BETWEEN WORLDS: VISIONS AND VIEWS FOR THE FUTURE OF CAD

ELLEN YI-LUEN DO AND MARK D. GROSS
Design Machine Group, University of Washington, Seattle 98195

Researchers in computer aided architectural design are in many ways "between worlds." To become an expert, a student in computer-aided architectural design must master (at least) two core competencies: design and computing. The cultures of these competencies are widely, wildly, different. Despite some tensions that arise from this odd juxtaposition, this area of research also offers tremendous opportunities at the various intersections of computation and architectural design.

1. Introduction

It has been almost forty years since the publication of Herbert Simon's influential chapter "the Science of Design" in *Sciences of the Artificial*. It is an appropriate time to examine critically the field of computer aided architectural design, as Professor Flemming has done in his trenchant article in this volume (Flemming 2004). For the most part we agree with Flemming's analysis—its successes notwithstanding, our field indeed remains in a pre-paradigmatic state. We too have been dismayed at many of the same phenomena that Flemming, Maver (Maver 1995), and others have observed.—tremendous variation in the quality of published work, the apparent lack of effect that CAAD research has had on tools in the marketplace, and the tendency of each generation of CAAD researchers who are largely unaware of the work that has gone before, to re-invent old ideas in new technological contexts. We agree with Flemming's advice as well—as regards, for example, knowing work that has come before, clearly identifying the purpose of a project before beginning it, and the importance of evaluation in the development of software prototypes.

We would take stock in a somewhat different way, as though to reconsider the enterprise anew. After forty years of research and nearly as many years of development, it is fair to ask some questions. What is this field about? What do researchers need to know? What is the relationship of computer-aided design research to the discipline and practice of architecture? What are the opportunities for the future?

We have organized our stock-taking into three main sections. In the first section, we examine the expertise that one might expect a computational design researcher to have, a picture that in broad outline remains the same as in the early days of the discipline, but one that has expanded dramatically in its details. In the next section, we outline some of the tensions that researchers must answer to, in the institutional context of architecture, which we believe contribute to the difficulties that Flemming has identified. In section 4, we sketch some of the many areas of opportunity for computational design research in architecture.

Overall, we argue that our discipline inhabits an interesting, but peculiar, space that is between the worlds of science, engineering, and the arts, and (within the university) between the worlds of professional training and education within the discipline of architecture.

2. Knowing design, knowing computing

It is a truism that computational design researchers draw on knowledge of two quite different subjects: design, and computation, but an important fact nonetheless. There is a lot to know.

DESIGN

We can identify several ways of knowing design. We might call them the way of the designer, the way of the cognitive scientist, and the way of the theoretician or methodologist. Each of these approaches has its own methods, culture, and goals. Each contributes in a different way to our understanding of design.

The *practitioner* brings to the discipline of computer aided architectural design a first-hand understanding of the task and domain for which there is no substitute. On the whole we have found it easier to teach designers to program than to teach programmers to design, although there are certainly individuals who defy this generalization. Knowing the problems of designing firsthand is certainly an advantage when it comes to designing software to support the process. Although it is no substitute for careful analysis of the problem to be solved, the researcher who has also been trained as a designer can draw on his or her own experience to understand more deeply what computational support is needed, for what purpose, and in what modalities.

The *cognitive scientist* (and to some extent the ethnographer and anthropologist) studies the efforts of designers, both individuals and groups, and analyzes processes of design as carried out by human beings. This empirical mode of investigation observes designers at work, for example capturing and analyzing the protocols of designers at work. Eastman was among the first in our field to mine this vein (Eastman 1968), and others (Akin 1978; Schon 1984; Goldschmidt 1989; Cross, Christiaans et al. 1996) have followed. Although experienced designers may look at this work with some amusement (what, they wonder, is the point of studying such a skill empirically when it can simply be practiced?), empirical, cognitive science style studies can lead to important insights about how designers function individually and in teams. How do designers explore, define, elaborate, and solve or resolve problems? How do novice designers behave differently from experts? What is the role of various external representations? More generally, what cognitive and social constraints structure the way that human beings do design?

