KEEPING UP WITH THE CODE

Communicating the Decision Making History of Architectural Scripts

W. HAMILTON,
University of New South Wales, Sydney, Australia
william.hamilton@cox.com.au

Abstract. There is often a disconnect between users of scripts and their understanding of the decisions made within a coded process, either due to a lack of knowledge regarding the contents of the script or poor communication of its intent. In a script where there are multiple potential outcomes, a decision must be made, either through a human informed rationale or a mathematical optimisation in a logic system. When architectural decisions and choices are made there is often little documentation of this criteria and the potential for this to affect design thinking is undervalued. This issue is exacerbated when the act of decision making is embedded and hidden within complex code. This research aims to develop a method that successfully traces scripted decision making history (DMH). A relational database titled Huginn has been developed to test the feasibility of tracing decision making history in scripting. This was achieved through a Python Web framework that has data sent to it in a JavaScript Object Notation (JSON) format from Grasshopper. The result of this research is a system that can effectively link a series of objects and their decisions back to their origins. This research contributes to developing theoretically grounded coding protocols and expanding an understanding of the possibilities that an alternate scripting convention can present to improving design practice.

Keywords. Decision making; Database; Communication; Visual programming; Design rationale

1. Introduction

"Any sufficiently advanced technology is indistinguishable from magic" (Clarke 1973, p.236). Despite scripting becoming increasingly pertinent as a tool to aid design practice and operational workflows in the architecture industry, Clarke's classic quote still applies to the modern professional world. Through consultations with staff from Cox Architecture, the industry partner of the research, a recurring issue arose; the inability to understand the procedure and logic of architectural scripts that colleagues had developed, accompanied by a reluctance to engage with emerging technologies. An architectural script can be defined as any piece of code, usually written in a visual programming language such as Grasshopper or Dynamo, intended to "automate repetitive activities…extend design experimentation…[and] improve file-to-factory protocols" (Burry 2011, p.8-9). The identified industry aversion to engage with scripting is a stark juxtaposition to the rate at which computational tools for the architecture, engineering and construction (AEC) industries are becoming increasingly powerful and accessible, calling for an investigation into the communication and interpretation of data from scripts (ibid.).

Compounding this issue of poor comprehension, is an industry-wide underappreciation of the value understanding decision making provides to architectural practice. As a result, there has been minimal established procedure for documentation of DMH (Peng et al. 2000). In this paper DMH is defined as the series of decisions and logical rules that are connected in a computational system that culminate in an outcome. The outcome of DMH can be either an architectural model or an individual geometry or value within. In other academic works such as that of Chachere and Haymaker (2011, p.86), the term Design Decision Rationale (DDR) is used as a definition of "a set of assertions that…support design decisions" which is the information this research intends to document and trace. AEC professions are experiencing an increased demand for their outcomes to be justifiable through the practice of evidence-based design and a routine documentation of DDR would provide strong evidence to substantiate and quantify reasoning (Criado-Perez, et al. 2019).

This paper seeks to address a prominent lack of involvement within architectural practice regarding scripting participation and DDR documentation. By partaking in an action research methodology, the challenge can be approached through an iterative design process grounded in feedback from industry professionals to ensure the outcomes are tailored and relevant to practice. Reluctance to participate in the automation of architectural practices can be derived numerous concerns, primarily originating from a lack of trust in an unknown system (Heumann & Davis

2020). Through the development of a prototype workflow, this research intends to contribute to overcoming this barrier of technological engagement by increase transparency in scripts and therefore the ease at which their processes can be comprehended. Furthermore, this work seeks to increase the success, accessibility and uptake of automated workflows, encouraging more people to be involved in using computational design tools.

2. Research Aims

The key aim of this project is to explore the viability of documenting decision making that is embedded within architectural and engineering scripts. This is an experiment in both the technical feasibility of the task and the ability to develop a scripting procedure that makes it possible. More specifically, this research aims to develop a workflow that reveals these decisions in an easily processable way by users. This takes the form of a prototype workflow for extracting DMH from a Grasshopper script and exporting it to a Web hosted database. Subsequently, a function is developed to compile and collate this data into a format that interpretable by humans.

By building on the work of others in the field, it is already achievable to get geometric data into a Web environment, but this research seeks to discover if it is possible to integrate a communication of decision rational into this form of system (Leung, et al. 2018). Proving this is achievable raises the question of the potential to make this system approachable by an inexperienced user to give it robustness for industry application.

3. Research Questions

To what extent is it possible to analyse the decision making history embedded in an architectural script to inform enhanced comprehension of the code?

Which can be broken down into two sub questions that must be addressed:

1. In what ways can embedded DMH be extracted from architectural scripting?

2. To what extent can a convention be established for the documentation of decisions that facilitates both human input and interpretation of scripted decisions.

4. Methodology

“Action research is a name given to a particular way of looking at your practice to check whether it is as you feel it should be (McNiff 2013)”. When engaging with emerging technologies, often the most effective way to understand the subject matter is to take an immersive approach that involves practitioners and industry professionals who are most familiar with the

realistic applications, successes and shortcomings of such technologies. Action research provides the ideal methodological framework for researching a topic that is so intrinsically tied to industry professionals and work place environments. This makes the methodology well suited for this investigation due to the nature of the case study being so intertwined with both emerging digital technologies and the fact that it is a direct response to challenges identified by those working in practice. The outcomes achieved in this research were only possible due to the close collaboration with the industry partner Cox, not only as a provider of technical support but by grounding the investigation in a relevant context.

