AUGMENTED INTELLIGENCE: INTERSECTIONS BETWEEN COMPUTER SCIENCE AND ARCHITECTURAL DESIGN

Ming Hu[1]
[1]University of Maryland, College Park, Maryland

ABSTRACT: In the 1960s, the "design pattern" developed by Christopher Alexander provided a promising starting point for reflection on architecture that is increasingly produced by digital and generative systems. Alexander's theory left a limited mark on the architectural design process, but it was radically adopted by computer science. Fifty years later, design algorithms and parametrics are a new field situated directly at the crossroads between computer science and architecture. We are in an era when digital intelligence can not only solve complex mathematical problems on the level of software, but also augment human intellect by generating architectural meaning. This paper compares computer science and architectural design, analyzing the fundamental differences and identifying the point of convergence.

KEYWORDS: Computer science, Augment Intelligence

1.0 CHRISTOPHER ALEXANDER AND COMPUTER-AIDED DESIGN

At one of the first international conferences on the relationship between architecture and computers, Christopher Alexander, then 30 years old, polemically confronted the pioneers of computer-aided design such as Steve A. Coons and Serge Chermayeff. In sharp contrast to their prevalent philosophy, which defined the computer in architecture largely as an intelligent drawing machine, Alexander represented the viewpoint of a structure-oriented, experimental design culture, whose primary focus was neither on forms of representation nor on a subsequent digitalization of finished design concepts. To Alexander, the computer's true strength was its extraordinary ability to calculate (Gleiniger, 2009).

A computer was nothing more to him than "a huge army of clerks, equipped with rule books, pencil and paper, all stupid and defined operations" (Alexander, 1965). He was vehemently opposed to the popular tendency to ascribe artificial intelligence to computers. The mainstream thought about computers among software developers and believers in computer-aided design is that the computers were able to quickly generate a mass of diverse plans and elevation alternatives from every conceivable perspective; in contrast, Alexander declared soberly: "At the moment, the computer can, in effect, show us only alternatives which we have already thought of. This is not a limitation in the computer. It is a limitation in our own ability to conceive, abstractly, large domains of significant alternatives" (Alexander 1965).

Alexander's insight into the limitation of computational design is not only incisive but also remarkable. He captured the potential and nature of the computer with regard to architecture application. Indeed, his perspective might illuminate some fundamental reasons that portend the integration of computers or other forms of digital tools into the design process. Even today, opinions diverge widely on whether a digital tool could, and would, be regarded as a design component and a fear that digitalization or computerization could jeopardize design creation. "Those that fear the computer itself are invariably those who regard design as an opportunity for personal expression. The computer is a threat to these people because it draws attention to the fact that most current intuitive design is nothing but an outpouring of personal secrets in elastic form" (Alexander, 1965).His major contribution was to view the computer's true application potential in architecture as largely at a structural level. His understanding of architecture and technology differed from that of the majority on visually oriented computer graphics at the beginning of the 1960s. This concept is still very relevant to today's context, as new computer tools are being created every year with the intention of helping architects and designers enlarge the boundaries of design creativity as well extend our capability to manipulate our own imagination—in another sense, to augment human intellect. Despite the original intention of new and advanced technologies, one can hear the lament for and agony over the architect's space or scope having been "encroached on" by fabricators and contractors. Other design-related disciplines have been progressively taking on a more leading and aggressive role by integrating those technologies into their practice in order to amplify their capabilities.

We clearly need to examine why we still fear automation and computerization, and on what philosophical level we, compared to related disciplines, are lagging behind in terms of broadening of our intellectual capability.

1.1 Algorithms and parameters: Computer science

Instead of examining closely related disciplines such as construction and engineering, this paper compares computer science and architectural design. The two disciplines were at a point of convergence in the 1960s but have followed

dramatically different development paths since then. Computer science has exploded in the last 50 years or so to become a critical component of societal infrastructure, while architectural design's development has stagnated. The following points illustrate the major differences between the two fields on a basic organizational and structural level.

1.10. Christopher Alexander's influence on computer programming
Christopher Alexander's pattern language has had a bigger impact on computer science than on architectural design. Even though his theory caused quite a stir when it was introduced, it was not integrated into architectural practice. However, 10 years after the publication of A Pattern Language, American computer scientists Ken Beck and Ward Cunningham applied Alexander's theory to problems in software engineering (Beck et al. 1987) That is how pattern language, originally conceived as a system for architectural design, was eventually applied to the world of computer science, which at that point was experiencing a paradigm shift due to what are known as object-oriented programming languages. (Gleiniger, 2009) This remarkable cross-disciplinary transfer of knowledge happened because software engineers gained an insight into the work of an architect that proved very useful. Ultimately, the software that was developed based on programming languages now helps to power Macintoshes, iPhones, games, etc.

