emerging building that can be used in computer-based rendering as well as for interactive “walk-through” exploration. Ideally, some of the rooms and features in this building model are hyper-linked to appropriate entries in the issues data base (below).

Issues Database: Generate a catalog of concerns and dependencies that are important to the current design project, as well as generic issues that could be re-used in future building projects. For instance, this database (a hypertext document) would retain in a structured manner important concerns to the users of the building, formal code requirements and constraints, as well as things that have actually gone wrong in Soda Hall or in other recent construction projects. It might also list some obvious dependencies, such as that windows that cannot be opened must be washed from the outside, which necessitates suitable provisions on the roof of the building, which in turn has implications on roof accessibility and the need for safety railings.

CAD: Exercise the use of computer-aided design tools, and test their efficacy for developing, representing, and communicating design solutions. The tools used by the students included state-of-the-art CAD software for modeling, rendering, presenting and communicating their designs. FormZ was used as the primary modeling tool, StrataVision as the rendering engine, and PhotoShop was used to make presentations.

New Tools: Develop improved CAD tools that will ease or enhance the design decision-making process, handle much of the routine bookkeeping, explicate the implications of design decisions, and make it possible to visualize the emerging design solutions early on, thereby facilitate the overall design process (qualitatively and quantitatively).

Obviously most, if not all these objectives are linked and rely on each other. For example, the collaboration between the architects and the clients made it possible to have ‘observers’ who could record the issues that were raised during the process of design, but who were not at the same time the designers themselves. New design tools emerged from the observation, and contributed to the process and communication of design. The primary computer-based communication tool used among all the participants was the World Wide Web. It provided a simple yet powerful multi-media recording tool, as well as easy multi-platform accessibility to each other’s records and to an

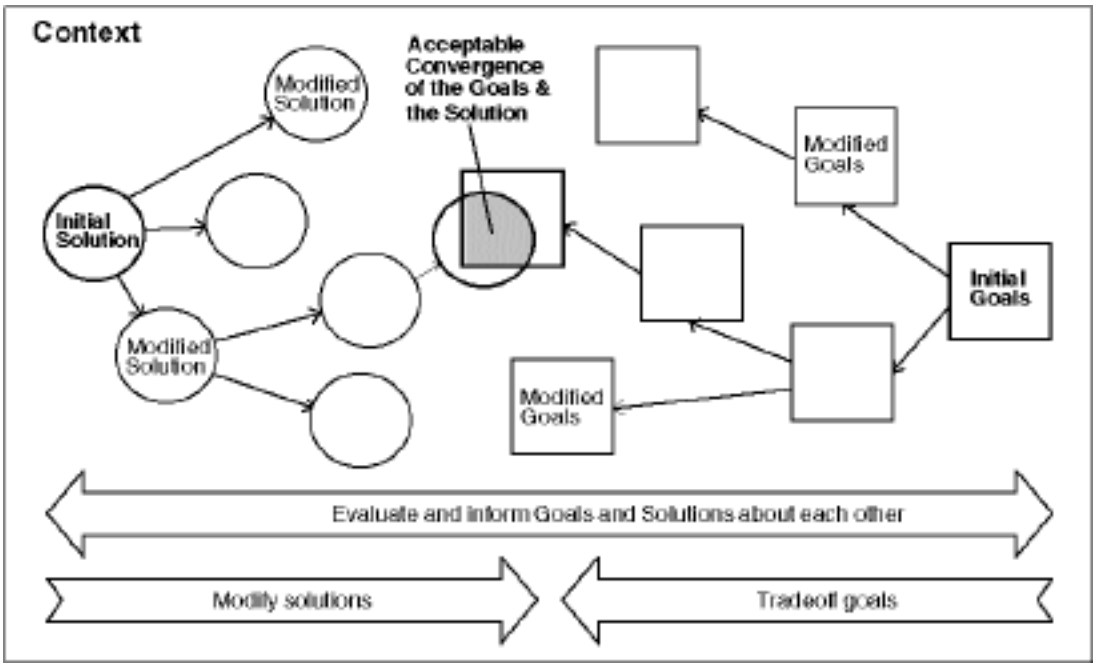

Figure 1. Design as a bi-directional search for a 'satisficing' solution.

informal distributed database containing model geometry as well as design issues.

METHOD

The architectural design process in this studio was structured as an iterative search for formal solutions (i.e., forms) that meet desired functional goals within a specific context, following the Performance-Based Design paradigm [Kalay & Carrara, 1995]. As depicted schematically in Figure 1, this paradigm assumes no causal relationship between form, function, and context. Both groups of students began their design exploration by analyzing the building program from a functional point of view. They developed adjacency matrices, and discussed issues that constituted the initial goals for their design. In parallel, the students learned about the context of the project, through site studies and a lecture by the campus architect. At the same time, the architecture students began their development of a conceptual solution, by looking at examples of existing buildings and by following their own inclinations and aspirations.

The three separate components (form, function, and context) began to converge after about three or four weeks. Within two weeks the building concept was adapted to the site, and the various functions that were required by the building program were being mapped into the conceptual building form. Inevitably, changes were needed: the

building concept needed adaptation to site requirements, and in more than a few cases it was found to be incompatible with the functional requirements. The adaptations resulted in modified forms, relocation of functions, and sometimes even negotiations with the clients to relax certain functional constraints in the building program.

