2023
volume 20 | issue 2

ENQ
The ARCC Journal for Architectural Research

Re-constituting Precarity for the BIM-Architect

Alex Blanchard

Abstract

As architecture, engineering, and construction (AEC) practices become broadly mediated by computational methods, this article considers the modes of precarity implied for the architect adopting BIM as a medium of modelling and design. Situating the computational apparatus as a prosthesis to the BIM-architect, the article outlines the degree of agency configured for operators of BIM applications while they utilize the structures and methods of software pre-programmed by the application's original developers. Exploring the structures of Autodesk Revit's database via the Application Programming Interface (API), the paper interrogates the rationale and logic of building encoded by the program through a reading of its operative code in textual form.

Situating an interplay between the Revit-architect and application, who programmes a building model while their intention and conceptualization is programmed in turn, the conditions of precarity installed for the Revit-architect as operator are considered as a result of their limited capacity to modify the programme's operative methods. Drawing from a political history of technology to interrogate the distributed agency between the Revit-architect and technical apparatus, the article ultimately explores how the architect might adopt the phenomenal experience codified by the procedural operations of algorithms through alternative means. It concludes by drawing from autoethnographic practice and situated experiences at the site of the author's studios, offering material from which to construct an alternative and differentiated notion of algorithm-aided modelling and design according to a nuanced attention to the depth of building.

Keywords: building information modelling, programming Autodesk Revit, codeworks, computational site-writing.

INTRODUCTION: THE BUILDING INFORMATION MODEL AS PROSTHESIS

The Building Information Model

Spurred by benefits of efficiency and reliability, contemporary AEC practices are assembling around the use of digital BIM models to develop a script for construction, envisioned to perform as a digital twin and management tool for the constructed artefact. Situated at a Graphical User Interface (GUI), the BIM-architect adopts the structures and logic of a building model

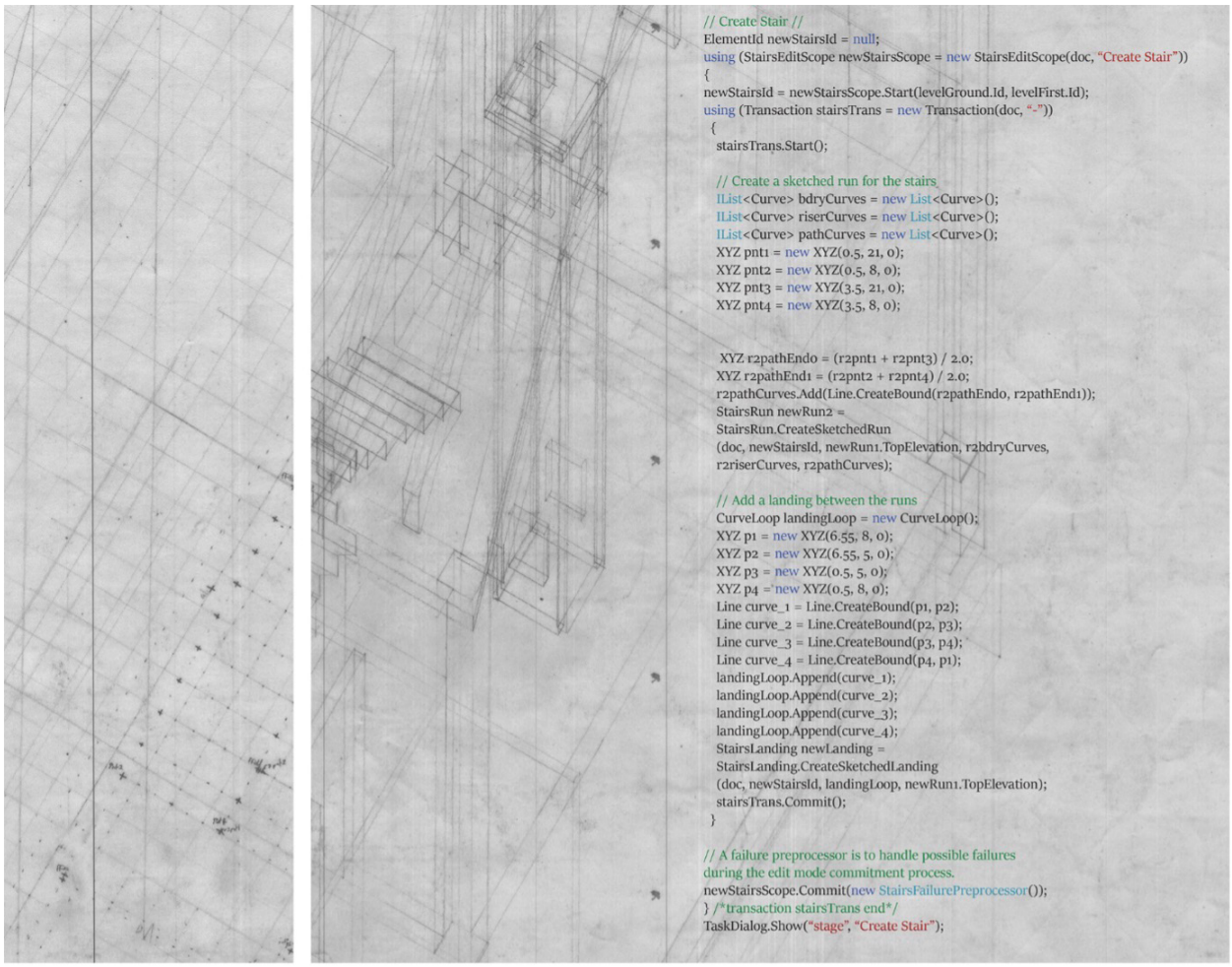
Stable URL: https://arcc-journal.org/index.php/arccjournal/article/view/1157
DOI 10.17831/enq:arcc.v18i1.1157

