arq (2017), **21.1**, 65–73. © Cambridge University Press 2017

Nathalie Bredella talks to Robert Aish, a major figure in the development of new computational technologies in architecture since the 1970s.

The evolution of architectural computing: from Building Modelling to Design Computation

Robert Aish and Nathalie Bredella

In recent decades, the interrelations between architecture and computation have fostered new design concepts and design methods, which challenge existing design and building processes. Specifically, the increased use of parametric methods and scripting allow for the development of modelling and fabrication techniques, which in turn challenge the role of drawing as one of the main tools for conceptualising and realising architecture. In 1975 Chuck Eastman proposed the conceptual basis for a more complete and effective role for the use of the computer in architecture with his 'Building Description System'.[1] During the 1980s, software developer Robert Aish contributed to the development of RUCAPS – one of the earliest applications based on the concept of 'building modelling'. Later developments of building modelling envisaged further modes of collaboration between architecture, engineering, and construction in the (AEC) industry. Jerry Laiserin pointed out in his introduction of the first Building Information Modelling (BIM) Handbook, that Aish's description of 'building modelling' published in a paper in 1986, was 'the first documented use of the term Building Modelling in the English language', which 'set out the augments for what we now call BIM and the technology to implement it'.[2] In his paper 'Building Modelling: The Key to Integrated Construction CAD', Aish put forward a systematic description of the functionality of the Building Modelling systems and pointed to the ways CAD systems determine our thinking about building.[3] In light of more recent discussions on the impact of digital models that simulate the design and construction environment the following conversation addresses how Building Modelling came about, and the ways in which software engineers were looking at architecture at the time. We also discuss the transition from Building Modelling to parametric design and how the ways of design thinking offered by algorithmic tools introduced new opportunities and challenges for architecture.

Nathalie Bredella (NB): Professor Aish, you were involved in the development of the first Building Modelling systems during the 1980s. In your opinion, what are the intrinsic characteristics of Building Modelling, and how were Building Modelling systems conceived?

Robert Aish (RA): Developing a computer application is quite similar to any other design task – for example, designing a building. The software developer has to combine pre-existing ideas about software engineering, computational geometry, human-computer interaction, and data structures with his insights into the application domain – in this case the architectural design process. In observing the application domain, the software developer might ask a number of questions: first, what exists? This might refer to tangible artefacts – such as a drawing – or something more intangible – such as concepts and ideas, which the architect creates through the drawing. This might even refer potentially to ideas and concepts, which are manipulated independent of the drawing. Second, the software developer might ask people who work in this application domain to describe operations they perform. Third, the software developer might ask what the expected consequences are. As such, the prior state (the inputs), the transformations (the processes), and the subsequent state (the outputs) gain relevance to the software developer and often end up becoming encoded into the applications. The concepts and operations encoded in the software then influence the use of the application and effectively encourage or institutionalise a particular way of thinking and working. The perceptive software developer does not simply encode existing representations and processes, but rather imagines and realises new alternative representations and processes that harness computation possibilities and give some practical or creative advantage to the user.

NB: In your paper 'Building Modelling: The Key to Integrated Construction CAD' you talk about the impact of software on our understanding of buildings. How would you describe the interrelationships between architectural knowledge and software practices?

doi: 10.1017/S1359135517000185

https://doi.org/10.1017/S1359135517000185 Published online by Cambridge University Press

RA: Well, the more perceptive software developer might take the following into account: what is it about the entities and operations of CAD systems that reflects some underlying cognitive structure of the potential user? What skills must the potential user possess to be a successful operator of the application? If I created an application for this potential user, what aspects of her or his conceptual knowledge would be automated by the application I am proposing to develop? What new concepts or new tasks will the user have to acquire in order to become a proficient user?

In the context of developing an architectural design application, the software developer is evaluating existing visualisation practices and the way these underpin the conception, or the description, of a building. While developing new applications, these insights and potential advantages need to be balanced against the new skills, which might be required by future users of the application. The perceptive software developer understands that for her or his application to be adopted in architectural practice, the prospective users will want to assess the balance between the short-term 'costs' – such as the disruption associated with learning how to use a new application – against the long-term benefits, for example improvements in design quality, competitiveness, and productivity.

NB: Can you explain what assumptions software developers made about the representation of architecture? How did these assumptions, as amplified by the software, impact the way we understand building?

