Simultaneous Translation in Design: the Role of Computer Programming in Architectural Education

Megan Yakeley

In this paper it is proposed that architectural design involves simultaneous translation between several different languages and their corresponding systems of notation. The process of educating architects involves teaching fluency in these systems both separately and together. To improve pedagogical efficiency the physical manifestation of the languages – the graphical product – should be separated from the continuous expression of ideas in these languages – the conversational process. Digital media offer the opportunity to learn the process of translation between these systems, and thus form a strong foundation for the ability to design. Here a course taught at MIT by the author is described whose central theme is the development of design process through the use of the intermediary system of notation of a procedural programming language.

Keywords: Architectural design education, emergent rules, systems of notation, grounded theory.

Introduction

The process of designing in architecture involves the ability to converse in both verbal and graphical languages. Prior to the completion of the building, the architectural implications of these languages of representation must be understood and exploited. The architect, when she is designing, must use the systems of notation involved with each of these languages to produce “objects to think with” (Papert, 1980) – representations of the design at each stage must be used as the basis for the next design thought to be explored. To do so effectively, she must learn to interconnect the languages involved and move freely between their respective systems of notation. Architectural education traditionally promotes this ability through a studio setting in which students practice the process of designing through projects proposed by the studio professor. There is an assumption that by designing, the student will learn the skills needed to design. The inherent problems such a Catch-22 situation have previously been recognised (Schön, 1985, Schön, 1988a), but are accepted as an inevitable initial condition that will eventually be overcome through practice and repetition. The thesis of this paper is that digital media now offer the opportunity to circumvent the mutual dependency of the requirements of learning to design. This opportunity is being exploited in a course currently offered in the Department of Architecture at MIT, taught by the author. The aim of the course is to provide a foundation for design thinking by teaching the student to move freely between different systems of notation. It uses a graphical programming environment to do so. The end result is the development of a cognitive strategy that forms the basis for architectural design, with or without digital intervention.
Utilising Design Research for Education

There has been much research in architecture and design in the latter half of this century, much of it under the heading of “what is design?” (Archer, 1963–4, Cooke, 1975). Attempts have been made to define universal criteria for the criticism of architecture, with little success (Alexander, 1965, Silver, 1967). Often this research has had the ultimate goal of automating the evaluation of designs, or optimising their creation (Alexander et al., 1966, Galle & Kovacs, 1992a, Mitchell et al., 1992, Negroponte, 1967, Whitehead & Eldars, 1964). There has been little success in such automatisation, principally because it has proved impossible to predetermine every eventuality in such a complex discipline as architecture (Winograd & Flores, 1986). Other researchers have turned away from defining design, and instead looked at the way in which architects work—asking not what is design, but “what is designing?” (Akin, 1984, Davies, 1987, Lawson, 1980, Lloyd & Scott, 1994). They have attempted to understand how architects are able to make decisions when they too are unable to predetermine all possible complications and qualities of the current project. Each design project has qualities that are unique and cannot be neither predicted nor replicated. The architect must search out answers to these qualities to proceed effectively with the design project. There is a constant learning process involved in designing (Guarra, 1974). Architects ask questions (Rowe, 1987), look for answers, and use that information to guide the design process.

Although there has been little success in digitally emulating the learning process of designing, there has been consensus upon some of the cognitive strategies employed by architects in their work. It is these strategies that are encouraged and promoted in schools of architecture. The architecture student is motivated by the professor to continuously evaluate what has already been achieved, and to find ways of looking at the problem in a new light (Berkeley, 1968). Often the professor’s role is to encourage the student to explore her ideas in more depth. She is encouraged to proceed with the development of the design solution despite the absence of a complete problem definition – problem definition and solution develop together and culminate in the finished building (Beheshti, 1993, Portillo & Dohr, 1994, Schön, 1985). The student is encouraged after initial exploration of many ideas to find one possibility – the primary generator (Darke, 1979, Lloyd & Scott, 1995) – that has potential, and to explore it with consistency and depth.