Corresponding Author: Alex Blanchard <a.blanchard@newcastle.ac.uk>

translated into a computable program. The computer is built upon the platform of electro-mechanical hardware, which David M. Berry suggests encodes the task of building information modelling as a calculable problem (2011, 47).

In the case of Revit, the digital model is assembled from a lexicon of architectural types and methods. Developed from the basis of Euclidean geometry and parametric meta-data to script components such as a wall, floor door, or roof, information-rich BIM models are defined according to a set of properties and variables by which an operator can develop a record of existing built fabric or a script to guide construction (Figure 1). BIM models are proposed as a centralized site of collaboration for a wider team of specialists, including structural engineering, specification of mechanical and electrical equipment, and environmental analysis, predominantly performed using specialized toolsets integrated into the application (Kolarevic 2005). Revit's owners and current developers Autodesk position the model as a "single source of truth" (Autodesk 2022). The simulation of building by a centralized digital model is purported to reduce the risk of errors on site, streamline production through file-to-factory processes, and offer an indexical link between the components scripted through computational means, indexed to standardized components and computer-aided manufacturing processes (Carpo 2014, 8, 9; Garber 2009).

Situated in a broader context of digitally-mediated construction, BIM applications are also envisioned as a management tool for the material life of the built artefact, constituting a "digital twin" to reality (Pan and Zhang 2021). BIM applications such as Autodesk Revit have been situated as a tool for facilities management, utilized for space and maintenance planning, linking building product data and documents, and creating an inventory of building equipment pre-occupancy (Meyer and Spencer n.d.). Much of this information is embedded



```
// Create Stair //
ElementId newStairsId = null;
using (StairsEditScope newStairsScope = new StairsEditScope(doc, "Create Stair"))
{
newStairsId = newStairsScope.Start(levelGround.Id, levelFirst.Id);
using (Transaction stairsTrans = new Transaction(doc, "-"))
 {
  stairsTrans.Start();

  // Create a sketched run for the stairs
  IList<Curve> bdryCurves = new List<Curve>();
  IList<Curve> riserCurves = new List<Curve>();
  IList<Curve> pathCurves = new List<Curve>();
  XYZ pnt1 = new XYZ(0.5, 21, 0);
  XYZ pnt2 = new XYZ(0.5, 8, 0);
  XYZ pnt3 = new XYZ(3.5, 21, 0);
  XYZ pnt4 = new XYZ(3.5, 8, 0);


   XYZ r2pathEndo = (r2pnt1 + r2pnt3) / 2.0;
  XYZ r2pathEnd1 = (r2pnt2 + r2pnt4) / 2.0;
  r2pathCurves.Add(Line.CreateBound(r2pathEndo, r2pathEnd1));
  StairsRun newRun2 =
  StairsRun.CreateSketchedRun
  (doc, newStairsId, newRun1.TopElevation, r2bdryCurves,
  r2riserCurves, r2pathCurves);

  // Add a landing between the runs
  CurveLoop landingLoop = new CurveLoop();
  XYZ p1 = new XYZ(6.55, 8, 0);
  XYZ p2 = new XYZ(6.55, 5, 0);
  XYZ p3 = new XYZ(0.5, 5, 0);
  XYZ p4 = new XYZ(0.5, 8, 0);
  Line curve_1 = Line.CreateBound(p1, p2);
  Line curve_2 = Line.CreateBound(p2, p3);
  Line curve_3 = Line.CreateBound(p3, p4);
  Line curve_4 = Line.CreateBound(p4, p1);
  landingLoop.Append(curve_1);
  landingLoop.Append(curve_2);
  landingLoop.Append(curve_3);
  landingLoop.Append(curve_4);
  StairsLanding newLanding =
  StairsLanding.CreateSketchedLanding
  (doc, newStairsId, landingLoop, newRun1.TopElevation);
  stairsTrans.Commit();
 }

// A failure preprocessor is to handle possible failures
during the edit mode commitment process.
newStairsScope.Commit(new StairsFailurePreprocessor());
} /*transaction stairsTrans end*/
TaskDialog.Show("stage", "Create Stair");
```

Figure 1: Orthographic drawing of a model programmed via the Revit Application Programming Interface. (The Author).

in the model through the use of proprietary products that include a wealth of documentation categorized by meta-data, ostensibly allowing a smooth transition from the design and construction process through to facilities operations (Meyer and Spencer n.d.). The BIM model is proposed to be combined with principles such as the Internet of Things in order to manage the built artefact, combined with technologies such as blockchain to manage the specification and procurement of discrete parts through to the built project and for the duration of its material life (Lagkas et al. 2018; Dounas, Lombardi, and Jabi 2021).

