THESIS FOR THE DEGREE OF LICENTIATE OF ARCHITECTURE

Realtime & Development

modes of knowing design computation in architectural practice

FRANS MAGNUSSON

Department of Architecture and Civil Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2017

Realtime & Development
modes of knowing design computation in architectural practice
FRANS MAGNUSSON

Department of Architecture and Civil Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone + 46 (0)31-772 1000

[Chalmers Reproservice]
Gothenburg, Sweden 2017

ABSTRACT

This thesis examines theoretical, methodological and organisational implications of design computation for architectural practice - from an insider perspective. It also proposes a conceptual model for knowing within this practice, in an approach that interrelates theory and actionable knowledge.

Dsearch is a research & development network focused on the introduction of design computation at White arkitekter, a large scandinavian architectural firm with over 900 employees. The research studies Dsearch from an insider perspective, providing a description and discussion of an organisational entity involved in the paradigmatic shift towards computationally augmented design thinking - design computation - in architectural practice. This entails the introduction of new tools, methods and competencies.

This thesis contributes to the knowledge in architectural design computation practice, conceptualising documented methodology so as to establish a deeper theoretical understanding. Reciprocally it provides the wider field of design theory with insights into the problems and methods that occur in design computation in architectural practice. It also proposes a conceptual model for how to attain and share knowledge in an architectural practice augmented by design computation. This model articulates Realtime and Development as two approaches to time that are distinguished by design computation. The model builds on practical experience intertwined with notions from cognitive science, philosophy of the mind, design methodology, action research, and management- and learning-theory.

Keywords: Design Computation, Design Methodology, Design Theory, Design Systems, Architectural Practice, Action Research, Computational Design, Project Workflow Management

ACKNOWLEDGEMENTS

I want to thank Jonas Runberger, for bringing me into development at Dsearch. What for me began as speculation, has become stimulating work. Thank you for equal parts of avuncular patience and critical friendship - and just plain old friendship!

At Chalmers there is of course a long list of persons I am grateful to: My main supervisor Nina Ryd for bringing her professionalism and scholarly craftsmanship to this project. It seems you can teach an old practitioner a few new tricks! Karl-Gunnar Olsson for early attempts at bringing me to Chalmers and later success in bringing SmartGeometry. Klas Moberg and Mikael Frej for bringing design computation (and me) to Chalmers. Stefano Delia, Daniel Norell and Jonas Lundberg for taking it (and me) to the next level. Fredrik Nilsson for supporting my academic ambition all the way from bachelor level, and for supporting my practice ambition all the way to licentiate seminar. Also for the *Architecture in the Making* research environment, and bringing me into it.

There are also other academic relations I am grateful for: Professor Alexander Styhre for agreeing to lead the licentiate seminar. Tim Anstey for last seminar before licentiate, for patience with a work in progress and constructive advice. Michael U. Hensel for the Changing Shape of Practice symposum at the AHO in Oslo, with the following PhD workshop. Also, for cautioning me of the word this. Andrew Morrisson for seeing a PhD in there. Henric Benesch for Gretsky quotes. Pablo Miranda for Knuth.

At White I would like to thank: Tobias Hesselgren for being part of the Bridge account. Ola Delsson for being part of early experimentation. Nina Borgström for understanding my need to stay at Chalmers and live in Gothenburg. Nonna Klasander for establishing a seminar for practice based research (in practice!), where you get to talk to people like Johan Dahlberg, Elise Gross, Saga Karlsson, Erik Linn, Marja Lundgren, Stefan Lundin and Malin Zimm.

I am of course also heavily indebted to the brilliant people at Dsearch, thank you: Malgorzata Zboinska, Vladimir Ondejcik, Jonas Runberger and the network.

Malin, Siri och Ada for keeping my priorities straight, I love you.

Frans Magnusson Gothenburg, November 2017

APPENDED PAPERS

Paper 1

Runberger, Jonas, and Frans Magnusson. 2015. “Harnessing the Informal Processes Around the Computational Design Model.” In *Modelling Behaviour: Design Modelling Symposium 2015*, edited by Mette Ramsgard Thomsen, Martin Tamke, Christoph Gengnagel, Billie Faircloth, and Fabian Scheurer, 329–39. Springer International Publishing Switzerland.

Reproduced with permission of Springer

Paper 2

Magnusson, Frans, and Jonas Runberger. 2017. “Design System Assemblages: The Continuous Curation of Design Computation Processes in Architectural Practice.” In *Professional Practices in the Built Environment*, edited by Rowena Hay and Flora Samuel, 194–204. University of Reading, UK. https://www.reading.ac.uk/architecture/soa-professional-practices-conference.aspx.

Paper 3

Magnusson, Frans, Jonas Runberger, Malgorzata A. Zboinska, and Vladimir Ondejcik. 2017. “Morphology & Development - Knowledge Management in Architectural Design Computation Practice.” In *Proceedings of the 35th ECAADe Conference - Volume 2*, Sapienza University of Rome, Rome.

CONTENTS

INTRODUCTION

This thesis is written in the middle of things. While undertaking this research education, I have seen my oldest daughter's nervous last day before school, transitioned from a generalist architect to a researcher and developer of design computation, rushed between being home with my youngest daughter at day and teaching digital design methods at night, returned home from international conferences to do some floor sweeping and accounting for the kindergarten co-op, etc.

The research has been conducted through immersed engagement more than cool observation and contemplation. The experiences and questions discussed in this thesis have come up as reactions and reflections to practice, education and teaching - specifically against a background of introducing design computation to these domains. Over the course of my education, this phenomenon of reacting and reflecting on an ongoing situation has become interesting to study in and of itself. I see this as a potential for a practice based knowledge production, in need of methodological development. Doing this double-take, I have come to realise that my position in the middle of things is a strong determinant for what I can study - and how. Fortunately there are several academic disciplines addressing this problem from various perspectives, such as action research, social learning, materialist ontology, and cognitive science.

Automation

When writing this, robotics company Boston Dynamics has just released a video showcasing the latest version of Atlas.* This is a humanoid robot slightly smaller than an adult human. In this video, Atlas quietly and elegantly executes a back-flip from a low platform. This can be compared with their first video - nine years ago - in which BigDog, a quadrupedal robot with a screaming two-stroke engine, just barely negotiated a downhill slope without falling over. I am not particularly scared that Atlas now has acquired the acrobatic skills needed to usurp its human overlords; what is ominous is the speed of its development.

Manual dexterity, along with intellectual flexibility, has long been the high ground of one side in the debate on technological unemployment - the potential threat to human labour from automation. The argument of this faction is that although much of physical labour has already

* https://www.bostondynamics.com/atlas 2017-11-27

been replaced by more or less advanced machines, these exclusively human attributes can never be automated. The workers that are automated away from one industry can use their human talents in another. On my phone, Atlas has just delivered a striking - if not conclusive - counter-argument.

Economists Erik Brynjolfsson and Andrew McAfee detail the theory behind this argument (2014). The increased productivity from automation in one industry, will lower its prices and subsequently increase demand for other goods and services. According to this theory, the result will be a net increase in the overall demand for labour. Brynjolfsson and McAfee also counters that theoretical argument with a question coming from data. Up until the late 1990s employment grew alongside productivity. After that, employment has decoupled from productivity - automation does no longer drive job growth. Their question is: "Which history should we take guidance from: the two centuries ending in the late 1990s, or the fifteen years since then?" (2014, p180). For futurist Martin Ford it is clear that we are facing a societal shift where "..robots, machine learning algorithms, and other forms of automation are gradually going to consume much of the base of the job skills pyramid. And because artificial intelligence applications are poised to increasingly encroach on more skilled occupations, even the safe area at the top of the pyramid is likely to contract over time" (Ford 2015, p252).

Topics such as robots, machine learning and artificial intelligence are addressed also at architectural events, such as the *Design Modelling Symposium* in Versailles 2017 under the theme of *Humanizing Digital Reality* (De Rycke et al. 2017). *ACADIA* - one of the major conference series for computation in architecture - held its 2016 event in Ann Arbor, Michigan under the heading *Posthuman Frontiers - Data, Designers & Cognitive Machines*. *SmartGeometry* - another highly influential symposium series mixing students, practitioners, and academics - will 2018 be held in Toronto. Its theme of *Machine Minds* is introduced like this: "Whether humans are directly collaborating with, or indirectly authoring, such computationally intelligent agents, they have the power to be an active partner or tool in design creativity. Machine learning can blur the traditional relationship between a designer and their tools, each learning from the other,

adapting, and changing their behaviours or beliefs”*. There has also been research into the practical use of techniques such as machine-learning (Davis 2016).

So while technological aspects are finding their way into architectural discourse and practice, I believe that also the political implications of the automation of intellectual labour are worth contemplating. In a world of artificial general intelligence (AGI) - manufactured minds that can cheaply perform all conceivable tasks with super-human skill, at super-human speed - there will be no economical incentive to hire architects, or indeed any kind of human professional. Such a world is weird and hard to imagine - and experts widely disagree on when, or even if, we can expect its arrival. The political, social and economical implications of AGI will be highly dependent on the path humanity takes towards this breakthrough, and it is wholly outside the scope of this thesis to address such a historical paradigm shift.

What we can use AGI for, is as a kind of reductio ad absurdum for the automation of architectural practice. As the job skills pyramid shrinks from the bottom, the concentration of intellectual flexibility demanded from architects will increase as more and more routine tasks are automated. This is the underlying phenomenon that motivates this thesis. While I strongly feel that rote and repetitive tasks have no inherent value, I believe that there lies a risk in eliminating them without a corresponding expansion of scope for architecture. If we regard the architectural discipline as static, the human agency of architects can only be diminished by automation. This scenario will play out as a zero-sum game of eliminating all but the brightest individuals. Understanding and augmenting the cognitive capacities of designers is a core responsibility for design methodology, and in the light of AGI - an imperative for architectural practice. Engaging in computation as a designer is my contribution to an expansion of scope for architecture. Without this the direction is clear - whether or not we ever reach absurdum.

---

* https://www.smartgeometry.org/challenge 2017-11-06

Design Computation

The discipline of architecture has been exploring the implications of computers since the popularisation of cybernetic theory in the 1960s. This new view on man-machine systems influenced architects such as Christopher Alexander (1964), Nicholas Negroponte (1970), and Cedric Price.*

In his PhD thesis on the digital design field, Jonas Runberger points to this moment as the start for the oldest of three intellectual trajectories, each relating computational thinking with architecture differently (2012, p12). Along with many other disciplines, architecture has since been trying to integrate and/or disentangle intra-disciplinary issues and computational. Daniel Davis for instance, argues in his PhD thesis that issues in parametric design "resemble problems software engineers encounter when programming" and that there are lessons to be learnt from the field of software engineering (Davis 2013, p13).

The disciplinary disentanglement has within methodology revolved around the notions of computation and computerisation. Computational design scholars Achim Menges and Sean Ahlquist write in their anthology *Computational Design Thinking* that: "the distinction rests in the approach towards design, rather than in particular skill sets or knowledge" (2011, p11). Where computation increases the amount and specificity of information, the mere computerisation of external processes only contains as much information as is initially supplied. They find a true expression of the computational approach to design already in Ivan Sutherland's pioneering graphical computer system Sketchpad from 1963. "Sketchpad unfolded a logic to capture inter-related geometries, and how associations cause ripple effects in the development and manipulation of form. The description of design was no longer symbolically the representation of the physical elements as they lie against each other but rather a summation of the forces, pressures and constraints which realise the form" (Menges and Ahlquist 2011, p13).

Runberger observes another trajectory emerging in the 1980s. This is a view on the building industry from a product development perspective, emphasising rationalisation of information - main reasons being

* https://www.moma.org/collection/works/864 2017-11-27

productivity and quality assurance. The tangible outcome of this trajectory is the concept of building information modelling (BIM), which has entered the mainstream of the construction industry. The current notion of BIM encapsulates just that symbolic representation of physical elements that Menges and Ahlquist connotes with the computerisation approach to design.

In the early nineties, a disciplinary exploration of new digital architectural processes and aesthetics was triggered by the access to affordable personal computers and software. Runberger marks the so called paperless studio at Columbia University, associated with the pioneering digital architect Greg Lynn, as one initiating factor of this trajectory. Software tools were mainly borrowed from the movie industry resulting in a preoccupation with the animation of form. This early emphasis on the digital potential for form-generation gradually shifted towards material effects and fabrication, with a curiosity for the industrial processes of the auto- and military-complex.

Computation has found its way into general architectural practice via various paths. An early example is the Foster + Partners Specialist Modelling Group, set up by Hugh Whitehead to solve geometrical problems posed by for instance the Swiss Re Headquarters project commissioned in 1997 and finished 2004.* In a 2013 review, Xavier De Kestelier - then co-head, says that the group has grown to a team of more than 20 and "…diversified into two teams, one dealing with computation, geometry and fabrication, and the other with environmental analysis and simulation" (De Kestelier 2013, p24). Christian Derix, writing on the integration of computational design at Aedas, posits that "Computational Design often tends to be either Architecture or Computation but seldom leads to a feasible synthesis between the two disciplines. 'True' Computational Design must sit *in-between* the two fields and therefore demands new standards for design thinking, its professional workflows and the use of algorithms" (Derix 2009, p567). These are two examples of how computational specialist teams relate to their respective general practices, either through increasing organisational specialisation or by working in-between domains, inserting specialist knowledge into design processes via customised tools. In the case of Dsearch that is studied in this thesis, computation

* https://www.fosterandpartners.com/projects/30-st-mary-axe/

is treated as one architectural skill among others - albeit a specialism. This is a highly specific kind of computational skill, adapted to the conditions of commercial practice.