The *design theorist or methodologist* investigates the principles of design, irrespective of human cognitive capacities or predilections. The approaches here are many and we can list only a few. Simon's analysis of design from an operations research perspective (Simon 1969) examined the structure of design problem solving—applying the mathematics of game theory, observing the nature of search in "ill-defined" problem spaces and the need to account for the cost of exploration in the utility function that measures good and not-so-good designs. These ideas continue to reverberate in software for design in all the engineering disciplines. Rittel and Webber's analysis of the "dilemmas" of planning (Rittel and Webber 1969) (under the rubric of which we may surely list design) points to the aspects of "wicked problems" that make design an open challenge. Rather than calling for a problem-solving approach, though, as Simon does, Rittel proposed an approach of elaborating and deliberating design issues, which led to the idea of issue based information systems, (IBIS) in which a computer program is used to capture, structure, and manage an argument among stakeholders in a design or planning problem (Rittel and Kunz 1970). This approach is used today in requirements engineering, knowledge management, organizational memory. Alexander's *Notes on the Synthesis of Form* (Alexander 1966) proposed that design knowledge could be represented as nodes in a graph, which could then be partitioned and worked in tightly coupled clusters. Detailed studies of how people inhabit the built environment led to *A Pattern Language* (Alexander, Ishikawa et al. 1977), a (graph-structured) collection of design issues and partial solutions. The pattern language approach to design has been taken up in software engineering and in human-computer interaction (Gamma, Helm et al. 1995). Habraken's analysis of the problems of mass housing led him to propose a method for designing, the SAR method (Habraken, Boekholt et al. 1976) based on recognizing the hierarchies of form, territory, and control that are inherent in the built environment (Habraken 1998). Habraken's methodical approach, which (among other things) evaluates the capacity of a design alternative to support variation at a lower level, has been applied in architecture as well as urban design. Recently there has been interest in the software engineering community to understand the implication of Habraken's ideas for the design of complex software artifacts. Finally, Stiny's explorations of the mathematics of shape and the use of production systems to generate form (Stiny and Gips 1972) has led to a productive stream of research in generative CAD, some of which is represented in this volume.

COMPUTATION

There is also much a computational design researcher should know about computation. We might divide this territory into hardware, software, and the human-computer interface.

Apart from any *hardware* implementation, computing machines exist as mathematical entities, as a reading of Turing (Carpenter and Doran 1986), von Neumann (von Neumann 2000), and other early computer scientists makes clear. In practice however it is useful to understand what is going on inside the box. A computational design researcher must understand hardware architectures, for example the so-called von Neumann architecture that has dominated computing for the past fifty years, but also models such as cellular automata (also explored by von Neumann (von Neumann 1966)) and other distributed models that are becoming useful in pervasive computing. Basic knowledge of computer components (cpu, mass storage, input-output devices,) and network devices and familiarity with various protocols (serial i/o, for example, or TCP/IP) is also valuable for the computational design researcher who builds systems that run on real computer hardware. Those working in physical computing application areas will find an understanding of basic electronics as well as radio frequency (RF) and infrared (IR) communication helpful. A detailed understanding of hardware architectures is not necessary but most CAD researchers will find it helpful to know more or less what is going on inside the box.

Second, building *software* artifacts to demonstrate ideas has been a common practice since the earliest days of computer aided design research. Over the past forty years software environments have become rich and complex as they have become powerful, and the knowledge needed to build even a proof-of-concept working demonstration program has grown tremendously. Certainly, one needs to understand data structures and algorithms—for example, B-trees, linked lists, hash tables—and breadth-first, depth-first, and A* search—Horn clauses and unification—pattern matching and forward (or backward) chaining, and so on. One must not only understand these computational constructs conceptually, but one must know when to use them in designing software, and of course master the skills of debugging. One must also be able to write working computer code to implement them. This in turn entails knowing the elements, structure, and idiom of (at least) a programming language as well as the practical details of editing, compiling, and debugging software in a particular development environment. These days, to build software prototypes one must be familiar with large class libraries, their methods, and functionality.

That's just for starters. The serious computational design researcher may focus in a particular area that demands more specific computational expertise. For example, one who specializes in computer graphics and geometry must know the various algorithms and data structures involved in creating graphic representations, 3-D models, and animation. One who specializes in artificial intelligence aids to design decision making must know how case based reasoning works, genetic algorithms, expert systems, constraint programming, and so on. One who specifies in building product models, for example, must know about databases and query languages. There is a significant literature in each of these areas of specialization that must be surveyed, if not mastered, in order to apply its knowledge in computational design research.