In the context of climate and ecological crises, the close mapping of material components through a digital model as script could facilitate targeted re-use of fabric and parts at the end of the building's life. While utilized as a design tool however, the possibilities of building are prescribed and made precarious. As the model becomes adopted as an indexical script, the design and construction of built artefacts are modulated while being encoded by the structures of the BIM application used. This article takes the Revit-operator, whom I call the "Revit-architect", as its focus. Utilizing the application, the Revit-architect may relatively quickly and efficiently combine components to develop a model. Drawing from the wider context of BIM adoption, and interrogating the relationship between one such as the Revit-architect and the programme they employ, I explore the implications for an individual designer whose agency in design might assume a precarious status as it becomes co-configured by their medium.

The Prosthesis

In view of the coupled and entangled relationship between the Revit-architect and programme they adopt, Revit may be situated as a prosthesis to its operator. The prosthesis denotes an artificial body part, such as a limb that acts as a (supplementary) addition to the body "proper". Becoming structurally coupled with the body, once in place it might radically shape the user's relation with the world. French philosopher Bernard Stiegler interrogates the relationship between the human and technology, arguing that one's rationale and logic are co-configured by their technical supplements. In Stiegler's account, an individual "exteriorizes" themselves into prosthetic technical supports such as tools and media, which are "not a mere extension of the human body", but "the constitution of this body" in the form of the human (1998, 152–53). The tool is "no longer merely inert matter, but neither is it living matter", as the human (the *who*) exteriorizes themselves into inert matter (the *what*) (1998, 49). While engaging a technical object, Stiegler suggests an individual is structurally coupled with technological objects and apparatus in a play of mutual influence between entangled cortex and matter (1998, 158).

PROGRAMMING A BUILDING MODEL

Turning Stiegler's thesis to BIM, the Revit-architect is situated in close coupling with the application's structures. Through learned engagement with the program, and their intimate knowledge of its methods, the Revit-architect conceptualizes and imagines building according to the manner in which it encodes the task of modelling.

The Revit-architect's phenomenal experience of design is primarily formed by the minimal interface of the click. Once they have learned and become attuned to the software, through a combination of keyboard shortcuts and the icons offered by the GUI their mode of production holds the sensation of rapid flow, as their task is characterized by inputting essential variables to the algorithmic methods of Revit, which partially automates calculation of the model. While the Revit-architect's work is expedited, it simultaneously follows the pathways pre-programmed by the application's original developers. Their gestures are accelerated according to essential input to the programme's methods, directing tasks related to modelling and representation which are calculated through rote operations rapidly performed by the central processing unit (Robinson 2008).

Translating the model into a form of digital data enables its resampling and analysis by algorithmic means, drawing from highly ordered and labelled information to automate production of schedules, drawings (or more properly, images), and renderings from the virtual model space. Considering the character of "signalized" computational methods in *Signal. Image. Architecture.*, architect John May argues that the contemporary architect's character of work has from shifted from orthographic methods such as drawing toward the *processing* of digital images (2019, 97).

As the Revit-architect calls upon the pre-programmed properties of the application, Florian Cramer and Matthew Fuller suggest they are assimilated into the structure of the computational program themselves—the operator being cast as a computational object (2008, 151). Stiegler's notion of a prosthetic coupling between human and technical apparatus suggests that while utilizing Revit as a modelling medium, the Revit-architect enters into a co-constitutive relationship with the program. The hard categorical boundary between the "who" and the "what" is blurred, as an individual thinks *through* as much as *with* their modelling medium.

Contemporary BIM methods extend cultures of design and construction characterized by distributed, specialized labour. Indexed to historically evolving and contingent patterns of work, the use of BIM codifies a digital model as the mediating interface for a

decentralized network of parties using a central model and platform (Braun, Kropp, and Boeva 2022). In this context, the contemporary architect's role is positioned by proponents of BIM as an organizer of expertise, directing the labor of specialist designers, fabrication and construction professionals remotely via the model, while being directed by them in turn (Carpo 2013; Garber 2014).