Upon identifying the issue, the need to be able to understand the relevance of decision making in scripts, action research calls for an intervention to the problem to be taken (Azhar, Ahmad, & Sein 2010). Following this, and for a development to be considered research, observations and data must be collected about the intervention. The data must then be reflected upon and used to inform the decisions made about future iterations and generations of the developed solution. This is a cyclical process that should be continued until there is a sufficient understanding of the problem, potentially leading to further research or a different approach to the topic. These principles manifest themselves in this research in the form of an iterative design process, with future decisions being informed by their predecessors. In both the Web database and Grasshopper workflow, the effectiveness and usability were gauged and used to inform changes that should be made in following iterations. Furthermore, these two components were developed in parallel, allowing the discoveries and hurdles from one to inform the necessary course of action for the other. By following a cyclical procedure of ‘plan’, ‘action’, ‘observe’ and ‘reflect’, the research utilises the iterative design principles of action research to improve the outcome through meaningful assessment (Kemmis 2009).

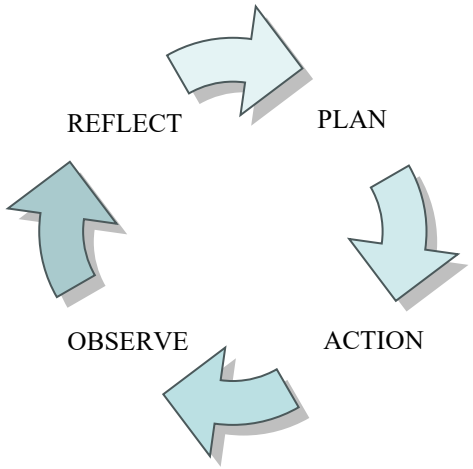

Figure 1. A diagrammatic representation of Kemmis’ action research theory

5. Literature Review

SCRIPTING IN ARCHITECTURE AND THE RELEVANCE OF DATA

"Twenty years ago we thought computers were machines for making things; today we find out they are even more indispensable as machines for thinking" (Carpo 2018, p.135). With the rise of computing power, access to technology within the AEC field and increased understanding of the capabilities of computational design practices to improve workflows, the industry is developing new methods of automation and optimisation. In an architectural context, this often takes the form of using visual scripting languages such as Grasshopper and Dynamo as well as conventional coding platforms like Python and VBScript (Leitão & Santos 2011; Cichocka, et al. 2017). Optimisation spans a wide scope of applications from solar access optimisation scripts already abundant in industry to experimental automatic compliance checking scripts, a contemporary point of focus in research (Balaban 2012; Patlakas, et. al. 2017; Guedes & Andrade 2019).

While data is often the driving factor of automated computational systems, practices within the AEC industry have minimal precedence of utilising the potential of data to inform systems or to achieve creative outcomes. Sectors such as finance and manufacturing have historically developed on a framework of "technological change captured under the rubric of automation" (Pardo-Guerra 2012, p.568), performing with high levels of engagement with data to inform decisions and improve efficiency. On the contrary, creative fields like architecture have been slow to adapt to the use of data manipulation in creative workflows, at least consistently at a wide scale. The works of Hua & Jia (2010) and Nagy, et al. (2017) are showcases of exploratory work with generative data systems as a method for optimising layout planning but this is not a reflection of standard practice.

The applications of data for the architectural industry aren't restricted to creative and/or optimisation solutions however, with its primary application manifesting in the utilisation of Building Information Modelling (BIM). A survey taken in 2017 highlighted that 87% of employees in architecture firms identified with using Revit (Gardner 2018), a singular BIM software, in their workplace suggesting that BIM has a near ubiquitous integration within the industry. BIM is essential to modern AEC project workflows due to its ability to "improve productivity and quality of project delivered, curtail the project delivery time and cost" (Kushwaha 2016, p.100). When considering the extensive application of BIM it suggests that any successful creative or optimisation solution informed by data must be able to be integrated into a BIM workflow to mesh with architectural industry procedure.

ARCHITECTURE AS DECISION MAKING

"Design consists of many interdependent decisions" (Lewis, et al. 2007). Fundamentally, the practice of architecture is a culmination of decision making as a result of numerous changing conditions, influences, objectives and constraints. The process of decision making is complex and requires individuals to formulate some degree of assessment criteria to do so, regardless of if they are conscious of it or not. Decision Theory (DT), the process one goes through when making choices, rationalises the best course of action to maximise expected utility (Chachere & Haymaker 2011).

With decision making being "argued to be the principle activity" of architecture and engineering, (Lewis, et al. 2007) it is logical that this process should be well documented and recorded. Multiple scholars have explored approaches to this including Chachere and Haymaker's rationale clarity framework (2011) and Peng et. al.'s object-oriented information management framework (2000) which both provide suggested structures for workflows which mandate a formal documentation of design decisions, which: (1) provide a justification and DT grounded rationale of choices and (2) provide the opportunity to retrieve a record of DMH. Peng et al. suggest this can be achieved by linking decision history to rationalised CAD items through the object oriented programming language C++.

By utilising data driven design, logic systems and even artificial intelligence within design, architects are 'handing over' a large portion of decision making to the computer. When considering that any logic embedded into code and scripts is ultimately just instructions to the computer on how to make decisions, it is a reasonable assertion that a method of documenting and tracing DMH should be as equally sought after as a manual recoding method.

