Computation/Performance

Problematizing education_ integrating computational methods in relationship to ‘performance’ within a new undergraduate curriculum

Anna Pla-Catala
IE University, Spain
anna.pla@ie.edu

Abstract. Setting up a completely new architectural academic curriculum for a brand-new school of architecture. Elaborating, critical space within the new curriculum for the teaching and learning of ‘digitalotechnologies’ (DT) through the integration of them with its core subjects, rather than understanding the teaching/learning of such technologies as an add-on set of skills that comes a posteriori. How to articulate the potential of the ‘Computational-Architect’ as a professional capable of being a productive agent within society; that is, capable of adding Value. The nature of such is what it’s at stake here, if we want to avoid to become or be reduced to mere providers of services.

Keywords. Computation; performance; ecology; code; maker.

INTEGRATING COMPUTATION_PROBLEMATIZING EDUCATION
To have the opportunity of setting up a completely new architectural academic curriculum for a brand-new school of architecture is an extremely exciting challenge that does not appear often.

Within this task, even more exciting for the discussion at place here, is the project of intentionally dedicating, as well as elaborating, critical space within the new curriculum for the teaching and learning of ‘digital-technologies’ (DT) through the integration of them with its core subjects, rather than understanding the teaching/learning of such technologies as an add-on set of skills that comes a posteriori.

Besides summarizing the above mentioned experience, this paper tries to also evaluate the outcome of the implementation of such DT after its 5 years of incipient existence (2008-2013). This period is equivalent to getting a Bachelor’s degree, and in consequence, the level of produced work is of a Basic Level. However, and maybe precisely, it is considered especially relevant as is the result of establishing DT’s basic-structural disciplinary seeds at the very foundation of the development of a practice.

The paper will try to evaluate the failures and successes, and the reasons behind those. Examples of work accompany the text.

NUMERIC CULTURE
The question of DT’s autonomy -or not- within architectural making, together with the capacity of this technology to form an independent corpus of work and discourse by itself, is key.
In the light of the European higher education area as prescribed by European policies, together with the practice of an architect being a regulated one by the Government (in its final form of professional associations needed to practice/perform/sign), it is more than legitimate to affirm that there is no actual spot allocated for DT (computation+fabrication) in an undergraduate/eec curriculum today. And hence it is also legitimate to question why that is.

Since the inception of DT into architecture during the 90s, DT have come into play as a radically disruptive form of knowledge for the architectural discipline. Over the years, there has been an increasing acceptance of them on every front. However, the myriad of courses of all sorts of formats that have appeared since then, still place the teaching/learning of DT at the ‘End’ of an architectural academic degree (post-graduate levels) or at most, as a specialization (end of bachelor or workshop).

This paper aims at posing critical questions as well as describing a critical reflection on the education of the ‘digital-native’-architect.

A clear bias is placed on an Education Model as a platform for research (as a focus of investment in education as much as a true cultural and productive innovation), over a model based on teaching as a mere form of transfer of previously accumulated knowledge.

Implementing DT undergraduate program 2008-2013 implied settling down the very foundations of a new kind of culture. What’s important is the creation of a ‘not-analogue’ kind of culture, but a numeric one (Figure 1).

PART 01_COMPUTATION / EDUCATION
‘I think everybody in this country should learn how to program a computer because it teaches you how to think’ (Steve Jobs, The Lost interview).

This academic curriculum aims at establishing the proper intellectual environment for active engagement in building-up a set of skills and techniques based on the embracement of Digital Technologies as the relevant tooling apparatus for current and future generations of architects.

Considering the current socio-economic European current context; i.e. the complete re-organization of the global-economy and the new emerging cultural paradigm slowly appearing in the midst of a crisis that particularly affects spatial practices, the matter becomes:

How to articulate the potential of the ‘Computational-Architect’ as a professional capable of being a productive agent within society; that is, capable of adding Value. The nature of such is what it’s at stake here, if we want to avoid to become or be reduced to mere providers of services.

The underlying guiding principle in structuring this DT-CV, has been from its beginning, a very simple urge to convey to students the relevance of moving on from their initial (immediately gratifying) use of DT as representational tools, into generative methodologies.