RA: The most important assumption in Building Modelling is the idea that a building could be modelled as an assembly of components. We now realise that this assumption was based on a very specific way of thinking about architecture. Of course, it was a convenient abstraction that could be shared by both the architects and the software developer. On the plus side, it offered considerable advantages to the architect, particularly in terms of data management in construction and it was reasonably easy to implement with the available software development skills, geometry modelling, and database techniques. However, there are a number of researchers and practitioners who now consider what we might call the Building Modelling assumptions – that a building could be modelled as an assembly of components – to be highly limiting and an unwarranted simplification. For one thing, it reverses the natural order of the architect's design process and forces the architect to think about the design or the selection of the components before the overall form of the building has been explored. This issue is still relevant today, but we need to appreciate that this criticism of building modelling dates back to the 1980s. Even in the case of RUCAPS, there were already a number of researchers who suggested that Building Modelling was quite limiting.

NB: Could you describe some of the applications developed by software engineers who were working during the initial period, and the challenges they faced working within architecture?

RA: Of course, the first effective demonstration of Computer Aided Design was Ivan Sutherland's SketchPad system at MIT. The key idea was later systematically described as the 'model-view-controller' paradigm. In the case of SketchPad, the 'view' was the computer graphics screen, the 'controller' was the light pen, and the 'model' was the constraint solver. During the 1970s, software developers were developing new computer 2D drafting applications, which they thought would reduce the time-consuming drawing process, as well as ease other tasks such as the revision of drawings. These applications allowed architects to draw lines, edit them, and to plot out new pristine versions. These systems used the 'view' concept (the display) and the 'controller' concept (the mouse) from the model-view-controller paradigm. But what happened to the 'controller' concept? Of course these 2D drafting systems provided some convenience to the architectural user, but the computer system had no knowledge, no model of what was being designed. These systems were variously described as an electronic pencil or as a graphical word processor. Therefore, the idea of 'aiding' – which is central to the idea of 'computer aided design' – was lost, at least for a while. We might claim that these 2D drafting systems democratised technology because they enabled many more users to harness some form of computation, but effectively these applications were a shadow of SketchPad's potential: a completely neutered version of the model-view-controller paradigm.

NB: What where the consequences of the more widespread use and accessibility of digital technologies on future software development?

RA: It is difficult to argue against the general idea of the democratised technology, but that general principle should not inhibit a thoughtful evaluation of the different technologies being democratised, and the consequences of this democratisation. The problem is that each time a technology is created and becomes popular, the use of that technology institutionalises a certain way of working – indeed a certain way of thinking. This is because the tool starts to become a standard way of operating, and therefore people start to think in terms of the concepts and operations that the tool offers. The availability of 2D drafting systems really extended the use of 2D drawings as a representation of architecture well beyond the point where the architectural profession should have switched to more appropriate 3D modelling techniques. Essentially, these applications (2D drafting and 3D modelling) represent different technology plateaus. The problem is that we get stuck on a particular plateau when we should have moved onto the next plateau. Then, when we do switch, we are late in

https://doi.org/10.1017/S1359135517000185 Published online by Cambridge University Press

'We ought to remember that there were enlightened researchers – principally Chuck Eastman – who were instrumental in bridging the divide between architecture and software.'

taking advantage of the technology on that next plateau.

The problematic nature of the history of the development of architectural computing is characterised by the production of a series of partial solutions. The deployment and use of these partial solutions seems to trap the architect into using software that essentially fails to support the full scope of architectural thinking and creativity.

NB: When looking at the development of this architectural computation, how did software engineers observe the architect's work, and how in turn did architects impact subsequent software development?

RA: We ought to remember that there were enlightened researchers – principally Chuck Eastman – who were instrumental in bridging the divide between architecture and software. In his original paper from 1975, 'The Use of Computers Instead of Drawings in Building Design', Eastman suggested a far more holistic approach to the representation of buildings and envisaged a far deeper engagement of the architect in creating and querying such representations. It is acknowledged that creating a representation of a building in the system, which Eastman proposed might require more upfront effort on the part of the architect. But as a consequence it would enable the model to support much more interesting queries and interrogation. It would therefore return benefits to the architect whose value would be many times the cost of the initial model building.

NB: Already during the 1970s, architectural offices like Skidmore Owings & Merrill were developing their own software. What were the challenges architects and engineers faced with such in-house software advances, and what role did the marketing of software play?