Third, knowledge about *human-computer interaction (HCI)* rounds out the knowledge that a computational design researcher must have about computational hardware and software. Human-computer interaction has been a concern in computation since Sutherland's interactive pen based drawing program, Sketchpad (Sutherland 1963). With the development of bitmapped window displays at Xerox PARC in the 1970s and the adoption of Engelbart's idea of the mouse (among other ideas about using computers to augment human intellect!) (Engelbart and English 1968), HCI rose to prominence in the 1980s and 1990s, drawing on work in ergonomics and human factors, psychology, ethnography, and system-building efforts. One should be familiar with (among other things) methodologies of user testing, user-centered design (Norman and Draper 1986), Fitt's law, an understanding of Gibson's ecological theory of perception (Gibson 1950). HCI research is relevant to any computational design researchers who is exploring virtual reality, gestural interaction, pen based computing or sketch recognition, speech and natural language interaction, video, collaborative design, tangible user interfaces, or pervasive (ubiquitous) computing.

3. Tensions in computer aided architectural design research

From this somewhat daunting (yet incomplete and sketchily anecdotal!) list of expertise that the well-dressed computational design researcher may be expected to master it should be clear that researchers in computational design are awkwardly positioned between the "worlds" of various disciplines. This is, needless to say, also the case for any of the interdisciplinary fields, from bioinformatics to ethnomusicology. Although interdisciplinary work requires the researcher to be expert in a wide range of topics, it also offers rewards for applying insights from one domain to the other. Some of the tensions and opportunities are in the nature of any interdisciplinary work. (See (Kim 2002) for an interesting perspective on collaboration among computer scientists and biologists in computational biology). Others, though, are specific to the domains of architecture and computation. These are 'contextual' tensions that we have experienced, and observed others to experience, in conducting computational design research in schools of architecture. These tensions are peculiar to computational design research in architecture, as opposed, to the study of the same issues in, for example, mechanical engineering. Much of this derives from the larger tension between training for professional practice versus research and advancement within the discipline of architecture (Anderson 2001). Apart from scholarly work in the history, theory, and criticism of architecture, research is generally a foreign idea in a school of architecture, no matter how dearly the larger University may value it.

IMPLICIT VS. EXPLICIT KNOWING

The first tension is between the need to make knowledge and expertise explicit, in order to deal with it computationally; versus the culture of tacit (or implicit) knowing (Polanyi 1966) that characterizes architectural design. Bringing in the computer requires that we understand—and represent explicitly—knowledge and expertise that designers employ. This explicitness is the requirement of building computational tools. Indeed, one of the reasons to build computational models of designing, beside building better tools for practice, is to gain insight into design processes. Regardless of how we think this knowledge is best expressed (through rules, constraints, cases, geometry, etc.) most computational design researchers would agree that eliciting knowledge about design and designing is at the heart of the discipline.

Yet in schools of architecture expertise remains defiantly implicit. The culture of architectural design resists the idea that design processes can be examined, understood, and ultimately made computable. Many practicing designers (not only in architecture) maintain that subjecting design expertise to scrutiny is tantamount to destroying it; or at best, that it is a hopelessly complex, subjective, and quintessentially human activity. This resistance was certainly common in the early days of computer-assisted design, most frequently manifested in opposing the use (or even the concept) of computers in any but the most menial phases of architectural practice. Yet although drafting, modeling, animation, and presentation software are common in firms as well as in schools, the culture of research still clashes with the culture of architectural design as a primarily artistic endeavor.

RATIONALIST VS. ARTISTE

A second tension, related to this opposition between explicit and implicit knowledge, is between rationalism and creativity. Within an architecture school, computational design researchers are sometimes viewed as small-minded rationalists who seek to "reduce" design to a science. Yet somewhat incongruously, when we walk across the campus to collaborate with our colleagues in engineering and computer science, we are seen as artists, who can lend an air of creativity to the more serious engineering side of a project. This tension between what we really do, and how our colleagues in design on one hand and on the other in engineering—can lead to a kind of professional schizophrenia.

CAD TRAINING VS. EDUCATION FOR INNOVATION

In a school of architecture, there is a third tension: between professional students' need for training in using computer applications in design and the computational design researcher's focus on innovating future design systems. On one hand, professional students, by far the lion's share of students that architecture faculty serve, expect training in using CAD software. Quite reasonably and pragmatically, they have little interest in developing the design systems or software of the future. On the other hand computational design researchers seek to replace, not celebrate, the still impoverished tools of the present. Computational design research faculty members may know how to design better CAD applications, but they are typically less experienced users of the software than professional colleagues who employ the programs every day in their practice. Nonetheless, it is the CAD researchers on the faculty who are most often called to teach the basic "how-to-use" software application courses.