All of these strategies, and more, are used by architects in their design work and taught to students through practice in the design studio. All of them are developed through a continuous series of sketches, models and other external representations of the current state of the architect’s design thinking. The novice must learn to think in design terms, an unknown process, and simultaneously express those thoughts on paper in a graphical language, which is itself uncomfortable and unfamiliar.

External Expression of Internal Ideas

Swimming in the confusion faced by all students of architecture, the novice student looks for the most concrete aspect of the design education process – the creation of the drawings and models used in designing. An experienced architect understands the multiple values of these external artifacts, both as means for communication to clients and others involved in the project, and as external renditions of internal thoughts to be utilised in the personal inquisitional conversation of design (Goldschmidt, 1994, Schön & Wiggins, 1992, Schön, 1983). But to the novice, they are primarily the means by which their projects are judged, and as such have a value that often inhibits their use in any ongoing personal design conversation of inquiry. The student focuses her attention on the creation of these products of design, and in so doing their true pedagogical significance is obscured.

The creation of such external expression of
internal ideas in graphical form, is the basis of the design process (Galle & Kovacs, 1992b). Often these thoughts and ideas are expressed in a verbal language, but they must be translated into a graphical language that in turn must represent an architectural one. The architect must be conversant with each of these languages; she must explain the architectural significance of an idea in verbal or graphical notation with equal ease. The architectural student in her design studio must practise both the creation of the products of design— the drawings and models that represent her thoughts – and the process of translation between verbal and graphical languages with their corresponding architectural implications. An inability to create the products of design will inhibit the process of translation and ultimately the design process itself. Often the beginning student must concentrate on their creation and has little time to face the difficulties of simultaneous translation. It becomes easier to concentrate on their physical manifestation than face the difficulties of simultaneous translation. However, such concentration invests the products of design with a personal significance that inhibits the student from learning to be at ease with the translation needed in design thinking. These representations become “objects to look at” rather than “objects to think with”.

**Developing a Personal System of Notation for Design**

The key to architectural designing, and with it architectural education thus lies in the development of the sketches and models, the external representations of the thinking process. The student must learn that these representations are to be translated into architectural or spatial implications, and be simultaneously verbally explicable. A line drawn on paper might imply a wall, a boundary, a change in surface texture or height. Whatever its current meaning, the student must understand the architectural implications of applying that particular meaning in that particular context. If it is to be a wall, how high, or thick is it? Is it a continuous height or does it vary? What is it made of? How might it be built? What is supporting it? If, on the other hand, the line implies a more ambiguous notion of boundary, what might that consist of?

All of these questions and more are possible from a single line on paper. To support such a wealth of potential interpretations, marks made are usually ambiguous, particularly at the beginning of the process. Mapping between graphical representation, verbal intent, and architectural meaning is never static. As a result, the marks in this continuously fluctuating system of notation are only temporarily distinguishable, and the experienced architect is able to keep the compliance class for each mark very large (Goodman, 1968). The marks are usually created with specific intent, but ambiguity plays a role in their interpretation. Any single element in the system may be replaced with a new, different one with the same meaning. The architectural meaning of each mark is also open to change. Reducing the size of the compliance class is a key aspect of further design development, culminating in final working drawings made in a universal unequivocal language of formal architectural notation, given to builders and others involved in the creation of built form. Ambiguity is not possible in construction. But in the beginning when nothing is certain, the role of ambiguity and interpretation is critical (Stiny, 1994). In the initial process of designing the architect must remain open to the possibilities of interpretation suggested by the work she has recently produced. She must remain open to the dialogue that occurs between her and her work. This dialogue must oscillate constantly between clarity of thought and ambiguity of graphical interpretation, something students have great difficulty with.

Many architecture students have difficulty in remaining clear about their design intentions whilst simultaneously permitting ambiguity to enter into the interpretation of the sketches being made. Flexibility of the notation system used by the experienced architect is therefore critical. For this reason the ambiguous nature of a pencil is almost always
preferred in early design phases to the fixed nature of lines drawn through digital means, whose endpoints are fixed and difficult to change (Stiny, 1994, Winograd & Flores, 1986). Architecture students are often encouraged to work in rough pencil sketches at the beginning of a design. But in the work of many students the ambiguity of the pencil can sometimes represent a lack of clarity of design idea rather than a potentially large compliance class to be explored. In some cases the student draws with minimal or no understanding of the implications of the marks made: the compliance class is empty. Design education aims to clarify the expression of the student’s individual design steps whilst encouraging the increase in size of compliance classes during each stage of the design process.