The contemporary BIM architect is indexed to evolving practices of construction where the designer programmed the work involved with building. Historically, various technological means of developing a record of design to guide or script construction included methods such as the production of 1:1 mould forms to direct the cutting of stone at a quarry, representation of the built artefact through orthographic drawing, and production of physical models alongside written specification. Computational programmes developed since the mid-twentieth century extend existing cultures and AEC practices while translating the building model into a computable problem. Developed in the early 1960s Ivan Sutherland's Sketchpad bridged orthographic means of design with the computational programme, translating drawing into a computational simulation whereby an operator could input data using a light pen to designate points and vectors on-screen (Sutherland 1963). Graphical drawing methods were combined with algorithmic operations such as "draw", "move," and "delete", re-mediating the modes of (orthographic) design associated with a drafting board into the capacity of the computer to automate tasks (Müller-Prove 2002). Sketchpad represented an early architectural use of the computer's capacity to perform labor "without the exercise of thought" (Davis and Davis 2005, 82), a thread continued in contemporary algorithm-aided design which adopts parametric model structures where a user can change "only a few parameters and the remainder of the model can react and update accordingly", extrapolating from the operator's point of input (Jabi 2013, 9). While the Revit-architect adopts a similar series of algorithmic methods, their conceptualisation of building might be made compulsive according to the relatively fixed structures of the application's underlying database. The following section draws from the compulsive experience of the Revit-architect to explore the tendencies of industrial objects materialized by contemporary computation.

INDUSTRIALIZED TECHNOLOGIES AND PRECARITY

Translating the model into computational media enables its re-sampling through automated methods that utilize labor crystallized in the form of algorithmic code. Situated as a prosthesis, the experiential affect of coding—which Joseph Weizenbaum noted reinforces a compulsive way of thinking through the translation of human intent into logical steps—is replayed through the Revit-architect's engagement with the program predominantly through the GUI, with limited agency to modify the software's structures (1976).

John May reflects on the new modes of engagement configured by computational technologies as practice becomes "signalized"—that is, re-mediated into automated digital forms (2019, 80). The nature of human-computer interaction entails a shift from hand-drawing toward *programming*, calling upon and directing algorithmic operations via a user interface. On these *signalized* modes, John May asks:

> is it possible that the original" copy command, in the first commercial release of AutoCAD in 1982, constituted a fundamental and decisive rupture in architectural reasoning? A rupture in which a whole series of incredibly labor-intensive (that is to say, time-intensive) orthographic gestures were subsumed within an algorithmic logic whose aim was to automate that labor in the name of efficiency? (May 2019, 82–83)

Revit generates sensations of acceleration and compulsion. I once left a project to retro-fit the Van Nelle factory late. Over a few frantic weeks, with a fixed, hunched posture over the screen, I relied on the model as site of design. Revit codified my gestures, but in return would automate production of most of my images at the last moment (Figure 2). The fabric of the Van Nelle lent itself to digital modelling. As I rolled out mass produced components in multitudes using copy and paste, I felt the sensation of engaging industrial

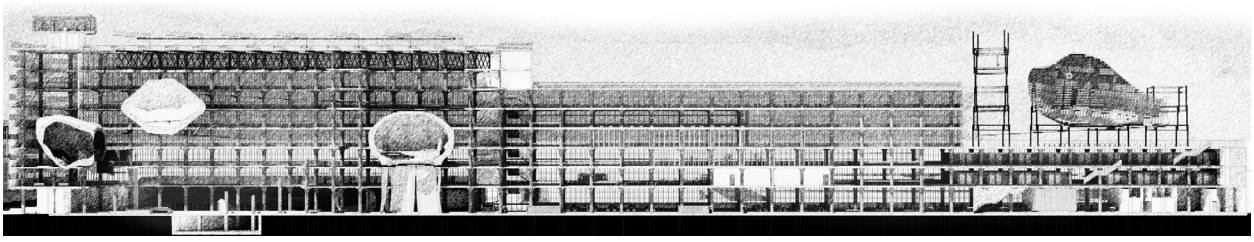


Figure 2: A project to retrofit the Van Nelle factory in Revit. (The Author).

machinery related to those originally housed by the Van Nelle factory for the processing of goods. My physical gestures were made routine according to the set means of engagement with a computer, my imagination becoming compulsive to accord with the methods prescribed by Revit's underlying database.

Parallels extend from the industrial nature of the Van Nelle, whose spatial design and engineering would rationalize and streamline processes of sorting, refinement, packaging, and logistics it housed, into the rationalized task of the Revit-architect. As industrialized forms of production broadly accelerated processes of mineral extraction, refinement, and manufacture, sophisticated technical apparatus of the early twentieth century re-composed relations between their operators and users. The forms of repetitive labour emerging around Ford and Taylorist manufacture typically situated the worker at the site of one fragmentary task within divided production as a whole. Gilbert Simondon responded to the industrialized forms of modernity such as those at the Van Nelle, locating a shift in the dynamic between technology and its modes of use. Simondon posited that the sophisticated automated apparatus of the mid-twentieth century assumed increased agency and took on the role of "technical individual", re-situating the human's role as an organizer of the ensemble of apparatus, or relegating them as a helper with reduced agency across the technical and social milieu, "[they] grease, clean, remove detritus […]"(2017, 78).