Meanwhile, the CS students were introduced to the architectural design process and were asked to examine their building from an informed user's point of view. They were asked to list their likes and dislikes, and to formulate specific goals for their environment. A couple of CS students were assigned to each architect as clients. They followed the architectural design process, and contributed to it their own criticism, goals, and advice.

About four weeks into the semester, the CS students began to form plans for design tools that may aid the architects' work. This role-reversal provided both architecture and CS students with the opportunity to experience 'the other side of the coin': architects became clients, and CS students became designers. The architects had the opportunity to form their own wish-lists for the CAD tools they desired, and the CS students had the opportunity to design and develop these tools and test them with 'real' users on 'real tasks.'

Process

To facilitate the pedagogy, as well as the administration of the courses, the design process was partitioned into several phases:

development of a library of case studies
analysis of the building program to derive functional requirements
analysis of the context (physical and cultural)
generation of conceptual building forms
matching form and function
evaluation and refinement of form and function in context.

Case studies

A hypertext document was developed by each architecture student, describing existing computer science and similar buildings on campus and elsewhere, such as Evans Hall,

Soda Hall (Figure 2), the Mathematical Sciences building, and UNC-Chapel Hill. These documents included text and photographs, with specific attention to the salient qualities of important spaces, as identified by the architects and their clients (e.g., student offices, lecture halls, computer labs). The documents were linked to the Web home pages established by each design team, as well as to specific locations in the home page for the two courses, to be shared by everyone involved.

PROGRAM ANALYSIS

A condensed version of the building program that was given to the students is depicted in Figure 3. Adapted from the actual building program for Soda Hall, it included the

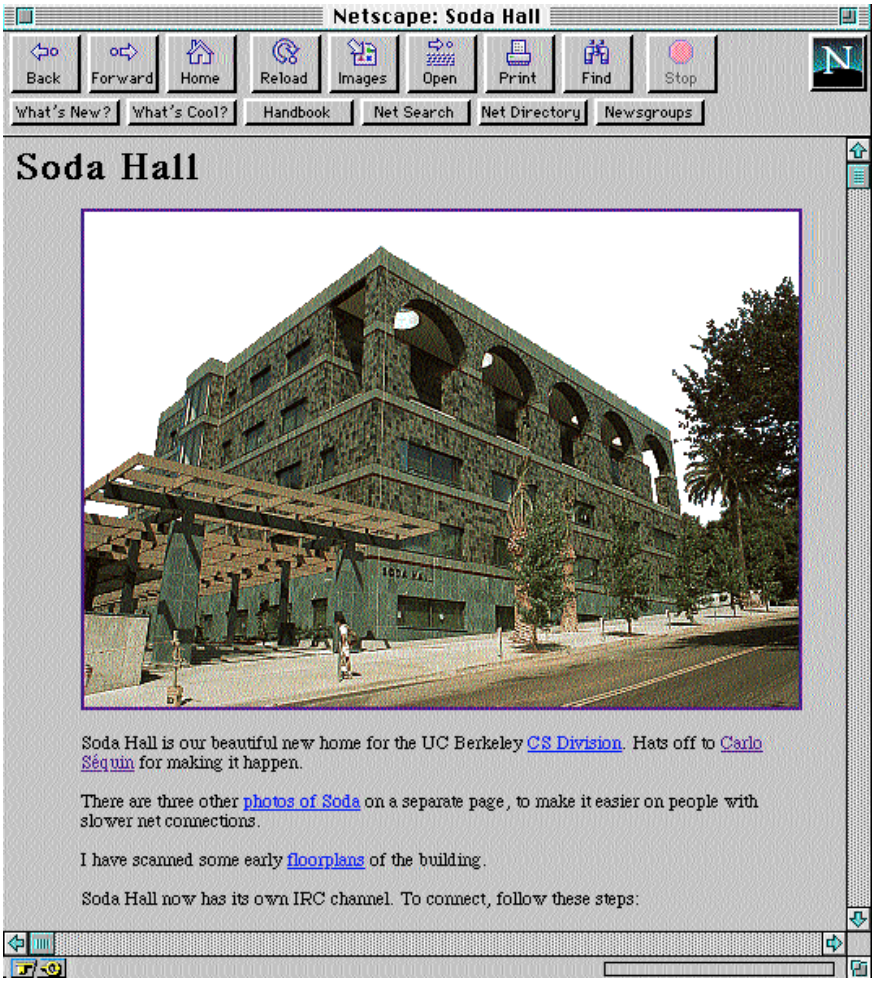

Figure 2. The original Soda Hall.

required spaces and their sizes, as well as an overall description of the building functions as desired by the clients. It is apparent from this list that the students were given a rather complicated building, with tight tolerances in terms of areas: the building was to have a total floor area of 100,000 square feet, with net usable area of 60,000, and rather specific amounts of floor space dedicated to various functions such as faculty offices, instructional labs, and an auditorium.

