Abstract. The linkage between the worlds of Architecture, which involves the design and construction of the built environment, and Computer Technology, which involves practical applications of computation, still has a vast, as yet untapped potential. What if the implications of the linked term, ‘computer-architecture,’ are explored to reveal its full scope? This paper describes a unique method to analyze and code works of Architecture in a way that enables one to discover hidden computational structures in the works of Architecture. The case being made here is that the inherent structures of architecture may be computational structures as well.

1. Introduction

The term ‘computer architecture’ is often used in the computer industry and refers specifically to the design of computer systems, both hardware and software. Even Bill Gates, the head of Microsoft, prefers the title Chief Software Architect. This linkage between the worlds of Architecture, which involves the design and construction of the built environment, and Computer Technology, which involves practical applications of computation, still has a vast, as yet untapped potential. What if the implications of the linked term, ‘computer-architecture,’ are explored critically to reveal its full potential?

A work of architecture is created after an intense design process. The resultant architecture has embodied in it various formal structures (i.e., structures that articulate a particular form). The really interesting question is, are these formal structures, feasible computational structures as well? If the answer is yes, this will truly bring the world of Architecture into the world of computation! This project sets out as its main goal to discover and verify if the formal structures embodied in works of Architecture could serve as computational structures as well.
This project is the next in line of a long list of investigations completed by Mahalingam in the last decade linking the worlds of computation and architectural design. For his doctoral work Mahalingam successfully created an algorithm for the design of proscenium-type auditoriums. The algorithm was incorporated in object-oriented software for the design of proscenium-type auditoriums using the Smalltalk programming language and the VisualWorks software development environment. (Mahalingam, 1998, 2000). As a part of his doctoral investigation, Mahalingam also proposed a paradigm for the representation of architectural design entities as virtual computers (Mahalingam, 1997). This was a significant attempt to look at architectural entities as computational devices. In a subsequent investigation, a model was proposed for the parallel computational processing of load transfer in rectangular structural systems for architectural design (Mahalingam, 1999). A project was also completed where a programming language was proposed for architectural design with the complete Backus-Naur notation for the language (Mahalingam, 2000). In a more recent project, a new model was proposed for the sensor-based control of the propagation of sound in spatial enclosures based on an algorithmic model for sound propagation simulation developed earlier (Mahalingam, 1999). This project involved the modeling of the components involved as an elliptical graph called an optimaton (Mahalingam, 2005).

In a recent seminal paper, which has generated the main idea for this research project, a paradigm was presented for the representation of different aspects of architectural design using connections-based models (Mahalingam, 2003). The paradigm suggested a uniform representation of spatial layouts, circulatory systems, egress systems, structural systems and environmental control systems in architecture using three-dimensional networks or graphs. The argument was made that these three-dimensional networks or graphs reveal the architectonics underlying their composition, and by extension, could be the basis of computational frameworks. In this project, the author has simulated the behavior of a computational structure in the form of a virtual finite state machine (VFSM) that is based on works of physical architecture to see if the VFSM could be the basis of new computational tasks in architectural design such as the simulation of fire spread in a building, load transfer in structural systems, sound propagation in spatial enclosures, and heat transfer in buildings, to name a few.

2. Methodology

The way this was accomplished is as follows:
2.1. ANALYSIS

The first step was to analyze a work of Architecture (i.e., part of the built environment) so as to reveal its underlying systems, such as structural systems, circulation systems and arrangements of spaces. Three projects by the architect Frank Lloyd Wright from around the U.S.A. that were analyzed earlier by March and Steadman (1971) were selected by the author. Each of these works of Architecture had been analyzed to reveal the ‘invariant’ relationships in their arrangement of spaces. Other examples of some of systems that could have been included in the analysis are structural systems, circulation systems, egress systems, HVAC systems, plumbing systems, etc. These were not attempted in this initial implementation.

