ReAD: Representational Algorithmic Design

Renata Castelo-Branco
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa
Lisbon, Portugal
renata.castelo.branco@tecnico.ulisboa.pt

ABSTRACT

Algorithmic Design (AD) is a novel approach to architectural design based on the creation of form through algorithms. The inherent flexibility of AD encourages the exploration of a wider design space, the automation of design tasks, and design optimization, considerably reducing project costs. Nevertheless, current AD uses representation methods that radically differ from those used in architectural practice. This creates a barrier to the adoption of AD, thus, limiting the potential benefits. We propose to address this problem by coupling AD with complementary representation methods that are adequate for designing complex architectural projects and by supporting their synchronization in a dedicated design tool.

CCS CONCEPTS

• **Applied computing** → **Computer-aided design**; **Interactive learning environments**; • **Software and its engineering** → *General programming languages*; • **Human-centered computing** → *User interface design*; *Virtual reality*.

KEYWORDS

Algorithmic Design, Programming Paradigms, Program Visualization, Interaction Mechanisms

ACM Reference Format:
Renata Castelo-Branco. 2020. ReAD: Representational Algorithmic Design. In *Companion Proceedings of the 4th International Conference on the Art, Science, and Engineering of Programming (<Programming'20> Companion), March 23–26, 2020, Porto, Portugal.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3397537.3398478

1 ALGORITHMIC DESIGN

In the last few decades the architectural practice as been changing at an unprecedented rate [10]. One such change is Algorithmic Design (AD), a design method that defines the creation of architectural designs through algorithms [2]. Using AD, the architect does not create the building's digital model directly, but instead, creates the program that creates the model [17].

The inherent flexibility of AD not only encourages architects to explore a wider design space, but also supports the automation of design tasks, eases the integration of changes in later stages, and promotes design optimization, thus, motivating the creation of more sustainable buildings [15]. Figure 1 presents an example of an AD process for the case of the Astana National Library (ANL) project, a rather complex form that would be almost impossible to model using traditional design methods.

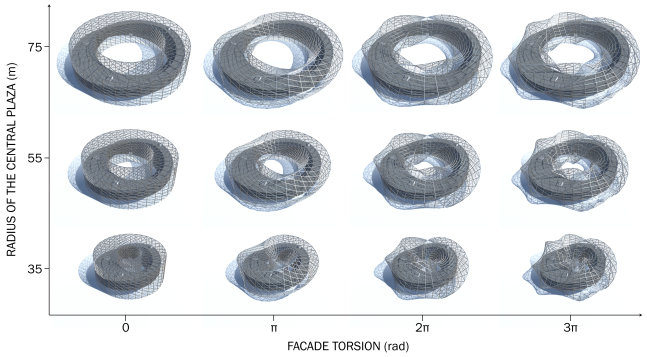

Figure 1: Variations of the ANL model produced by changing selected parameters in the code.

Despite the numerous advantages of AD, it relies on representation methods that radically differ from the ones currently used in the architectural practice. In order for architecture to truly benefit from AD, it is necessary to bind these different representation methods, closing the existing comprehension gap between the code and its behavior.

2 REPRESENTATIONAL AD

The Representational Algorithmic Design (ReAD) proposal aims to make AD more akin to the design process employed by architects, by removing the three main difficulties these practitioners find in the programming task: (1) the steep learning curve; (2) the difficulty in comprehending the program's structure and behavior; and (3) the unfamiliar interaction mechanisms.

2.1 Hybrid Programming

Given the graphical nature of this profession, visual programming environments [1] have become very popular in architecture. However, in most cases, as programs grow in complexity, they become hard to understand and navigate [6]. On the other hand, and despite being less visually attractive, the Textual Programming (TP) paradigm offers more scalability and portability [12, 14].

Our first strategy explored a hybrid programming paradigm, casting the advantages of the visual approach, while guaranteeing the scalability of TP. Our initial experiments [3] found that the system is most appealing for users already proficient in visual

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
<Programming'20> Companion, March 23–26, 2020, Porto, Portugal
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7507-8/20/03…$15.00
https://doi.org/10.1145/3397537.3398478

Figure 2: Live Coding in Virtual Reality.

programming who wish to transition to TP. However, as complexity increases, we find that those who can, prefer to rely entirely on TP since the visual features noticeably hinder performance.

2.2 Program Visualization

AD programs translate into 3D models and, regardless of one's programming abilities, it is always hard to relate textual descriptions to the 3D entities they generate with no further aid. Many Program Comprehension (PC) strategies that help developers understand complex software pieces [16] are inadequate for AD programs, since a program developed to describe an architectural design requires visual comprehension mechanisms [7, 13].

Our second strategy focuses on program visualization, i.e., the use of graphical representations to illustrate or explain programs [14]. Specifically, we propose integrating, within the program, the drawings architects produce when designing, and relating the annotations in them with identifiers in the program. We also plan on supporting traceability [13], i.e., establishing relationships between code pieces and model parts, and immediate feedback, i.e., quickly re-computing the model after changing the program, allowing designers to easily understand the program's behavior. Since architectural models tend to be computationally heavy, we are exploring game engines as a fast visualization alternative [4, 11].