Designing thus involves the creation of an individual, highly personal, flexible and continuously fluctuating system of notation, in which the architect is able to state her ideas with fluency. The two methods of notation, graphical and verbal, are used together to describe the intended architectural meaning of the designer. Both are needed in early stages to temporarily reduce the ambiguity for purposes of explanation, but the mapping between any of the three is not fixed.

Therefore the key skill required at the beginning of a student’s architectural education is the ability to translate between these different systems. Learning to translate between verbal and graphical notation forms the foundation for the cyclical design process of graphical representation, verbal interpretation, and development of design idea ready for further graphical representation. This translation is the basis of each architect’s personal design process.

**Separating Conversational Process from Graphical Product**

Although much architectural research has examined the process of designing, the practice of architecture is naturally predominantly concerned with the creation of products – the completion of physical buildings. In a similar tradition, schools of architecture tend to focus predominantly on product, but in the academic context the products are representations rather than built form. When there is a focus upon process, it often takes the form of standardised techniques developed by the professor and taught to the student. Such courses usually result in products that are as similar in nature as the processes taught to achieve them, and are often difficult for the student to replicate in another setting. Although valuable, it is not clear whether such practices enable the student to develop an individual process of design thinking. Such an individual ability is generally assumed to be learned as a by-product of the studio system. But in the studio the student can be confused between design production and the process of design understanding. Studio work is judged primarily through the quality of the drawings – the products. It is assumed that beautifully crafted drawings and models reflect a similarly highly developed design understanding (Oxman, 1999). Process and product are not clearly differentiated in the curriculum, although the techniques involved in design production are usually taught through model making, drawing and CAD skills courses. But learning the mechanics of graphical notation in this manner – the syntax of various graphical languages – does not give the student a chance to develop the semantics of the language by linking the graphical notation to a verbal notation system with corresponding architectural meaning. The missing element is a course to develop the student’s ability to translate between different languages of design and converse fluently with the different systems of notation involved, and in so doing develop an understanding of the processes needed in designing. Instead, the student is expected in the design studio to simultaneously develop both the syntax and the semantics of a personal, fluctuating system of notation, a highly difficult task.

It is with this goal in mind that a course has been developed by the author in the Department of Architecture at MIT. The underlying aim of the course is to equip the students with the conscious cognitive
strategy of simultaneous translation between systems of notation. In so doing, the students are taught the process required to develop architectural ideas. The course uses a third system of notation, a graphically based computer scripting language as a bridge between the different systems of notation, giving the student the opportunity to understand their significance both individually and together. Now in its third year, ‘Digitally Mediated Design’ is a course aimed at students of architecture who are beginning the process of understanding what is required in designing. It aims to teach the link between design generation and exploration, and in so doing promotes design as a process of discovery, an inquisitional conversation. It does not attempt to impose one model of the design process on the students, nor does it impose one model of learning. Instead, it provides the framework and a set of skills that support and promote the efforts of the students to construct their own cognitive tools for the process of designing, and with it their own understanding of what it means to design. The framework is the digital intervention, and the skills include conscious simultaneous translation between different systems of notation. These skills can then be applied in the design studio. The course is intended as a foundation course for architectural understanding, not an alternative to design studio education. It does not attempt to provide answers to the question of what constitutes design thinking. It takes as its starting point the cognitive strategies employed by experienced architects and designers, and asks instead how such strategies might best be taught to novices. The message to the students is not “this is design” but “this is designing”.