The discussion in this section can serve as a background for the articulation of the definition of *design computation* as it is used within this thesis. In professional and popular discourse, the wider used terms *computational design*, *parametric design* and *algorithmic design* are often mistakenly interpreted as denoting a specific kind of aesthetic. The debate around Patrik Schumacher's concept of *Parametricism* has played a part in this, not least because of Schumacher's polemic prowess and the prolific high-profile work of Zaha Hadid Architects, where he is principal. In *The Autopoiesis of Architecture: A New Framework for Architecture* (2011). Schumacher proposes Parametricism as the appropriate architectural style for our time. Schumacher's definition of style goes beyond aesthetics to include methodology, socio-cultural aspects and knowledge production:"…styles provide medium-term programmes that frame whole clusters and series of works and forge them into a collective effort, where all innovative advances are mutually relevant to each other" (2011, p321). Still, it is hard not to see the production of Zaha Hadid Architects adhering to a highly specific and homogeneous aesthetic.

Putting the word design first is a statement. This makes the wording read as computation for the sake of design, and on the conditions of design. From my practice perspective I also charge it with instrumental value, design computation is a dedication to solve real problems in all their messiness. Another point is to align the term with the computational approach to design from Menges and Ahlquist. Following their discussion I intend to mean that design computing augments the cognitive capacity of a designer to create information. This is distinctly different from computerisation and automation.

Practice

I spend half of my working time in practice. To be specific I develop design computational workflows and methods at Dsearch - a research and development network at White arkitekter AB. The notion of practice has a lot of different definitions and connotations. Therefore I will define my usage of the term within this thesis.

This is especially important since I need to discuss practice from multiple view-points. My main frame of reference when referring to architectural *practice,* is the common distinction is between academia and practice. Below, I will describe White as a *commercial* practice, this is not intended as a value judgement of the politics, aesthetics or skills of that firm - I simply refer to the presence of a client. This presence impacts the organisation and methodology of a design enterprise greatly, compared to an artistic or research-driven practice. The relationship between client and consultant can go beyond conventional monetary compensation; what really defines this relation travels the other direction - the consultant ends up in obligation to the client. In this view, practice can be seen as under constraint - durational, economical, physical, political, social, etc. Practice deals with reality, it is strongly connected to the practical and the pragmatic. There are always trade-offs to consider, never the pure pursuit of a singular aspect.

When addressing Dsearch in relation to *design computation*, I need to employ a wider definition of practice. If architectural practice is defined by a common contractual model, this community is formed around a common domain of interest, mutual engagement and a shared repertoire. This is an example of what management and learning scholar Etienne Wenger calls *Communities of Practice*, social formations that "evolve as shared histories of learning" (Wenger 1998, p87). To keep these two notions separate I will use *practice* and *community* respectively.

Setup

White arkitekter AB is the third largest architecture firm in Europe with more than 900 employees. The headquarters is located in Gothenburg, and there are 15 other offices in Sweden, Norway and England. Founded in 1951 by Sidney White och Per-Axel Ekholm, it is now fully employee owned with 616 share holders of which the 122 partners hold the majority vote.* White has a history of research and development reaching back to the 1970s, when a foundation was established, supporting also actors outside of the company - both from practice and academia (Nilsson and Lundgren 2016). This history

* http://whitearkitekter.com/about-us/ 2017-11-17

has also formed strong ties to academia, especially Chalmers. For instance 3 out of 4 heads of research since 1989 has had dual affiliations - Claes Caldenby, Fredrik Nilsson and current head of White Research Lab, Anna-Johanna Klasander, are all Chalmers professors along with Jonas Runberger, Head of Dsearch and Artistic Professor of Digital Design.

The engagement with academia also includes a small group of industrial PhD candidates, a number of research educated colleagues and numerous part time lecturers and tutors. White is an environment with experience in learning and knowledge development - in practice and academia. This is expressed through an established infrastructure of networks for specialist competence, exhibitions seminars, and internal research funding. The strong presence of research competency also allow White to participate as partners in various research and education projects.

In addition to the knowledge networks, White Research Lab (WRL) comprises four development networks: Light, Wood, Tectonics and Dsearch. These are thematically focused, transdisciplinary responses to societal challenges, building on architectural experience. These distributed networks employ professionals with different specialisms for work in relation to ongoing projects, often through bottom-up initiatives from involved employees. “Light focuses on the architectural qualities of daylight and lighting design, taking into account health aspects as well as energy and quantitative analyses of light in architecture. Wood discovers environmental and design aspects of wooden structures, as well as design strategies. Tectonics is where building and construction details and architectural key features are developed to meet new functions and sustainable solutions.”*

Dsearch was initiated in 2010 by PhD Jonas Runberger, with the objectives of developing and implementing computational design workflows at White, and strengthening the relation between advanced design and research. Experience from over 50 project engagements have established an internal environment for computational development and experimentation. Operations within WRL are based on an internal co-financing model where certain project issues can be

* http://whitearkitekter.com/wp-content/uploads/2017/11/White-Arkitekter-Research-Programme-2017-2019.pdf 2017-11-17

investigated, partially free from the constraints of practice; in this way the risks of developing project specific methodology are mitigated. Dsearch could in that sense be described as a symbiont, able to infuse a project at White with new knowledge and perform new kinds of work while dependent on project knowledge, goodwill and funding. This setup relies heavily on trust and a common understanding of the conditions for each engagement. Project specific problems, findings and developments are then integrated with a continuous method development, facilitating White as a learning organisation. Project engagements and methodology also inform a strategic organisational development of Dsearch itself. Dsearch has also pushed a research agenda from the beginning; Jonas Runberger was writing his doctoral thesis during the first years at White and early Dsearch projects are included in that research (2012). My thesis and the Dsearch collaboration on the included papers continue the research effort.

At the time of writing, the Dsearch core team comprises Jonas Runberger, me, Vladimir Ondejcik and Malgorzata Zboinska. The tradition of sharing dual affiliations between White and Chalmers lives on also here: Jonas Runberger is professor and leads a research group on digital design, and Malgorzata Zboinska holds a PhD in the area of digital architectural design methodology. At Chalmers, she leads the research project *The Architectural Convertibles*. Surrounding the core team of Dsearch there is a wider network of colleagues, pursuing various aspects of design computation, such as advanced geometrical modelling, parametric design, structural and energy engineering, etc. This network is heavily involved in projects and developments, both in the form of active participation and in reflective discourse. Surrounding the network is a large group of colleagues with interest in design computation. This includes regular attendees to seminars, tooling courses and other events as well as collaborators at the various White offices.

Research

The aim of this work is to explore and exploit my dual position as a doctoral student, also working professionally with research and development at a commercial architectural firm. The conclusions presented here are based on experience from design computation in practice, supplemented by a rudimentary model for how this kind of insider

knowing is attained and enacted. The format for this thesis is therefore in part an action research account where I have worked collaboratively with my stakeholders towards a preferred change in practice. The other part is a conceptual modelling exercise employing a diverse literature to critically discuss my experience of this work.

What difference does computation make in architectural practice?

This question drives my curiosity in research and in practice. I try to keep it present at the back of my mind when engaging in often quite specific and mundane interactions in my daily practice. I am not actively pursuing difference, but readying myself to be surprised when it presents itself. The question overlays my attention towards hands-on problems, as well as wide-ranging strategic discussions. As a research question it is too sprawling and and fluid to be satisfactorily answered by any one person at a given moment. My approach is therefore to be flexible and reflexive - specific questions are asked under specific circumstances, and answers are directed towards specific audiences. The initial question can be further articulated in terms of design computation in architectural practice. Questions that I address in this thesis are:

- What new architectural tasks can we identify?
- What new architectural knowledge can we identify?
- What new organisations of architectural practice can we identify?

These questions are all aligned with my theoretical objectives for this doctoral research project:

- Produce knowledge materials relevant for selected audiences.
- Develop my research approach, and articulate it as a model.

The second objective can be used to reformulate the initial question:

How do I find out what difference computation makes in architectural practice?

I also have a set of practice objectives that support the research:

- Gain experience in practice at Dsearch.
- Facilitate collective learning at Chalmers and White.

The collective learning comprises seminars, tutoring, critiques and

lectures. Included in this thesis is a series of papers collaboratively written by Dsearch from a practice perspective. Their objectives and audiences are introduced in the *Discourse* section below. One component of this discourse is the establishment of a vocabulary for the emerging design computation practice. This is instrumental both for practice and research.

These objectives and questions force a choice regarding the design of my research project. My immersion in design driven practice affords me vantage-points of the design process, beyond the scope of a single artifact. In fact, the step away from being in immediate control of the design process - towards facilitation - is itself a design activity that I believe is relevant to study. In my position as research and developer at Dsearch, facilitation is an activity I, within the constraints of practice, can engage with more experimentally. By adopting a designerly attitude in practice here, the research approach could be described as research by design, albeit with the design artifacts being primarily organisational in nature (Hensel 2013).

Setup

I applied for this research education with a letter stating my research interests. This was formulated as a response to my time in conventional architectural practice, in the light of my interest in parametric design and scripting. It presented a scenario for architectural practice - driven by the societal trend towards increased digitalisation described above. It anticipated increased attention towards the augmentation of the conventional capacities of architects, and towards the effort to establish intellectual infrastructures for the facilitation of this augmentation. This polarisation still underlies my understanding of the difference computation makes in architectural practice, and has become part of my own career with my move to Dsearch. In this thesis the polarisation is further articulated through the *realtime* and *development* modes of knowing.

I hold a position as a part time doctoral student in the department of Architecture and Civil Engineering at Chalmers University of Technology in Gothenburg. If the environment at White has academic experience, my supervision team at Chalmers has a good understanding of the conditions for research in and into practice. Examiner is Fredrik Nilsson, professor of architectural theory, and as already

noted - long time head of research and also chief research strategist at White. Primary supervisor is associate professor Nina Ryd, with a long experience researching into the relationship between architects and clients. Secondary supervisor is Jonas Runberger which puts us in a dual relationship; as head of Dsearch he is my closest superior at White. While this fact strengthens the connection between research and practice, it also raises some concerns regarding the autonomy of my research. This issue is addressed in the *Action Research Cycles* section.

I also formally belong to the research environment *Architecture in the Making: Architecture as a Making Discipline and Material Practice*.* This is a Swedish nation-wide architectural research effort, taking interest in the increased complexity within the design of the built environment due to ICT in design and production, sustainable development and re-use of the existing building stock. The architectural discipline as a starting point, with design thinking as means to create "significant contributions to theories and methods in architectural research as well as practice"**. Four research themes are identified: *Material mechanisms*, *History*, *Alteration*, and the one I am connected to: *Investigative modelling*. This theme investigates a situation where: "ICT technologies and digital tools have fundamentally affected architectural practice during the last decades. They have changed the organisation and process of design and have reconditioned the conceptualisation of architectural projects and their materiality"***.

In this thesis I address the issues of organisational and processual change in architectural practice in the *Design Computation Practice* part. In the *Knowing Practice* part, I engage with theme of Investigative modelling as a conceptual and theoretical exercise - building a model for knowing practice based on realtime and development modes of knowing.

---

* http://architectureinthemaking.se/about 2017-11-25

** ibid

*** ibid

Audiences

Hillary Bradbury, editor-in-chief of the academic journal *Action Research* urges all scholars to develop facility in communicating with both *local* practitioners and *cosmopolitan* scholars (Bradbury-Huang 2010). I believe that for my research the audiences need to be further specified. Dsearch is a community among many at White, and supporting knowledge exchange and collaboration between these is core to its mission. There is also a vivid international community around computation in architecture, with a strong tradition of exchange between practice and academia. This means that the practice mode of this research is not necessarily more local than the academic. In some respects communication within the international community is even stronger than within the respective firms or institutions.

On one level my research focuses on developing specific theory, method and organisation in practice; on another it studies how this knowledge is developed in practice. This means that the same issues can be addressed with different audiences in mind. In the design computation community, the same audience may also be interested in both approaches to the same issue. Even if features of the methodological developments at Dsearch are specific to White, that kind of documentation is still potentially relevant for similar practices. Here it is a matter of curating the documentation of practice based and highly action oriented knowledge, so as to be relevant for the new audience. If re-contextualised, this kind of work can provoke reactions and reflections in and on practice that are potentially relevant for a wider community - architectural design computation, architectural practice or practice in general. An example of this would be *Paper 1* that led to further collaboration and development - as described in the *Discourse* section.

In my experience, the growing community of design computation is still in a position where the overarching architectural discipline requires its explanation and justification - both in terms of academic disciplinarity and professional credibility. Very often I find that cultural resistance towards design computation springs from its very explicitness. As a new methodology it needs to explain and justify itself, but it also poses a threat towards established and thus mostly tacit design methodology - that is made explicit by the comparison. One way to address this issue academically is to apply a theoretical

framework external to architecture, in order to discuss new and established ideas in a neutral language. In that way we can articulate the difference computation brings; but more importantly - by making new and old explicit we can also articulate their interrelations. This the approach of *Paper2*.

This thesis is also aimed at an audience specifically interested in knowledge production in practice. A conceptual model for learning, acting and knowing is presented and discussed. Formulated as a model and methodology for realtime knowing, and the development of action oriented knowledge, it is intended to be used also outside of architectural design computation practice. This model is articulated in the *Knowing Practice* part below. It is presented in *the R&D-model* section as one of the research results , but it is designed for use in practice.

Thesis Structure

In previous sections I have provided the impetus for this thesis in the form of automation of intellectual labour, of which I believe we have yet to see the peak. I have summarise the history of engagement with computation within the architectural discipline and justified my belief in its continued relevance. I have detailed my position in research and practice, the research objectives, questions, choices and approach resulting from that, as well as the potential audiences for this work.

The two following parts both present one aspect of the research behind this thesis. Each part starts with a vignette setting the stage.

In the *Design Computation Practice* part, the *Bridge Engagement* provides a 1st-person perspective of a typical social interaction in design computation practice at White - providing concrete context for further discussion. The *Project Engagements* section highlights some recurring issues when introducing design computation in the projects at White. It also presents a strategic device developed by Dsearch to address these issues. The next section provides examples of project, method and strategic *Developments* at Dsearch, and deliberates how these can form a basis for academic research. *Insider Research* is articulated as a form of action research - negotiating the implications of being native to the studied environment as a researcher. The *Discourse* section provides a vocabulary for design computation developed through the research presented in the papers of this compilation thesis. Each paper is also presented in terms of context and impact.