Our goal has been to add the necessary skills that allow for both the conception and control of higher geometrical orders as well as software interoperability. But, above all, to convey: Why all this matters, i.e., to be able to fully communicate Why it’s important to expand the limits of ‘conventional’ CAD concepts and ‘user-friendly’ interfaces that, nevertheless, do nothing much more than constrain architectural language through predetermined architectural elements. In short, the aim of this project has been to create the intellectual foundations for a design culture based on algorithmic thinking and digital fabrication (Figure 2).

Epistemology
This alone, and perhaps simply said, implies nonetheless an enormous and unparalleled epistemological change. It definitely is the most radical shift of mentality that has occurred since Modernism and its Post-s... i.e., the fact that within an associative design framework, students are forced to address the problem of how to design and fabricate architectural components that are ‘Programmed’ rather than ‘Drawn’.

Probably the most obvious of the proven consequences of introducing DT in undergraduate archi-
architecture school (programming-fabrication, whether hard-coded or graphic-scripting), are the questions and discussions on ‘process-driven’ design that inevitably and immediately arise. Even if ‘rule-based’ design systems have been mainstream for decades already in some design contexts even in analogue form (Eisenman), there still exists an extremely high resistance (even fear) towards the reformulation of design Authorship and what constitutes such notion today.

This might explain why implementing DT has been (in our specific context) reasonably achievable and successful (by being accepted and willing to implement it) in almost every area of the academic CV.
but design studio, which is the area where we have found most resistance from. Before jumping into obvious criticisms however, this fact might have a very simple explanation. One that lies at the core of the problematics that emerge out of the profound shift in architecture-making due to the impact of DT (Figure 3).

**Instrument vs Method**

If structure, construction and representation classes have welcome DT’s corpus of knowledge in collaboration with their own, it is primarily because parametric modeling, programming and digital fabrication are mainly valued as ‘Instrument’ and not ‘Method’. To be more accurate, as an instrument for improving: a) workflow, b) variable input/output and, c) delivery of precise geometric data to be taken to digital fabrication and/or performance analysis.

Nonetheless, this fact alone we argue, merely constitutes a slight automation device of otherwise traditional and conventional design procedures, bypassing the truly essential foundation of parametric and algorithmic thought.

The degree of control necessary to develop an initial intuitive hunch by means of the hard-core rigor that computational tools entail is such, that the designer must be skilled first, and above all, in the ‘Logic of Design’ of highly complex systems that comprise geometric, algebraic and logical relationships.

As educators, a two-fold task presents ahead of us; on the one side, to keep up with the fast rate development of DT’s as intrinsic to themselves (Computer Science), and on the next, to focus on the relationship with the corresponding culture of ‘use’ within Design Practices. What is key, is how to trigger the combination of ‘Intuition and Logic’ both of ‘Ideas and Skills’ in one single but multidimensional dynamic ensemble.

Experience over the past 5-years has proved that prejudices as to what architectural design ‘is’ or ‘ought to be’ still exist. And the introduction of programming and digital fabrication within architecture’s education has still to overcome an extensive set of deep-rooted classical values. Most surprisingly is the fact however, that these prejudices do not always come from some of the more established layers of the profession (as perhaps expected),
but also, from the collection of ‘a-priori assumptions’ that young candidates arrive at architecture school with . . . not only about the discipline, but also in respect to the the digital, and the radical change that is involved in making a highly ‘strict’ use of what they otherwise have known to be ‘playful’ devices.

At an institutional level and in contrast with the type of architectural education’s resistant attitude we have tried to convey, a couple of non-architectural examples are here worthy of noting. Such projects are born out of a true honest belief in the capacities of computer code and the new epistemological paradigm opened-up by DT. Those are: Code.org [1] in the USA, and the recent enterprise taken on board by Code Club [2] in the United Kingdom (an afterschool voluntary initiative that aims at teaching computer programming to 10 year-old kids).