Familiar Software, Unfamiliar Role

The computer program used to teach the course is a professionally orientated computer aided drafting (CAD) package, VectorWorks. Within VectorWorks is a procedural scripting language whose commands are predominantly recognisable English and whose output is screen-based graphical elements common to any CAD program – lines, rectangles, three dimensional solids and so on. For the software to form the bridge between verbal idea and graphical expression it is critical both that the written code is recognisable and builds upon the architecture student’s existing knowledge of CAD software, and that the output is instantly graphical (Papert, 1980). Many students assume that computer programming cannot help with the more qualitative aspects of design. This prejudice often inhibits students at the start of the course, and is further exacerbated by the steep gradient of the initial learning curve. English language and graphical elements similar to those produced by many CAD programs help ease the difficulties ahead.

Teaching this new system of notation is as liberating as it is dangerous, however. Once the student has overcome her fears of the process, and begins to see the qualitative aspects possible with such a system, she has little or no expectations that she should be able to use such a language. In the design studio, students are often inhibited from asking for assistance with graphical or model building skills, incorrectly assuming they are expected to have these abilities and not wishing to reveal ignorance. There is no such fear in the programming course, providing
the professor an opportunity to guide the student through the difficulties of conversing in several languages at once. For this reason it is critical that such a process of teaching architecture students to program be conducted by an instructor whose predominant speciality is the teaching of design, not the teaching of computer programming skills. Every conversation with the student must be weighted predominantly with design; the fundamentals of programming are subservient (Schön, 1988b).

The students begin the semester by selecting an idea for their design project. There has been considerable variation in content for these first projects. They have ranged from strongly design orientated ideas, such as developing aspects of the student’s concurrent studio project, to projects with a stronger computational component, such as a partial recreation of Conway’s Game of Life. In the first semester, the subject matter chosen is less important, since it is merely the vehicle by which the student is learning the new techniques. In the second semester, when she is fluent in the new language and methodology, the focus becomes the design project they are applied to.

Preliminary results from the current research study of the class, outlined below, are suggesting that there is a possible relationship between the students’ expectations of digital interventions in design, and the subject matter they choose to explore. Typically, those students who have a less well defined understanding of the role of digital media in design have chosen projects linked to more qualitative, creative aspects of their work. Those with a more confined notion of digital media have often gone to great lengths to find projects that link mathematical “computational” constructs with architectural objects. Part of the data to support this includes several students who begin with more computation based projects, and change part way into the semester to more design orientated projects. Often these students have given as reasons for the new project a change in understanding of the role of the computer, particularly computer programming, in their design thinking process. However, the current study is too small to reach any definite conclusions.

In the first weeks of the course, the students attend a weekly two hour lecture on the language used, VectorScript, given by the author. Here they are introduced to basic computational constructs and the structure of the language itself. However, the emphasis of the course is on the practical application of the language, so the students begin to program from the very beginning, in the separate, weekly, four hour recitation period. With little or no idea of what is possible, it is very difficult for the students to take the first step, particularly when their chosen project is design based. They are encouraged to consider the manner in which they might represent their first rough sketches on paper, and then to write a simple script that produces the same shapes and patterns on screen. Once the first script is written and executes successfully, the students are then encouraged to examine the graphical output, and use it to decide what the next step will be. The programming is thus taught as an incremental, ‘tinkering’ process described by Turkle and Papert as ‘soft’ (Turkle & Papert, 1990). Unlike the ‘hard’ process taught in many computer science courses, this incremental fiddling, or hacking, bears a striking resemblance to the cyclical nature of the design process, and is something many students find familiar. Students with extensive programming background, however, have occasionally found this informal and unstructured approach to programming somewhat alien. These students have instead preferred to formalise the process, describing from the beginning the final intended product and the steps they will be taking to achieve it. This focus on final ‘product’ places the emphasis on the computer programming, and not on the designs created through the programming process. The ‘product’ of such a process is the final script produced, with its associated, theoretical, future users. This is not the intent of the course, where the product is the development of the design idea through the combination of scripting code and interpreting graphical results. The code developed in the course is intended for personal use. The
difference between the two approaches is similar to the difference between the creation of an iconographic presentation drawing, and the use of rough sketches to develop an idea. The sketches are merely the vehicle for the idea, and are of use only to their author, and usually only at the time of their creation. They are “objects to think with.” The presentation drawing is usually independent of the design process, and is usually intended for communication to others. It is an “object to look at.”