MACHINE DECISION MAKING

The DT of a computer needs to be rationalised in the form of numerical values, which is used to create a decision matrix (Arroyo, et al. 2012). In the work of Arroyo et. al., they specifically explore the various mathematical models of multiple-criteria decision making and their relevance to the AEC industry. All the methods are characterised by their dependency on a decision matrix to inform choices, coincident with the data type a computer requires to complete decision making processes. As discussed earlier, a body of scholarship already exists in which the importance and methods of documenting decision making history are explored, as well as the procedure for embedding DT into code but there has been little to no work done on the combination of these two concepts, providing an opportunity for further research. It should be noted that having history linked to an object is not a new concept, as showcased in modelling software like Maya but there are no standardised systems of tracing scripted DMH.

A major challenge in documenting scripted design decisions is the quantification of design qualities. While architecture evokes feelings and experiences that would conventionally be described through feelings and emotive language such as “organic” or “kitsch”, a computer demands a finite taxonomy for processing calculations (Stott 2015). Durmisevic et. al. (2001) approached this challenge by creating ‘aspects’ of design traits within the categories of attractiveness, wayfinding, daylight and physiological to quantitatively measure qualitative design elements. An alternative method was explored by Berry & Park (2017) where sensory equipment was used to produce numerical data such as temperature and thermal comfort to rationalise the experience of architectural space. Some architectural contexts naturally lend themselves to rationalising qualitative experience such as sports architecture, where the success of geometric decisions in the design process is intrinsically linked with profit and therefore a machine-legible value of success (Joseph, et al. 2015).

HUMAN INTERPRETATION OF MACHINE DECISIONS

Beyond the challenge of interpreting and processing the DMH of a script, there is also a significant hurdle of both turning the data into human-readable content as well as allowing architects to engage with and make use of the information to improve design decisions. Because architecture is inherently a visual discipline, both in its procedure and outcome, “it is no surprise that many students, architects and academics consider themselves ‘visual thinkers’” (Austin & Wajdy 2016, p.831) and thus struggle to understand raw data and conventional coding methods. Furthermore, a major challenge posed by increased computational involvement and complex coding solutions in design workflows is the disconnect between users and the outcome produced by scripts. A study by Davis, Burry and Burry (2011) found that architects struggled to comprehend the function of unfamiliar visual scripts despite being familiar with all individual functions involved. This research highlights the need for data to be presented to architects in a visual and simplified format to ensure comprehensibility and usefulness.

6. Case Study

This case study seeks to address the research aims through two primary developments; A Web-based database titled Huginn and a localised Grasshopper workflow. Both elements were developed simultaneously and in collaboration with Cox resulting in an iterative design process that was grounded in industry relevance. While the decision was made to work in Grasshopper for this experiment due to it being most familiar to the researchers, the principles employed in this research are software agnostic and could theoretically be applied to other platforms such as Dynamo. Furthermore, the Grasshopper workflow should be applicable to all projects

and jobs but a pre-existing case study script was selected for the sake of testing and proof of concept.

6.1 CHOOSING A CASE STUDY SCRIPT TO OPERATE ON: THE STADIUM BOWL

As there was a need for a base script to host the DMH tracing workflow that was developed, it was chosen to use a portion of a stadium bowl script written by Cox. This choice was made due to the large amount of decisions that are made throughout the design of a stadium bowl and the fact sports architecture contains many logical and geometric dependencies meaning decisions have direct influences on the form and therefore future decisions (Joseph, et al. 2015). By using an existing script to test the workflow on, it meant that time could be allocated towards new investigations as well as ensuring the research had relevance to industry standard procedures and the types of decisions that were recorded would have substance as plausible and realistic outcomes.

6.1.1 Features of the Stadium Bowl Script

The portion of the stadium script that was isolated can be divided into multiple clusters, each performing a primary function in creating the output geometry and data. These clusters are titled as follows and their geometric outputs listed in Table 1. Figure 2 depicts example output geometry from each of the clusters.

TABLE 1. Clusters within the stadium script

Cluster:	**1. Gridlines**	**2. Field of Play**	**3. Plats**	**4. Bowl + Aisles**	**5. Seats**
Outputs:	X Centreline	Field	Lower bowl plat profile	Lower bowl	Seat outlines
	Y Centreline	Boundary line	Suite bowl plat profile	Suite bowl	Seat count
	Gridlines		Upper bowl plat profile	Upper bowl	Row count
			Line of sight	Aisle lines	Aisle count
				Steps	