In *Knowing Practice*, the *Realtime Monologue* is highly abstract and composed as a synthesis of the discussion in this part. The Realtime & Development *Modes of Knowing* are articulated as an epistemological basis for knowing practice. *Conceptual Modelling* is discussed as a research activity. The R&D-model is then gradually developed using theories on cognition, design and learning. *the R&D-model* is presented as diagram and text.

The final part provides a *Discussion* on the research questions together with a proposal for furthering practice-based knowledge production, and a potential direction for further research.

Frit Pattern

Tobias comes by my desk with a piece of acrylic in his hand. He wants to talk about a frit pattern for a glazed bridge, connecting the old part of a hospital with a new building that the office is designing. There is a need to regulate visibility between the corridor on the bridge, and the hospitable rooms behind the windows of the adjoining facades, as well as the courtyard below. Other factors are over-heating from solar insulation and daylight quality. Tobias believes that the answer to this delicate balancing of parameters is to accommodate a heterogeneous density of fritting within one distinct motif. It is also important for the energy analysis to monitor the opacity, the ratio of fritted or otherwise opaque area to the overall glazed area.

On the plastic, which is modelled after one of the bridge elevations, is etched a rhombic pattern. Floor and roof slabs are marked with lines along with the structural truss just behind the glazing. Every other rhombus is etched opaque using a standard cad hatch. This model triggers a slight feeling of dissonance in me - but it soon dissipates. Tobias tells me that rhombic patterning is a recurring theme in the existing 1920s building and also picked up in various ways throughout the new design. This fact has led him to consider the diagonals of the bridge truss as rhombi, establishing the large scale of the pattern. Here ensues a technical conversation regarding screen printing on glass, where I share previous experience about printers and their concerns, and Tobias recounts his dialogue with the energy specialist on choice of glass and targets for opacity. Tobias can then conclude that the minimum dot size of the fritting should be rather large compared to most frit patterns. Each rhombus is to be visible in the

interior of the corridor to establish the small scale of the pattern. The relation in size between dot rhombi and truss rhombi leads to the decision that only one intermediate scale level is relevant: rhombus fields, either empty or filled with dot rhombi - just like the etched acrylic of the sketch model.

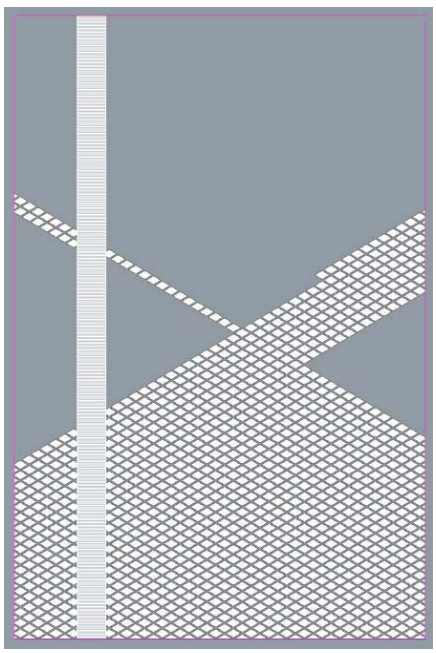
Sketch model

We agree on that a critical issue is to set the size of the dot rhombi so that it is large enough to be visible from the corridor, and small enough to be interpreted as a rhombus field surface when seen from the courtyard. This dependence on human perception makes this issue impossible to test in scale, so Tobias anchors the decision to use Dsearch for support with developing mock-ups. Our first use of design computation is to simply generate a pattern comprising thousands of small rhombi in order to plot a full height pane of glass. The pragmatic testing method involves me and Tobias pinning up various versions of the pattern on a wall, and then slowly walking towards them in order to determine when we can make out individual dot rhombi. While immersed at this level of scale, we also see a need for a kind of framing of the rhombus fields. Made out of continuous lines of dot rhombi, the thickness of these frames also inform the decision of dot size and grid spacing. Our pacing up and down the studio, combined with the giant plots, triggers engagement and spontaneous discussion, so this precious decision is informed by the collective expertise of our colleagues.

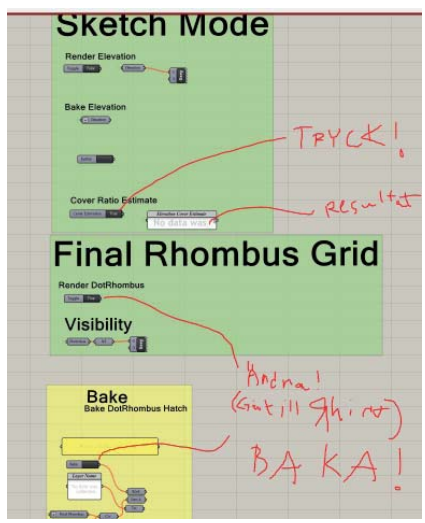
Another starting point is to define the parametric logic of the pattern, as applied to the specific context of the bridge. Tobias sends me a cad file to use as an underlay. The definition of a rhombic grid is a straightforward parametric procedure, however I can't seem to fit the grid into the large scale rhombus made out of the truss diagonals. When I investigate its

dimension I realise that it is skewed. In fact, the whole bridge is skewed due to the fact that it negotiates differing floor levels in the new and the existing building. This fact about the project is of course only new to me, but for the rest of the design team it is not really a fact - it has not caused any systemic problems until now. My problem is that the grid looks unpleasant, or even defective, if skewed to fit the bridge. The other unsatisfying option is to tilt the grid to fit the angle of the floor and ceiling of the bridge. This instead results in problematic meetings at every glazing joint and a failure to cover the vertical members of the truss.

This setback makes me pick up the acrylic model again. When I know what to look for, I now see that while the rest of the bridge is skewed, Tobias's cad hatch is aligned with the world-XY axes. Having just recently been involved in a development where this very issue was meticulously avoided, I readily have the problem definition in mind. The misalignment of the hatch pattern causes its edges to look jagged where it meets the borders of each rhombus field. This is why I felt intrigued when I first looked at it. I suddenly get jolted into action as an interesting line of thought appears to me; if we use the skew as a design concept, how would that play out? I send a screen dump to Tobias and walk over to his desk on the other side of the atrium.



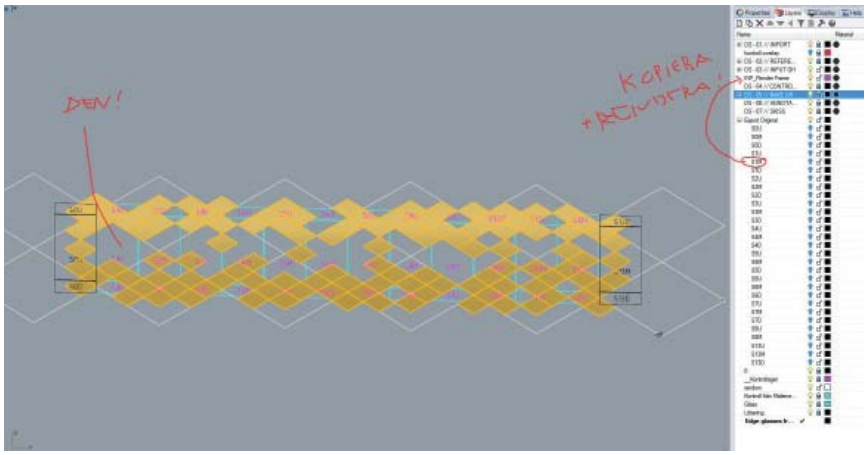
Screen dump



Sketch and Detailed modes

Tobias likes the idea and decides that this is all we need to break the dullness of a too perfect geometric pattern, without losing the restrained elegance sought for in the design brief for the overall building project. This

puts an end to a series of parallel investigations: individual scaling of dot rhombi, random reduction of dots within a field, etc. Instead we focus on trimming the size of dot rhombi and their underlying grid, so that jaggedness occurs from misalignment as desired. The effect is most articulated in the frames between two empty fields, so this part of the solution is where we focus our attention. After a brief stand-up meeting in front of the latest 1:1 plot with the design principal, we get the permission to go ahead with a new phase of the Dsearch engagement. I get a few days to develop a design workflow for Tobias, in which he can compose the pattern on the two facades.



Facade workflow

This work takes place on the actual elevation drawing. Tobias chooses which fields are filled with rhombi and which are left empty. At this stage he continuously gets feedback on the intermediary scale, a visualisation of field- and frame-surfaces along with an estimation of opacity to guide his composition. This feedback informs the composition where Tobias wants to ensure the privacy of the patients lying down in the hospital beds by blocking most of their line of sight between each other, but still allow for sufficient daylight in the bridge.

In the next stage he can execute a detailed process that renders the final print originals in 1:1 precision - ready to send to the contractor. This also triggers the exact opacity calculation for each facade segment. The reason for this two-step process is that the detailed facade drawing contains hundreds of thousands of small rhombi polylines, some of which are cut by the edges of each glass sheet. This makes for a heavy load on the computer. Waiting for several seconds is acceptable for the late stage of producing production documents and verification of measurements, but for the design stage it is vital that the system provides rapid response to the designer.

the Bridge

Project Engagements

The vignette above is an account of how a project engagement between Dsearch and an ongoing project at White can play out - in this case by me and an architect colleague. While each of the circa 50 engagements so far has had unique circumstances and specific aims and conditions, some recurring issues highlighted by the above *Bridge Engagement* are worth further deliberation.

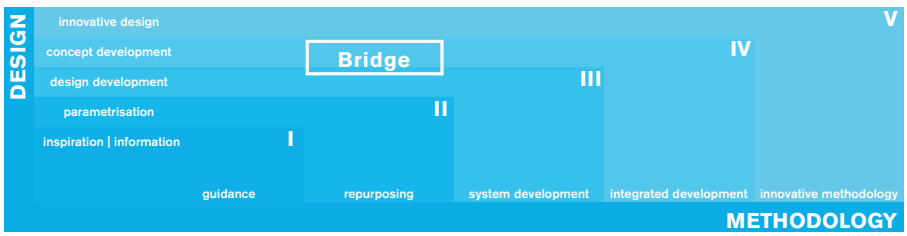
Tobias is part of a specific design discourse taking place in the project; he has a native understanding the full interplay between design considerations and practical issues. What I bring to the situation is the ability to address a specified set of considerations and issues with bespoke tools and methods. As an outsider to the project this set has to be curated for me in some form. Our collaboration is an opportunity to leverage computation from within the design process, where this curation can be interactive and iterative - rather than a one time presentation. The skew of the bridge was discovered as a problem and subsequently informing a solution through a collaborative inquiry of

the project materials. Such a conceptual development would probably not have occurred if computational methods were applied post-factum, as a parametrisation of a fully formalised design concept.

Free-floating rationales is a concept from philosopher and cognitive scientist Daniel Dennett, describing how evolution blindly ascribes organisms affordances in relation to their current context. An organism can find itself fit to survive in a new situation because of a feature that was developed for another 'purpose' in the old situation. "Natural selection is an automatic reason-finder; it 'discovers' and 'endorses' and 'focuses' reasons over many generations." (Dennett 2013, p234) This dynamic is also common in design processes - suddenly a proposal solves a problem that was hitherto unarticulated. The skew of the bridge was embedded as a free-floating rationale in our common material. A conventional architectural presentation could not express this fact, but when a new medium was introduced that rationale could be made explicit and productive. Such integration with also the most abstract levels of the design process, is what I see as the core aspect of *design computation*.

In the Bridge Engagement I develop a workflow specifically for Tobias. I take care to connect a set of capacities in Tobias - to compose a facade pattern responding to the project conditions, with a set of capacities in the *design system* - to produce visualisation and evaluation of the facade, along with detailed instructions for fabrication. The bespoke nature of this development implies that Tobias and the design system form a new whole with emergent properties that are not found in either part. This phenomenon is described in *Paper 2* under the name *design system assemblage*. That name arrived as a consequence of discussing the implications of expanding the notion of the design model. In the Dsearch discourse, terminology has not yet settled for this widened scope. The notion of systems carry undesired connotations of a technical nature, and assemblage is an esoteric word for our (predominantly Swedish-speaking) target audience. Important for Dsearch in this widening of scope, is to emphasise the role of project and development aspects such as design concepts, time constraints, competencies, infrastructures, etc, when discussing methodology in practice. For this thesis, I use the term *workflow design* to describe my contribution in the Bridge Engagement. In addition to the reasoning above, this is to get away from the more narrowly interpreted terms *tool*, *method*, or *project*.

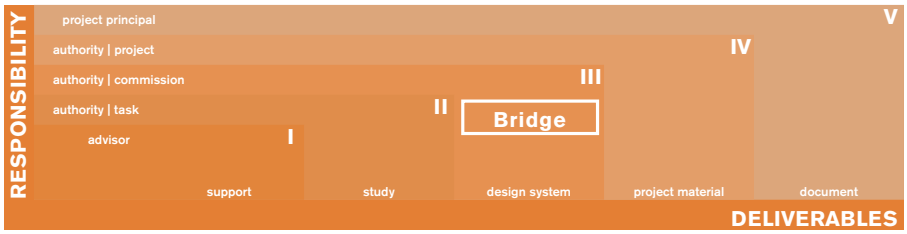
At several points in the Bridge Engagement there are hints of a management level beyond the design process. Commitments are made between me as a Dsearch representative and the project principal - often with Tobias as a spokesperson. The plain fact is that the cost of my time in this case has to be covered by the client of the project. This necessitates a cost-benefit estimation of the *project engagement* with Dsearch. Development within a project follows an investment dynamic where time has to be spent on work that is not immediately productive, but potentially more-so in the end. This dynamic introduces new kinds of risk that are compounded by the introduction of new methodology - especially when involving scripting. In this account, the risk and commitments of the engagement is managed through a series of informal agreements over design scope and time-frame. This kind of trusting relationship is possible by the general culture at White in combination with the personal relationships involved. For Dsearch, development within projects is a matter of aligning the interests of the local project with the global development at White and the conditions for the actual realtime design collaboration.