Stiegler's theoretical framework can be characterized as a critical theory for technology, interrogating the political implications of the relationship between one such as the Revit-architect and the programme they operate. In his later work on automatic society, Stiegler argued that an industrial character of technology can be situated according to the separation of producers from consumers. In the case of Revit, the user effectively consumes the labor of others, materialized in the methods and operative algorithms of the programme to develop a building model. Stiegler located a crisis emerging through the rapidly evolving sophisticated technologies of recent history, arguing that a disequilibrium occurs when the possibilities for an individual to differentiate their experience are over-determined by increasingly automated apparatus. The architect's reason and desire become structured by the BIM package they engage (Stiegler n.d.). According to Stiegler's critical theory of technology, the Revit-architect—relatively distant from the computer's operative algorithms and holding limited agency to modify the programme—can be situated as an *operator* rather than a *programmer*. As broader cultures of building rely on the centralized BIM model, a precarious condition is installed for the Revit-architect, their capacity to differentiate possibilities of building from the modulatory conditions of the application resonant of the factory worker reduced to an operator of industrialized apparatus.

CODING A BUILDING MODEL

The Revit application and model are comprised of object-oriented code, which discretely scripts the parameters of architectural types and their methods of interaction (Goffey 2008). The architect accesses these objects and methods via the abstracted graphical interface, *programming* a building model according to formal structures of the application. David M. Berry describes the mechanism by which computational practices perform a translation of the world. Parsing it into "symbolic sets of discrete data to represent reality [once encoded, the data] can be resampled, transformed, and filtered endlessly" (Berry 2011). While the architect programmes a digital model via a BIM application, they call upon the computer-as calculating device to co-develop a design, feeling sensations of immediate production. Tapping the keys "W" and "A" to open the wall method, and with two mouse clicks, a fully detailed wall is generated in plan. Control-P. Enter, enter, enter. Three drawings exported as PDF format. The calculations of the software performed through algorithmic methods are accessed from the distance of the graphical user interface, automating the work involved with modelling and representation.

A user may also engage with Revit via the Application Programming Interface (API), which is offered for practitioners to develop custom tools within the application, or by third-party developers to create macro extensions that supplement the functionality of the program. The text-based programming console uses words and phrases structured by syntax to develop or engage with a model. The code offers another means to read the underlying database of Revit along with its operative algorithms. Programming a model through the API demonstrates the manner in which Revit encodes the building model from basic classes such as point coordinates and vectors, building in complexity toward architectural types such as a wall, floor, or door. The following samples are presented from a dwelling model developed through the API, exploring how the types and methods typically accessed through icons are structured by code.

Coding a floor slab
A slab at ground-floor level might be coded by first declaring a set of four points:

```
XYZ ia = new XYZ(a.X + 0.5, a.Y + 0.5, a.Z);
XYZ ib = new XYZ(b.X + 0.5, b.Y 0.5, a.Z);
XYZ ic = new XYZ(c.X 0.5, ib.Y, a.Z);
XYZ id = new XYZ(d.X 0.5, ia.Y, a.Z);
```

Each coordinate point then informs vectors which delineate the plan form of the slab:

```
Line iab = (Line.CreateBound(ia, ib));
Line ibc = (Line.CreateBound(ib, ic));
Line icd = (Line.CreateBound(ic, id));
Line ida = (Line.CreateBound(id, ia));
```

Via the API, types of floor slab can be accessed through a search function rather than graphical menus. In Revit's C# programming a language a filtered element collector obtains a specific class to be used as floor type according to name:

```
FloorType groundFloorSlab = new
FilteredElementCollector(doc).
OfClass(typeof(FloorType))      .First<Element>(e
=>              e.Name.Equals("FloorGrndBearing_
65Scr90Ins125Conc50SBld150Hcore"))
as FloorType;
```

Once the floor family type is selected, the set of vectors that delineate its boundary in plan are listed as an array (a multi-dimensional list) of curves:

```
CurveArray slabCurves = application.Create.
NewCurveArray();
slabCurves.Append(iab);
slabCurves.Append(ibc);
slabCurves.Append(icd);
slabCurves.Append(ida);
XYZ normal = XYZ.BasisZ;
```

Finally, the floor plan, the floor type, and its level are used as input values to model and create the floor:

```
tGroundSlab.Start();
Floor grndslab = doc.Create.
NewFloor(slabCurves, groundFloorSlab,
levelGround, false, normal);
grndslab.get_Parameter(BuiltInParameter.
FLOOR_HEIGHTABOVELEVEL_PARAM).Set(0);
tGroundSlab.Commit();
```

The code to program a floor slab illustrates the discrete conceptualisation of building encoded according to type. While the user's possible input is modulated to a relatively high degree by the application's structures and means of programming, their agency in modelling is made precarious, contingent to the fixed database that underlies the programme.

Seeking to respond to the compulsive experiences conditioned by Revit according to my situated experiences in a studio space, I set out to explore alternative uses of BIM techniques, turning the operative algorithms of Revit toward an alternative.

AN ALTERNATIVE NOTION OF ALGORITHM-AIDED DESIGN

Programming code holds a double nature, performing as both a form of machine language highly ordered according to syntax and classes for execution by the CPU, and as a means for human interpretation and scripting. Programming a building model through Revit's API, I found myself experiencing the compulsive effects of coding, replaying the sensations of Revit-modelling via the GUI. The final component of this article explores how situated autoethnographic practice can adopt a mode of site-writing to critically engage with the structures of computational programmes.