RA: Skidmore Owings & Merrill's Architecture Engineering System (AES) is a very interesting example of architectural software, particularly because the development was undertaken by a significant architectural practice. As I recall, AES had many innovative features, and it was a surprise to the so-called 'industry watchers' that SOM abandoned AES at a critical moment and switched to regular commercially available 2D drafting applications. I am not familiar with the reasons why this decision was taken, but we might imagine that perhaps the senior management did not consider software development to be a core business of an architectural practice. This decision most probably reflects one of the dilemmas faced by those pioneering architectural practices and also engineering consultants who had originally developed their own software. These organisations most probably had to decide whether the software was exclusively for internal use. In the internal use scenario, the software might provide an advantage over competing practices, but the practice would have to bear the complete costs of development. Alternatively, in the external use scenario, the software would be marketed and used by potentially competing practices. This would generate revenue to offset the development costs, but the architectural office would have lost some of the advantages provided by the exclusive use of software.[4]

NB: Could you say something about how the conception of existing drawing based representational systems in architecture were addressed by the early Building Modelling systems such as RUCAPS?

RA: Summarising my 1986 paper, we can now view the development of RUCAPS as a 'proto-BIM' system. It was conceived as a response to the perceived limitations with existing drawing based modes of representation and communication in architectural practice. These limitations can be described as follows: first, a set of drawings is essentially a description of a building using multiple independent and incomplete representations. Second, the different drawings can be changed by different people, or by the same person, at different times. This leads to the third limitation, namely that the drawings within the set may be inconsistent and this might lead to errors in construction. Fourth, there is no mechanism to automatically maintain the consistency of a set of drawings, except in the mind of the architect. Fifth, the set of 2D drawings may not adequately describe the essential 3D geometry of the building.

RUCAPS was conceived as a Building Modelling system and in response to the drawbacks in the use of 2D drawings. Building Modelling systems are based on the principle of the clear separation between defining data and reports that can be derived from the model. In these systems, the defining data is the building model in 3D. 2D drawings are considered derived reports from this 3D model, together with other text-based reports. Of course, drawings as derived reports cannot be edited. If the building design has to be changed, this editing is effectively a transaction, which is applied to the building model as the defining data. When a modelling transaction on the defining data is complete, then all reports – including drawings – can be re-derived. They are therefore consistent at the time of derivation. The defining 3D building model can be shared among multiple users, allowing for real-time multi-user collaboration. The components within the defining 3D building model were time-stamped, so that different sequences in the evolution

https://doi.org/10.1017/S1359135517000185 Published online by Cambridge University Press

of the design could be modelled. These might include the removal or introduction of different components, potentially representing phases in the development or in construction. Components could be computationally generated via definitions written in Basic and with individual components defined by the unique value of driver parameters. The Building Model could be interfaced to relational databases, thus allowing additional reports to be generated based on relational queries, which in turn could be based on the value of defining or reported attributes. The model could also be interfaced to analysis applications.

NB: Jerry Laiserin points out that while the RUCAPS Building Modelling system was applied to the refurbishment of Terminal 3 at Heathrow Airport, it was not until twenty years later that construction of Terminal 5 at Heathrow Airport was cited as one of the pioneering case examples of this technology. Can you explain the network of ideas and industry strategies that led to the more popularised use of Building Modelling today?

RA: There are three other important factors that significantly influenced the adoption of BIM compared to the adoption of the original proto-BIM systems, such as RUCAPS.

First, the cost of computer hardware fell significantly and the performance improved, as described by Moore's law. In the 1980s, the hardware costs to run a RUCAPS software were many times the cost of the hardware to run a 2D drafting system. By the time Revit was available the cost/performance ratio of hardware had fallen so that there was practically no difference between the cost of a computer to run a 2D drafting application and a BIM application. Second, having used 2D drafting systems and gaining firsthand experience of their limitations, many architectural practices were ready to accept the arguments for BIM. Third, there has been a policy shift. Government agencies and other building owners have been persuaded that construction costs and operating costs will be lower if buildings are more appropriately represented as an information model, not only including 3D geometry but also attribute data.

NB: Software marketing certainly had an impact on popularising certain design approaches and developments too. Besides software development, what other factors impacted the more popularised use of software applications?