Method/Design Matrix

As a response to the issues discussed above, Dsearch has devised the *service matrix* in order to determine the ramifications of the engagement with the project principal. The matrix is one of the objects forming the *strategic framework* described in *Paper 1*. The Bridge Engagement is marked in the image above as *concept development* and *repurposing*. This means that Dsearch was involved in formulating the design concept, and that the project development could proceed without development of new methods. Since *Paper 1* Dsearch has also developed a corresponding matrix for determining responsibility in project management and deliverables. The Bridge Engagement would here be considered a *task*, in that the design process was carried out between Tobias and me. A larger or more complex issue would perhaps have necessitated my presence in the

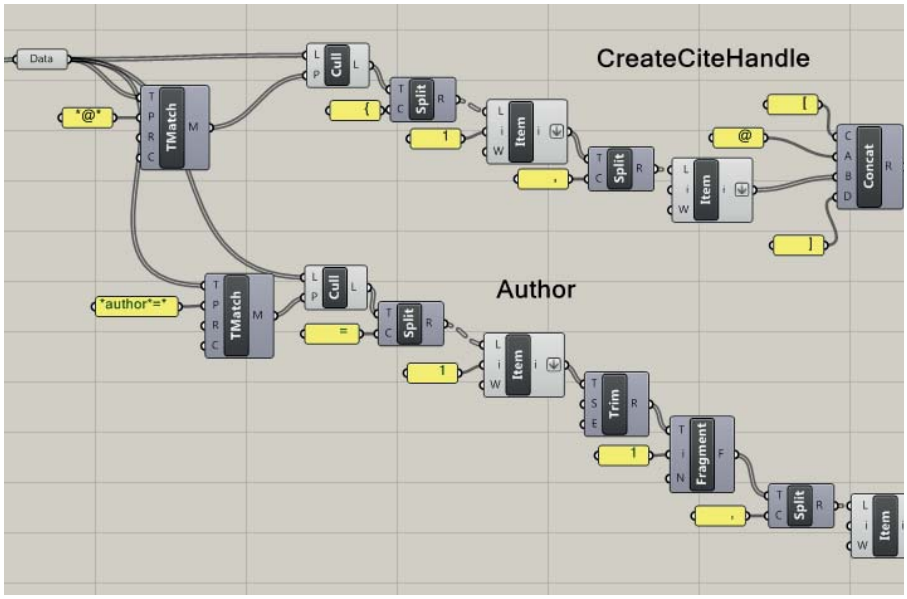
internal meetings of the larger project team - a *commission* level engagement. I delivered a *design system* to Tobias, with which he produced the documents needed for production.



Responsibility/Delivery

Developments

The service matrix provides an example of how specific project engagements are systematically reflected on, generalised into strategic issues and developed into materials that can guide further activity. These materials can be said to embody organisational learning in the sense provided by Zuber-Skerritt and Perry: “a process of collaborative action learning and action research in an organisation with the aims of solving complex problems and achieving change and improved performance at the individual, team and organisational levels” (2002, p172). This form of strategic development is complemented at Dsearch by a more general form of methodology. The current primary platform for this effort is Grasshopper (GH) - a visual programming interface for creating parametric geometry within the widespread 3d-modelling application Rhinoceros (RH). In this environment, geometry can be generated and evaluated by scripts that perform design actions. In Grasshopper terminology, these are called *definitions*.



Grasshopper Definition

Method development at Dsearch entails the documentation and generalisation of project experiences, but also exploration in anticipation of coming project needs. Knowledge sharing occurs, informally through personal support for projects and spontaneous workshops and briefings, and formally through organised meeting- and seminar series as well as written and illustrated method sheets. The variety of formats and venues allows for rich feedback from project work to network, and from informal project conversations to strategic and methodological discourse. Method sheets cover standards regarding project responsibilities, procedures and communication. They also address technical issues such as program installations, guides to certain plug-ins or modelling techniques etc. This introduction to the *Grasshopper Developer* sheet can serve as an illustration:

> This document covers standards and recommendations for working with Grasshopper as a *Developer* in a project at White. Emphasis is placed on documentation and communication.
>
> White has two modes of working with Grasshopper:
>
> - The *User* is a member of a design team at White using a definition as part of a design process. This definition can be developed at Dsearch or within the team.

- The designer of the definition is defined as a *Developer*. This mode brings a larger responsibility to uphold quality standards for project development at White.

> Please make sure that you are reading an up-to-date version of this document by accessing it from White Help. For an overview of available grasshopper documentation at White, please read Grasshopper Help.

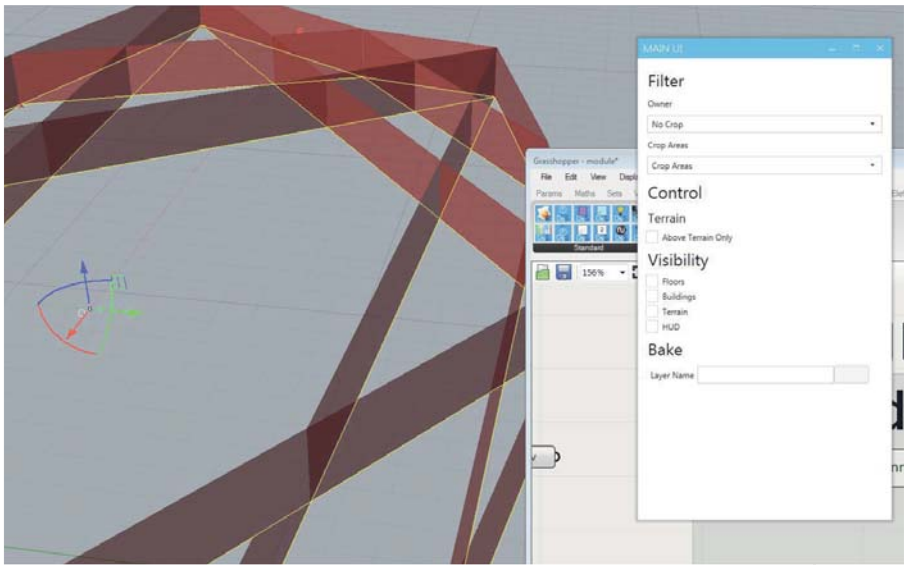
This standard defines the role of the *Developer* of Grasshopper scripts. Designing a script for use within a project infers a set of responsibilities beyond the designed artifact. Because this part of the project involves a competency that is not generally available in the design team, the developer must maintain communication around scope and complexity of the script. File versioning and other quality assurance aspects are also covered here.

Another document is aimed at the *User* of the script. This is written more as an introduction to Grasshopper and a guide to how it is used at White. The user should be able to involve a script in the design process without any prior experience of Grasshopper, or programming in general. Therefore it starts at the very basic level:

> Grasshopper definitions are created by connecting *Components* of different functionality through *Wires*, very much like a switch board. The wires convey data - geometries, numbers, lines of text etc, downstream (to the right) through the network. Data enters each component from the left, is processed, and exits to the right.

This distinction between user and developer is an example of a highly practical management issue that can be layered with a wider reflection on new architectural roles and competencies to describe a difference that design computation does in architectural practice. The notion of a user, leveraging a script within the design process, requires the developer to pay attention to aspects beyond the design process of the resulting artifact. Corresponding to that process, there is a design process of the script itself, where considerations of the user is not geared towards the inhabitant of a building, but instead the architect using the script in a design workflow. Here cognitive aspects such as clear connections between cause and effect in the script, and timeliness and clarity of the feedback it provides, are introduced as professional concerns for architects. While these aspects are intuitively approachable for a designer, there exists a large body of specialist knowledge that could be imported from for instance the fields of

GRASSHOPPER USER
WHITE / GRASSHOPPER / STANDARD



This document will guide you through basic Grasshopper use. It covers how design workflow interfaces are set up at White and how project relations with in-team developers or Dsearch are managed. The intention here is to prepare you as a User of an existing Grasshopper definition.

White has two modes of working with Grasshopper:
- / The User is a member of a design team at White using a definition as part of a design process. This definition can be developed at Dsearch or within the team.
- / The designer of the definition is defined as a Developer. This mode brings a larger responsibility to uphold quality standards for project development at White.

Please make sure that you are reading an up-to-date version of this document by accessing it from White Help.

For an overview of available grasshopper documentation at White, please read Grasshopper Help.

Intended readers

/ Grasshopper Users
/ Grasshopper Developers

Covered aspects

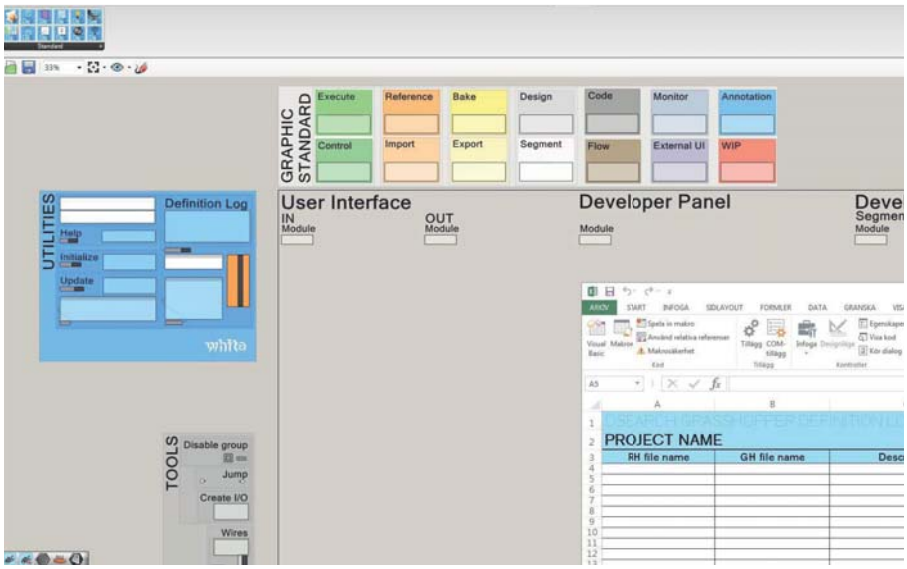
/ Grasshopper workflow
/ Grasshopper Basic Operation
/ User Interface
/ Project workflow

References

Dsearch | White Intranet
White Help | http://whitehelp.white.local/index.php
Grasshopper Setup | GH/Plugin Installation & Setup
Grasshopper Help | Standards, Vocabulary & GH resources

GRASSHOPPER DEVELOPER

WHITE / GRASSHOPPER / STANDARD



This document covers standards and recommendations for working with Grasshopper as a Developer in a project at White. Emphasis is placed on documentation and communication.

White has two modes of working with Grasshopper:

- The User is a member of a design team at White using a definition as part of a design process. This definition can be developed at Dsearch or within the team.
- The designer of the definition is defined as a Developer. This mode brings a larger responsibility to uphold quality standards for project development at White.

Please make sure that you are reading an up-to-date version of this document by accessing it from White Help.

For an overview of available grasshopper documentation at White, please read Grasshopper Help.

Intended readers

- Grasshopper Developers

Covered aspects

- Developer Setup
- Grasshopper Template
- Versioning & Log
- Export & Patch
- Modularisation
- Graphic Standard

References

Dsearch | White Intranet
White Help | http://whitehelp.white.local/index.php t
Grasshopper Setup | GH/Plugin Installation & Setup
Grasshopper Help | Standards, Vocabulary & GH resources
Grasshopper User | GH Basic Operation, UI & Workflow

human-computer interaction and user experience design. Even in the most common case, where the developer of a script is also the user, this discussion brings the notion of workflow as a design task in its own right to the fore.

For a practice-based research and development team, one opportunity to pursue academic research lies in the critical discussion and theorising of material primarily developed to support practice. Design and organisation theorist Donald Schön has developed a theory for how practitioners learn and leverage knowledge. He has authored the terms *reflection-in-action*, and *reflection-on-action* (1983). The Bridge Engagement, the service matrix, and the user/developer distinction, are all examples of issues emerging from practice, and subsequently developed into knowledge materials by Dsearch as a form of reflection-in-action. The next level of reflecting-on-action, for Dsearch is directly based on these strategic and methodological materials. This hybrid perspective has not only yielded academic output in the form of the three papers presented in this thesis; the papers show how research findings in turn have been instrumental to further strategic and methodological development.

Insider Research

The primary objective for Dsearch to engage in projects is to address project-specific issues where design computation workflows are relevant. The secondary objective is to situate knowledge production so that project team, design computation specialists, and White as a whole, can learn from a unique situation. The tertiary objective is the production of generalised knowledge for the wider architecture profession and the academic audience. The role here for a practice-oriented researcher, is not only to observe but to proactively arrange this situation to be fruitful for all parties. This activity is in itself also potentially fruitful for academic pursuit.

Dsearch has access to an insider view of development processes and engagements with new tools and methods, often in the form of personal experience combined with developed knowledge materials such as documentation, generalised methods and organisational strategies. The research strategy for Dsearch has been to build on this unique material and layer it with critical reflection and academic discourse

that is most often unavailable when working under the constraints of practice. In *Paper 2* Dsearch frames that approach as a designerly form of the *insider action research* methodology put forward by organisational research scholars Teresa Brannick and David Coghlan (2007). In defence of researchers being native to the situation under study they argue that an insider position is especially conducive for producing knowledge that is useful within the studied situation itself.

For Dsearch as designers, developers and scholars it is vital to integrate all of these modes of operation in research. This entails an experiential, experimental and iterative attitude towards knowledge production. Being native here, does not only refer to the position of the researcher within an organisation; in order to produce knowledge that is useful for White, it is also vital for the individual researcher to experience, and be experienced in, the architectural practice they are engaging with. Therefore, all members of the Dsearch team are educated as architects and have prior experience as practising architects.

The insider perspective also impacts the point of view a researcher can take. "Traditionally, research has focused on third person: researchers doing research on third persons and writing a report for other third persons." (D. Coghlan and Brannick 2014, p7) This norm is challenged by education and management scholar William R. Torbert. Together with a long list of collaborators he has developed *Action inquiry* as a kind of social science that facilitates timely action. Comprising a matrix of 81 research modalities, action inquiry distinguishes between "first-, second-, and/or third-person research voice, studying first-, second-, or third-person practice, in the past, present, or future, with single-, double-, or triple-loop learning" (Torbert and Taylor 2008, p241). Torbert describes learning as a set of recursive loops. Responding to an event with *single-loop* learning leads you to adjust future actions in similar events. *Double-loop* is your transformation of the strategy that led to the event. *Triple-loop learning* triggers a qualitative change in your attention to the event, and the actions and strategy involved.