WHAT	ASF EACH	HOW MANY	TOTAL ASF	COMMENTS
Faculty -ind	180	40	7200	
Supervisors -ind	150	4	600	
Secretary	75	20	1500	
Visitors	90	20	1800	
Students	60	250	15000	
Chair -ind	300	1	300	
Sen. Admin -ind	150	2	300	
Div Staff	75	8	600	
Tech Supervisor -ind	150	4	600	
Tech Support	90	8	720	
Reception Area	500	1	500	
Div. Lounge	1600	1	1600	
Open Discussion -each flr	250	6	1500	
Auditorium	1500	1	1500	
lg. Classroom	800	2	1600	
sm. Classroom	500	4	2000	
Seminar room	200	4	800	
Instr. Labs	1000	3	3000	
Terminal room	400	3	1200	
Comp. Machine rooms	750	6	4500	
Research Labs	600	6	3600	
Student Consulting	200	6	1200	
Printer/Copier -each flr	150	4	600	
Kitchenettes -each flr	100	4	400	
Storage -each flr	200	4	800	
Central storage	1000	1	1000	
Receiving area	1000	1	1000	
.....				

Figure 3. Excerpts from the initial building program spreadsheet (simplified).

To understand the spatial implications of this program, the students represented the relative sizes of the required spaces as rectangular or circular shapes (Figure 4a). Different colors were used to designate functionally related groupings of spaces (e.g., student offices, labs, faculty offices). In the second half of the course, an interactive bubble diagram editor was developed by the CS students to expedite this rather tedious task (Figure 4b). In its final version, this tool automatically translates the numbers in the spreadsheet into labeled circles of corresponding scaled sizes, and provides some visual links between them to represent different kinds of relationships (strength of attraction or repulsion) gleaned from information in an adjacency matrix.

The translation of the numbers into geometric shapes was accompanied by discussions between the architects and the clients, intended to explore the nature and mean-

ing of each space. For example, the clients described to the architects what a student office ought to be like, and what they expected to see in a small lecture hall. To record this valuable information, the CS students produced a database of goals and issues that

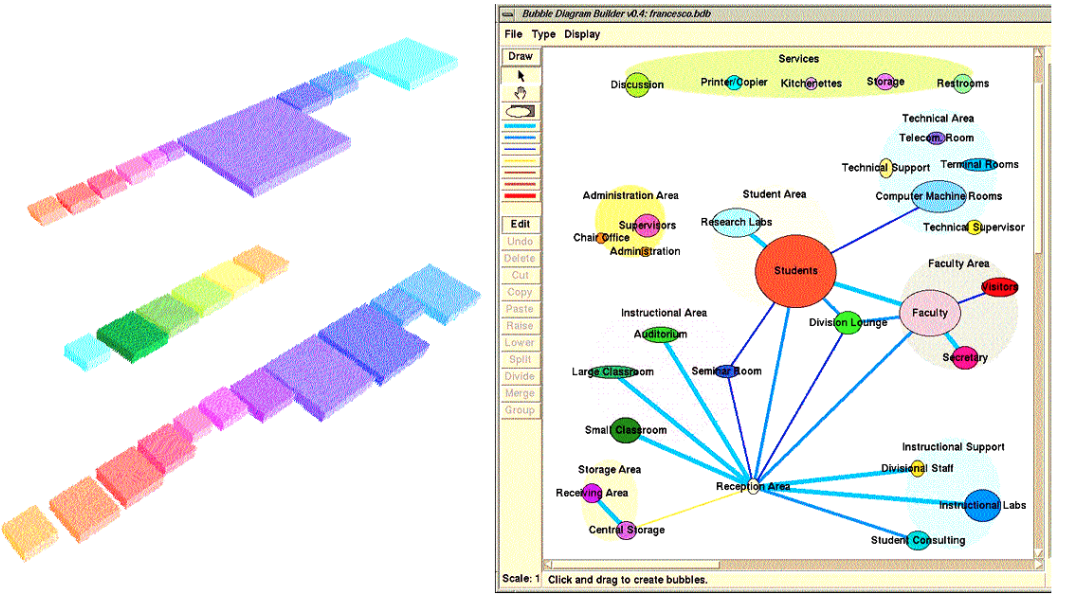

Figure 4. (a) Rectangular areas as first developed by the architects;
(b) bubble diagram editor, as developed by the computer-scientists.

allowed any concern to be recorded in textual form and then linked to relevant spaces or to other related concerns. The database was made accessible to all course participants by implementing it as a collection of linked pages on the Web (Figure 5). A special browser was developed, which facilitated the access to related issues.

CONTEXT ANALYSIS

The students were introduced to the architectural composition of the Berkeley campus through a lecture given by the campus architect, and were asked to investigate the site that was selected for this project on their own. They used digitized terrain maps, as well as photographs and an existing site model. Their investigations were reposited as hypertext documents in the Web (Figure 6). Among other findings, the students discovered that the stated campus architectural objectives were not well respected by the existing buildings, which allowed them to develop their design ideas more freely without strict adherence to the stated campus policy.