2.2. CODING

The next step was to code the spatial arrangement system as a diagram comprising nodes and links, i.e., as a graph. The spatial arrangement system uncovered in the analysis phase was coded as an adjacency graph comprising ‘nodes’ and ‘links.’

![Figure 1. Encoding of an architectural plan as a graph showing how different features are embedded hierarchically.](image)
2.3. VFSM GENERATION

The next step was to use the graph that was uncovered in the previous step to model a virtual finite state machine (VFSM). The graph was used as a template for the generation of a VFSM using commercial software (StateWORKS for VFSM simulations).
Figure 3. The state diagram of a virtual finite state machine (VFSM) based on the 3 architectural works by Frank Lloyd Wright that share the same adjacency graph for the spaces that they contain. The finite state machine is used to determine computationally if fire has spread from one space to another, given the occurrence of fire at the various locations.

2.1. SIMULATION

The next step was to simulate computations using the VFSM and see what computational structures could be derived from the works of Architecture. The VFSM was used to simulate the spread of fire in the buildings, a computational task in architectural design that could be mapped easily onto the VFSM.

2.1. TESTING

The last step was to test to see if the computational structures could be used to form the basis of new computer software for architectural design (i.e., the spread of fire in a building). The efficiency of the VFSM in performing the computational task attempted was demonstrated. The suitability of the VFSM for new computational tasks in traditional computation as well as
other computational tasks in architectural design will be explored in the future.

3. Implementation

The spatial arrangement of three works of architecture by the architect Frank Lloyd Wright which were analyzed earlier and coded as adjacency graphs were used in the implementation. Incidentally all three works had the same underlying adjacency graph. The software StateWORKs (Wagner et. al., 2006) was then used to generate a virtual finite state machine (VFSM) that was based on the adjacency graph of the spatial arrangement.

A particular computational implementation was then mapped onto the VFSM. This was a computation that would determine if fire spread to a particular space given the occurrence of a fire in another space. The nodes of the graph (the spaces) were each assigned a range for a flammability value. This flammability value was modeled as a ‘switchpoint’ that would switch on and off based on whether the fire in that space crossed the high or low threshold value. If the intensity of a fire in that space exceeded the flammability value’s high threshold then the space caught fire. Conditions were set for the fire to transmit from one space to another. This was modeled as state transition conditions in the VFSM. A system was then set up to input the intensity of a fire in each of the spaces. A simulation was then run, whereby one could input the intensity of a fire in each of the spaces using a numerical input dialog box and see if it spread to the other spaces, which was indicated in an output monitor that indicated that a fire had occurred in that space. The whole process of the spread of the fire was a computation of state transitions in the VFSM.

In a real world scenario, the system for the input of the intensity of the fires could be linked to a real digital input using a communication port in the computer, and the output signal that a fire had occurred could be used to activate an alarm using another communication port in the computer. This capability to link digital inputs and outputs to communication ports on the computer is inherent in the StateWORKs software system. This VFSM could effectively form the engine of a real fire alarm system in each of the buildings analyzed.

If one had to develop software for the prediction of fire spread in the architectural design by inputting flammability values for each of the spaces, starting fires of various intensities in the various spaces, and predicting where the fire would spread, then this VFSM could be used as an engine for the development of the software. The StateWORKs software system allows you to generate such software engines for runtime control systems with full control of I/O (input/output) such as WinStExec, StExec, LinuxExec and a diskless RTOS (real-time operating system) environment, which can be used
for software development using other IDEs (integrated development environments). The conditional transitions from state to state in the VFSM could also be used to model systems such as Bayesian networks that are based on the VFSM. The state transition conditions could then incorporate probabilistic triggers.

Also other computational systems, such as heat transfer from space to space, could also be mapped onto the same VFSM. Instead of the ‘flow’ of fire, the ‘flow’ of heat from space to space could be computed using the same VFSM. The conditional transitions in the computational ‘flow’ from space to space could be modeled based on the heat transfer properties between the spaces.
Figure 4. Screen shots of the VFSM runtime computation monitor in StateWORKS that monitors the Wrightian VFSM. The indicators in green show where the fires have occurred and the numerical values are the intensity of fires that have been mapped to the various spatial locations in the Wrightian houses.