RA: In retrospect, it is quite amazing to observe that the original arguments for Building Modelling that the developers of RUCAPS first proposed in the 1980s are at last being taken seriously by the majority of stakeholders in the construction industry. The principle argument used then and still relevant today is that architectural design process should be viewed as an information flow system. So while it is possible to retrofit automation to some aspects of this process in isolation (2D drafting) without affecting other aspects, the real benefits only come through a complete automation of the information flow and in viewing this information flow and the actions of the different stakeholders as an information system. What is critical is that this information system approach cannot simply be achieved by technology adoption alone, but rather requires the adoption of a significant process re-engineering by all the stakeholders.

'In retrospect, it is quite amazing to observe that the original arguments for Building Modelling that the developers of RUCAPS first proposed in the 1980s are at last being taken seriously by the majority of stakeholders in the construction industry.'

NB: In light of the different shifts in design, what role does BIM play?

RA: So we can see that on the one hand, there are the technology changes of BIM, and on the other hand, there are important non-technology changes. These include: process changes (creating consistent and detailed design models upfront rather than by relying on ambiguous 2D drawings), organisation and contractual changes (the use of joint ventures rather than adversarial contractual relationships between stakeholders), and policy changes (government and other building owners mandating particular workflows and deliverables). It would be interesting to discuss how these technology changes mesh with the non-technology changes. Would the industry-wide process, re-engineering, contractual, and policy changes have happened without the catalyst of the BIM technology? I doubt it.

NB: You were central to the development of the parametric form-generating platform GenerativeComponents, which allowed for the transformation and generation of geometry based on associative relationships. How were the relationships between the structure of the design process and GenerativeComponents conceived? What impact did GenerativeComponents have for the reconfiguration of the design process?

RA: The critical question with any form of architectural computing is, what is being represented? With conventional systems, such as 2D drafting and Building Modelling, what is being represented within the application is the result of the design thinking. This is the state of the 2D drawing or the 3D building model at the point at which the architect saves her or his work. What is not represented are the various steps within the design process that lead up to this state. As such, the conventional Building Modelling applications only represent the state of the model and not the design process. Apart from some linear undo/redo

https://doi.org/10.1017/S1359135517000185 Published online by Cambridge University Press

command buffer, the sequence of modelling operations that the architect used to construct the drawing or model is lost.

The innovation in GenerativeComponents (and its predecessor, CustomObjects) is that what was being represented switched from the result of the process to a description of the process. This representation of the process takes the form of what we might call an editable, re-executable design history. What this means is this: as the architect designs using the operations within the application, the application records her or his actions. This design history is not internal to the system; it is not hidden from the user, but rather is completely accessible. Subsequently, the architect can inspect this history. She or he can rewind the history of their actions, can edit one action, or remove it and replace it with another action, and then can re-execute the design history. This re-working of the design history is neither simply linear, nor is it based on the sequence of the original operations. Rather, it is a live graph-based system. The graph is effectively a symbolic model of the design logic of the architect. Changing a single point or input parameter may change the whole configuration of the building design. This gives the architect tremendous freedom to explore design options, which would be practically impossible with manual modelling techniques associated with conventional BIM applications. As you can imagine, to implement this new approach to architectural design required a completely different software configuration and performance, compared with a conventional Building Modelling application.

'The innovation in GenerativeComponents (and its predecessor, CustomObjects) is that what was being represented switched from the result of the process to a description of the process.'

NB: And how did the parametric applications change the concept of design, as well as the work process of the architect?

RA: With parametric design, software architects are not directly designing the building. Instead, they are operating on a representation of the design process which, when executed, generates the design of the building. For this change in approach to be successful, the way architects think about designing has to change subtly. If the computational representation is the process, then architects are not just designing a building: they are designing the process.