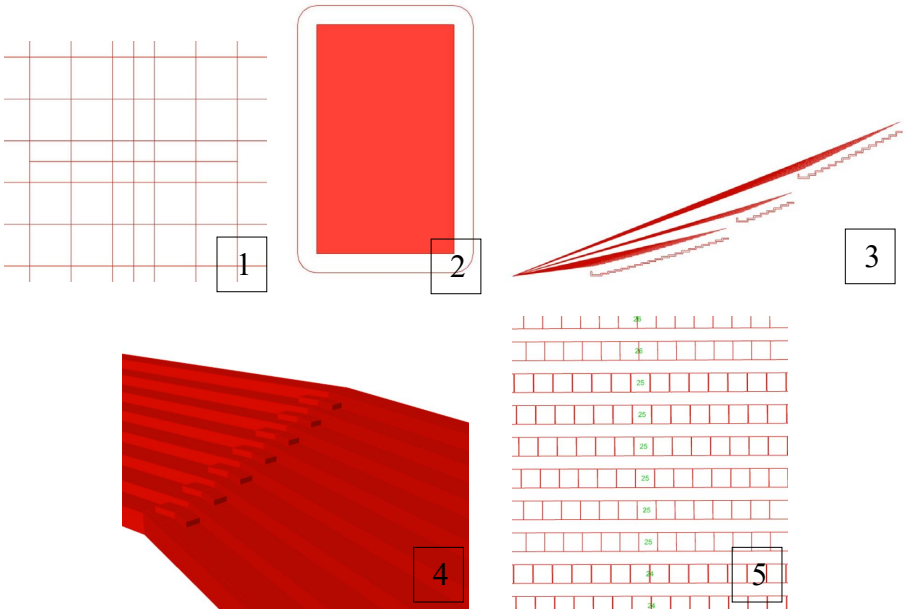

Figure 2. Example output geometries from the clusters

6.2 DEVELOPING HUGINN, A WEB-BASED RELATIONAL DATABASE FOR STORING JSON OBJECTS

6.2.1 Using Django REST framework to build the Web API

To achieve a system in which DMH can be traced, a method had to be established for linking decisions to each other in order to comprehend the 'history'. The chosen solution to this was to write a series of Python code modules using Django REST Framework. Django is a tool for building Web application programming interfaces (API). The Python code was written in a way that utilises classes, where a class is a set of instructions for creating objects that have assigned properties. One possible feature of classes is the ability to link two objects together which will be referred to as mapping, effectively creating a child and parent object (Figure 3). This structure of mapped objects stored together is known as a relational database, meaning that any decision influenced by a previous decision will have it linked as a parent in the database structure.

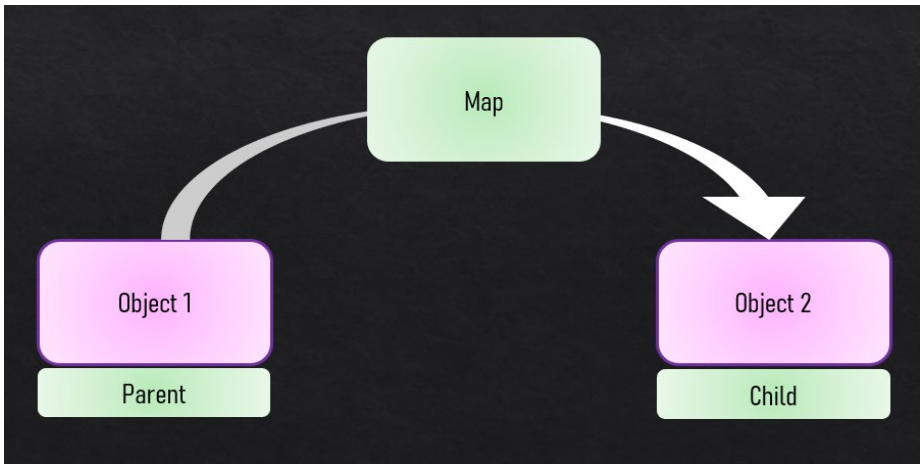

Figure 3. Diagram of object mapping with Python classes

Two classes were created, one called parameterObject for storing objects with decisions attached to them (Figure 4) and one called paramaterMapThroughObject for storing the mapping of two objects (Figure 5). Django also required the setup of numerous serializers and definitions to allow the database to not only receive data sent to it, but to check that the information is correctly in a JavaScript Object Notation (JSON) format and make this readable by the class functions.

```
13 class parameterObject(models.Model):
14     param_created_at            =    models.DateTimeField(auto_now=True)
15     parameterIdentity    =    models.CharField(max_length=255, unique=True)
16     parentIdentity       =    models.CharField(max_length=255)
17     parameterVal         =    models.CharField(max_length=65535)
18     parameterType        =    models.CharField(max_length=200)
19     data_text            =    models.TextField(max_length=65535, default="[]", blank=True, null=True)
20     sourceParameter      =    models.ManyToManyField('self', through= 'parameterMapThroughObject',
21                                through_fields=('object_to', 'object_from'), symmetrical=False)
22     def __str__(self):
23         return self.parameterIdentity
```

Figure 4. parameterObject class for constructing objects

```
25 class parameterMapThroughObject(models.Model):
26     map_created_at          = models.DateTimeField(auto_now=True)
27     object_from          = models.ForeignKey('parameterObject', on_delete=models.CASCADE, related_name='through_from')
28     object_to            = models.ForeignKey('parameterObject', on_delete=models.CASCADE, related_name='through_to')
29     # function           = models.ForeignKey('functionOb', on_delete=models.CASCADE, related_name='maps')
30     function             = models.CharField(max_length=255)
31     data_text            =    models.TextField(max_length=200, default="[]", blank=True, null=True)
32
33     def __str__(self):
34         return self.object_from.parameterIdentity +" -> "+self.object_to.parameterIdentity
```

Figure 5. parameterMapThroughObject class for constructing objects

6.2.2 Hosting the code on a Pythonanywhere server

On its own, Django only provides a local API to interact with, so a server was set up with Pythonanywhere which allows Python code to be hosted online. This allows anyone with the URL to send information to the Huginn database as well as view what is currently being stored (Figure 6). By hosting Huginn online, a collaborative platform was developed meaning multiple colleagues or even stakeholders can participate in the same system.