Dsearch is arguably active in all cells of this matrix, but the published papers are still authored in a relatively conventional voice. The voice called 'we' in the papers is still the third-person objective 'I', but collated from several researchers. This is an attempt to separate

Dsearch - the researcher (we), from Dsearch - the studied situation (they). The documents that comes closest to a genuine second-person voice are the method sheets described below. Building on feedback from network and project engagements, Dsearch - the method developer (us) - describes methods for White (us). The Bridge Engagement above is written in a 1st-person voice, but during realtime it played out as an inter-subjective dialogue - a 2nd-person practice. Both participating individuals of course draw personal lessons from this experience, but a set of materials can also be developed in the engagement that enables for instance 2nd-person method development. In the case of this thesis, my documentation of the situation allows me to reconstruct a 1st-person account, that in turn leads to a 3rd-person discussion in relation to research methodological literature.

Coghlan & Brannick points out that there are political ramifications of doing research within an organisation - especially your own. The questioning nature of action research can be regarded as subversive: “It examines everything. It stresses listening. It emphasizes questioning. It fosters courage. It incites action. It abets reflection and endorses democratic participation” (D. Coghlan and Brannick 2014, p151) In the case of Dsearch this list of provocations is valid for development as well as research. The introduction of new methodology can be perceived as an inherent criticism of current practice, which for architects can be taken personally. Practice is often intimately bound to personal experience, competence and preference. For development this means that new methodology must either be enforced on current practice through standardisation, or emerge as a new practice with altered capacities. Dsearch is following the latter trajectory. For research, this means that not all questions can be asked. It is near impossible to carry out quantitative experiments in a situation set up in anticipation of the unique. It is also politically undesirable for Dsearch to investigate relations and transactions of power between individuals and organisational entities at White, even if this could have bearing on the dynamics of change surrounding design computation. With a mission-statement to move White in a certain direction, Dsearch is an actor in these political transactions and cannot credibly maintain objectivity. Furthermore, charged with the responsibility to further the interests of White as a commercial practice, Dsearch does simply not have full academic autonomy. Such organisational studies could perhaps be carried out at White, also from an insider

perspective; but such a study would be more credible if carried out by a colleague without a pre-existing agenda. For the purposes of my thesis, my position at White is too connected to Dsearch to make social analysis part of my research objective.

Action Research Cycles

David Coghlan presents the *action research cycle*, a common methodological model for action research. This is a cyclical, four-step process that requires deliberate application of "(a) planning; (b) taking action; (c) evaluating the action (d); leading to further planning, and so on" (D. Coghlan and Brannick 2014, p6). While he acknowledges the importance of a cyclical approach where results can inform further planning, this process is essentially sequential in nature. In my position as an insider within a design-based practice, this sequentiality is challenged by several factors:

- Practice is hectic and messy. Project schedules, even when successfully adhered to, are out of my control as a researcher and developer. This makes it a challenge to align project specific and strategic development, and unfeasible to establish an independent research time-plan.

- Design is not sequential. If architectural practice is messy, the core activity of design is itself inherently non-linear. In the words of design scholar Bryan Lawson: "It is central to modern thinking about design that problems and solutions are seen as emerging together, rather than one following logically upon the other" (2006, p124). Each project starts out in a state where it is more or less unclear what the problem really is, and the process towards a proposed solution does not follow any prescribed step sequence. It is therefore hard to predict the quality, or even quantity of results from a research study, and even harder to schedule when these results can be expected.

- I am a doctoral student. The fact that I am developing new skills along with generating findings, at times makes it unnecessarily limiting to stick to a plan that was developed by a less informed and competent version of me. This fact of course applies to every researcher to a greater or lesser extent.

Coghlan problematises the sequentiality by referring to pioneering action researcher John Heron's distinction between *apollonian* and *dionysian* cultures of inquiry. The Apollonian adhere to "a more rational, linear, systematic, controlling and explicit approach to the

process of cycling between reflection and action" while "Dionysian inquiry takes a more imaginal, expressive, spiralling, diffuse, impromptu and tacit approach to the interplay between making sense and action" (Heron 1996, p50). Heron defines these cultures as complementary and as a source of creative tension for a research project. Framing the concepts as cultures, he implies that the method of inquiry emerges from the content of the studies. Heron encourages the researcher to stay flexible in approach; too apollonian "and the inquiry will lose depth, range and richness"; too dionysian "and the inquiry will lose its focus and cease to be an inquiry" (1996, p50). For Dsearch the apollonian approach is present in the need to follow the business planning cycle at White with yearly revisions to strategic plans and budgets. Here the conditions for the research efforts are fixated. When it comes to actual research and development, a Dionysian flexible and serendipitous approach is taken, grabbing the opportunities to take part in interesting projects and developments when they occur.

My dual position as a practice insider and a doctoral student, could be separated into a core action research project and a thesis action research project. Management scholars Ortrun Zuber-Skerritt and Chad Perry, present this model comprising two interconnected cycles of planning, action and evaluation - with different aims and outcomes (Zuber-Skerritt and Perry 2002). Where the core project is geared towards enacting positive change within the context being researched, the thesis project is concerned with producing relevant new knowledge about this change. Zuber-Skerritt and Perry builds this model from the perspective of research education and proposes that the core project is pursued as a form of field work - possibly revisited, but scheduled as a excursion from the thesis project.

In addition to the challenges to sequentiality in my case, I also see a asymmetric relationship between my core project as employee at Dsearch, and my thesis that needs to be addressed if this model is going to work. My practice takes the form of project engagements and method, organisation, and knowledge development. This presence is integrated into the culture and organisation of White, it cannot be summarised in a report. The core of my action research is in that way simultaneously richer and more intangible than a project with a concise and tangible deliverable. For the thesis, this project of course

must adhere to a standard format. Subject to conventional academic discipline, this level of my research enforces a certain apollonian rigour to my research approach along with a peculiar outsider perspective on my own insider position.

Related to the core/thesis distinction is my intricate relation as an individual to the Dsearch team. Not only am I researching myself and my work in practice; the rest of the team includes fellow researchers with separate research interests and we also conduct academic research together as part of Dsearch operations. As a doctoral student it is vital to maintain autonomy of the thesis project, and at the same time hard and possibly also undesirable to distance myself from Dsearch in my insider position. For the duration of my research education this autonomy has been a matter of aligning the research interests of Dsearch, me and my main collaborator Jonas Runberger. This negotiation has been relaxed but intricate due to the fact that Runberger is the director of Dsearch and in the function of artistic professor of digital design at Chalmers, also my thesis co-supervisor. Our common starting point for alignment is the mission statement for Dsearch to introduce design computation at White. From this follows the need for a deeper understanding of this emerging practice. This has been addressed by focusing on the description of methodological development and organisational issues relating to new professional roles and workflows stemming from the introduction of new methods and technologies.

My main influence as an individual researcher on my core level at Dsearch has been the construction of a materialist ontology to interpret and inform the development activities. I have also managed to align the scope of each paper included here, to form a coherent whole. On my thesis level, Dsearch project engagements, developments, and research forms a new whole to be studied.

Discourse

This thesis includes three papers collaboratively written by Dsearch, in order to articulate and reflect on its own emerging design computation practice. The series of papers starts from a strategic perspective, zooms in on project workflow management, and ends with connecting the very detailed level of tool- and method development

back to strategic considerations around knowing and learning in practice. The materials presented, includes what Dsearch defines as a *strategic framework* of management devices in *Paper 1*, comprising for instance the *design system* - an expanded notion of the architectural model. *Paper 2* further articulates design systems and how their qualities are differentiated by various project contexts. *Paper 3* scrutinises the visual design script and how it relates to the overall design and project development process.

While these papers are authored by members of the core team of Dsearch, they are based on a discourse involving a wider network of colleagues, clients, professional peers and academic partners that has formed around Dsearch, sharing an interest in design computing and its ramifications. The closer network of colleagues is part of a continuous conversation as part of the practice that grounds it. Emerging from practical issues, a community of practice is nurtured within White in order to reflect-in-and-on-action. A crucial component of this discourse is the collective convergence on a vocabulary for this emerging practice. On a practical level at White, this standardisation of communication is instrumental to quality assurance and knowledge management. From an academic perspective, each new term is an opportunity to clarify and systematise experiences from practice. Before introducing each included paper and its context, it will be useful to present this vocabulary in its current state.

In *Paper 3* we state that computation is determined by *code* and define it as an *instructional notation*. Bryan Lawson gives instruction drawings as an example of an instructional notation in architecture. These are “intended as an unambiguous one-way form of communication from designer or design team to constructor or supplier”(2004, p34). We also define the *algorithm* as a *descriptive notation* and a representation of the computation that the code performs. We argue that as scripting designers, we may push our engagement with the algorithmic further away from mathematical precision and comprehensiveness to better address the poorly defined problems we often face.

Code is authored, usually as text in a specific programming language. In visual programming, the author arranges the code as a schema - connecting components with wires, as Dsearch phrase it in their introduction to Grasshopper. This is a kind of *dataflow programming* where the author interacts through the visual interface with a *graph*,

an organisational construct associating specified computational processes with their respective input and output data. This abstraction lies in-between the visual interface and the various layers of code on top of the processor, enabling realtime feedback when scripting. This last feature is vital for designers and other creatives, that rely on fast iteration between exploration and evaluation in their process. Because of this, visual scripting is a very common form of programming in architecture,* but also in music and visual arts.**

The act of programming could also be nuanced in the context of architectural practice. There is a difference to be made between a *script*, written in the context of a specific software environment, and a *program*, designed as an autonomous software entity. A script does not have to include management information, such as how it should interact with the operating system of the computer. Commonly, a script is run in order for its own author to perform one specific task, whereas a program usually is designed to anticipate an arbitrary number of use cases. For the designer, a script is a tool, albeit sometimes specifically created for a unique situation. A program could then be considered an artifact, and as such a design task in its own right. Here, attention lies on how users, not the developer, will interact with the artifact. These modes of programming could be used as directions on a gradient. The design systems described in the papers below, due to their heterogeneous makeup, show up as distributions on this gradient. However, all of the examples could be described as more of a script than a program. A design system could involve an arbitrary number of scripts, some interacting directly with a user, some requiring a developer for safe execution.

There are examples of architectural practices that develop more program-like code. In the Grasshopper community this commonly takes the form of plug-ins. These programs can only be used within the Rhino environment, so they are not fully autonomous, but they still have to be able to handle a full range of use cases. Dsearch pursues a strategy that stays closer to the script end of the gradient. Attention

---

* for instance Grasshopper for Rhino, Generative Components for MicroStation, Dynamo for Revit, Flux IO

** for instance pure Data, Max/MSP, Quartz Composer, vvv

is focused on developing design concepts and bespoke workflows, rather than making the code universally stable. This closer engagement with project dynamics explains the need to expand the notions of model and project into the *design system assemblage* of *Paper 2*. It also feed into the visual script as a design medium. *Paper 3* proposes a visually enriched form of code comment, the *note* that can tie together the algorithmic aspects of the code with project development issues.

Paper1

Harnessing the Informal
Processes around the Computational Design Model

This paper presents a collection of managerial materials formulated by Dsearch. The purpose of these is to minimise the risk involved in introducing design computation in an established commercial architectural practice. These materials are described and theorised using the concept of *boundary objects* originating with sociologist Susan Leigh Star. Such materials "inhabit several intersecting social worlds […] and satisfy the informational requirements of each of them" (1989, p393). The boundary object concept is leveraged to discuss how organisational change and learning can be managed from the bottom-up.

The paper was presented by Jonas Runberger at the 2015 *Design Modelling Symposium* in Copenhagen, a recurrent conference gathering researchers and practitioners from the computational design field with special interests in modelling, fabrication and simulation. Our paper, with its management perspective and theoretical framework, was a bit of an outlier, but the presentation at the symposium gathered interest - predominantly from practitioners in organisations similar to Dsearch. One tangible consequence of the paper was a collaboration between Dsearch and structural engineer Julian Lienhard at HafenCity University of Hamburg. There a group of students applied and developed some of the materials presented in *Paper 1* (Lienhardt and Runberger 2017). This work in turn informed new development at Dsearch presented in *Paper3*.

Jonas Runberger was the main author for this text, describing strategic development work carried out together with me. In the initial

stages I acted as a discussion partner and text editor. I also detailed some of the materials presented. At this stage this was a straight-forward description of how we organised Dsearch in relation to White, and how we designed and deployed materials for this purpose.

My main contribution as co-author occurred in the later stages of the review process when we were asked to clarify the implications of deploying these materials in practice. Here I proposed to revisit Susan Leigh Star's notion of boundary objects (1989) that Runberger had earlier discussed in his PhD thesis (2012). We then casted our material as boundary objects and discussed how this view impacted earlier experience of using them in practice. The paper also employs the concept of *reification* to describe how a computationally augmented design process can solidify into a boundary object and enable cooperation between teams with and without computational expertise. Attention to this kind of object has remained a priority for Dsearch. Reification is a concept strongly coupled with management scholar Etienne Wenger's theory of *communities in practice* which is taken up as a vital component of the R&D model proposed in this thesis (1998).

This introduction of an extradisciplinary theory to analyse the technical and managerial materials we present, has been influential on subsequent discourse, research and development at Dsearch. In *Paper 2* this approach is discussed as a method to achieve reflexivity. Management scholar Mats Alvesson formulates this as the use of "theories which challenges common sense, not only for the direct application but also for encouraging perspective on one's own lived reality and thus facilitating looking upon things in a more all-sided way" (Alvesson 2003, p186).

Paper2

Design System Assemblages - Continuous Curation of Design Computation Knowledge in Architectural Practice

This paper further details one of the boundary objects described in Paper 1 - the *Design System* - and exemplifies with two cases from Dsearch practice. The theoretical frame here is Manuel DeLanda's development of Assemblage Theory that emphasises that any given individual, object or system - assemblage - does not have essential qualities or a permanent boundary. This fits very well with the authors view that the current design computation practice at Dsearch cannot be explained, or successfully managed, within traditional categories such as *projects* or *specialisms*.