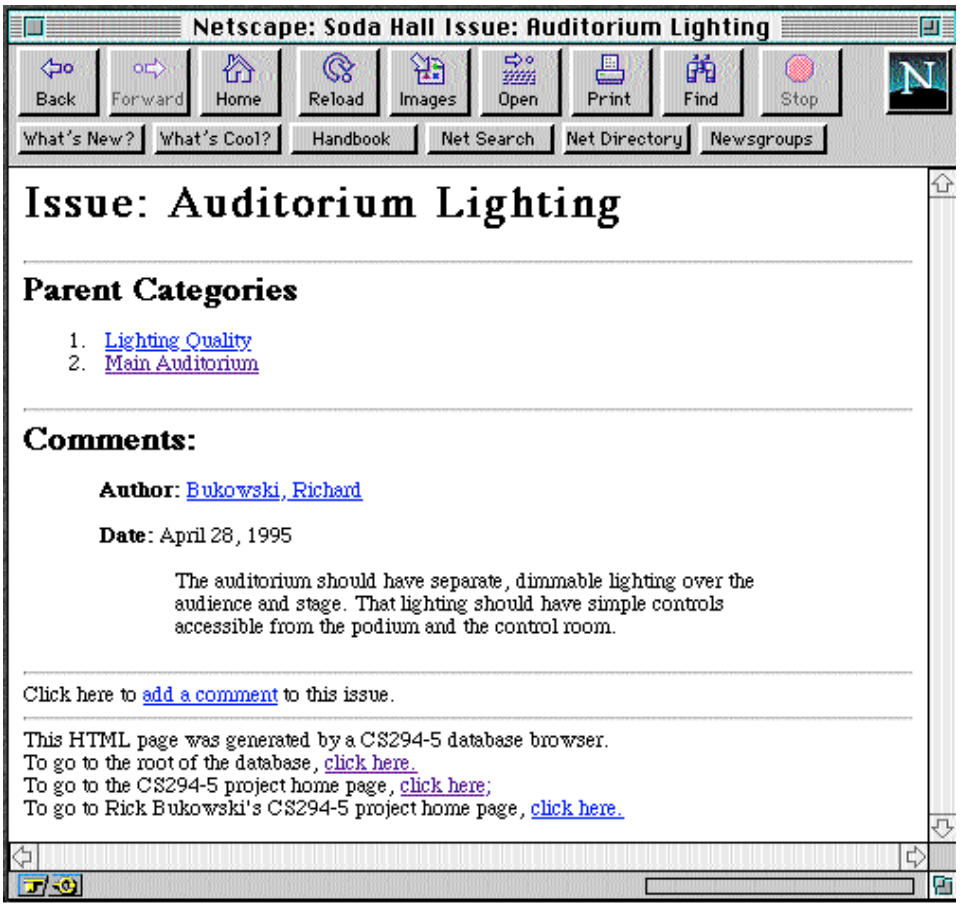

Figure 5. Sample entry from the database of goals and issues.

FORM GENERATION

The architecture students spent by far the most time on the development of a concept for the building. After all, this is a key responsibility of the architects, and it is clearly the most rewarding aspect of the design process. Each architect selected different forms, based on his/her own inclinations, background, and aesthetic ideas. Every architect, often more than once, changed the initial concept completely and started with new constellations of building wings and towers. In retrospect, as will be discussed later, the students where allowed too much freedom in developing and revising their formal ideas, and therefore ran out of time before detailed design could be developed.

The students used a variety of tools to model their forms: sketches, cardboard models, or modeling foam, but only reluctantly employed the suggested computational means, such as FormZ, indicating to us that more improvements are still required in these modeling tools and in their user interfaces for the purpose of developing early building concepts. All the different models were recorded electronically (by scanning,

Figure 6. Part of a site analysis document.

video capture, or by direct computer input) and placed in the emerging Web document (Figure 7).

Matching form and function

Given the list of functions and the bubble diagram that interpreted it on one hand, and the schematic form of the building on the other hand, the students' next task was to match the two and mold them into a building. It might have been assumed that starting the form-making process after both program analysis and site analysis were completed would ensure that the chosen forms had a good chance of meeting both needs. As it turned out, most designs were good matches to the particularities of the site, but the functions, with their specified floor areas, typically did not fit well into the shells of the conceptual building forms. In fact, the students were reluctant to address this unrewarding and tedious task, and by the middle of the course their floorplans still did not fulfill the giving building program in a quantitative manner.

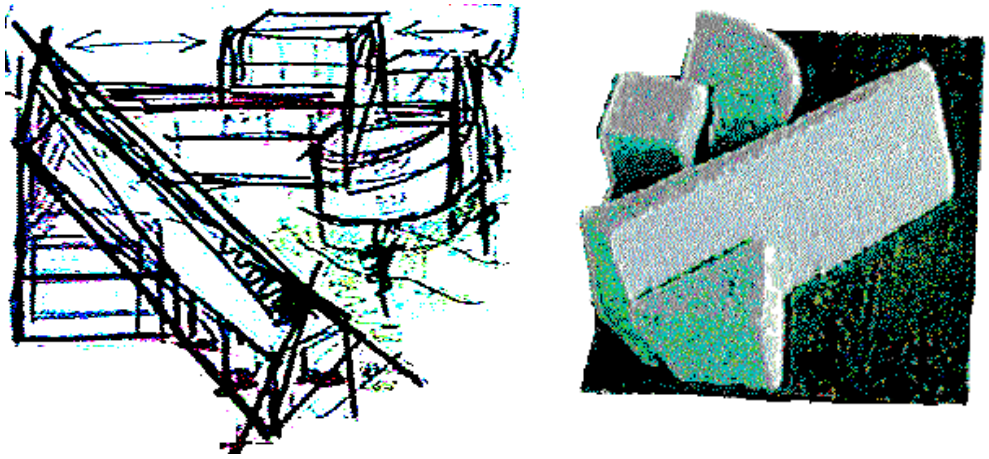

Figure 7. One of the early building concepts: (a) Sketch; (b) foam.