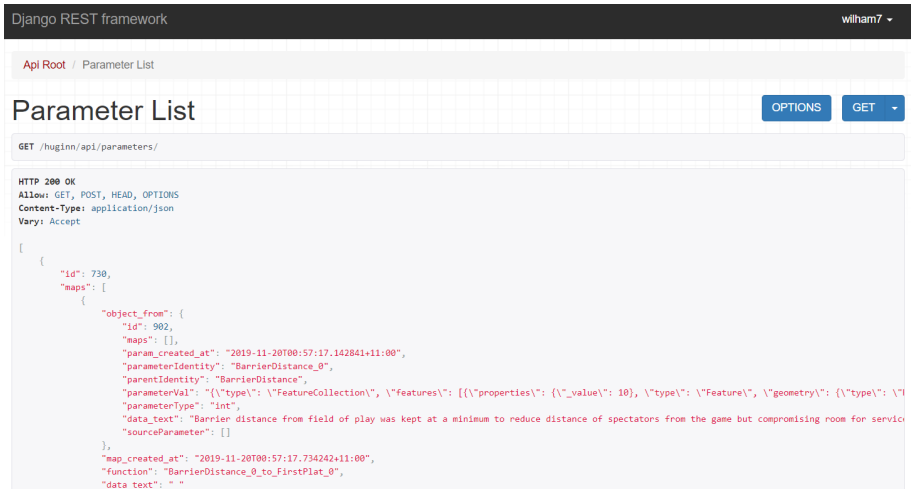

Figure 6. The Huginn database

6.2.3 Posting and mapping objects on Huginn

To get information onto the database, a ‘post’ request must be made in which the database checks if the information is the correct type for it to accept. Initially, to test if the database was working, objects were posted directly from the Django API. The required input variables are documented in Figure 7. The variable ‘Data text’ is where the DDR linked to the object is stored. Once the localised posting method had been resolved, a service called Postman was used to test sending a post request from an external computer.

Figure 7. The input to post directly from Huginn

6.3 DEVELOPING THE HUGINN SUITE FOR GRASSHOPPER AND A WORKFLOW FOR BRIDIGNG DATA BETWEEN THE TWO PLATFORMS

6.3.1 Posting text from Grasshopper

Once Huginn was capable of receiving data inputs, work was able to simultaneously commence on the development of the Grasshopper suite. This was started by manually typing a JSON formatted text box (Figure 8) and feeding it into a GHPython component, a feature which allows the writing of Python code within Grasshopper. Figure 9 depicts the Python script that executed a post request, successfully sending data from Grasshopper to the Web.

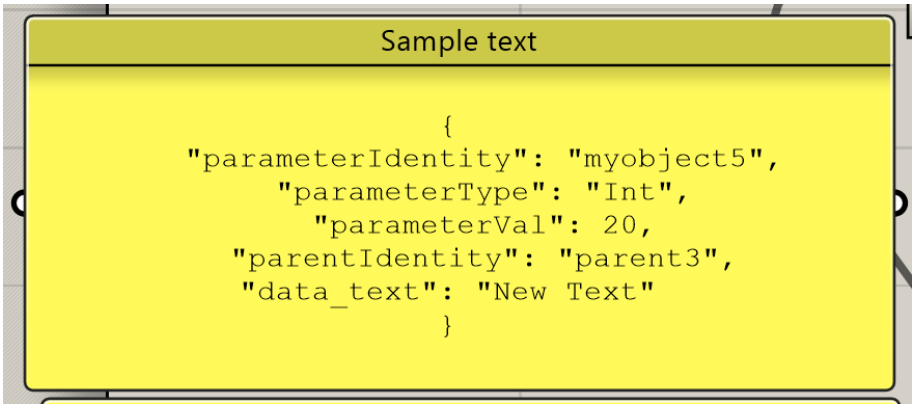

Figure 8. Test data sent from Grasshopper to Huginn

```
try:
    data = urllib.urlencode({'data' : data})
    req = urllib2.Request(url, data)
    response = urllib2.urlopen(req)
    a = response.read()
except Exception as e:
    a = str(e)
```

Figure 9. Python code to post data

6.3.2 Converting geometry to a text format

To send geometry objects with assigned decisions to the Huginn database, they first had to be converted into a text format that could be stored within the JSON that gets posted. Building on the existing work of Cox, a component was employed that uses GHPython to convert geometry into ArchiJSON files. These are JSON format files with specific instructions on how to construct a geometry including its data type and the coordinates of the points used to construct it. Due to time limitations of this research, the component was only capable of processing certain geometry types (Table 2), simply reproducing the name of the data type if it couldn't deconstruct it into core elements.

TABLE 2. List of data types the ArchiJSON component can process

Accepted Data Types	Incompatible Data Types
Point	Mesh
Point collection	Surface
Curve	Polysurface
Curve collection	Point cloud
Plane	
String	
Boolean	
Integer	
Float	

6.3.3 Creating the components for a 'plug-and-play' workflow including the documentation of DMH

In order to make the workflow intuitive and practical to use, it was essential to simplify the amount of manual inputs required. By using an assortment of native Grasshopper components, most of the information required by the Huginn database could be automatically extracted from the input geometry. This was packaged into a cluster called object_to_Huginn that had only three inputs as follows:

1. The geometry or data
2. The name of the object
3. The decision making process associated with the object

The inputs were collated and packaged into a JSON format that is posted to Huginn. A second cluster called map_to_Huginn was developed for creating the mapping between objects and their histories, only requiring two inputs as follows:

1. Parent object
2. Child object

6.3.4 Deploying the Huginn Suite for Grasshopper into the stadium script

After verifying the success of the Grasshopper clusters and refining them through a series of iterations to improve simplicity of use, the workflow was retroactively inserted into the stadium script. Every decision that affected a choice of outcome was documented and linked together within the Huginn database. An example of the mapped DMH tree for a given object is shown in Figure 10. Deploying the workflow into the stadium script highlighted several issues with its state, including poor syntax for naming conventions and extremely slow posting when handling large quantities of objects which resulted in several new iterations of code being written.