We can argue that all forms of design involve a certain degree of premeditation. And in the design of the process, architects have to anticipate how they might want to go back and edit and re-execute this design history. Let's explain this by considering the difference between using a word processing application and using a spreadsheet application. The word processor is equivalent to the regular BIM application, while the spreadsheet is equivalent to the parametric design application. With the word processor application we can write anything, and apart from some annoying spelling and grammar checking, the word processor does not know what we have written: it does not maintain any consistency over what has been expressed. With a spreadsheet application, we can define driver or input data in some cells and expressions in other dependent cells that explicitly reference and use this driver data. So when we change the driver data, the dependent cells update automatically. But to get to the point with a spreadsheet where we can do some useful *what-if* experimentation, the user had to invest time in constructing the spreadsheet with the expressions and dependencies between cells before it could be used. And of course, having constructed the spreadsheet, we can now play multiple *what-ifs*, experiment with different input values, and observe the resulting out values. In doing so, we gain some intuitive insight into the logic we have encoded in the spreadsheet, and potentially we get some new ideas about how to refine or extend this logic. We can go back and change the logic of the spreadsheet and continue with our *what-if* exploration.

As with the spreadsheet, and with a parametric design application the architect explicitly has to define the logic and dependencies between input parameters (and defining geometry) and the dependent expressions or geometric operations. But this allows for the architect to explore design *what-ifs* in a way that would be impossible with manual design and modelling approaches. Yes, the architect has to make a greater initial investment and explicitly declare the design logic. But the rewards in terms of design exploration are many times greater than the investment. We can also argue that the process of explicitly externalising and generalising the design logic has a beneficial effect on the architectural process. When the architect is editing and refining the graph she or he is operating at a higher level. And there is the opportunity to think about a range of alternatives, which can then be explored. She or he is not editing one specific solution.

NB: Can you retrace the evolution of design computation starting with drawings?

RA: There is the progression from 2D drafting to 3D BIM and then onto design computation. If we think about this progression from a systems or data flow perspective, we can see four phases in the evolution of architectural computing. In phase 1 (2D Drafting), the architect directly creates drawings, which are used in a craft-based or mechanically-assisted construction process. 2D Drafting has all the attendant problems of the drawings, which are likely to be inconsistent, potentially resulting construction errors. Essentially with 2D Drafting, we are retrofitting advanced computer-graphics technology to an archaic 2D design process. In phase 2 (BIM modeling), the architect creates a 3D BIM model from

https://doi.org/10.1017/S1359135517000185 Published online by Cambridge University Press

which drawings can be extracted, with the advantage that the building is represented in 3D and all the drawings are consistent. However, the construction process is still craft-based or mechanically assisted. Essentially, we are harnessing computation in a more appropriate way, for example using proper data normalisation to deal with construction information, but the final connection to construction still relies on the use and manual interpretation of 2D drawing. In phase 3 (Design computation as a BIM generator), the architect is creating scripts or graph node diagrams as generative representation with which to automatically generate the BIM model. In this scenario however, 2D drawings are still being extracted to drive what is essentially a craft- or mechanically-assisted construction process. Effectively, advanced design computation is being retrofitted to an archaic craft-based construction process. In phase 4 (Design computation as a direct building generator), the architect is creating scripts or graph node diagrams as a generative representation of the building, which can now directly include the generation of g-codes to drive digital fabrication and robotic construction bypassing BIM modelling and 2D drawings. Now for the first time we have a complete end-to-end tool chain and workflow which are truly computational.

NB: Can you give an example of a project that embodied parametric design concepts?

RA: Of course there is the now famous parametric stadium designed by Luigi Moretti, which was part of the *Parametric Architecture* exhibition at the Twelfth Milan Triennial in 1960. But the first fully computational application of parametric design to architecture involved the Grimshaw designed Waterloo International Railway station. What is interesting about this project is that the parametric basis for this design was developed by the architectural team independently of parametric computation, and prior to the availability of this technology. It was only subsequently that their design was reinterpreted using parametric software. So this building demonstrates that ideas of a parametric architecture preceded or were arrived at independently of the parametric software development. If you like, it is a form of technical and cultural bow wave.

The Waterloo building can be described as a series of arches constructed as banana trusses, which all share a common set of design rules based on tangential arcs. The span of the arches progressively increases along the length of the station to accommodate the passengers on the platforms. Therefore the radii of the arcs are increased, but within the constraints of a specific design rule, which required that a consistent tangent arc relationship common to all the arches was maintained. Although each arch is potentially different, they all share a certain commonality. In fact, the architects conceived the building as a set of variations on a common geometric theme. So we have this duality between commonality and variation: what part of the architecture is common; what part varies? With parametric design software, the geometric relationships allow the architect to express the commonality. The different values for the parameters allow the architect to express the variation. What is really significant is that the CAD systems available at that time Waterloo was designed had no facilities to represent these underlying geometric relationships. The only way the architectural team could express this idea was to directly model each separate arch. Of course this representation was incredibly awkward, not just in terms of the effort required initially to construct the arches, but because any change to the underlying rules would have required each of the arches to be remodelled manually. The effort involved effectively became a disincentive to explore design options.