This again harks back to the more general contrast between the typical architecture school's main mission of educating the practitioners of tomorrow, and the university's research mission to create, not merely to pass along, disciplinary knowledge.

ARCHITECTURE VS. DESIGN RESEARCH IN GENERAL

A recurring theme in design research (computational or otherwise) has been the question whether there are overarching commonalities among the various design disciplines (as one might think from reading Simon), or whether each design domain must be considered uniquely. The interest that software engineers have shown in Alexander's Pattern Language (for example) would suggest that there is some generality to design. Yet on the other hand, it is devilishly difficult for domain-independent ideas about design to find traction. If we use architectural design examples to convey research about designing in general, willy nilly our work is viewed as domain-specific architectural design research. On the other hand, what business (or credibility) have we using examples from other domains?

One might worry less about this loss in generality, but for the facts of funding research in architectural design. For, as we note below, the professional architectural design community is not, ironically enough, in a position to sponsor computational design research, so we must look elsewhere for a constituency for our research.

Yet for all that, we do belong to wider communities, perhaps most importantly the community of researchers who study design (whether empirically or computationally) in other fields, for example in industrial, mechanical, civil engineering, and even software engineering. (Bringing this community together was the beauty of the ten-year (1985-1995) NSF-funded Engineering Design Research Center at Carnegie Mellon University). These colleagues are engaging many of the same questions that we do, although where we may deal with walls and windows, they deal with gears and pulleys; where we deal with pedestrian circulation, they deal with timing and power. Hence we find journals like *Design Studies, Research in Engineering Design,* and *Computer Aided Design* that publish work in design, independent of the specific discipline.

SOME IMPLICATIONS

These tensions have implications for *how* we conduct research as well as *what* research we conduct. Funding is a major concern, but also where students will come from, and where they will find work, and how those who choose to remain in academic positions will find their way to tenured positions.

Let us begin with funding. First, is funding necessary? In the sciences and engineering, there is simply no question that research costs money, but in architecture and the arts, students pay for their professional education. Most of our faculty colleagues in architecture do without funding—the professional students pay tuition for their education; small grants suffice for publications, project materials, and travel. However, that is professional education; this is research. Our colleagues in computer science and engineering, in contrast, are engaged in an endless effort to secure research funding. The money they secure for research pays for student tuition and stipends, hardware and software, and travel to conferences. For example, Carnegie Mellon's tuition (2004/5) is ~ $29K; a small stipend and summer salary brings the direct cost of each student to approximately $40K/year. The University charges approximately 50% overhead on this figure, so the grant must request $60K just for one student research assistant. Add equipment to support the project, a month of summer salary for the faculty supervisor or course release time to work on the grant, and even the most minimal proposal comes in at $100K per year. Where might this come from?

The USA provides no federal funding for research in architecture. Architecture is considered an art and is supported through the National Endowment for the Arts in amounts so small as to be useless for research in computer aided design. Computational design researchers seeking funding must apply to other agencies: for example, the National Science Foundation (NSF), the Department of Education (for education related projects), Department of Energy (for energy and sustainable design projects), on through the alphabet soup NIST, NIH, NASA, DARPA, and the other military agencies (ONR, ARI, AFRL). As these agencies are not chartered to support architectural design research, the researcher must identify topics of interest to the sponsoring agency that also advance the researcher's agenda. One might think this is hopeless, but it is not. It merely requires some creativity. For example, a project to model pedestrian behavior in cities might look to the Centers for Disease Control, who would be interested in reducing obesity by encouraging urban design that fosters walking. Thus, although NSF does not explicitly support research in architecture, recently awarded NSF grants include "Procedural Modeling of Urban Activity and Form", "Improving Design and Construction Engineering Education with Virtual Reality", and "Collaborative Research in Immersive Design Environments."

Although it's clearly possible to obtain funding for research in computational design, it's increasingly competitive (NSF funded 31% of the proposals it received in 2001; some programs were considerably more competitive; only 25% of engineering proposals were funded; and in some programs funding rates are as low as 10%). If a typical proposal supports two students for three years, then the researcher must expect to submit two or three proposals per year to maintain the two-student team. Apart from funding rate, programs at NSF correspond more or less to the academic disciplines in the university. Architecture researchers compete with colleagues in other departments for whom the funding is principally designated. Add that architecture faculty are not expected to find research sponsorship, so teaching loads tend to be higher than in engineering and the sciences. Therefore architecture faculty members have less time to work on obtaining research funding than their engineering and CS colleagues.