2.3 Interaction Mechanisms

Modern digital design tools typically entail a mouse-based manipulation of geometry using a flat and unnatural visualization [9]. This explains why architectural design processes still heavily rely on the production of physical models [8]. The flexible nature of AD makes it impossible for architects to produce physical models for every instance of the design space. However, the farfetched design method presented by AD, combined with the design complexity it allows, demands better visualization mechanisms.

To that regard, ReAD's third strategy proposes the integration of Virtual Reality (VR) technology in the design process, for a more tactile interaction with models, along with scale control up to real size. Furthermore, in the context of AD, we can not only use VR for model visualization, but we can also transform the design process in VR into a real-time and interactive one, allowing architects to code their designs while immersed in them (see figure 2). Our preliminary research on this topic can be found in [4, 5].

3 CONCLUSION

The goal of this research is to provide architects with the methods and tools they need to fully benefit from AD. For that, we propose coupling AD with representation, visualization, and interaction mechanisms that appeal to architects and that are adequate for designing complex architectural projects. We expect ReAD to make AD not only an advanced architectonic representation method that allows for a flexible and comprehensible design exploration, but also a more accessible design process, allowing the industry to benefit from AD's potential for promoting synergy between design creativity and design optimization.

ACKNOWLEDGMENTS

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with references UIDB/50021/2020 and PTDC/ART-DAQ/31061/2017.

REFERENCES

[1] Margaret M. Burnett. 1999. *Visual Programming. Wiley Encyclopedia of Electrical and Electronics Engineering.* John Wiley & Sons, Inc., 275–283. https://doi.org/10.1002/047134608X.W1707
[2] Inês Caetano, Luís Santos, and António Leitão. 2020. Computational design in architecture: Defining parametric, generative, and algorithmic design. *Frontiers of Architectural Research* (2020). https://doi.org/10.1016/j.foar.2019.12.008
[3] Renata Castelo-Branco and António Leitão. 2020. Visual meets Textual: A Hybrid Programming Environment for Algorithmic Design. In *Proceedings of the 25th CAADRIA Conference.* Bangkok, Thailand.
[4] Renata Castelo-Branco, António Leitão, and Catarina Brás. 2020. Program Comprehension for Live Algorithmic Design in Virtual Reality. In *Companion Proceedings of the 4th <Programming> Conference).* ACM, New York, NY, USA, Porto, Portugal. https://doi.org/10.1145/3397537.3398475
[5] Renata Castelo-Branco, António Leitão, and Guilherme Santos. 2019. Immersive Algorithmic Design: Live Coding in Virtual Reality. In *Proceedings of the 37th eCAADe Conference*, Vol. 2. Porto, Portugal, 455–464.
[6] Gabriela Celani and Carlos Eduardo Verzola Vaz. 2012. CAD Scripting and Visual Programming Languages for Implementing Computational Design Concepts: A Comparison from a Pedagogical Point of View. *International Journal of Architectural Computing* 10, 1 (2012), 121–137. https://doi.org/10.1260/1478-0771.10.1.121
[7] Stephan Diehl. 2007. *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software.* Springer Berlin Heidelberg.
[8] Nick Dunn. 2014. *Architectural Modelmaking.* Laurence King Publishing.
[9] Enrico Gobbetti and Riccardo Scateni. 1998. Virtual reality: past, present and future. *Studies in health technology and informatics* 58 (1998), 3–20. https://doi.org/10.3233/978-1-60750-902-8-3
[10] B. Kolarevic. 2003. *Architecture in the Digital Age: Design and Manufacturing.* Spon Press – Taylor & Francis Group.
[11] António Leitão, Renata Castelo-Branco, and Guilherme Santos. 2019. Game of Renders: The Use of Game Engines for Architectural Visualization. In *Proceedings of the 24th CAADRIA Conference*, Vol. 1. Wellington, New Zealand, 655–664.
[12] António Leitão, José Lopes, and Luís Santos. 2012. Programming Languages for Generative Design: A Comparative Study. *International Journal of Architectural Computing* 10, 1 (2012), 139–162. https://doi.org/10.1260/1478-0771.10.1.139
[13] António Leitão, José Lopes, and Luís Santos. 2014. Illustrated Programming. In *Proceedings of the 34th ACADIA.* USC School of Architecture, Los Angeles, California, USA, 291–300.
[14] Brad Myers. 1990. Taxonomies of Visual Programming and Program Visualization. *Journal of Visual Languages & Computing* 1, 1 (1990), 97–123.
[15] Anh-Tuan Nguyen, Sigrid Reiter, and Philippe Rigo. 2014. A review on simulation-based optimization methods applied to building performance analysis. *Applied Energy* 113 (2014), 1043–1058. https://doi.org/10.1016/j.apenergy.2013.08.061
[16] Margaret-Anne Storey. 2005. Theories, Methods and Tools in Program Comprehension: Past, Present and Future. In *Proceedings of the 13th International Workshop on Program Comprehension (IWPC '05).* IEEE Computer Society, Washington, DC, USA, 181–191. https://doi.org/10.1109/WPC.2005.38
[17] Robert Woodbury. 2010. *Elements of Parametric Design.* Routledge.