One of the first applications of parametric design was retrospectively to model the Waterloo roof, in which the underlying geometric relationship was recovered, or reverse engineered, from the series of explicit arch models. With the underlying geometric relationship recovered, it was then possible to rebuild the station roof as the architects had originally conceived it: as variations on a common geometric theme. With this model of the geometric design rule properly represented in the computer, it is now possible to explore options, or refine the design, by playing the architectural equivalent to the spreadsheet *what-if*. What I find really interesting is that the idea of the parametric building came from the architects, not from the commercial software developers responsible for architectural applications. In fact, the software used for this retrospective parametric model of the Waterloo roof was based on the ship building software developed by Intergraph.

I was part of the development team for Intergraph's parametric ship building software and I also worked onsite at the Gdansk Ship yard in Poland helping their naval architects to work with this software. What was also extremely interesting was that when we arrived in Gdansk we were handed the parametric design manual of the shipyard. So, the naval architects had already been conceptualising their design and fabrication process as a rule-based parametric system long before the arrival of parametric software.

'One of the first applications of parametric design was retrospectively to model the Waterloo roof, in which the underlying geometric relationship was recovered, or reverse engineered, from the series of explicit arch models.'

NB: According to your view, how did the concept of parametricism change the view of architecture more generally, in particular when historic architecture

https://doi.org/10.1017/S1359135517000185 Published online by Cambridge University Press

was rediscovered and reinterpreted anew under the perspective of paratmetricism?

'Gaudí had used as a series of parametric and geometric relationships. Mark could then rebuild the construction geometry using these relationships. This gave him the opportunity to play 'what-if' with the exact values of the dimensions and parameters.'

RA: During the time when I was part of Intergraph's software development team in Paris, architect Mark Burry visited me. This was his first introduction to fully functional parametric design software. Mark's interest was to recover retrospectively the design principles within Gaudí's drawings and scale models of the Sagrada Familia. His task was to take the remaining models and drawing left by Gaudí, and from these, to create the accurate full-size geometry required for construction. His concern was that it would not be possible to create this construction quality geometry by simply scaling up Gaudí's original models, because such a scaling would amplify errors and imperfections in the models, some of which were damaged. He recognised that rather than simple scaling, a better approach would be to use the remaining models to recover the underlying design principles Gaudí had used as a series of parametric and geometric relationships. Mark could then rebuild the construction geometry using these relationships. This gave him the opportunity to play *what-if* with the exact values of the dimensions and parameters. For example, at critical intersection of these different geometric constructs, the exact parameter values could be adjusted to achieve the 'fit' required. Mark's initial efforts using conventional non-parametric software to recover these relationships and to fit the geometry were exhausting, hence his interests in the ship building software. Of course Mark went on to use parametric software as a form of design archeology, allowing the hidden parametric design concepts of Gaudí to be revealed and the building to be realised in all its glory.

So each of these examples – the ship building design and fabrication process at Gdansk, the roof at Waterloo International, and the Sagrada Familia – were all conceived in terms of underlying geometric relationships and parametric principles, but independently of and prior to the availability of functioning parametric software.

NB: I am interested in whether parametric design concepts were project-specific or more universal. Could you speak to that?

RA: The question for the software developers was how to generalise project-specific experiences to create new parametric design tools, which could be deployed and made more accessible to a wider audience of creative architects. The experience of these three projects was the inspiration for GenerativeComponents, and so it was natural that one of the first test cases we used with GenerativeComponents was to remodel Waterloo. Of course, with the design computation systems such as GenerativeComponents, we adopted a far more architectural approach to the parametric functionality and this brought us into close contact with architectural firms such as Grimshaw, Foster, KPF, and engineers such as Arup and Buro Happold. There were also lectures and workshops at the Architectural Association and MIT, and this extended into the founding of the SmartGeometry Group. From the detailed conversation with these architects and engineers, we were able to understand that architecture is, or has the potential to be, completely open-ended and required computational design tools which were similarly open-ended. As software developers, we shared the views of our architectural colleagues who told us that they simply do not know what a building of the future will be like. Instead of particularly hard-coding architectural rules, the whole principle of parametric design as realised by GenerativeComponents and other equivalent tools, is that the architectural user has to define her or his own rules. GenerativeComponents has no rules of its own. It provides the facilities for the architect to express her or his own rules; once defined, GenerativeComponents will faithfully execute these rules. This frees the architect from mundane modelling operations. Instead, she or he can focus on the higher-level issues of geometric principles and relationships such as proportion, symmetry, adaption, and differentiation and link the resulting forms to building performance and explore issues of constructability.