What about industry? After all, researchers in other areas benefit from the largesse of their corporate constituencies. Computer science design research may receive sponsorship from Microsoft, Intel, or Sony, for example, and mechanical engineering from the automobile and aerospace companies. Architecture, alas, has no Sugar Daddies. The professional firms that hire our graduates, even the largest ones, don't have large enough budgets or profit margins to sponsor research. They may benefit, in the long run, from research in computational design, but they cannot afford to pay for it. Construction companies could be interested in supporting work directly related to the construction process; yet here architecture competes with civil engineering. CAD companies would seem plausible sponsors too. For most technical tasks, though, why not hire a computer science graduate to do the software design and development and perhaps consult some domain experts to tailor the software to architecture, mechanical engineering? Especially if, as is the case for many CAD companies, architectural design represents a small fraction of the installed base, it is easy to see why they may overlook CAD research programs in schools of architecture, even if they do have some funds to spend at the university.

Finally, what happens to the graduates of computational research programs? Graduates might infiltrate industry and eventually turn the funding tide; if they could form a concentrated mass in the middle management of industry. It is difficult to obtain more than anecdotal information, as university graduate programs keep quite incomplete records of the career paths of their graduates. However, it seems that graduates of computational design research programs have done quite well. A considerable portion (both MS and PhD) pursue careers in teaching—some are international students who return to their home countries to help build academic programs in CAD. Among those who go on to industry, only a few remain in the AEC industry; others work in software development, in HCI, or in industry computer science or engineering research labs. What is clear is that, unlike mechanical engineering or computer science, there is no waiting industry ready to harvest all the students that computational design research programs produce; yet at the same time, the graduates of computational design programs find that their education has prepared them for a diverse range of interesting careers.

TENSIONS AND THE PRE-PARADIGM STATE OF CAAD RESEARCH

From this list of tensions, we can see that the "pre-paradigmatic" state of our field of research, as outlined by Professor Flemming, is part of a larger picture. In this picture, the specific context of academic architecture departments, and the domain of architecture within computational design research, provides a set of conflicts or tensions that frame our work. The tensions have to do with the institutional setting within the university and the broader professional culture of architecture; the opportunities for funding both from government and industry sponsors; and the career path of graduates of computer aided architectural design graduate programs. From this larger perspective we can see that although the problems within our research culture that Flemming alludes to are real (and assuredly must be addressed), they are part and parcel of a larger set of institutional challenges that we must also address if our field is to thrive.

4. Opportunities in computer-aided design research

Despite the tensions and challenges mentioned in the previous section, our position between worlds provides many compelling opportunities in computational design research. The past decades have seen many advances in computer hardware and software. Many problems that rested only on computing technologies have been solved. For example, the visualization and animation of proposed built environments, once an area of research in computer aided design and computer graphics, is not only a solved problem, it's a commercial product. Still there is much to do.

Today we find the field at a fascinating juncture. On one hand, many ideas dreamed up in research labs over the past decades have come to fruition. Information technology has forever changed the way designers design, and every design firm depends on computer aided tools to do its daily work. On the other hand, what has been done represents largely the low-hanging fruit. Many more difficult but potentially revolutionary projects, some long anticipated (Negroponte 1970) but tantalizingly not yet achieved, remain the subjects of ongoing research. Despite years of work on computational design support, most commercial CAD tools are still dumb, difficult to control, based largely on a view of design as drafting, a collection of disparate tools that exchange information poorly. A great deal of work remains to be done on the original problems of design computation. This includes basic design research, both empirical and theoretical; the design and implementation of innovative CAD systems based on modern software architectures. It includes the application of artificial intelligence, and building computer systems that reflect the cognitive characteristics of human designers and the interactions of collaborating members of design teams. It includes research on design space navigation (Chien and Flemming 2002) and management of design spaces (Woodbury and Burrow 2003). Embedding "intelligence" into the objects and elements of design representations, (Aly and Krishnamurti 2002), for example, will enhance the usefulness of next-generation CAD software. Generative methods, including geometry and shape algebras, will surely play an important role as well. In short, despite its tremendous impact on the industries of design, manufacturing, and construction, the CAD revolution has only begun.