To help overcome this obvious hurdle in the design process, the CS students started to develop a couple of tools that specifically addressed this task. The Model_Slicer takes as input the conceptual form of a building and slices it into individual floor outlines, which are represented as properly dimensioned polygons (Figure 8a). The Floor_Space_Allocator then helps the architects to position the various functional groups onto the different floors, as expressed in the bubble diagram, by converting the circles into form-fitting rectangles that represent rooms in a schematic floorplan. This tool also kees track of and continually displays the allocated areas, the remaining areas yet to be allocated, as well as the ratio of the circulation areas to the useful area (Figure 8b).

With these tools, the architects could more readily test the efficacy of the conceptual forms they had chosen against the functional and spatial needs represented by the building program. In many cases they had to make significant changes to the building forms, or even completely revamp their concepts. At the same time, the process revealed inconsistencies or semantic ambiguities in the bubble diagram. For instance, the fact that the bubble types “faculty office” and “secretarial space” are connected with a strong positive link does not mean that every secretarial office has to be close to every faculty office. The discovery of such difficulties led both parties, the architects and the clients, to rethink the allocation of areas and the relationships between them.

Evaluation and refinement

The development of various building designs was accompanied by the usual visual inspections and critiques by the clients and by the teachers. In addition, the CS students

developed some tools that could subject the emerging designs to a more thorough evaluation. One such tool was the Adjacency_Checker, whose purpose was to verify that the adjacency relationships specified in the original building program were indeed still respected in the final designs (Figure 9). When conflicts or inconsistencies were discovered, they had to be corrected or negotiated away in discussions with the clients and then documented in the Log that continued to describe the building. For example, if the form of the building did not allow for all offices to face the preferred San Francisco Bay views (West and South-West), a way was sought to compensate for this 'deficiency.' As it turned out in discussions with the clients, some students and faculty actually preferred to look in a North-East direction at the green Berkeley hills, because of the reduced glare. A corresponding entry was made in the Log.

Another important evaluation, from the clients point of view, is what the building looks like from the inside. Modern computer technology will make it more and more common to provide clients with an early preview of the interior. However, this preview

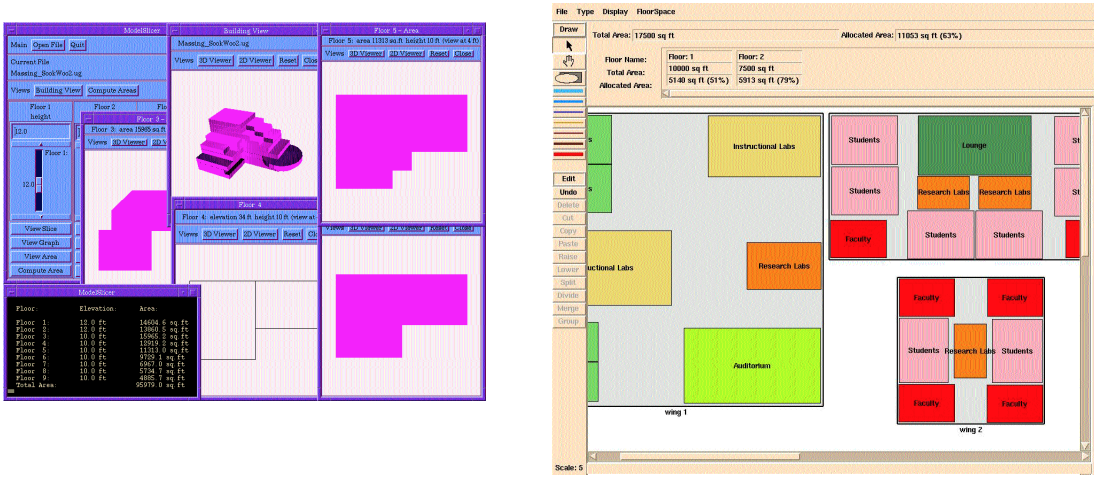

Figure 8. (a) The Model_Slicer; and (b) Floor_Space_Allocator.

option is predicated on the existence of a consistent 3D solid model of the building interior. Creating such a computer model is normally a labor-intensive and expensive task. The CS students tried to shorten the path to achieving such a model by developing a couple of programs that convert a symbolic floor layout quickly into a more realistic floorplan with walls of finite thickness in which some doors and windows are incorporated in reasonable default positions, and to extrude such floorplans into a 3D solid model which can be explored in a virtual reality walkthrough (Figure 10). However, it appears that the models generated by these programs have to be made more detailed, and possibly furnished, before the resulting visualization is of any real benefit to prospective building occupants.