The scripting language is used by the students to create a script whose function is to generate three dimensional form. Usually referred to as a generative system, it can be loosely defined as a set of rules that act in some manner upon a set of elements to produce a set of potential solutions (Henke, 1990, Krishnamurti & Giraud, 1986). A digital generative system might therefore be seen as an engine for the development and testing of a set of rules and a set of elements. In this context, the potential set of elements is the set of graphical primitives possible in VectorWorks. The rules are defined by the student and are developed within the context of the project she has chosen. Each rule emerges from the interpretation of the most recent code’s graphical output. Such emergent rules or decisions are an implicit part of any design process. The designer examines her sketches, and makes decisions based on her interpretation of what she sees. Each emergent rule may only apply once in that context at that moment, or it may be more generally applicable. Whatever its applicability, each of these emergent rules can subsequently be coded into the scripting language. New graphical output is created, and further rules emerge to be coded. The cyclical design process is recreated, but the cycle is expanded to include the additional step of writing code.

The possibility for variation is built into the generative system through the selective use of random numbers to encourage the process of exploration (Guarra, 1974) and to reduce the student’s personal investment in the graphical output. Her investment is in the code, not the resultant graphics, permitting a greater sense of detachment from the latter. The concentration upon the coded set of rules with restricted variation rather than a single design project more typical of the design studio promotes the process of exploration that uses the different systems of notation. It is the development of the rules, and their role in the translation process from one system of notation to another that is the fundamental aspect of this design process. The demands of programming in a procedural language, inherent in the limitations and rigorous structure of all computer programming environments, force the student to search her verbal intentions carefully, and to give deeper consideration to the link between the verbal and graphical languages. The student learns to write code with specific intent, but to interpret the results with a greater degree of ambiguity. For example, the development of a simple “if this then that” rule requires the programmer to state precisely the conditions under which the rule will execute, and the resultant algorithm that is to be carried out if the initial conditions are met. There is no room for ambiguity in computer programming (Bucciarelli, 1988). The graphical output of the code is produced as a result of running the script, and has not been directly drawn by the student designer. It can therefore be viewed with some detachment by the student. Where variation has been permitted in the code, unexpected results can occur that stimulate the student’s thinking and help promote new ideas. The detachment from the graphical product also promotes ambiguity in its interpretation, despite the rigidity of the digital system used in its creation. The student learns the role of ambiguity in design, and with it the importance of large compliance classes for marks made in early stages of designing. This is digital sketching. A further pedagogical advantage of the process is the architecture student decelerating her own design process, permitting a greater understanding of both its procedural nature, and the individual steps taken, whether potential or actual.
Digitally Mediated Design 1999: Changes in Design Thinking.

The Spring 1999 cohort of the course agreed to be the basis of an ongoing formal research study by the author into the effects of the processes they learn on their design process, the development of metacognitive skills (Nickerson et al., 1985), and their understanding of design. This study has begun to demonstrate that differences in the effect of the course on individuals depends on their experience, strengths and weaknesses in design. All of the students who have taken the course in the three years it has been taught at MIT have given indications of the course’s effects in other areas of their education. With all the students the key to such changes is an increased awareness of the role different languages play in the design process. Students arrive to study architecture fluent in a verbal language and its corresponding notation, albeit one that does not yet contain architectural concepts. Graphically they are less fluent but nevertheless usually articulate in representation skills using many different media. On the whole their representational skills are employed in the presentation of objects rather than concepts. Architectural education must teach the ability to represent abstract concepts in multiple languages and their corresponding systems of notation. To do this effectively it must empower the students by giving them greater control over their own cognitive tools. This course promotes a framework within which unique cognitive strategies for design can be developed. Each student finds a different aspect of his or her design working in which the course can assist. Students begin at different points on the spectrum of development, and proceed at different rates and to different levels of understanding, but preliminary results are showing that there are common benchmarks of achievement in understanding. They begin with some naivete of the computer, its possibilities, and its potential role in design. Data from the Spring 1999 course, primarily from the formal in-depth interviews, is indicating changes in the students’ understanding of all of these. More importantly, however, many of the students have expressed an awareness of a change in their ability to think like designers. In all cases the primary focus of their progression lies in the link between the different languages of design and the role the software used in the course plays in bridging or augmenting these languages. Ultimately the students expand their understanding of design from an exclusively studio based activity to include the processes developed in the course itself. Design is an activity of exploration, of learning, and of creation that requires an ability to express ideas in more than one language and to use those languages to explore ideas from more than one viewpoint. Software creation is a design process. The understanding this concept is as much the aim of the course as it is of architectural education in general. The difference lies in speed: the course offers this understanding within a year of an education system that takes anywhere from three and a half to seven years to complete.