At the intersection of architecture and computation there's more than the historically "pure" research in design. Add to the unresolved challenges of the CAD revolution (better computational design environments) a new set of research opportunities and challenges. The advent of faster and smaller processors, wireless communication, and inexpensive video cameras and sensor networks has ushered in a new era of computational design research. The work to be done focuses on physical interaction with computational processes in the built environment, work that will call for the expertise of architectural designers who are also virtuoso computational performers.

Human-computer interaction research over the past decade has resulted in new modalities for interacting with software that go beyond the traditional command-line and windows-menu-mouse interfaces that have dominated desktop computing. The application (and development) of new interaction modalities in design software—from understanding freehand sketching to gesture to speech and tangible interfaces to systems supporting design collaboration—is a rich and growing research area. It will have an enormous impact on tomorrow's design tools as well as every other software application.

Virtual environments—online places where people play, learn, do business, and participate in community life—is another area that needs computationally capable architects. Already many citizens spend time (and money) in online places that are designed and built without the benefit of architectural expertise. Mitchell has discussed the profound changes in community that derive from this development (Mitchell 2000) Others have used virtual environments to capture, represent, and provide enhanced information about architectural heritage (Maver 2001). Although the worlds of bits and bricks are different, architecture has much to offer the design of virtual environments.

Ubiquitous (pervasive, invisible, wearable, tangible) computing is another area that demands the expertise of computationally fluent architectural designers. (Weiser 1996) It is here that computation meets the built environment, and requires new and subtle understandings of how to integrate information technologies into places and things. The design of intelligent, responsive, cognitively assistive environments demands expertise in both architecture and computation. Today's examples are simple—technology augmented classrooms, assisted cognition living spaces for Alzheimer's patients, buildings that respond to climatic conditions to conserve energy. As the technologies of sensor networks and smart materials mature, architects will learn to design with an entirely new set of sensing, responsive, communicating, and dynamically changeable building components. Windows will sense light levels; rooms will determine who is inside and what they are doing; every surface will be potentially a display and rooms will sense human gestures; structural members will sense loads and perhaps vary their stiffness. In short: The built environment is the new locus of computation (McCullough 2004). Architects have a stake in shaping the changes that will inevitably come.

Another area that clamors for expertise in architectural computation is *design process management and production of the built environment*. The management of information flows throughout the design and construction lifecycle, including facility management and renovation, demands better computational support; an approach that demands the adoption and application of building product models (Eastman 1999). We have moved from passing around paper drawings to exchanging and marking up CAD files, but a great deal of work remains to be done in integrating design, construction, and facilities management.

Taking these ideas a step further, *computer-aided design and manufacturing* (CAD/CAM), robotic construction methods, and rapid 3-D prototyping are already producing buildings that could not have been conceived, let alone built, using traditional methods. In building the Guggenheim museum in Bilbao (and other similarly constructed buildings), architect Frank Gehry applied CAD/CAM techniques developed in the aerospace industry to building construction. Hundreds of titanium-coated steel panels, each with a unique three-dimensional curvature, were made from specifications sent directly from a CAD program to the manufacturer, who then shipped them to the site for assembly. Experimental buildings like these herald new ways to design and construct our built environment. It remains to be seen how these explorations in avant garde architecture will affect the production of ordinary environments, but increasingly, practioners are becoming aware of these technologies (Kolarevic 2003).

Finally, our ability to *simulate and visualize the performance of built environments* has reached the point where it can actually be useful to designers. Since the earliest days of computer aided architectural design, simulation of building performance in various dimensions has been a consistent theme of research. Each year the state of the art advances, though this aspect of computer aided architectural design has largely been unsung. More accurate and precise models of light, energy, sound, airflow, and occupancy are leading to better predictive ability, earlier in the design process (Kalay 1992). Low-cost, high-performance computer graphics also make it easier for architects to visualize these performance behaviors, and communicate the results of simulations to clients, stakeholders, and members of the design team. All this means that we can ask more "what-if" questions, understand the answers more easily, and rely better on simulation results in making design decisions. Although commercial CAD software has emphasized 3-D rendered imagery and animation, the application of these simulation and visualization technologies to performance prediction and evaluation has barely scratched the surface.

SUMMARY

In summary, research at the intersection of architecture and computation is a rich and challenging domain with many potential impacts on the production of the built environment. One can classify work in this domain into three categories:

- basic research in design
- computation in the built environment
- simulation and visualization

The first includes the 'traditional' topics of computational design research: basic research in design, including computational geometry, empirical studies of design process, design knowledge representation and artificial intelligence in design. Insofar as this category is about building computational environments for doing design, we also include relevant topics in human-computer interaction (HCI), such as calligraphic, gestural, tangible and multimodal interfaces, and computer-supported collaboration. In addition, we include here the management of design information in the structure of organizations producing the built environment.