I was the main author with strong support and collaboration from Jonas Runberger, who also wrote the SOfFTA case description. I presented the paper at the *Professional Practices in the Built Environment* conference in Reading 2017. The conference was "conceived as an event to bring built environment practitioners together with academics to discuss the development of collaborative research on practice" (Samuel and Hay 2017, p2). We addressed this topic in terms of the designerly version of insider action research approach deliberated above.

The paper is a reworked version of an earlier submission to ACADIA 2016 that was not accepted. That text was based on Dsearch discourse, continuing the ambition to explain design computational development through extradisciplinary theory. For that particular event this proved to be the wrong approach. The reviewers did not see a benefit in establishing new concepts for design computation discourse - wanting deeper detailing of technical innovation. Shifting the focus of the paper towards the implications of design computation for knowledge production in professional practice, marked a corresponding shift for the overall thesis work. For me, this formative rejection, led to an interest in the preconditions for conducting genuinely practice-based and practice-oriented research.

In the ACADIA version of the paper, assemblage theory took centre stage. This was an ambition to contribute with a theoretical framework for an emerging computational design thinking tradition (Terzidis 2006, Coates (2010); Woodbury and others 2010, Menges

and Ahlquist (2011); M. Burry 2011). Lost in the revision was a more in-depth discussion on DeLanda's two analytic parameters of *territorialization* and *coding*. Used to discuss various stages of the development of the *SOfFTA* and *Urban Values* cases, they are defined as follows: "The more homogeneous the internal composition of an assemblage and the better defined its outer boundaries the more territorialized its identity may be said to be" (2011, p187). "Coding then determines the potential for change in territorialization" (Magnusson and Runberger 2017, p196).

Genetically imprinted instincts are coded, whereas a capacity to learn decodes the assemblage - freeing it to shift its level of territorialization in response to external conditions. DeLanda provides the example of object oriented programming languages for computers that decodes the identity of the software so that it can perform situation-dependent operations on input data, depending on the input data itself.

When trying to determine the territorialization of a specific design system assemblage, we often felt that half of the relevant situation was left out of the description. "Applying this theory to the case material gave rise to the insight that the analysis of an object is also always an analysis of its immediate situation" (Magnusson and Runberger 2017, p203). We also realised that the act of setting system boundaries is just as important for a design process, as for an analysis. This is a more apparent condition when it comes to the second order design of design workflows, where the design object is one step removed from any physical artifact, but it applies equally in first order design processes.

Materialist philosopher and originator of the term *object oriented ontology* Levi Bryant addresses this phenomenon in his work on pleating: "Bodies dwell in a field that constitutes the horizon of their existence. There is no body that does not dwell in a field that constitutes both the "wherein" that it dwells in and that is that from which it exists" (Bryant 2016).*

* Cited from Bryant's translation found at https://larvalsubjects.wordpress.com/2016/11/01/for-an-ethics-of-the-fold/ 2017-11-21

Paper3

Morphology & Development - Knowledge management in architectural design computation practice.

In this paper we discuss the design script, referred to as a 2nd order design model in *Paper 1*. We argue that it is clarifying to distinguish between design and development processes on the one hand, and the computational and morphological processes performed by a script on the other. The paper then describes a set of tools and conventions developed by Dsearch in order to makes use of this distinction. Design develops over time, while morphological code executes in realtime according to a predefined logic. To draw an analogy from biology: evolution can be seen as a design process of searching through a solution space for viable options. This exploration and evaluation is distinct from shape generation during foetal development - morphogenesis. The deterministic morphological design graph is thus not to be confused with an open ended design process; the graph is rather subjected to modification through iterations within the design process.

The notion of a design graph as an object to be designed in and of itself, in tandem with the artifact that it generates, is an emerging topic in Dsearch discourse. This paper concludes that the "graph, defining a computational process, is a genuinely new kind of representation for architecture" (Magnusson et al. 2017, p689). To extend that line of thought I would like to argue here that the visual script, for design, is a new kind of medium for the production and reproduction of graphs. Designing a morphological process by scripting its graph, results in an artifact that can be saved, copied, edited, etc and still retain its capacity to process form. This is different from the reproduction of finalised process results. Describing a morphological process by annotating its graph results in a design algorithm, a descriptive layer distinct from the code. This is a kind of design process documentation that is more easily reproduced alongside its morhopological code. Graphs, scripts and algorithms are clear examples of the difference that computation makes in architectural practice.

Based on ongoing discourse and methodological development within Dsearch, this paper was mainly written by me with strong support and collaboration from Jonas Runberger and Malgorzata Zboinska. Vladimir Ondejcik is credited for his contributions to development and discourse. The paper was presented by me at eCAADe - a large

annual conference on education and research in computer aided architectural design. The heading for the 2017 event in Rome was *Sharing of Computable Knowledge*. If the organisers really intended the specificity of the wording to go beyond just a general call for computation, sharing, and knowledge (Fioravanti 2017, p6) - our paper in a sense inverted the sentiment of that theme. The intention behind the Dsearch development, and our reason to reflect over these tools and conventions in academic discourse, is to facilitate for designers to make sense of computation. To challenge the eCAADe theme, tongue-in-cheek: this entails sharing of knowable computing.

KNOWING PRACTICE

Realtime is now, but longer. Now hits you unevenly - it washes over a topography of ridges and runnels. Sensors at strategic positions continuously pick up incoming signals, informing you in different arrivals. Now thus stretches backwards up causal chains, up every information tether anchoring you to the world. Your situation is buffered and synchronised, forming a delayed but coherent realtime experience. This means that realtime lags.

To minimise response time, incoming information is part prefab. You cast anticipation on your sensorium. What your sensors produce is not perception but the discrepancy between anticipation and perception. Surprise demands better speculation to dampen the signal. Realtime is your participation in reality, the surprise-fuelled reciprocity between a body and its situation. Also your own actions are anticipated - mental and physical. Until they are successfully executed, you remain surprised.

Only now is real - past is re-enactment, realtime is fabrication and future is speculation. Your situation also contains the inner territories. Cognition is physical, your experience is carried by matter. Past, realtime and future do exist, but as models. With the experience of your now being fabricated in realtime, rather than witnessed, your models are just as real.

In the situation you find materials - objects or environments providing you with realtime affordances. In this sense, realtime draws on development. Matching the materials capacity to affect with your capacity to be affected, you alter your physical or mental powers. Knowledge materials enter realtime through your knowing; they leave through your making.

Development has always already happened. Past realtime activities have developed knowledge materials accessible for coming realtime use; artifacts and arrangements; media, tools and infrastructures. Users are anticipated, development is aimed at specific practices. Abusers appropriate materials for unanticipated use. Any artifact has an unlimited set of capacities to affect and be affected by the situation - exceeding the intentions of its developer. Materials are used, abused, produced and altered, then aggregated and sedimented. In this sense, development draws on realtime. Development is a preparation for the future, having been done through engagement with relevant parties - a continuous communal arrangement of your situation leading up to now. Development is you, having been co-developed. Now, you know your way around your model, computer, desk, colleagues, workshop, library…

Modes of Knowing

The above vignette is an introduction to *realtime* and *development* as two modes of knowing design computation practice. In choosing the word knowing over knowledge, I imply that I am pursuing a certain attitude towards learning and acting on knowledge. The *Realtime Monologue* above is written in the 2nd-person, present tense voice. It is me - the writer, addressing you - the reader. I believe that this point of view best captures the sensibilities I want to add to the discourse on practice with these modes of knowing.

Donald Schön is interested in the professional competency of architects and describes knowledge as a description of what is essentially a feel for things: "One must use words to describe a kind of knowing, and a change of knowing, which are probably not originally represented in words at all" (Schön 1983, p59). Building on Schön and a pragmatist tradition starting with John Dewey, cognitive scientist Henrik Gedenryd frames the design process as a form of *inquiry*.

> Pragmatism prefers the term "knowing" to knowledge. It is a label not for a thing but a capacity, something that manifests itself in an individual's actions and which is not assumed an existence beyond that: knowing is thereby primarily an activity, and this is reflected in "knowing" being principally a verb; knowing is an entity only in a derived sense, and this is reflected in "knowing" also being a verb used as a noun, and not a noun *per se*. "Knowledge" is a noun, pointing out a thing stored in the mind. (Gedenryd 1998, p78-79)

This distinction between knowing in action, and knowledge as stored knowing, is fundamental to the realtime and development modes of knowing. It highlights the distinction between realtime knowing and developed knowledge material. Gedenryd stores that knowledge as a thing in the mind, but in this thesis part I will argue that an embodied mind, can also make good use of knowledge stored externally - in the development mode of its situation.

Complementing the pragmatic epistemology, this thesis is based on a materialist ontology. This is expressed most explicitly in *Paper2* with its use of Manuel DeLanda's Assemblage Theory (2016). Another philosopher that has shaped my sensibilities for the interrelations between material and immaterial objects is Levi R. Bryant, although he would phrase it 'my *structural openness* towards the *mediation* of *corporeal* and *incorporeal machines*' using the terminology of his

Onto-Cartography (2014). Any comprehensive introduction to his thinking is out of scope here, but what he stresses is the importance of a flat ontology where no entities exists above or beyond the others, controlling or preceding their being in the world. Incorporeal ideas and intentions are as real as concrete, but any cartographer needs to attend to how they are mediated by corporeal machines.

Bryant, along with DeLanda, belongs to a school of thought called new materialism (NM) that has has found it's way into design computation discourse. Architectural theorist Neil Leach frames their common focus on design process and material performance as a Gothic logic, building on French philosopher Gilles Deleuze.

> "From an architectural perspective, it is Deleuze's distinction between the Romanesque and the Gothic spirit that highlights the key difference between Postmodernism and NM. Whereas Postmodernism privileged the Romanesque logic of representation and symbolism, NM focuses on the Gothic logic of process and material performance" (Leach 2016, p345).

Leach argues that NM challenges not only the linguistic turn in twentieth-century philosophy, but also the dialectic materialism of Karl Marx. This is accomplished not by completely overturning the former project, but by situating semiotics within material reality. The realtime and development modes of knowing aligns with NM in its ambition to understand design as a interrelation of ideas and concrete reality.

Realtime

Realtime can generally be defined as the "actual time during which a process or event occurs."* Computer scientists Kang G Shin and Parameswaran Ramanathan writes how the concept is used in their domain (1994). They present three major components that characterise realtime computing: time, reliability, and integration with the environment. "For example, for a drive-by-wire system it is meaningless to consider on-board computers alone without the automobile itself" Shin and Ramanathan (1994), p6]. Because of their use in precarious environments, often ensuring the safety of humans, realtime systems need to reliably respond to events within specified time frames.

Computer games are realtime systems, a classical example being

* https://en.oxforddictionaries.com/definition/real_time

StarCraft.* In that realtime strategy game you are simultaneously expanding your infrastructure, unlocking technological capacities, and fighting a skirmish war with your opponents. All actions have a specific duration and the goal is to strategically outmanoeuvre your opponents while also make fast tactical decisions in the heat of battle. In contrast with the solemn contemplation of Chess where you are in full control over what move to make, and when to finalise that decision, StarCraft happens to you at a rapid pace - decisions must be made quickly, without perfect situational knowledge, and they must be well executed in time and space. Chess is a turn-based game. During your turn, you can take your time to figure out the one - and only one - best possible move considering the state of the game at that turn. There is no penalty for delaying your response. Your opponent can only wait and prepare hypothetical moves in her head. That wait is frustrating enough for the chess community to have come up with the chess clock to limit the amount of time available for the players; but that does not change the fact that Chess is asynchronous - it is not played in realtime.

Writing this thesis I find myself inserting comments to the text. These are notes where my current, rested and focused self, is ordering my later, tired and dense self, to carry out rote tasks - such as correcting citations or updating references to chapter names etc. To minimise distraction, these messages are brief, explained mostly by their context and peculiarly mostly in English. Even if Swedish is my native tongue, I find switching output language for my thoughts cognitively straining. This method, analogous to the GH *devNote* object described in *Paper 3*, is an example of how realtime can be intertwined with development. By temporally outsourcing work, I can focus on the task at hand - no need to perform auxiliary tasks while I remember them. Computation here provides an external cognitive infrastructure. The comment leaves the realtime of my mind and becomes part of the development mode of the situation. My later self can then acknowledge it as a feature of a new realtime situation - for that self, it has already happened.

---

* https://en.wikipedia.org/wiki/StarCraft

Development

The notion of Development, as a mode of knowing, is specified here so as not to be conflated with the general definition of the word. My aim is not alter the concept but enrich its meaning with a new layer of appreciation. The Realtime Monologue above is a description of Development as seen from realtime. This view can be contrasted with the *Developments* section that presents the activities involved in design computation development at White. These are sensibly described from a 3rd-person viewpoint - in a neutral voice used to collect, summarise and discuss found phenomena. In design methodology, this is a common perspective to describe projects and processes. The Bridge Engagement instead uses a 1st-person voice in order to relate a development process as it is experienced. It is not a timeline seen from the side; it is a narrative sequence of scenes that I - the writer - inhabit.

Development has a realtime of its own. The Bridge Engagement shows a first implication of this statement: development work takes place in realtime. The Realtime Monologue then aims to communicate what follows from that implication: a sense of temporal continuity in that you find your last sentence just where you wrote it on the screen, and simultaneously a sense of disruption in that it does not invoke the same feeling as the one that went into writing it. That disruption would unsurprisingly be even stronger if someone else wrote the sentence you are reading. But even in this case there would be some form of continuity between you. Development is the salient parts of history that protrudes into realtime, the infrastructure of the realtime situation. It is not a timeline seen from the side, but from the front view.