The design of the study into the course draws upon current general education research methodology (Denzin, 1989, Glaser & Strauss, 1967, Lincoln & Guba, 1985, Strauss & Corbin, 1990), in particular the qualitative traditions of grounded theory. Primary data have come from in-depth conversational interviews at key stages in the first and second semesters of the class. The design of the study is emergent: the questions of each of the interviews have been developed as a result of data gathered in previous interviews. As with all grounded theory research, the questions are also based upon an intimate understanding of the aims and objectives of the course. These data are supported by audio recordings of all of the students’ presentations, and their individual tutorials. The data are being coded and analyzed. The aim of the study is to use the results of this analysis to demonstrate changes in the students’ understanding of design and the role of digital media in that process. The study is very small, and as with many similar educational research projects it has not been possible to create a control
group of students for comparison. Studies of this nature cannot produce results that can be generalised. They describe what has happened, and to whom, and suggest but do not define reasons for the results observed. They are important, however, as preliminary work in the development of more formal hypotheses that warrant further study. The course described here has in the past given results consistent enough to suggest a research study of this nature. It is hoped that the results from the study will indicate which aspects of this intervention into teaching the design process might benefit from further research.

Preliminary results from the study are promising, indicating changes in the students’ understanding that have occurred during the semester. One important change already observed with several students is their understanding of the role of computer programming designing. Computer programming in architecture is often seen by the students at the start of the course as a bridge between mathematical and graphical languages; verbal expressions of conceptual ideas are considered to be the realm of the design studio. This view is often exacerbated by previous formal programming training (Turkle & Papert, 1990). The understanding that computer programming can be used to express, and more importantly, to develop, conceptual qualitative ideas is fundamental to the success of the course.

The aim of the class for beginning design students is to teach them the different roles of verbal and graphical notation in architectural design development. For students who are relatively far down the road of their design education the course appears to offer an interesting interlude of exploration of new tools on established practices. More experienced students are already able to translate between verbal and graphical notation. The study has indicated that these students initially view programming not as a bridge to help them but as an abyss that hinders a process that is already familiar. Yet even for these students the demands of programming as a part of their design process appears from the study to cause them to consider each move more carefully. As a result they express a sense of greater control on their design thinking. It remains to be seen whether this is supported by evidence of greater exploration in design thinking in later studio work.

Although the course is open to everyone, the students upon whom it appears to have the greatest influence are those at the beginning of their design careers. The intermediate notation gives the novice student a chance to separate her verbal thinking from the graphical expression of those thoughts. This permits the consequences of each move in each of the different notations to be examined from an architectural viewpoint. The student appears to be able to more clearly separate and therefore better develop each language, and to bring them together using the techniques from this course as a bridge. The course has other less significant benefits, such as establishing a more creative role for computers in design than many assume possible. However, the most important aim of the course is to develop a sense of design thinking and understanding, and to act as a bridge between the two languages needed in this process. From this description it is possible to see that the role of the computer in the course is secondary to the development of the notations of design thinking.

References


Goodman, N. (1968) Languages of art; an approach to a theory of symbols (Indianapolis, Bobbs-Merrill).


Megan Yakeley
MIT
Mass, U.S.A.
yakeley@mit.edu