In the second category of research are all the places where computation meets the built environment (physical or virtual). For example, we include ubiquitous computing—specifically the ways that computation embedded into built and manufactured environments (re)structures our experience of place. We include here also the design of habitable virtual environments, whether for entertainment, education, commerce, community, or democracy. We also include the use of rapid 3-D prototyping for design and CAD/CAM production processes in manufacturing and constructing the built environment.

The third category of research deals with applying the power of computational simulation and visualization to decision-making in a range of problems in the built environment—from energy, to acoustics, to pedestrian flow, to traffic, to crime and safety. Although research on these topics has been underway for several decades, faster processors, better graphics, and new interaction techniques make this an especially promising area for payoff over the next decade.

In summary we can certainly say this. Research in architecture and computation is broadly interdisciplinary, or (as we began) "much between worlds". It draws on expertise in fields from engineering to cognitive psychology and computer science to management. It brings this diverse range of expertise to bear on the design of the built environment, and on the design of computational tools for doing architecture.

5. Discussion

We have outlined some tensions and opportunities for research in computer aided architectural design. Further, we have argued that the tensions arise, in part, from the fact that research in general, and particularly computational design is awkwardly placed in schools of architecture. It is ironic that among the disciplines on a university campus, architecture is the one in which the largest fraction of the curriculum is devoted to teaching the practice of design. There is a dissonance between design research and design education for practice that stems from the structure of institutions rather than from fundamental differences.

In his chapter, *The Science of Design*, Simon outlined the elements of a university curriculum for the study of design. The former Head of Carnegie Mellon's School of Design, Richard Buchanan, describes the shift, beginning in the seventeenth-century university, toward the primacy of studying the natural sciences over the creation of artificial things. Design, at that time, was "part of the old learning." Today, Buchanan argues, echoing Simon, design is regaining its importance in the university and that there is "a new kind of university that is information today and that will emerge more clearly in the next century." Elaborating, he says, "Fragments of the human power or ability to create have, indeed, moved into universities in the past century of more, particularly in the form of engineering, "decision science," and most recently in the form of computer science. Furthermore, design education, too, has begun to find a place in a few universities—and in some of the leading research universities" (Buchanan 1999).

Research in computational design is at the center of this optimistic, but (we think) plausible, view of the future. Computation has changed fundamentally the way physics, mathematics, and biology is done. This has not happened yet in architecture, despite the widespread adoption of digital media in practice. Yet we believe that in architecture as well as in other design domains the more profound changes at what Allen Newell called, "the knowledge level," are, we think, inevitable. This calls for the kind of researchers that schools like CMU have been producing for years—people who combine the insights of disciplinary design practice with the rigor of research—who can contribute to the study of design from a cognitive and computational point of view, and who, in the long run, will further the development of tools and working methods for design practice.

References

Akin, O., 1978. How Do Architects Design. *Artificial Intelligence and Pattern Recognition in Computer Aided Design, IFIP*. E. J.-C. Latombe. New York, North-Holland Publishing, pp: 65-104.

Alexander, C., 1966. *Notes on the Synthesis of Form*. Cambridge, Harvard University Press.

Alexander, C., S. Ishikawa and M. Silverstein, 1977. *A pattern language : towns, buildings, construction*. New York, Oxford University Press.

Aly, S. and R. Krishnamurti, 2002. Can Doors and Windows Become Design Team Players? *Agents in Design 2002*. J. Gero and F. Brazier, Key Centre of Design Computing and Cognition, University of Sydney, pp: 3-22.

Anderson, S., 2001. The Profession and Discipline of Architecture: Practice and Education. *The discipline of architecture*. A. Piotrowski and J. W. Robinson. Minneapolis, University of Minnesota Press, pp: 292-305.

Buchanan, R., 1999. Design Research and the New Learning. *Design Issues* **17**(4): 3-23.

Carpenter, B. E. and R. W. Doran, Eds.,1986. *A.M. Turing's ACE report of 1946 and other papers*. Cambridge, Mass and Los Angeles, MIT Press and Tomash Publishers.

Chien, S.-F. and U. Flemming, 2002. Design space navigation in generative design systems. *Automation in Construction* **11**(1): 1-22.