In the Bridge Engagement, Tobias hands me a piece of acrylic that he has developed. With the laser-cutter he has engraved his intentions, entangled with other information resulting from a development process. In my hand it becomes a knowledge material, layering the conversation we are having on the design process, augmenting my capacities to understand Tobias. At the same time it has agency of its own; after Tobias has returned to his desk, me and the piece of plastic continue the conversation and comes up with the idea of misaligning the grid.

Tobias was the one to hatch the cad drawing that controlled the etching, but he did not intend anything with the hatch, or even attend to it. Still - the hatch was part of his development that protruded into my realtime.

Conceptual Modelling

In contrast with the insider action approach taken in the *Design Computation Practice* part of the thesis, here I take a step away from practice. I will build a conceptual model for knowing in practice, based on the experiences from my action research and structured by the realtime and development modes of knowing I have defined. As an architect, design computation specialist, designer of design workflows, and method developer - constructing conceptual systems is a professional skill. This kind of modelling is a research activity that allows me to make the most of my particular skill set.

The *Realtime & Development* model will end up a mongrel - aligning, integrating and mobilising theories from several fields of knowledge. Most precariously, it interweaves a model from cognitive science with concepts from the social sciences, the humanities and the making disciplines. The literature is disciplinarily diverse, but the texts share a sensibility in explaining phenomena through process rather than essence - as well as a view of learning as a highly situated and embodied phenomenon.

As an architect I have both formal knowledge and embodied knowing, as for cognitive science, I am an extradisciplinary scholar. Therefore I make no claims to fully comprehend this wide and vibrant field and its many ongoing debates. Here, cognitive scientist Henrik Gedenryd with his PhD thesis on *How designers work* can function as a inter-disciplinary bridge between the two knowledge domains (1998). His theoretical writing about design thinking, resonates strongly with my practical experience of performing it, validating my intuitions. His wider perspective on design as a domain-specific instance of interactive cognition provides the link that allows for making comparisons with the other approaches presented in this part of the thesis.

In the book *Surfing Uncertainty*, philosopher and cognitive scientist Andy Clark challenges the traditional view of perception and action (2015). Instead of regarding cognition as responding to incoming

information, he presents a hypothesis stating that the human understanding of the world is primarily driven by prediction. Clark frames his Predictive Processing (PP) model as a mid-level organisational sketch, supporting a set of tools and concepts without fully engaging in how the principles are implemented in the nervous system. As a kind of conceptual modelling, the R&D model incorporates PP at an even higher level of abstraction and interweaves it with other theories. This weaving is in equal parts a conventionally academic writing exercise, and a visual design process. The process iteratively engages the capacities of both modes of thinking, and is continuously evaluated against my practice-based experience. The objective is to embed this knowledge in a model that can be used as a basis for action in practice.

French sociologist Bruno Latour, a major actor in the network of Actor-network theory (ANT), shares the sensibilities towards situatedness and non-human agency with the theoretical framework of the R&D-model. He also agrees with the ambition to address the metaphysics behind social interaction, but not with the goal to generalise knowledge in order to support action. This is first and foremost a call against the construction of disciplinary terminology.

> ANT prefers to use what could be called an infralanguage, which remains strictly meaningless except for allowing displacement from one frame of reference to the next. In my experience, this is a better way for the vocabulary of the actors to be heard loud and clear - and I am not especially worried if it is the social scientists' jargon that is being downplayed." (Latour 2005, p30)

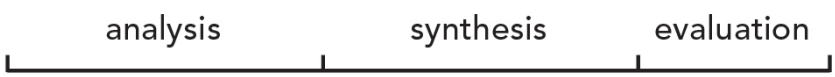
Latour is concerned with giving a correct account of social interaction in all its minute detail, and is worried that knowledge embedded in the studied relationships will be lost in translation. In this conceptual modelling exercise, I aim for a different kind of knowledge. ANT relates a social dynamic in descriptive high fidelity to an academic audience, able to perform the displacement from one studied frame to another. In contrast, the R&D model is intended as a guide for action.

As an architecture scholar I cannot make any claims that the R&D-model correctly maps the physiological processes behind learning, action and decision-making. It should be read as an abstraction. The claim here is rather one of relevance for guiding action;

justified by my insider experience from design computation in architectural practice. By interweaving the theories presented in the sections below, I believe that the R&D-model can draw on a larger body of knowledge and experience than if I had confined it to the design disciplines. By leveraging existing disciplinary theory, the model can potentially facilitate communication within and between communities such as design computation, and architectural academia and practice.

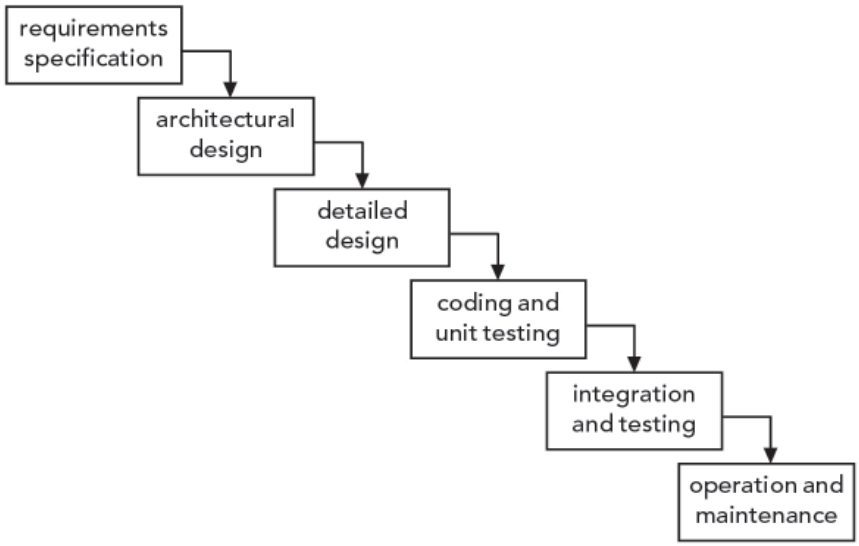
Rationality

Henrik Gedenryd finds the history of leaning towards rationality in design- and planning-theory problematic (1998). He argues that cognition makes greater use of the external world than accounted for in the rational tradition. Support for this claim can for instance be found in the seminal paper *The Extended Mind* where Andy Clark and fellow philosopher of the mind David Chalmers (1998) propose that cognition leans on processes external to brain and body - to the degree that it can be seen as partially spread out into the world.



Typical rational schema

Gedenryd traces a long tradition of rational problem solving: from the Greek mathematician Pappus of Alexandria, via French philosopher René Descartes to the early years of the design methods movement (1998). A typical rational schema would start out with an effort to understand the problem at hand, before solving it through *analysis*, and subsequently executing the solution through *synthesis*. This is then followed by an *evaluation* of the solution in relation to the problem. The larger design process undergoes *separation* into distinct phases and a *logical order* of these activities is explicitly specified beforehand during *planning*. Effort is also spent on maintaining *product-process symmetry* so as to make the structure of the design process reflect the structure of the sub-components of the resulting design product. Gedenryd exemplifies this linear and sequential rationality with the waterfall model within software engineering, which he claims hides all significant cognitive work in the black-box of the analysis phase.



The waterfall model of software engineering

Gedenryd characterises the rationalist tradition as *intra-mental* - a view of cognition as a process contained entirely within the mind. Intra-mental cognition is therefore strictly separated from action and perception, as well as all material, social or cultural aspects of the surrounding world. He posits that "evidence from design is quite conclusive that purely intramental performance is very poor, but the evidence is not restricted to design. On the contrary, this is probably the most well-documented fact in all of cognitive science" (Gedenryd 1998, p203). This is mostly a descriptive problem for design theory, according to Gedenryd - in practice, designers do not approach their work as rational problem solving. Problematisation of linearity is mirrored from also within the design disciplines, for instance by established design scholar Bryan Lawson:

> Design solutions and problems do not map onto each other in predictable or theoretically describable ways. This means that designers cannot really break down problems in the way classical natural science researchers do. Designers have no way of knowing in advance which aspects of the problem can be integrated into which solution ideas. For this reason the designer seems to have a special way of thinking which is integrative. (2004, 52)

Levi Bryant relates the intra-mental view to the *hylomorphic* logic, an account of creation prevalent in philosophical discussions on art and technology throughout history. "Under this model of fabrication, the artisan *first* has a sort of blueprint of what he wants to produce in his mind (the form), and *then* imposes that model on matter giving it form" (Bryant 2014, p18). In hylomorphic creation, the intra-mental idea is the *true* design, and the realised artifact a more or less flawed copy.

Bryant examines Jean Paul Sartre's parable of the introduction of steam engines in industry.* In Bryant's version, development of this new technology, forced factories to grow in size due to the number of maintenance staff required per engine. A bigger engine could power more production units with the same amount of labour, thus moving the economical optimum towards bigger operations. Bryant argues that the hylomorphic view disregards all the unanticipated material agency of a world, and the negotiation with these that an artisan must engage in to realise his intentions. "The machine itself ends up contributing to the design in a way not intended by the designer" (2014, p19).

Design theorist and educator Alain Findeli couples the rationalist tradition with a view on professional practice as the mere application of science. He finds one positivist culprit in the Bauhaus heritage, perceiving design as *applied art* or *applied science* (2001). Findeli would rather speak of *involved*, *situated*, or *embedded* science, and turns to complexity theory to establish a more productive connection between design and science. He proposes a model for how design activity can be described as complex systems interaction:

1. Instead of a problem, we have: state A of a system;
2. Instead of a solution, we have: state B of the system; and
3. The designer and the user are part of the system (stakeholders). (2001, p10)

---

* Sartre, J.P., 2004. *Critique of dialectical reason: Theory of practical ensembles* (Vol. 1). Verso.

Findeli's model situates the designer and the researcher within the inquiry - where subject and object can reciprocally influence each other. "Donald Schön's concept of 'reflection-in-action' thus is transferred from its mainly methodological to the epistemological realm" (2001, p10).

Also William Torbert wants to break knowledge production free from the modern technical-rational tradition. He positions the action inquiry research methodology as a kind of craft, design or art. "…action inquiry does not start from this separation of analysis and action, this separation of mind and body, this linear approach to inquiry. That is not to say that such off-line reflection is not useful, but simply that action inquiry is based in a holistic understanding that all supplies to act and inquire *at the same time*" [torbert2006action, p241]. The sentiments expressed in this section all align with John Heron's notion of a *dionysian* culture of inquiry, discussed in the *Action Research Cycles* section above, where care is taken to understand what knowledge a given situation affords the researcher (1996).

Problem > Analysis > Synthesis > Evaluation > Solution

Above, I initiate the Realtime & Development model with an adaptation of Gedenryd's figure of a typical rational schema for problem-solving. Note that the 3rd-person perspective does not allow for the modes of knowing as I define them. The realtime and development modes of knowing are premised on a situated agent. Replacing the objective, generic, 3rd-person view from nowhere and everywhere, with the subjective, specific and situated first-person - the agent is situated and can engage with the specific situation around it. The rationalist belief that problem-solving can be described as a linear sequence of distinct phases is abandoned; so is the architectural corollary that a design process could be comprehensively prescribed in the form of a flow-chart - leading from problem to solution. Instead, I introduce Findeli's notion of a situated agent, pushing and following a system through state changes. This is a first step away from the notion of this diagram as a timeline. The model instead focuses on the individual *Agent* - situated in *System A* - and its actions to bring about *System B*.

System A > Agent > System B

Embodied cognition situates the mind of our agent within a body within the situation that it is trying to know. The capacities of the body and its affordances within the specific situation determines what the agent can think and know - it is embodied. Bodies must take timely and advantageous action. Here, our model breaks with the tradition of rational problem-solving that presumes cognition to be performed by an objective mind, disengaged from the world. There, cognition is framed as passively receiving external reality, and planning the perfect response in full detail before its subsequent execution. Paradoxically, our plucking of a free-floating rational mind from space and binding it to a body, also sets it free. Embodied cognition can also extend out into the situation and make use of contextual specificity. This shift in perspective begs our model to reflect the more intricate relationship between the embodied agent, and the system it is pushing and following. The *Body* is now diagrammatically intertwined both with *System A* and *System B*.

System A > Body > System B

Design Inquiry

Henrik Gedenryd positions problem-solving and knowledge production in the design process as belonging to a pragmatist tradition of inquiry - drawing on Donald Schön's notion of the *reflective practitioner* (1983). A design methodology based on inquiry revolves around *design actions*, and how that cognitive craft is integral to a situated knowing. While a rational approach separates all phases of problem-solving in strict stepwise sequence and places all actual work in the analysis phase - all aspects of inquiry are inseparable. "They are not even distinct parts but only different points of view that can be taken; potentially even of the same, single action"

(Gedenryd 1998, p96).

Gedenryd argues that much work during a design process is geared towards a better understanding of initially vague and unclear requirements. This is performed by simultaneously challenging the current understanding of the problem, and trying to solve it. Schön proposes three design actions for the deliberate manipulation of a situation: *exploration*, *move-testing* & *hypothesis testing* (Schön 1983). These actions potentially reveal affordances hidden in the situation, inaccessible to intra-mental contemplation. Exploration is to perform design moves without any defined hypothesis of the outcome. The move-testing experiment specifically tests the validity of one hypothesis, whereas hypothesis testing compares several hypotheses against each other. Gedenryd dissolves a latent conflict between production and inquiry by making Schön's hypothesis-testing function implicit. This lets the inquiry move forward as long as the design action does not explicitly fail. "The purpose of action is no longer just to give the right result, especially not right away. Instead action is specified to also serve its inquiring purpose, as doing for the sake of knowing" (1998, p135).

One ability important for inquiry is to intuit familiarity within an unfamiliar situation. Schön calls this *seeing-as*, a knowing-in-action that finds problems and their solutions from a repertoire - enabling in turn a *doing-as*. This form of action does not necessitate conscious articulation of the situation. Skillful inquiry is thus never fully blind; while exploration may lack guidance from any consciously articulated hypothesis, the designer responds to familiarity by intuiting an explorative move. If it is cognitively less demanding to move and see, than to formulate an explicit hypothesis - then exploration is more efficient than experimentation. When testing ideas in the world is costly - measured in time, money, cognitive capacity, etc - it makes sense to consciously formulate one or more hypotheses. Experimentation is then more efficient than exploration.