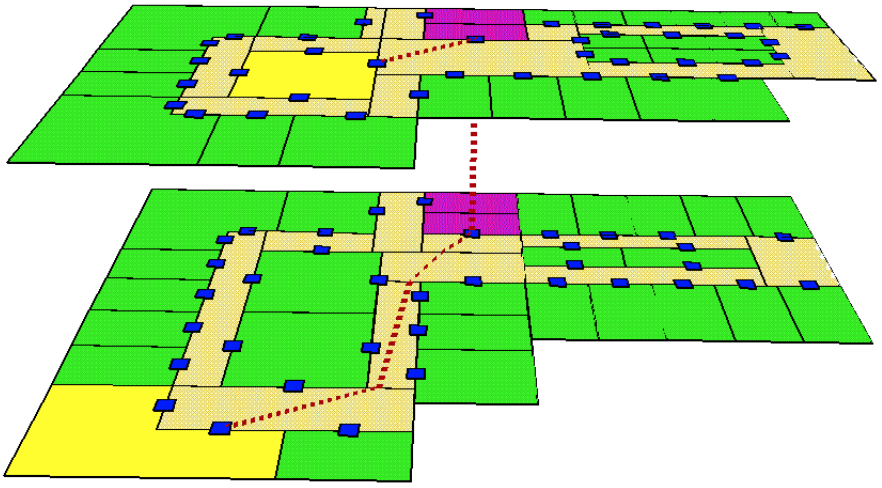

Figure 9. The Adjacency_Checker: between two spaces
(a) on the same floor; (b) on different floors.

PRODUCTS

The 'products' of this multi-disciplinary studio were numerous and quite diverse in nature. The most visible artifacts were, of course, the buildings designed by the architects (Figure 11). They were not developed quite as far as the instructors had hoped at the outset of the course, primarily because the architects were allowed to experiment longer than planned with the developments of various forms, rather than being forced to proceed with the best design they had available by some hard deadline.

Equally important are the tools developed by the computer scientists. These have the additional benefit that they are not just one-time artifacts, but can be reused and further improved in the future. In subsequent offerings of this design studio, these tools will be available from the beginning; they can thus be used to merge form and function earlier in the design process and to help the architects arrive at viable design solutions by a specified deadline.

Then there are some less obvious products that are, nonetheless, equally important. First, there are readily accessible, more-or-less complete logs of the design processes for the various teams, showing many of the intermediate products, such as the bubble diagrams, the site analyses, and early conceptual designs. It is very instructive to go back through this web of documents and ponder how some design issues have changed their relative importance as a function of time, and how some more flamboyant designs have simmered down to more practical solutions.

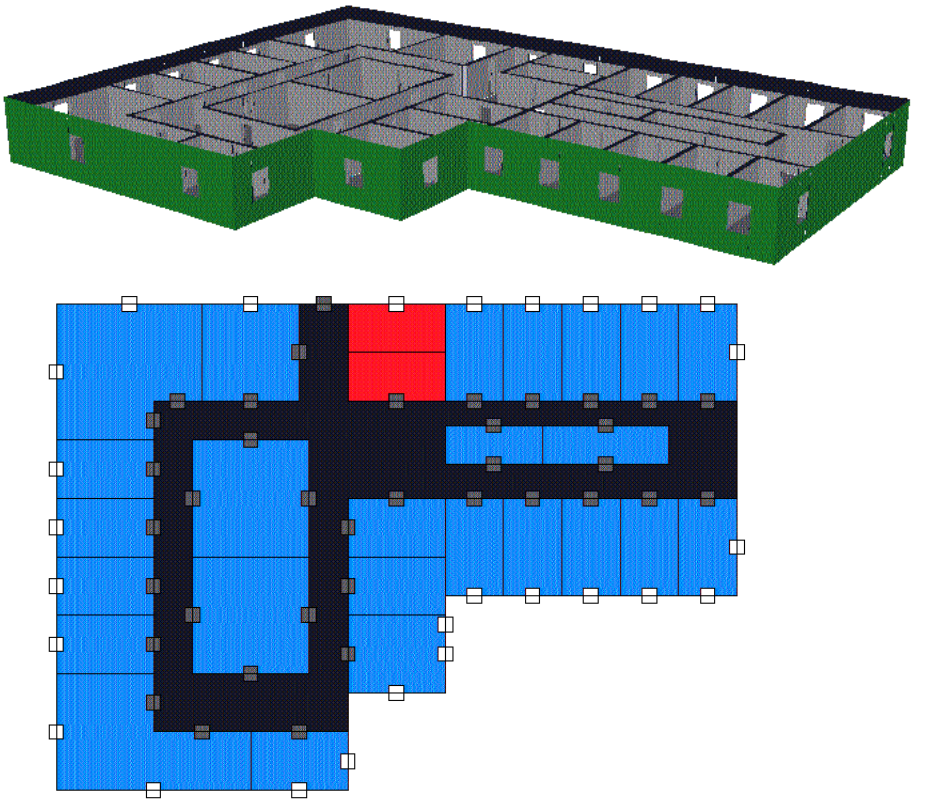

Figure 10. An extruded floor for interactive walkthroughs.

Furthermore, there is the issues-database, which contains information that should be of rather general interest for further design projects. While its structure is quite general, the content of the database at this point is still rather modest since only representative token examples of various kinds of issues were actually recorded.