Re-contextualising Revit's programming methods into print form and working with them as a text makes use of the double nature of code, creating a site to modify the structures of Revit where an individual can influence their media of design. Berry suggests the two-phase structure of code allows "its program to be read from its textual script form, and normative structures and intentionalities explored" (2014, 17). Adopting code as a typewritten text, I explored how it can open a site of negotiation between the Revit-operator and their technical prosthesis, offering means to fabricate other means of conceptualizing building. Spending approximately thirty-six months in artist studios that temporarily occupied buildings in the city centre of Newcastle upon Tyne in the UK, the compulsive experience of coding a dwelling model in the Revit API prompted me to write alternative (non-computational) scripts according to observations at my workspace. Seeking to adopt the algorithmic methods of Revit as a means to describe processes of decay and modes of habitation through procedural means, I documented the site of my studios through a combination of recording methods alongside a "textual model" of the built artefact and my experiences there.

One of my studio spaces was situated within Carliol House, a grand art-deco building constructed in the 1920s as headquarters for the North Eastern Electricity Supply Company. My initial tests sought to write the fabric in a textual form of code that explored the calculation of data performed by Revit's methods—such as the algorithm to model a stair—in longhand (Figure 3). Each point was coded manually though the use of lists as sets of coordinates, modified and transformed using "for-loops" to denote the built fabric. From this starting point, the code evolved toward a temporal and procedural re-construction of the events on site modelled according to my subjective encounters with them.

While the model sets out from the use of Euclidean geometry as a starting point, other means of describing

Figure 3: Writing the Carliol House main stair script. (The Author).

Figure 4: Decay and at the site of my studios. (The Author).

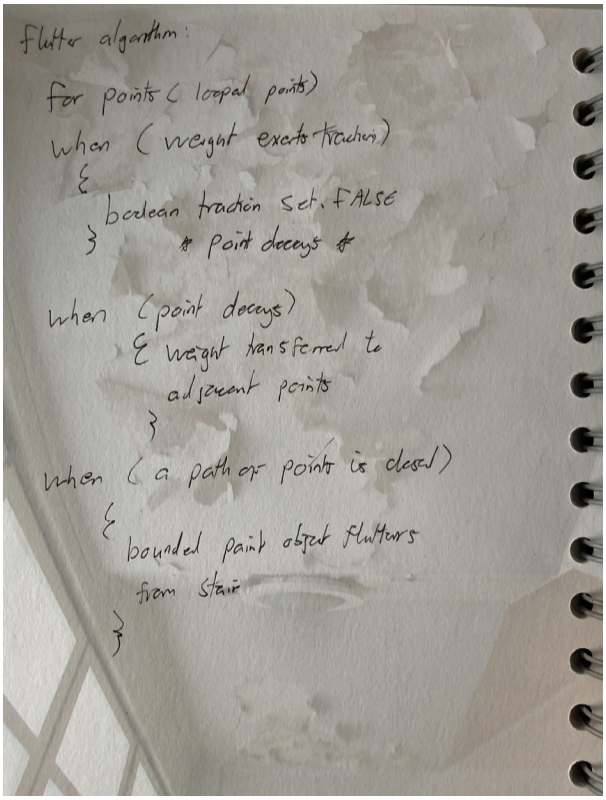


Figure 5: Initial notes for the flutter algorithm at the site of the stair. (The Author).

the experience of the building gradually emerged as I differentiated my means of describing the site *through* and *with* coding methods. The work began to portray different events and processes, such as the material decay observed within Carliol House. Paying close attention to delaminating paint to one of the landings—which was often remarked on by visitors—I scripted other methods that simulated the process of material decay in computational terms (Figure 4). The algorithm offers means of conceptualizing sites and buildings through temporal and procedural description. Working between the writing of a text and procedural methods, the flutter algorithm denotes a method by which segments of delaminated paint fall to the ground after losing contact with the plaster (Figure 5):

flutter()
{
find path from object datum to ground. Z at same position equal to X and Y. Divide path into discrete increments. For each path increment, object. Z equals object. Z minus length given by path division. Object X and Y plus or minus a random float according to degree of flutter. For X and Y, plus or minus value draws from a recollection of drift from previous flutter iteration.
}

The textual model evolved from these initial tests, its production rendered urgent by the notices of eviction served to the artist studios ahead of extensive demolition and redevelopment of the block, retaining only the street-front façade of Carliol House. Developing procedural means to describe situated experiences in daily habitation of multiple studios at the block, the text drifted from the quality of an executable script while articulating other phenomena more often concealed or omitted from BIM models, for example, the experience of pulling open the heavy oak doors each morning to enter the lobby of Carliol House, which was always cool despite the weather outside. Other methods pointed to alternative means of engaging computational programmes according to my observations elsewhere across the site, including an account of a studio member who opened a storage room by estimating its access code according to an observation of material wear to its keys. Some of the events, things, and people observed in the model were described in relation to upcoming redevelopment at the site, witnessed during the move out of studios. Other fragments of textual-code related to the close attention and intimate knowledge of the building fabric, ingrained over sustained daily use (Figure 6).