‘At age 10-11 (on average) children have the necessary numeracy, literacy and logic skills to learn the concepts of coding’ . . . ‘Some might argue that they have these skills even earlier than that. To be blunt, ICT lessons today mainly consist of learning Microsoft Office. Are we raising a nation of secretaries? I sincerely hope not. It’s insulting to children to think they can’t handle something more creative, inspiring and powerful than an Excel spreadsheet’ (Code.org Co-founder Claire Sutcliffe (Geere, 2012))

Such initiatives deserve our deepest respect indeed. It is most admirable to have achieved for computer code to surpass the ‘geek’ community in order to become a Country’s policy for children’s education; a generation, let’s not forget, that will still take 10 years approximately to get to Undergraduate Schooling.

And this is fact alone proves, that architecture schools should stop worrying about how to preserve traditional disciplinary knowledge modes and cease to have a conservative attitude in order to fully (and rapidly!) embrace programming and fabrication, as well as the rest of the vast array of DT.

Because, to put it very simply: These are our New Standards. And as such, this is the responsibility of architecture education today (Figure 4).

**PART 02_COMPUTATION / PERFORMANCE**

Even if the expected resulting final work to be delivered by an ‘Architect’ remains being a physical structure (a built design), it has indeed become more than clear that the contemporary architectural model we all participate in (every agent in the design-to-construction process), is an evolving one
that has become as much cybernetic as material. Computation has given the designer an unprecedented degree of control over the complete spectrum of design-build processes. As a design tool, it is capable of dynamically defining the global coordinates of a generic continuum, to then yield up to a specific (intentional) configuration. The criteria for evaluating which single instance is most suited for a particular design problem, is what drives us to the notion of Performance.

As a measure of the direct output of a driven process, performance is usually conceptualized as the increase-or-decrease in efficiency of such process. Although computation has been incorporated into the discipline of architecture, it has been mainly used for two main tasks: a) to generate complex geometries that intensify the function of the Formal; or, b) instrumentalised as mere optimization device without exploiting its ontological/cultural potential beyond technocracy.

Our mission has been to articulate a digital expertise for the 21C Architect whose practice is of a clear distinction from the one of the Engineer. Hence, computation in relation to performance is evaluated here with an explicit criticism towards statistical and self-referential efficiency models as sole alibis or testing-modes of resulting prototypes.

In biology, epigenetics studies how environmental factors affect genetic function (genotype). Similarly, ‘rule-based’ design processes have at their starting point the definition of a robust ‘genotype’ that can be subsequently refined according to feedback-loops that incorporate further information external to itself.

Ecology is not sustainability. In an effort to reconsider the Holistic ‘intelligence’ formed by the whole complex set of spatial components (digital, physical, material, economic, atmospheric, etc.), computational design ought to develop a model capable of strategically, tactically and synergetically relate to its environment. The utilization of Code as design method acquires full meaning only if it dynamically integrates the affects of the material context in which it develops (Figure 5).

Figure 5
NuDL_Digital Design_First Year.
PERFORMANCE_ (NON)OPTIMIZATION: NEW DISCIPLINARY CODES

Computer Code is the 21st century Architect’s Tooling Apparatus, and as such it is irreducible to any of Architecture’s traditional design taxonomies. Algorithmic thinking has entered design practices to irremediably transform them bottom-up. Code has become policy, albeit not in its restrictive sense but as generative protocol.

Performance needs to be theorized in order to move beyond modernist models based on a functionalist paradigm of efficiency and a mechanical approach to sustainability. Such a reductive notion of performance responds to a linear way of thinking that prioritizes the minimum use of material, structure and energy to fulfill single conditions. Paradoxically, it is also one that is in direct opposition to the non-linear dynamic nature of computational engines.

For the discipline of Architecture not to fully incorporate and reflect the enormous impact that Computer Science has had on all spheres of knowledge, is not a wasted opportunity but a retrograde act. If design research can attempt at becoming
scientific method, there needs to be a complete consistency between technology-tool-technique-episteme.

Although only 5 years old, this academic program has attempted to propose richer and more complex approaches to the generation and evaluation of built forms to the extent possible and by means of diverse methods and exercises of diverse difficulty for an Undergraduate level (Figure 6).

REFERENCES
Kuhn, TS 2012, *The Structure of Scientific Revolutions*, University of Chicago Press, USA.