Analysis

Pairing the two courses described above, and setting up the network-based Web infrastructure to allow electronic interaction between the two courses, required a somewhat larger-than-usual administrative overhead. The excitement experienced in the course itself and the final results made this investment well worth the effort. At first we had some concerns that the different levels among the students (undergraduate architects and graduate computer scientists) might pose some problems. We were pleased to find that they did not. Rather, a courteous professionalism developed, where each group respected the knowledge of the other. In fact, for the undergraduate architecture students it was very gratifying, if surprising, to find themselves in an authoritative position, where their opinions were respected as professionals rather than students. The computer scientists have learned to appreciate the difficult tradeoffs that are made when designing a building, and the less-structured, non-optimization-based design method typically

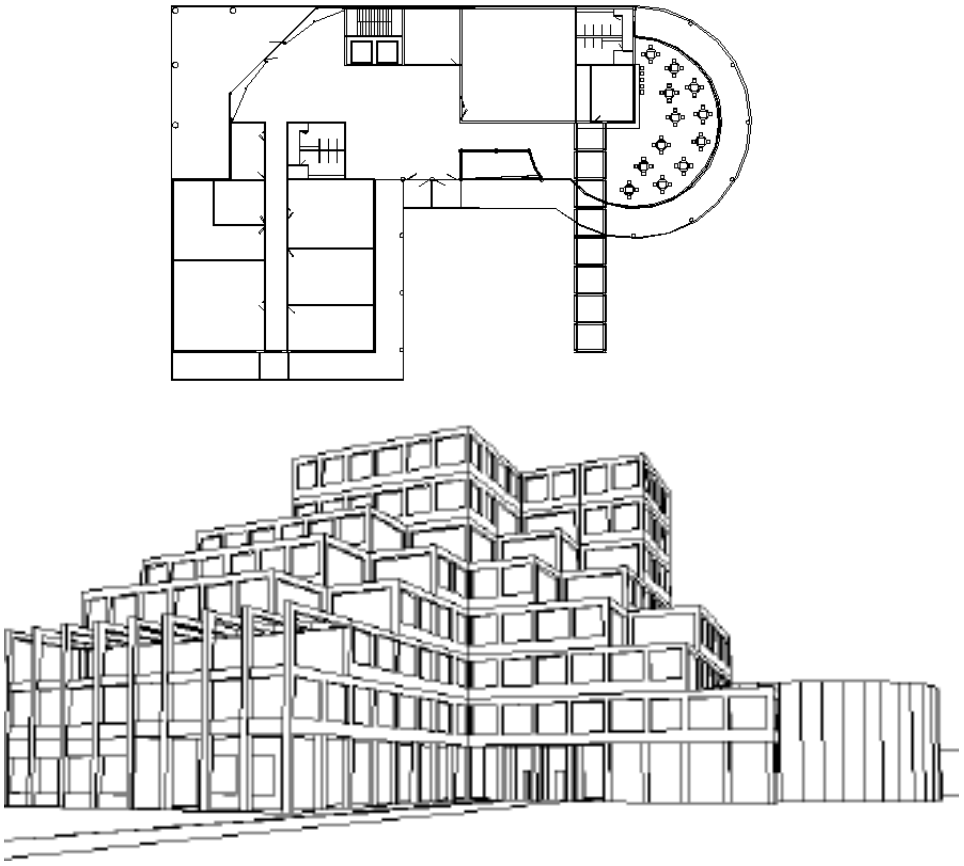

Figure 11. One of the buildings designed by architecture students (Sook Woo Park).

employed by architects. The built-in role reversal of the courses, where both groups were at the same time designers and clients, made each student appreciate the difficulties and positions of the other party.

Perhaps the success of the courses is due to the benevolent introduction of methods and tools. We did not attempt to revamp the entire traditional process involved in learning either architecture or computer science. That is, we did not attempt to make architects of the computer scientists, nor computer scientists of the architects. Neither did we force them into a mode of collaboration that is radically different from the practices they have been used to. Instead, we have provided them with an opportunity and a forum for collaboration, with a few structural guides and frameworks.

The collaboration between the two courses started out very well. The architects listened carefully to the requests of the clients and seemed to understand quickly what this building program was all about. The computer science students quickly grasped the initial phases of the architectural process leading to conceptual designs, and understood which aspects of this task lend themselves well for the introduction of computer-aided tools.

The tool-development followed a fundamental insight learned in the VLSI-CAD revolution of the 1980s: useful tools will emerge from the bottom-up. High-level, conceptual design is the domain that is least in need of better tools, because architects are good at it, enjoy doing it, and are not willing to give away this very personal and satisfying task to a machine. Many mundane and tedious bookkeeping tasks, however, can be readily relegated to the computer. For example, making sure that all required spaces have been accommodated in the emerging design and are allocated the specified floor areas, are such tasks, as are the tasks of ensuring that desired adjacencies have been observed, and that vertical connections (elevators, stairs) are compatible from floor to floor.

While the CS students were developing simple skeletal prototypes of these CAD tools, the architects were developing their designs, experimenting with many different massings, different forms, and different kinds of facades. In this process, the exact functional specifications seemed to get lost, since no CAD tools existed yet that could have kept track of the requirements automatically. When the 'final' conceptual designs were presented, there were substantial differences with respect to the specified floor areas, and the desired adjacencies were often grossly violated. Since the tools developed by the computer science students were not yet available, these violations had to be located by tedious examination of the floorplans, then corrected by the architects, or negotiated with the clients. Clearly, tools are needed to keep track of these issues in a way that detracts the architects the least from their creative conceptual design activity; they have plenty of other issues to be concerned about, such as esthetic issues and the relationship of the emerging design to the site and its surroundings-issues in which CAD tools can be much less helpful.