Cross, N., H. Christiaans and K. Dorst, Eds.,1996. *Analyzing Design Activity*. New York, John Wiley & Sons.

Eastman, C. M., 1968. On the Analysis of Intuitive Design. *Emerging Methods in Environmental Design and Planning*. G. T. Moore. Cambridge, MIT Press, pp: 21-37.

Eastman, C. M., 1999. *Building Product Models*, CRC Press.

Engelbart, D. C. and W. K. English, 1968. A research center for augmenting human intellect. *AFIPS Conference Proceedings of the 1968 Fall Joint Computer Conference*. San Francisco. **33**, pp: 395-410.

Flemming, U., 2004. *Computational Design Research: Looking Back, Looking Forward*. Pre-proceedings, Generative Computer Aided Design Conference, Carnegie Mellon University.

Gamma, E., R. Helm, R. Johnson and J. Vlissides, 1995. *Design patterns : elements of reusable object-oriented software*. Reading, MA, Addison Wesley.

Gibson, J. J., 1950. *The perception of the visual world*. Boston, Houghton Mifflin.

Goldschmidt, G., 1989. Problem Representation versus Domain of Solution in Architectural Design Teaching. *Journal of Architectural and Planning Research* **6**(3): 204-215.

Habraken, N. J., 1998. *The structure of the ordinary : form and control in the built environment*. Cambridge, MA, MIT Press.

Habraken, N. J., J. T. Boekholt, A. P. Thijssen and P. Dinjens, 1976. *Variations - The Systematic Design of Supports*. Cambridge, MA, MIT Press.

Kalay, Y. E., Ed.,1992. *Evaluating and Predicting Design Performance*. New York, Wiley.

Kim, J., 2002. Computers Are from Mars, Organisms Are from Venus. *IEEE Computer*: 25-32.

Kolarevic, B., Ed.,2003. *Architecture in the Digital Age*. London and New York, Spon.

Maver, T. W., 1995. CAAD's Seven Deadly Sins. *Sixth International Conference on Computer-Aided Architectural Design Futures*. M. Tan. Singapore, pp: 21-22.

Maver, T. W., 2001. Virtual Heritage:reconstructing the Past: Reconfiguring the Future,. *Enhanced Realities: Augmented and Unplugged; Proceedings of 7th International Conference on Virtual Systems and Multi-media*. Berkeley, pp: 168-176.

McCullough, M., 2004. *Digital Ground : Architecture, Pervasive Computing, and Environmental Knowing*. Cambridge, MA, MIT Press.

Mitchell, W. J., 2000. *e-topia*. Cambridge, MA, MIT Press.

Negroponte, N., 1970. *The architecture machine; toward a more human environment*. Cambridge, Mass., MIT Press.

Norman, D. A. and S. W. Draper, Eds.,1986. *User centered system design : new perspectives on human-computer interaction / edited by*. Hillsdale, N.J., Lawrence Erlbaum Associates.

Polanyi, M., 1966. *The tacit dimension*. Garden City, N.Y.,, Doubleday.

Rittel, H. J. and M. M. Webber, 1969. Dilemmas in a General Theory of Planning. *Policy Sciences* **4**(1973): 155-169.

Rittel, W. and W. Kunz, 1970. Issues as elements of information systems, Center for Planning and Development Research, University of California, Berkeley.

Schon, D., 1984. Problems, frames and perspectives on designing. *Design Studies* **5**.

Simon, H., 1969. *Sciences of the Artificial*. Cambridge, MA, MIT Press.

Stiny, G. and J. Gips, 1972. Shape Grammars and the Generative Specification of Painting and Sculpture. *Proceedings of IFIP Congress71*. C. V. Freiman. Amsterdam, North-Holland, pp: 1460-1465.

Sutherland, I., 1963. Sketchpad: A Man-Machine Graphical Communication System. *Proceedings of the 1963 Spring Joint Computer Conferenc*. Baltimore MD, Spartan Books, pp: 329-346.

von Neumann, J., 1966. *Theory of self-reproducing automata.* Urbana, University of Illinois Press.

von Neumann, J., 2000. *The computer and the brain*. New Haven, Yale University Press.

Weiser, M., 1996. Open House. *ITP Review 2.0*
http://www.ubiq.com/hypertext/weiser/wholehouse.doc.

Woodbury, R. F. and A. L. Burrow, 2003. Notes on the Structure of Design Space. *International Journal of Architectural Computing* **1**(4).