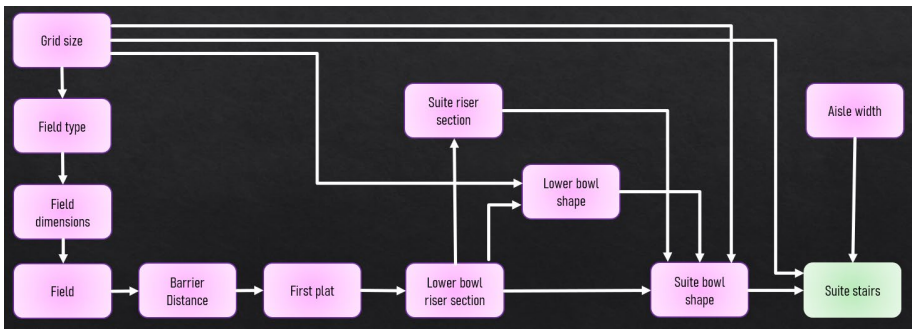

Figure 10. Example DMH tree of 'Suite stairs' object

6.4 PROCESSING, UNDERSTANDING AND COMMUNICATING THE DATA

Once the database had been compiled, it was then possible to identify any object and then observe all the linked prerequisites. This was achieved by calling a function which was only searchable via the localised Huginn ID which is randomly assigned, making it an impractical method of returning data. Furthermore, when an object was searched it provided all the variables for each of the prerequisite objects, which although useful for the preservation of data, could easily reach a surplus of 1000 lines making it difficult to process as a human reader.

Given this result, some additional Python code was written to iterate over the DMH map, just producing the name of the objects and their associated history in a comma separated values (CSV) file. With some simple post-production this information was able to be formatted into an easily readable table recalling all the relevant DMH of an object throughout a script (Table 3). This final step of the case study proved that it was possible to extract DMH embedded in a script and format into a series of prose for the consumption of users and designers.

TABLE 3. CSV table of full DMH of the 'Suite stairs' in the stadium script

ID	Parameter Value	Decision Making History:
Centrelines:	2x Lines	Location of site: Defined by client
Grid size:	15.3	Defined by the engineer due to column layout
Field type:	Soccer field	Defined by client
Field dimensions:	110 x 75	FIFA international requirements
Field:	Untrimmed surface	Shape defined by field type
Barrier distance:	10	Barrier distance from field of play was kept at a minimum to reduce distance of spectators from the game but compromising room for services in this area
First plat:	43x Brep	Barrier on the first riser is 400mm to avoid vision obstruction
Lower bowl riser section:	Closed Polyline	A C-Value of 0.085 was used for plat setout in lower bowl to balance reduced cost and size with quality sightlines
Suite riser section:	Closed Polyline	A C-Value of 0.1 was used for plat setout to give suite seats a premium viewing experience
Suite bowl shape:	43x Brep	Pre-cast risers used to reduce construction costs
Aisle width:	125	Set to be the minimum size still compliant with egress safety regulations to maximise space for seating
Suite stairs:	(1) 863x Brep	If plat step-up >= 300mm, use 2 steps

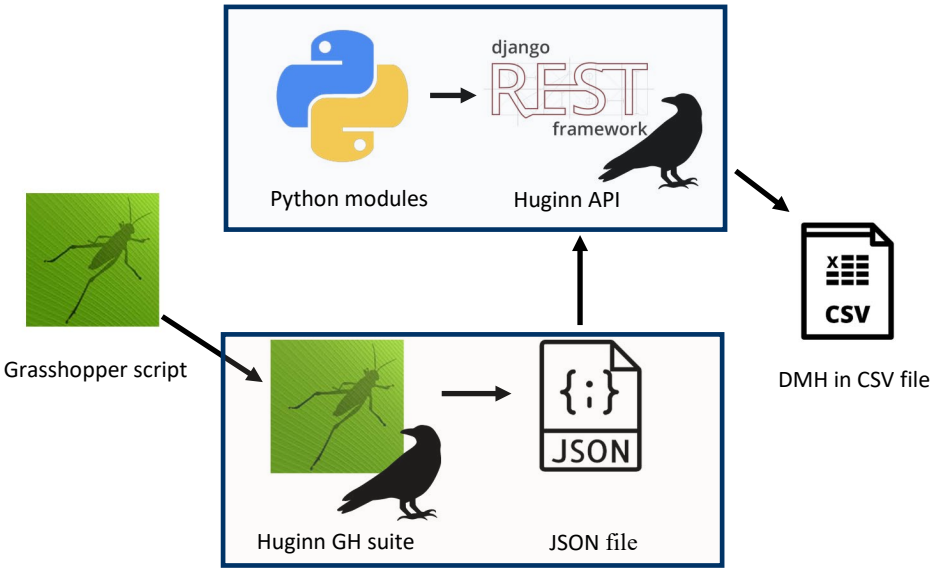

Figure 11. Overview of the complete Huginn workflow

7. Discussion

Through a series of iterative testing this research has successfully demonstrated the possibility to develop a system in which embedded decision making can be documented and recalled on demand. The viability of hosting geometry in a JSON text format in an online database has been shown along with the ability to assign decisions to specific objects. Perhaps the most critical and significant discovery of this research was the showcasing of the potential to draw links between decisions throughout the span of a project in an environment where numerous stakeholders can engage with the data. These discoveries have significant implications for future research potential as well as considerations for reformations in scripting protocols within industry, however the limitations of the research in its current state must also be acknowledged.