4. Intellectual Merit of the Project

The intellectual merit of this project is that it makes a unique proposal to analyze and code works of Architecture in a way that enables one to discover hidden computational structures in the works of Architecture. It is hoped that the project will provide valuable insight into the architectural basis of computational structures.

During the process of architectural design, various formal structures (i.e., structures that articulate a particular form) are generated and integrated to define the design of a building. These formal structures determine the spatial layout of structural systems, circulation systems, egress systems, arrangement of spaces, HVAC systems, plumbing systems, etc. in a building. All these formal structures are integrated in the design process to create the design of a functional building. These formal structures satisfy many constraints and meet many performance criteria in different domains. As such, they are very complex design constructs. If these formal structures could be shown to be feasible computational structures as well, then the rigor and complexity of the architectural design process could be brought to bear on the design of software systems. If a particular formal structure derived from a work of Architecture is shown to be a computational
discovering computational structures in architecture

structure as well, then the methodology of the architectural design process that resulted in that formal structure could be studied as a viable software design process. This will bring the whole body of design methods used in the architectural design process into the world of software design. Conversely, the research process will also yield computational structures for the design of architectural entities, thereby enabling the creation of new kinds of computer-aided design systems in Architecture.

The broader impact of this research project will be to amplify the interdisciplinary relationship between Architecture and Computer Science and provide practical benefits such as the creation of new kinds of software for both traditional computational tasks and for architectural design. Though the methodology described in this project aims at discovering hidden computational structures in Architecture, it can be adapted to discover hidden computational structures in other fields such as Engineering and Biology, thereby enriching the field of computation.

5. Conclusion

The project described in this paper has successfully shown how you can take a formal structure from Architecture and convert it into a computational structure. It has also shown how this computational structure can be used as an engine to develop hardware and software systems for applications such as the monitoring of fire spread in a building. This is the proof of concept for discovering computational structures in architecture. The project still has to demonstrate that these computational structures, which are derived from works of Architecture, can be feasible computational structures for tasks in traditional computation. They hold the promise of serving as meta-computational structures for computational applications in architectural design, but have yet to be shown to enable other computational tasks such as sorting and searching, which are often considered benchmark tasks in Computer Science.

In his landmark book, Hillier presented the case that “space is the machine.” (Hillier, 1996) This book has a strong connection to this project. However, Hillier was specific in referring to his theory as a “configurational theory of architecture,” and not a “computational theory of architecture.” In a chapter devoted to the topic, he made the case for “non-discursive techniques,” that were neutral in the analysis of space and form, thereby aiming for a “universal” understanding and the development of an “internal” theory of architecture. Is Hillier’s machine a computer? If this is the case, the ‘configurations’ of architecture become viable ‘computational structures’ as well. This project reveals the intriguing possibility that this may be the case. As this project unfolds, more involved issues related to discovering
computational structures in architecture are bound to emerge, which need to be thoroughly investigated.

The results of this research project are intended to be used as the foundation for an interdisciplinary Honors seminar course at our university titled, “The Architecture of Software Systems.” This course will extend this inquiry and develop it further. The Honors program at our university is based on selective admission and attracts the best and brightest students in the university who have a natural inclination for interdisciplinary studies. Courses are typically taught by a team of two or more faculty members from different disciplines. Mahalingam and a faculty member from Computer Science intend to teach the Honors course together. Their cross-disciplinary collaboration on the subject will make them effective teaching colleagues. The course will stimulate motivated students to pursue and extend research ideas in this area of inquiry further by exposing them to the state-of-the-art in this field.

Acknowledgements

The developers of the StateWORKS development tool have opened up this new avenue for research in the field of computer-aided architectural design. StateWORKS allows researchers to study computational modeling problems in architecture by building and testing tractable solutions in the form of virtual finite state machines that can be implemented in both software and hardware.

References