NB: What impact did this have for your software development?

RA: We agreed not to hard-code assumptions into design computation software that constrain architectural creativity. In regard to this, there are no components, but rather a library of completely general geometric primitives and modelling operations, which the architects can use to create any complex entity they like and with any behaviour they like. It is also up to the architectural user to define the inter-component relationships. As such, using scripting and other techniques, a single line of code or a few nodes in the graph diagram can be used to create tens, hundreds, or even thousands of geometric entities. Parametric and computational design tools require that architects explicitly reveal and externalise the design process, including the interdependencies, expressions, relationships, and design criteria to the computer. As a consequence, architects subsequently have the option to make careful and precise reconsiderations of the design process and allow the parametric application to take care of the re-execution.

https://doi.org/10.1017/S1359135517000185 Published online by Cambridge University Press

NB: With the emergence of new possibilities of software applications, were architectural design software also envisioned as open source programs where different ideas could be assembled and shared amongst a community of users?

RA: This is still an open question. On the one hand, one can argue that in the future there will be fewer applications, which will cover more applications. On the other hand, there is the development of lots of separate applications, each one tailored to a specific task. The problem when relating separate applications is that you have to move your data between applications and accordingly you have to rethink your problem in terms of these other applications. Sometimes no single application has all the functionality required for a particular project, so that the design team has to use many different applications to achieve their design ideas. The problem is that when lots of different applications are used on the same project, the design team can end up getting distracted by data management. So the question is: are we just transferring the geometric result, or are we also exchanging the logic defining how this geometry was constructed? What happens if the different applications have different ways to represent geometric relationships? Now we not only have a question of data management but also of logic re-engineering. Of course there are really expert users who take these kinds of issues in their stride, but unfortunately we are moving away from the ideal scenario where there is a fully comprehensive application, which is easily learnt and which lowers the barrier to access for regular architects.

NB: What I find interesting is that design computation provided a way to represent the intangible geometric and spatial relationships in an explicit form, and to use these rules and relationships to generate the representation of the tangible building fabric. How was this relationship expressed in the software, and what kind of knowledge did the focus on design rules require?

RA: Essentially the affordance offered by Design Computation tools is that they provide both a way for the architect to externalise design rules and relationships as a script or graph and then manage the execution of that script to generate the building description. So there is a twin process: externalisation and execution. The challenge for the architect is that writing such a script or creating such a graph required algorithmic thinking. Instead of just imagining the results and creating a visualisation of the imaginary building, the architect had to decompose the design idea into a sequence of discrete operations. These operations include explicit geometric processes, such as cross products, dot products, transformations, projections, intersections, etc. It may be surprising, but all these operations were actually used within the Building Modelling application but packaged into intuitive commands such as move, copy, and scale. In these applications, the fundamentals of the geometric and computational operations were often hidden from the architectural users in a benevolent but essentially misguided effort to protect the architects from the generality of architectural geometry. But of course, these new design tools throw down new challenges to architects. To work with this type of software requires mathematical knowledge. So future architects, at least by the time they finish their studies, should already have acquired most of the fundamentals of algebra and geometry on which programming and computer graphics are based.

NB: Are there approaches to design that are particularly suited to computation?

'Design computation is only applicable where there are underlying design rules that are worth abstracting into a script or graph, and where there is some anticipated value in executing such scripts and graphs multiple times to explore design options.'

RA: Design computation is only applicable where there are underlying design rules that are worth abstracting into a script or graph, and where there is some anticipated value in executing such scripts and graphs multiple times to explore design options. Of course it is recognised that effort and reward have to be balanced: that is the effort of creating the script or graph has to be balanced against the value of the resulting design exploration. Maybe in the design of a particular building there are no underlying rules. Maybe the architect's intuition is moving so fast and in so many different directions with ideas being imagined and dismissed a second later. Then, either design computation is not appropriate or is not appropriate until there is at least some stability of thought or some design hypothesis, which is worth externalising and worth formally exploring.