The most substantial constraint of this research was the 10-week timeframe in which it was conducted. Many challenges could have been overcome given more time, however, this also presents opportunities for further research and development. Given a longer time frame, an invaluable step would be to assess the effectiveness of the developed workflow and its potential to enhance comprehension and inform design. While it has been proven that it is possible to trace DMH, there has been no exploration into the usefulness of the information in a design workflow or the usability of the current interface. Iterations of user testing, surveys and comparisons would

be able to contribute an assessment of the practical applications and viability of this kind of system. In addition, there was an initial intent to incorporate the output data within a Rhino interface where the data could be viewed attached to the relevant geometry (Figure 12). Given data can be pulled back from the Web, this kind of system would be easily implemented given addition time and some skill in User Interface design.

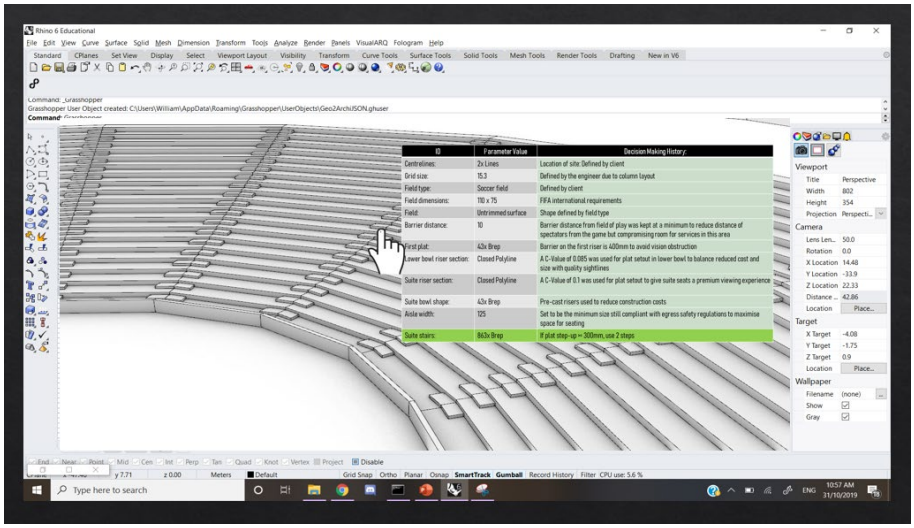

Figure 12. Mock Rhino interface design

Perhaps the biggest limitation encountered without an immediate proposed solution is the automation of logic based decisions. Throughout the research, it was discovered that decisions could be sorted in to two categories; human made and logic based. An example of a human made decision is “This will be a soccer field as opposed to an AFL field”. A logic based decision, while still initially defined by a human, is a criteria assessed by the script such as “According to regulations, maximum step height is $\varkappa$, therefore this stair should be broken in two”. In the current workflow, the need to document this manually is impractical and confusing, suggesting this needs further investigation.

Regardless of the current constraints, this research has far reaching implications for the future of scripting and coding for the AEC industry. Through the confirmation of additional testing, a need to document decision making and having the information accessible to designers could prove to be invaluable with the growing demand for evidence-based design (Criado-Perez, et al. 2019). Along with increasing the trust of colleagues, this research could be expanded to explore the implications of using a DMH tracing system between stakeholders as a means of ensuring accountability for design decisions. This could open investigation into the implementation

of blockchain or other data protection methods for the built environment to ensure the legitimacy of a DMH tracing system if it were to be used as a binding collaborative platform.

8. Conclusion

Using a Web-hosted relational database allows for the collection and communication of invaluable DMH that is otherwise lost within the complexity of architectural scripts. Widespread aversion in the AEC industries to engaging with poorly understood computational techniques is paradoxically occurring parallel to an era of unprecedented demands for data and evidence-based design to inform practice. This study has explored the viability of using a workflow within a localised scripting environment to document DMH that is accessible remotely by colleagues and stakeholders. Such a system serves to provoke thought and further investigation into addressing the friction point derived from the challenges of technical complexity and the need for access to data. By developing a workflow for sending DDR linked to objects from Grasshopper into the Huginn database and returning the relevant DMH of a selected entity, this research has proven the possibility of integrating a similar system into industry practice. The ability to view the DMH of elements in a script not only enhances the comprehension of a script's function but provides a tool to justify and defend decisions through the lens of data driven and evidence-based design. The contributions of this paper provide a definitive step towards a solution that simultaneously demystifies the complexity of scripting while communicating the significance of data and decision making within architectural workflows. In doing so, this project has laid the groundwork for future developments that may eventually overhaul how architects approach scripting procedures, empowering the designer with an arsenal of easily accessible data, reforming how we sculpt urban space.

Acknowledgements

Thank you to Cox Architecture for the workspace and research time provided to work on this project and an especially large thanks to Andrew Butler who provided an invaluable contribution of ideas, technical assistance and guidance through this research process.