CONCLUSION: RE-CONSTITUTING PRECARITY FOR THE BIM-ARCHITECT

As BIM methods are adopted across AEC, they configure

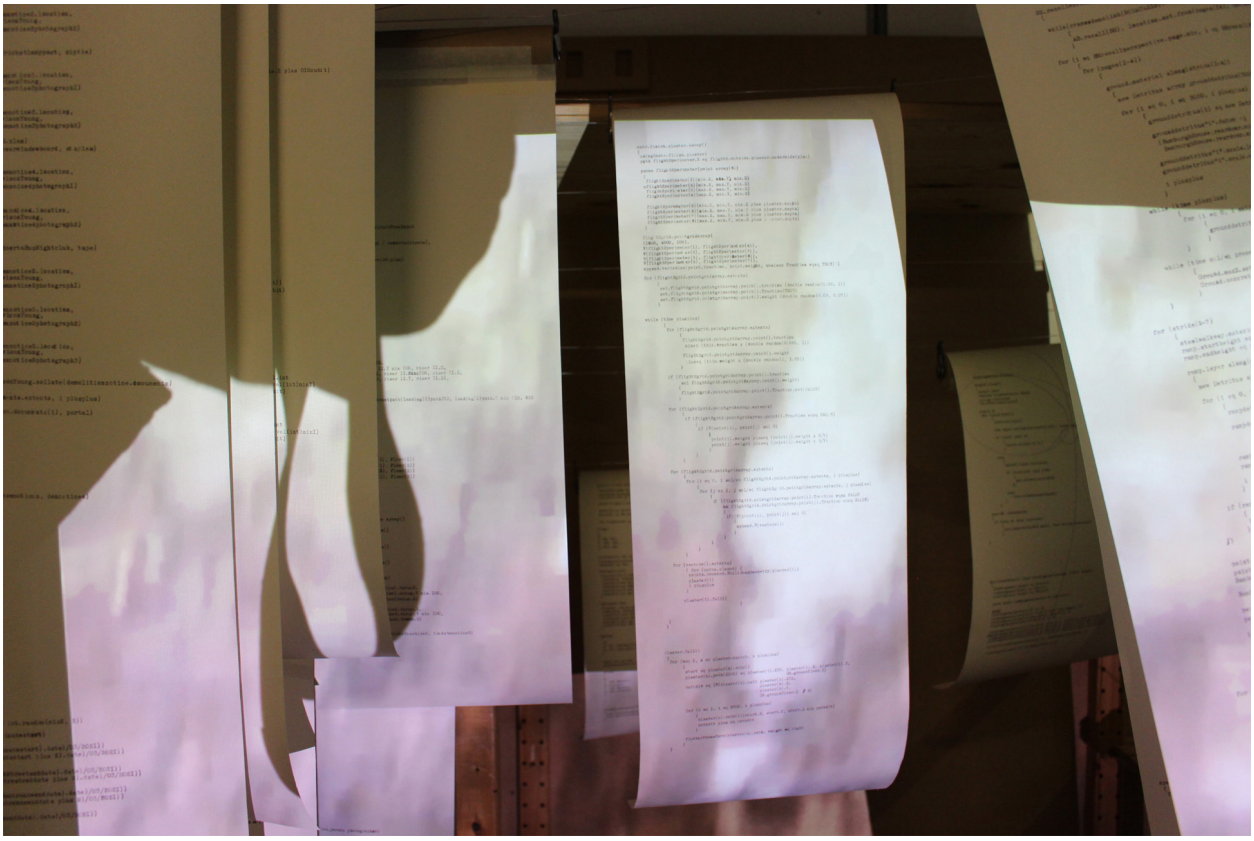


Figure 6: A fragment from the textual model of the site. (The Author).

new distributions of agency between professionals and the media they employ. In this context, the interfaces to modelling programmes retain potential for reconstruction as a site of critique. This article draws from Bernard Stiegler to demonstrate the agency held by technological apparatus in prosthetic co-constitution with their users, mediating and actively configuring their experience and perception of design according to the rationale and logic configured by computational tools. As industrial tendencies are extended through BIM platforms, the article demonstrates the precarity installed for users such as Revit-architects holding limited capacity to modify their tools. Drawing from autoethnographic research and my own relationship with Autodesk Revit, the article shows one possible route to adopt situated practices to interrogate the structures of BIM programmes and turn their methods toward other means of conceptualizing building. Re-contextualizing code in print is not foreclosed by the requirement to execute but assumes a productive mode of precarity in its interpretation, which remains contingent to a subjective reader. The alternative modes of algorithm-aided (textual) modelling that resulted from observations and experiences in the built environment offer one possible means to re-constitute a technologically-mediate attention to building according to a specific place.