Although design computation requires the externalisation of some underlying design logic, we need to dismiss the idea that somehow design computation is opposed to design intuition. Quite the contrary. In fact, the design logic itself may be the result of the architect's intuition. Once this intuition is encoded into the logic of the script or graph, the architect can then execute or exercise this script. But the way he does that – by changing the driver values, by moving the sliders, or interacting with the geometry – is also intuitive. So now we can see that design computation is essentially no more than a way of encapsulating and externalising one aspect of the architect's intuition into an executable form where it can be exercised by another aspect of the architect's intuition. What is fascinating is that design computation becomes a way in which architects can have a conversation between their logical self and their intuitive self. There are only two

https://doi.org/10.1017/S1359135517000185 Published online by Cambridge University Press

constraints: there must be some design ideas, which are encodable as logic and the architects must have the skill to make this encoding.

NB: As computation becomes an integral part of architecture, in your opinion what impact will algorithmic approaches have on the criteria, which are used when evaluating architecture?

RA: It is no longer tenable to be awed by computation and to assume it is some magic ingredient that justifies or redeems architecture. Instead, we need to construct an appropriate critique of computational architecture, where part of the critical process includes delving into how computation has been actually used in the design of specific buildings: what algorithm was used? How was it validated? Why was it thought to be appropriate? How was it used? What other algorithms might have been used to better possible effect? A building where the architectural form and configuration is the result of some highly logical computation is the Velodrome built at the 2012 London Olympics designed by Hopkins

'It is no longer tenable to be awed by computation and to assume it is some magic ingredient that justifies or redeems architecture.'

architects using GenerativeComponents. Here the computed form has been combined with some extremely subtle subjective ideas to create a true integration of computation, building function and aesthetics. When we see a building like this, it provides an opportunity to re-establish a well-known principle that it is not the technology that is important, but the way the technology is used. Not only is there still much to do to develop more effective and complete design computation tools, but there is also still much to do to develop a 'culture of use' so that these possibilities can be harnessed to create a truly computational architecture.

NB: Professor Aish, thank you very much.

Notes

1. Charles M. Eastman, 'The Use of Computers Instead of Drawings In Building Design', in *AIA Journal*, March, 63:3 (1975), 46–50.
2. Laiserin went on to comment that the description and these arguments are practically identical to those currently being used by BIM advocates more than thirty-five years later. See Jerry Laiserin, 'Forward', in *BIM Handbook: A Guide to Building Information Modelling for Owners, Managers, Designers, Engineers and Contractors*, ed. by Charles M. Eastman et al. (Hoboken, NJ: Wiley & Sons, 2008), pp. xi–xiv, p. xii.
3. Robert Aish, 'Building Modelling: The Key to Integrated Construction CAD', in *CIB 5th International Symposium on the Use of Computers for Environmental Engineering Related to Buildings,* Bath (July 1986), pp. 55–67.
4. The UK practice Gollins Melvin Ward Partnership (GMWP) also developed their own software called RUCAPS. But in this case, GMWP adopted a different business model by creating a separate software spin-off company, GMW Computers Ltd (GMWC). Essentially, RUCAPS was the first Building Modelling system, and its principles were laid out in the paper I wrote in 1986.

Acknowledgements
This interview is part of the research project 'Architecture and New Media' conducted by Nathalie Bredella. Thanks to the German Research Foundation (DFG) for providing the conditions and material support enabling it.

Authors' biographies
Robert Aish is Visiting Professor of Design Computation at the Bartlett School of Architecture at UCL, London. He was previously Director of Research at Bentley, where he led the development of GenerativeComponents, and Director of Software Development at Autodesk, where he led the development of DesignScript. He is also a cofounder of the Smartgeometry Group.

Nathalie Bredella is Visiting Professor of Architectural History with a focus on media and gender studies at the Berlin University of the Arts, where she previously led the research project 'Architecture and New Media', funded by the German Research Foundation (DFG).

Authors' addresses
Robert Aish
robert.aish@ucl.ac.uk

Nathalie Bredella
n.bredella@udk-berlin.de

https://doi.org/10.1017/S1359135517000185 Published online by Cambridge University Press