Meanwhile, Alexander's theory drew criticism from the architecture profession for oversimplifying the design elements: the 253 design patterns he identified would not be sufficient for a built environment. More to the point, however, many architects and urban planners were fearful that that identifying the patterns could diminish the complexity of the built environment. (Mehaffy and Salingaros 2011)

1.11. User-driven vs. generator-driven
The design patterns for architecture concept development software development differ a great deal in dimensionality, the degree of abstraction, and the target group within their own field of interest. Alexander defined design patterns based on criteria that are important to users, no matter whether they are the residents in a house or pedestrians on a street. Ward Cunningham, a founder of "Object-oriented programming" whose design patterns were fundamentally conceived independent of the type and dimensions of the software and its specific application. Basically, they relate to the dependencies and the flow of information and control within the software, but not to individual users. In Wassim Jabi's book of Parametric Design of Architecture, he demonstrates the capacities of different digital tools to create user generated pattern. (Jabi 2003)

1.12. Scale differences
Alexander's theory and design patterns cover an enormous range, from urban planning to construction. The examined structures span multiple orders of magnitude, from kilometers to millimeters in size. Some of the patterns used in construction scale can be replicated in urban scale and vice versa. On the other hand, in computer science, an interesting self-similarity exists. A basic principle of computer science is the subdivision of complex, complete systems into compounded, interdependent layers, as, for example, in hardware, system software, and applications. (Gleiniger, 2009) The layers do not present any shift in dimension, but rather serve as the principle of order, and thus neatly separate individual responsibilities and make them accessible only via a defined interface. To certain extent, software or information exchange can be scaled up or down, at least while it is being programmed, without alteration of basic patterns, the software and program could naturally evolve into next state—unlike architecture, where methods change fundamentally with the respective different scale and dimensions.

2.0. BUILDING INFORMATION MODELING (BIM)
Both architecture and computer science deal with specific segments of reality, and both aim to fulfill an actual purpose: one in the built environment and the other in efficient programs. Standard computer-aided design plans are merely a collection of lines and symbols, as are plans on paper; even three-dimensional Computer-aided Design (CAD) models are simply geometric objects. As Rob Woodbury pointed out in his book Element of parametric design "Language is what we say; design and making is what we do. Computers are simply a new medium for this ancient enterprise" (Woodbury 2015). In a typical set of CAD file only the geometric representation of the design process is stored., beside the geometry, it does not explicitly explain the information concerning the means and method of building process and construction methods. In this type of representation lines, objects are merely symbols.

The conceptual underpinnings of the Building Information Modeling (BIM) system go back to the earliest days of computing. As early as 1962, Douglas C. Englebart provided an uncanny vision of the future architect in his paper "Augmenting Human Intellect" (Englebart 1962). He suggested object-based design, parametric manipulation, and a relational database—dreams that would become reality several years later: "The architect next begins to enter a series of specifications and data—a six-inch slab floor, twelve-inch concrete walls eight feet high within the excavation, and so on. (Englebart 1962) When he has finished, the revised scene appears on the screen. A structure is taking shape. He examines it, adjusts it... These lists grow into an ever more-detailed, interlinked structure, which represents the maturing thought behind the actual design."
BIM has played a paradigmatic role in changing the traditional way computer-aided design suited the design process.

The use of BIM is aimed at much more than just creating a model to see what the building should look like; BIM intends to create a model that contains all kinds of information and "meaning," from spaces and geometry, to cost, personnel, programming, quantities, specifications, suppliers, and other information types. To be well managed and qualitative, this information is contained in such a way that it is all related and built on the different kinds, in order to provide the best solution, enhance decision-making, improve production to the high level of quality, enhance prediction of building performance, be a major time saver, and control the budget in a safer environment within an organized, collaborative way of working. (Saravanan 2016) BIM can currently store the origins of the objects in the model, along with their properties and meaning. This is done by evaluating individual interdependencies between objects: if, for example, a door object knows that a wall object surrounds it, the door object is able to automatically adjust the thickness of its frame to match that of the thickness of the wall. The properties of the objects thus become parameters that can be manipulated and adjusted by other objects; the model becomes an associative, parametric model (Gleiniger 2009).

The next step in this development is to use stored information to simulate performance in order to minimize conflict and optimize interdependencies between objects and attributes. For instance, if the construction schedule depends mainly on the concrete superstructure's volume and construction time, then what could the best alternative options to concrete construction be?