References

Arroyo, P., Tommelein, I. D. & Ballard, G., 2012. Deciding a sustainable alternative by 'Choosing by Advantages' in the AEC Industry. San Diego, Proceedings of the 20th Annual Conference of the International Group for Lean Construction.

Austin, M. & Wajdy, Q., 2016. I'm a visual thinker: rethinking algorithmic education for architectural design. Melbourne, Proceedings of the 21st Annual Conference of CAADRIA, pp. 829-838

Azhar, S, Ahmad, I & Sein, MK 2010, 'Action Research as a Proactive Research Method for Construction Engineering and Management', Journal Of Construction Engineering And Management-Asce, vol. 136, no. 1, pp. 87–98.

Balaban, Ö., Kilimci, E. S. Y. & Cagdas, G., 2012. Automated Code Compliance Checking Model for Fire Egress Codes. Prague, Proceedings of the 30th Annual Conference of eCAADe, pp. 117-125

Berry, J. & Park, K., 2017. A Passive System for Quantifying Indoor Space Utilization. Cambridge, Proceedings of the 37th Annual Conference of ACADIA, pp. 138-145

Burry, M, 2011, Scripting cultures : architectural design and programming, Wiley, Chichester, UK.

Carpo, M., 2018. 'Excessive Resolution: Designers meet the second coming of artificial intelligence', Architectural Record, vol. 206, no. 6, pp. 135–136

Chachere J., and Haymaker J., 2011. Framework for Measuring the Rationale Clarity of AEC Design Decisions, ASCE Journal of Architectural Engineering, 17(3), pp. 86-96.

Cichocka, J. M., Browne, W. N. & Rodriguez, E., 2017. Optimization in the architectural practice An International Survey. Hong Kong, Proceedings of the 22nd Annual Conference of CAADRIA, pp. 387-396

Criado-Perez, C., Collins, C.G, Jackson, C.J., Oldfield, P., Pollard, B., Sanders, K., 2019, Beyond an 'informed opinion': evidence-based practice in the built environment, Architectural Engineering and Design Management

Durmisevic, S., Ciftcioglu, Ö. & Sariyildiz, S., 2001. Quantifying the Qualitative Design Aspects. Helsinki, Proceedings of the 19th Annual Conference of eCAADe, pp. 111-116

Gabel, D., 1995. An Introduction to Action Research. San Francisco, National Association for Research in Science Teaching (NARST).

Gardner, N., 2018. Architecture-Human-Machine (re)configurations - Examining computational design in practice. Lodz, Proceedings of the 36th Annual Conference of eCAADe, pp. 139-148

Guedes, Í. & Andrade, M., 2019. Automatic Rule-Based Checking for the Approval of Building Architectural Designs of Airport Passenger Terminals based on BIM. Porto, Proceedings of the 37th eCAADe and 23rd SIGraDi Conference, pp. 333-338

Heumann, A., Davis, D, 2020. Humanizing Architectural Automation: A Case Study in Office Layouts, Impact: design with all senses : proceedings of the Design Modelling Symposium 2019 1st ed. 2020., Springer, Cham. pp.662-670

Hua, H. & Jia, T.-L., 2010. Floating Bubbles: An agent-based system for layout planning. Hong Kong, Proceedings of the 15th Annual Conference of CAADRIA pp. 175-183

Joseph, D., Kim, A., Butler, A. & Haeusler, M. H., 2015. Optimisation for Sport Stadium Design. Hong Kong, Proceedings of the 20th Annual Conference of CAADRIA, pp. 573-582

Kemmis, S. 2009. Action research as a practice-based practice. Educational Action Research. 17, pp. 463-474

Kushwaha, V., 2016. Contribution Of Building Information Modeling (BIM) To Solve Problems In. International Research Journal of Engineering and Technology, 3(1)

Leitão, A. & Santos, L., 2011. Programming Languages for Generative Design: Visual or Textual?. Ljubljana, Proceedings of the 29th Annual Conference of eCAADe, pp.549-557

Leung, E., Asher, R., Butler, A., Doherty, B., Fabbri, A., Gardner, N., Haeusler, M. H., 2018. Redback BIM: Developing 'De-Localised' Open-Source Architecture-Centric Tools. Beijing, Proceedings of the 23rd Annual Conference of CAADRIA, pp. 21-30

Lewis, K. E., Chen, W. & Schmidt, L., 2007. Decision making in engineering design, s.l.: New York: ASME.

McNiff, J., 2013. Action Research: Principles and Practice. New York, Routledge, 3rd Edition

Nagy, D. et al., 2017. Project Discover: An application of generative design for architectural space planning. Toronto, Proceedings of the Symposium on Simulation for Architecture and Urban Design Article No. 7.

Pardo-Guerra, 2012. Financial automation, past, present and future. In: K. K. Cetina & A. Preda, eds. The Oxford handbook of the sociology of finance. s.l.:Oxford University Press.

Patlakas, P., Livingstone, A. & Hairstans, R., 2017. An Automated Code Compliance system within a BIM environment. Rome, Proceedings of the 35th Annual Conference of eCAADe, pp. 153-160

Peng, C. et al., 2000. Recording and managing design decision-making processes through an object-oriented framework. Nijkerk, 5th International Conference on Design and Decision Support Systems in Architecture and Urban Planning, pp. 289-306

Stott, R., 2015. 150 Weird Words That Only Architects Use. [Online] Available at: https://www.archdaily.com/775615/150-weird-words-that-only-architects-use [Accessed 6 October 2019].