A cohesive suite of design tools was defined, which would support, rather than take over, the conceptual design process. These tools would reduce the tedium of keeping track of a detailed and extensive set of floor area specifications for the various functions in the building and of the desired adjacencies between the various spaces. They would allow the gradual development of a conceptual set of spaces, first expressed as a bubble diagram, into a set of schematic floor plans integrated into floor outlines that result from slicing tentative massing models at the desired heights. These floor plans could then be converted into tentative floor layouts with thick walls, containing automatically generated token door and window openings. These layouts would then be extruded into a consistent solid model of the shell of a building, which in turn can be explored with a real-time interactive walkthrough program running on a Silicon Graphics workstation.

The repeated conceptual re-design activity went on much too long. For most of the

designers, the conceptual design only settled about a month before the end of the course, leaving just enough time to document the design and to create the computer-based models. There was no time left to do much of any design development or to deal with the interior design of some selected spaces. In subsequent course offerings, a hard limit on the time for conceptual design will be imposed, particularly since we also want to include a group of structural engineers in the process. Hopefully, the CAD tools developed in this course will make this possible.

In the last couple of weeks, suddenly things started to come together in both courses. In the CS course, a well-connected suite of tools had been developed, that would indeed allow a designer to develop a building program from an automatically generated bubble diagram into a schematic floorplan and then to extrude it and visit the empty building shell with the Berkeley WALKTHRU program.

On the architectural side, three realistic conceptual designs for this rather complex building program were completed. All three buildings look like designs that could, potentially, be constructed, and they seem to interpret the specifications sufficiently well so that they could indeed work as a home for the Computer Science Division.

Conclusions

Overall, we think the direction plotted by these joint courses holds much promise for the future. The combination has provided added value to both parties involved, and generated some useful tools which can be reused in future courses. We plan to repeat this experiment, perhaps even on a broader basis, by including the participation of other professionals, such as civil engineers and construction managers.

The architecture students have learned that they share the development of a building design with their clients, who often have quite strong opinions that may enhance or interfere with the architect's own ideas. The computer science students were introduced to a design process that is very different from the one they were used to. They too have found out that their tools may be held to different standards of evaluation once the users begin to apply them in their own practices.

The Architecture and the EECS departments at U.C. Berkeley have a long-standing tradition of combining research and education with positive results in both domains. For example, the interdisciplinary collaboration of researchers in integrated circuits and in computer science lead, during the 1980s, to the emergence of relatively simple yet

powerful CAD tools for VLSI design, many of which, after some refinement and embellishment, have become commercial products or even industry standards. In the next 10 years, the combination of architecture and computer science may well prove to be an equally fertile breeding ground for new CAD tools and for a revolution of the architectural design process. A combination of courses in architectural design and in CAD tool development seems to be an excellent environment in which experiments can be run and creative ideas can be explored and tested quickly.

Acknowledgments

The authors express their gratitude to the students who endured the experimental nature of the course, and without whose support and enthusiasm the experiment could not have succeeded. These students include David Bacher, Richard Bukowski, Laura Downs, Randy Keller, Rick Lewis, John Sefler, and Maryann Simmons from the Computer Science Department, and Ivan Azerbegi, Francesco Nerici, and Sook Woo Park from the Department of Architecture. The authors also wish to thank Gustavo Llavaneras, who supported the entire process as a teaching assistant, and taught the architects how to use the numerous CAD tools they were required to use. We also wish to acknowledge the lessons learned from a parallel course we collaborated on at Stanford University, which was supported by the National Science Foundation's Synthesis Coalition.

REFERENCES

Chen N., T. Kvan, J. Wojtowicz, D. van Bakergem, T. Casaus, J. Davidson, J. Fargas, K. Hubbell, W. Mitchell, T. Nagakura, P. Papazian, 'Place, Time an the Virtual Design Studio,' ACADIA '94 (1994).

Cuff, D., Architecture: The Story of Practice, MIT Press, Cambridge, Massachusetts (1991).

Fruchter R. and H. Krawinkler (1995). 'A/E/C Teamwork,' ASCE Second Congress on Computing in Civil Engineering, Atlanta, GA.

Funkhouser T. A. and C. H. Sequin, 'Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments,' ACM SIGGRAPH'92, Anaheim, and Computer Graphics 27(2), (1993): 247-254.

Kalay Y.E. and G. Carrara, 'A Performance-Based Paradigm of Design,' IFIP WG5.2 Second Workshop On Formal Design Methods for CAD (J.S. Gero & F. Sudweeks, eds), Mexico City, Mexico (1995).

Khedro T., M.R. Genesereth, and P. Teicholz, 'FCDA: A Framework for Collaborative Distributed Multidisciplinary Design,' Workshop on AI in Collaborative Design, 11th National Conference on AI, Washington DC, (1993).

Larson M.S. 'Emblem and Exception: The Historical Definition of the Architects Role.' Professionals and Urban Form (J. Blau et al eds.), State University of New York Press, Albany, (1983):49-86.

Novitski, B.J., 'Architect-Client Design Collaboration,' Architecture, June (1994):131-134.

Shibley R.G. and L.H. Schneekloth 'Risking Collaboration: Professional Dilemmas in Evaluation and Design.' Journal of Architecture and Planning Research 5(4), (1988):304-320.