CONCLUSION

Parametric models are basically algorithms. On the one hand, the demands on the designer change, because much of the information—until now implicitly hidden in plans and with room for interpretation—has to be explicitly formulated into specific parametric models. On the other hand, designing parametric models is subject to the same rules of complexity and computability as software engineering. The approach thus far of storing as much information as possible in one model does not generally lead to a solution, but rather to models that are unmanageably complex and, ultimately, of no value. For the architectural context, this implies that the actual art of creating a parametric model is to find the correct level of abstraction, which means entering only the most necessary information and correlations into the model and omitting as many unnecessary details as possible. In the computer science context, it is about code and program: this particular objective of the minimal model is only partially compatible with the need to react flexibly to changes in the design. The choice of parameters and their dependencies predefines a certain solution space within which the model can operate. Any further changes require an adjusted or completely rebuilt model. Wide-ranging decisions have to be made in advance, without full knowledge of the later application. As is true in software engineering, design patterns can also be a helpful instrument here in determining the correct balance between efficiency and flexibility.

Design algorithms and parametrics are a new field situated directly at the crossroads between computer science and architecture. A new professional discipline is now developing precisely in this area—namely, on the border between architecture and engineering, as well as in independent consulting firms. Regarding architecture, working with parametric models constitutes a departure from the traditional of using computer-aided design as a digital drawing table. One might even go so far as to describe this departure as a paradigm shift, comparable to the shift that occurred when object-oriented programing was first introduced to computer science as an influence from Christopher Alexander. These innovative methods open completely new possibilities, such as overcoming standardization and designing complex forms for facades that have to be assembled from thousands of single components. (Salingaros) A new type of design/construction firm has emerged, such as ShoP. As one of the partners, Gregg Pasquarelli, said: "We make instruction sets for other people to build buildings." (Polsky 2014)The use of technology (particularly digital technologies) and the blending of computer science into the decision-making enabled ShoP architects to become the last great generalist profession, with specialties in multiple areas. Literacy in computer science and deep imbedded parametric thinking, with aims and purpose, is the key to their success.

The Alexander-influenced software design principle developed in computer science was an effective tool that helped programmers solve precisely these structural tasks so they can focus more on important, content-related issues. Though the general concept and technology behind BIM is approaching its thirtieth anniversary, the industry has only begun to realize the potential benefits of building information models. As we reach the point at which a majority of buildings are being crafted digitally, an existing building marketplace where building materials and structural components can be bought and sold locally will emerge. Sustainable design practices reinforce an attitude of designing for disassembly, and a marketplace of these parts is essential. (Quirk 2012) Currently, virtual reality, augmented reality, cloud computing, generative design, and human-computer interaction continue to expand their influence on the development of new version of computer-aid design tools. Looking back, it is easier to realize that the present moment is an exciting time for designers and programmers in this evolving industry, and that a new professional discipline suited to the in-between might provide architectural design another chance to catch what we missed half a century ago.

REFERENCES

Alexander, Christopher, 1964. A Much Asked Question about Computers and Design: in Architecture and the Computer. Proceedings of the First Boston Architectural Center Conference, Boston, December 5, 1964, from the archive of the Department of Architectural Center Conference, Boston, December 5, 1964, from the archive of the Department of Architecture, MIT, 1964, p. 52.

Douglas.C. Engelbart, 1962. Augmenting Human Intellect: A Conceptual Framework. Prepared for Director of Information Sciences Air Force Office of Scientific Research, Washington D.C.

Gleiniger, Andrea. 2009. Pattern: Ornament, structure and behavior. Basel: Birkhäuser.

Jabi, Wassim. Parametric design for architecture. Laurence King Publ., 2013.

Kent Beck and Ward Cunningham, 1987. Using Pattern Language for Object-Oriented Programs. Technical Report No. CR-87-43, 1987.

Mehaffy, Michael. Salingaros, Nikos A. 2011. The Pattern Technology of Christopher Alexander. Point of View, Metroolis. Web.

Polsky,Sara. 2014. How ShoP became NYC's go-to megaproject architects. http://www.curbed.com/2014/5/28/10095052/how-shop-became-the-goto-architects-for-nycs-megaprojects

Quirk, Vanessa. 2012. A brief history of BIM. http://www.archdaily.com/302490/a-brief-history-of-bim

Salingaros, Nikos A. Some notes on Christopher Alexander. http://zeta.math.utsa.edu/~yxk833/Chris.text.html

Saravanan, S. 2016. A Study of Role of Building Information Modeling in Life Cycle Based Integrated Project Delivery Process.

Woodbury, Robert. Elements of parametric design. Taylor and Francis, 2010.