ACKNOWLEDGEMENTS

The author wishes to thank the organizers and participants of the Precarity Conference held at the University of Pennsylvania in April 2022, whose generous feedback and guidance shaped the research presented in this paper. I extend my gratitude to Dr. Franca Trubiano, Weitzman School of Design, Professor Philip D. Plowright, College of Architecture and Design, Lawrence Tech, and the anonymous reviewers. Thank you also to Professor Adam Sharr and to my supervisors, Dr. Edward Wainwright and Dr. Stephen Parnell at Newcastle University, who helped to formulate a positive critique of precarious practice.

RERERENCES

Autodesk. 2022. “Revit Software.” https://www.autodesk.com/products/revit/overview.

Berry, David M. 2011. *The Philosophy of Software: Code and Mediation in the Digital Age*. Basingstoke, Hamp-

shire; New York: Palgrave Macmillan.

———. 2014. *Critical Theory and the Digital, Critical Theory and Contemporary Society.* New York: Bloomsbury.

Braun, Kathrin, Cordula Kropp, and Yana Boeva. 2022. "Constructing Platform Capitalism: Inspecting the Political Techno-Economy of Building Information Modelling." *arq: Architectural Research Quarterly* 26 (3).

Carpo, Mario. 2013. *The Digital Turn in Architecture 1992-2012*. AD Reader. Chichester: Wiley.

———. 2014. "Foreword." In *BIM Design: Realising the Creative Potential of Building Information Modelling*, 1st ed., 8–12. Chichester: Wiley.

Cramer, Florian, and Matthew Fuller. 2008. "Interface." In *Software Studies: A Lexicon*, edited by Matthew Fuller, 149–53. Leonardo Books; Cambridge, MA: MIT Press.

Davis, Martin, and Virginia Davis. 2005. "Mistaken Ancestry: The Jacquard and the Computer." *Textile: The Journal of Cloth and Culture* 3 (1): 76–87. https://doi.org/10.2752/147597505778052594.

Dounas, Theodoros, Davide Lombardi, and Wassim Jabi. 2021. "Framework for Decentralised Architectural Design BIM and Blockchain Integration." *International Journal of Architectural Computing* 19 (2): 157–73. https://doi.org/10.1177/1478077120963376.

Garber, Richard. 2009. "Optimisation Stories: The Impact of Building Information Modelling on Contemporary Design Practice." *Architectural Design* 79 (2): 6–13. https://doi.org/10.1002/ad.842.

———. 2014. *BIM Design: Realising the Creative Potential of Building Information Modelling*, 1st ed. Chichester: Wiley.

Goffey, Andrew. 2008. "Algorithm." In *Software Studies: A Lexicon*, edited by Matthew Fuller, 15–20. Leonardo Books; Cambridge, MA: MIT Press.

Jabi, Wassim. 2013. *Parametric Design for Architecture*. London: Laurence King Publishing.

Kolarevic, Branko. 2005. *Architecture in the Digital Age: Design and Manufacturing*. New York: Taylor & Francis.

Lagkas, Thomas, Vasileios Argyriou, Stamatia Bibi, and Panagiotis Sarigiannidis. 2018. "UAV IoT Framework Views and Challenges: Towards Protecting Drones as 'Things.'" *Sensors* 18 (11): 4015. https://doi.org/10.3390/s18114015.

May, John. 2019. *Signal. Image. Architecture.* New York: Columbia University Press.

Meyer, Bill, and Gareth Spencer. n.d. "Revit Modeling for Successful Facilities Management | Autodesk University." Autodesk. Accessed May 30, 2022. https://www.autodesk.com/autodesk-university/class/Revit-Modeling-Successful-Facilities-Management-2014.

Müller-Prove, Matthias. 2002. "Vision and Reality of Hypertext and GUIs: 3.1.2 Sketchpad @mprove." https://www.mprove.de/visionreality/text/3.1.2_sketchpad.html.

Pan, Yue, and Limao Zhang. 2021. "A BIM-Data Mining Integrated Digital Twin Framework for Advanced Project Management." *Automation in Construction* 124: 103564. https://doi.org/10.1016/j.autcon.2021.103564.

Robinson, Derek. 2008. "Function." In *Software Studies: A Lexicon*, edited by Matthew Fuller, 101–10. Leonardo Books; Cambridge, MA: MIT Press.

Simondon, Gilbert. 2017. *On the Mode of Existence of Technical Objects*. Translated by Cécile Malaspina and John Rogove. Minneapolis; London: University of Minnesota Press.

Stiegler, Bernard. 1998. *Technics and Time, 1: The Fault of Epimetheus*. Translated by George Collins and Richard Beardsworth. Stanford: Stanford University Press.

———. n.d. "Anamnesis and Hypomnesis: Plato as the First Thinker of the Proletarianisation." *Ars Industrialis* (blog). Accessed April 11, 2022. https://arsindustrialis.org/anamnesis-and-hypomnesis.

Sutherland, Ivan. 1963. *Sketchpad: a man-machine graphical communication system* (Technical Report). Lexington, MA: MIT Lincoln Laboratory.

Weizenbaum, Joseph. 1976. *Computer Power and Human Reason: From Judgement to Calculation*. New ed. San Francisco: W.H.Freeman & Co Ltd.