System A > < Test/Action > < System B

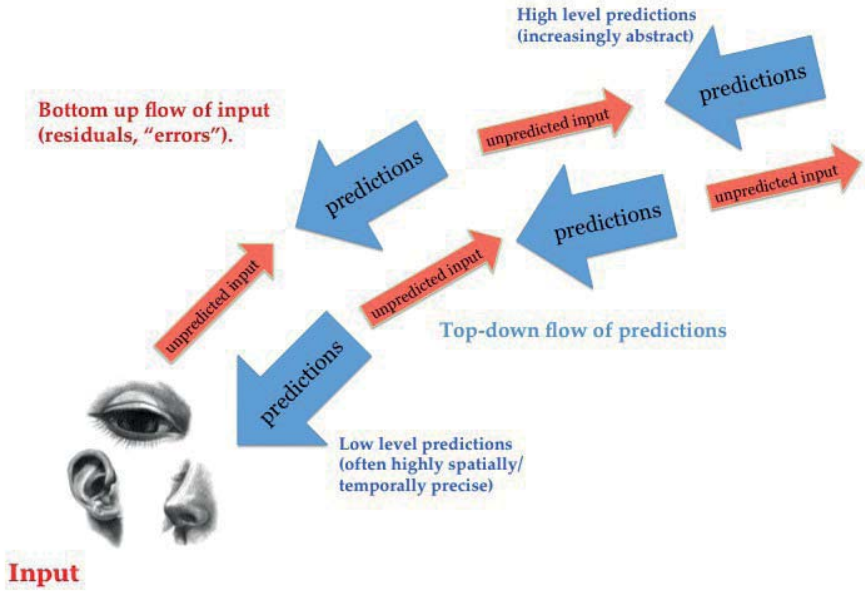
The R&D-model is adjusted to reflect the dual nature of the design action as productive, yet part of an inquiry. Therefore the notion of an agent is articulated into its compound actions. The notion of an

embodied agent is now represented by the dual heading *Action* / *Test*. The Test heading refers to Schön's framework of no-, single- and multi-hypothesis testing. These compound actions can be described using Findeli's terminology: a design action is part analysis of state A of a system, part change towards state B of a system and part hypothesis for how to mobilise that change as an involved actor (2001).

Predictive Processing

Andy Clark articulates the implications of a mind fully extended to its situation, and argues that the mechanism affording an extended mind is prediction. By synthesising a wide body of research within cognitive science and artificial intelligence, into the model of *predictive processing* (PP), he can state that cognition is fundamentally action-oriented and predictive (2013). For Clark, action is - contrary to the rationalist presumption - preceded by prediction, not planning. The main line of reasoning in PP, is that "[p]erceiving, imagining, understanding, and action are now bundled together, emerging as different aspects and manifestations of the same underlying prediction-driven, uncertainty-sensitive machinery (2015, pXIV). Prediction of dangers and windfalls is a vital survival trait for any living being, and a strong reward for energy-expensive cognitive processes.

According to Clark, the incoming signal from the sensorium is not information, but prediction-errors - harvested for the purpose of improving subsequent predictions. This process is tremendously layered in order to translate between low-level and highly specific raw sensory data, and a high-level heavily abstracted conceptual model. On each layer, top-down prediction is contested by bottom up prediction error arising from the layer below. Clark describes this layered modeller as a proactive guide for action and decision-making. While this guide is conditioned by prior knowledge, learning only occurs in hindsight - through the accommodation of prediction errors. The figure below shows how perception in PP is layered to form a gradient of abstraction. This implies that data informs the body on all levels of abstraction - bodily, sub-consciously and consciously. Prediction is not exclusively a conscious, intellectual activity.



The basic predictive processing schema

Internal sensations - proprioception (posture, movement, strength, etc) and interoception (pain, hunger, etc) - are part of the sensorium. In PP, this means that they are also predicted and generating prediction errors. This leads to the peculiar implication that actions are self-fulfilling prophecies. Body movement is not preceded by a pre-planned sequence of motor commands intended to manoeuvre extremities into place. Movement is the minimisation of prediction error, guiding limbs towards where they are predicted to be; it is the prediction of how it would feel if the extremities were already there.

Clark points to another peculiar implication of a model where the content of perception is generated, as opposed to received: perception and imagination are generated, using the same mental architecture. The predictive processing model describes how a machine can take action in realtime and pursue affordances in the world. In connecting imagination strongly with perception, it is related more to an external world than to a mental interior. This is how PP explains dreaming during sleep, but the model is inconclusive when it comes to conscious, deliberate imagining. Clark suspects that imagination "may require the use of self-cueing via language" (Clark 2015, p 94) but leaves this question open for further empirical studies.

I find the PP-model emotionally intriguing. On an intellectual level, it feels profoundly counter-intuitive to suggest that most of what I experience is generated within me. The notion that I don't passively receive reality, but instead actively produce a user-interface representation optimised for agency, runs counter to all my intuitions and schooling about perception and cognition. At the same time I can vividly recall the feeling of being drawn through the air by my badminton racket, a racket/body-machine carried towards impact with the shuttle in a smash. Physically and physiologically, this is obviously not a pertinent account of what really happens during a badminton shot; but this bodily memory of being guided by the anticipation of hitting the shuttle - a preconscious tingling in my wrist - helps me counter that intellectual prejudice.

Situation > < Prediction/Error > < Modeller / Memory

To represent the PP model, the notion of the diagram as a timeline is now completely abandoned. Instead of representing the current situation with System A, the *Situation* now represents a timeless state in-between events. Action is, according to PP the *Prediction* of action, and prediction *Error* the result of a test. What confuses the directions in the diagram is the lost dimension of time. Instead of testing the current system state A, error are reported to the *Memory* so that the *Modeller* can revise the next action based on experience. Instead of casting action towards the future system state B, this model version casts prediction towards the situation.

Action Inquiry

Action inquiry is based on the skillful analysis of subjective first-person experience. This is accomplished through triple-loop meditation-in-action, or:

"…consciously acting in a way that simultaneously inquires into the current awareness-mind-body-situation interaction…" (Torbert and Taylor 2008, p241). First-person inquiry takes place in several different states of mind. In this theory they are regarded both as analytic categories, and as phenomenologically accessible variations of consciousness.

These *territories of experience* can be summarised as (2004; 2008) :

- Outside World - object properties, quantities, events, results.
- Own Behaviour - bodily sensations, feelings, skills, patterns of activity
- Thought - distinctions and interrelations, strategies, reflection.
- Attention towards the integrated experience of all territories.

These territories are always accessible for an attentive mind, but in varying degrees of urgency and clarity. For the skillful inquirer they are all approachable, mutable, compatible and comparable. Torbert exemplifies them in order of human maturation. The outside world is mastered first, by learning how to run and play games. Next, your own performance is scrutinised through playing roles in social games, exploring status and power. The third territory of experience is often developed in higher education, through the training in creative or problem-solving capabilities in some cognitive field. Torbert argues that few of us go on to truly adult learning, seeking to directly engage attention itself, with its capacity for intentional movement among the other three territories of experience (Torbert 2004). These examples are mostly social or intellectual, but throughout his body of work, Torbert also stresses the importance of bodily and sensory feedback in first-person experience. This articulation of first-person experience expands the notion of a realtime situation, in that conscious attention can also be directed inward. The agent itself is of course always within the situation that it tries to understand, and action inquiry shows that also the mind of the embodied agent can be considered part of that situation.

Torbert uses the term *consonance* for the sensation that an experience aligns with your predictions of it; and *dissonance* for feelings of incongruity (Torbert and Taylor 2008). He emphasises that these are durational sensations unfolding in the present, rather than post fact reactions. This definition of dissonance fits well with the feeling I had when holding Tobias acrylic sheet in my hand, just as consonance describes my feeling of working out the misaligned grid as a design concept.

In Predictive processing terminology we can call these feelings variations of prediction error. Dissonance is the more straight-forward translation, whereas consonance could be seen as prediction perfectly aligned with the situation. At face value this must mean that consonance is not something an agent should waste energy on. PP tells us

that only the discrepancy between prediction and perception is fed forward - consonance would generate no prediction error, thus no information. Organisational scholar Chris Argyris relates error correction to learning:

> Learning occurs when we detect and correct error. Error is any mismatch between what we intend an action to produce and what actually happens when we implement that action. It is a mismatch between intentions and results. Learning also occurs when we produce a match between intentions and results for the first time. (Argyris 1993)

I interpret Torbert to posit consonance as a double positive: an actual and desirable experience, rather than merely the absence of surprise. One way to model this is to employ PP recursively: consonance could be seen as a prediction error stemming from the unfulfilled prediction of prediction errors. Simply put: while working on a problem, you are anticipating more issues to resolve but a pleasant surprise arises when you realise you are already unravelling the problem; in an everyday activity you are anticipating more of the everyday, but are pleasantly surprised when an action suddenly illuminates a connection to a previous experience.

Realtime > < Anticipation/Surprise > < Modeller / Memory

In removing the representation of time, this model version now frames the situation of the embodied agent in a way that is fully compatible with my conceptualisation of *Realtime*. The inclusion of a memory and a modeller that can learn, hints at a development mode - but PP does not provide us with a sufficient theory to fully model that. To stress that prediction is primarily not an intellectual exercise, the term *Anticipation* is substituted. Finally the notion of prediction error is replaced by the more phenomenologically rich *Surprise*. These changes in wording are intended to make the model more intuitively useful - also without a pre-understanding of the predictive processing model. This leaves us with a model where realtime action is the anticipation of action, guided by surprise, and consonance is the surprise of unfulfilled anticipation of surprise.

Realtime > < Anticipation/Surprise > < Stance/Organisation

To complete the model we need to further differentiate the modeller/memory part. Current scientific theory on how knowledge actually is managed by the brain is beyond the scope of this thesis; but a simple statement that memories are not immutable atoms of past experience is not controversial. “Rather than memory being an accurate video recording of a moment in your life, it is a fragile brain state from a bygone time that must be resurrected for you to remember” (Eagleman 2015, p22).

The mind needs to perform work - both to accommodate new knowledge, and to recall it again. This is a physical/physiological change in the *Organisation* of the agent. The realtime situation and the organisation co-determines a *Stance* towards the world, directing the agent in how it casts anticipation, and how it frames a situation. What you know about a situation determines what you can learn from it, and how you feel about the situation determines what you know about it. This “means that perception, attention, and embodied action work together to drive the agent in self-fuelling cycles of active perception in which we probe the world according to systemic ‘beliefs’ concerning that which our own actions are about to reveal” (Clark 2015, p70).

Communities of Practice

For Etienne Wenger, learning is to extend the mind into a social arena - it is fundamentally experiential and fundamentally social. Through shared practice, learning takes the form of a *negotiation of meaning* between a community and its members. “Learning in this sense is not a separate activity. It is not something we do when we do nothing else or stop doing when do something else” (Wenger 1998, p8). Wenger proposes the *Community of Practice* (CoP) as an environment conducive to the *acquisition* of knowledge - regulating access to newcomers and allowing for a personal experience of engagement; and to the *creation* of knowledge - forging strong bonds of communal competence and fostering respect for the particularity of member experience.

Wenger points to four deeply connected and mutually defining elements of a community (1998, p5):

- meaning - learning as experience
- practice - learning as doing
- community - learning as belonging
- identity - learning as becoming

For individual learning, Wenger means it is necessary to be able to identify with a community, and be allowed to negotiate new meanings within that social setting. Learning in that sense can be described as a transformation of individual and communal identities. For this to happen, the community must be malleable enough to allow members to engage and influence, yet rigid enough to maintain an identity relatable to all participants. Meaning must be achieved through interaction, and Wenger argues "that participation and reification are dual modes of existence in time, dual modes of remembering and forgetting, and dual sources of continuity and discontinuity" (1998, p91).

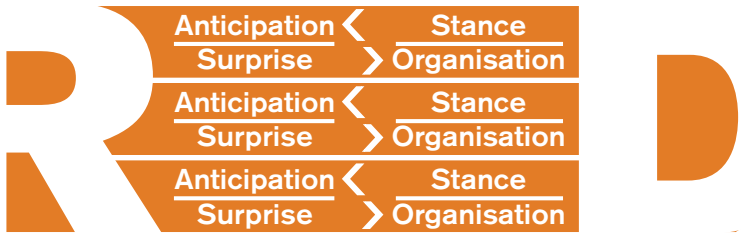
Member *participation* shapes individual and shared experience. This concept characterises individual learning as a negotiation of identity - a concurrent anticipation and mutual framing of a realtime situation. In order to share experience, it is vital not only to participate but to recognise yourself in the other participants. Wenger addresses the creation and agency of artifacts through the concept of *reification* - the production of materials that give form to individual experience. This activity produces anchor points around which negotiation of meaning within a CoP can be organised.

A CoP learns by mediating between the individual and shared experience of its members and the arrangement of materials and individuals that constitutes it; a mediation between its situation and its organisation. Participation and reification are formulated as a complementary duality, not as a dichotomy. Both of these activities must happen and unavoidably happens through the practice that maintains a CoP.

> "Whereas in participation we recognize ourselves in each other, in reification we project ourselves onto the world, and not having to recognize ourselves in those projections, we attribute to our meanings an independent existence. This contrast between mutuality and projection is an important difference between participation and reification"
>
> (Wenger 1998, p58).

Realtime > < Anticipation/Surprise > < Stance/Organisation > <Development

We can use the concept of reification to complete the R&D-model. Reification is the externalisation of individual experience into a communal domain - extending bodily organisation into the development situation. Wenger describes this as a projection, mirroring internal organisation to affect a change of the realtime situation. Adding the notion of an embodied and extended mind, also allows us to posit that both memory storage and cognitive capacity can be outsourced to the development mode. In both cases this material translates individual embodied knowing into communally accessible knowledge.



Realtime > < CoP > <Development

Wenger defines participation as a collective mode of negotiating meaning. In the R&D-model this could be reframed as a collective realtime casting of anticipation, and drawing of surprise. The knowledge materials made accessible by individual reification then structure the practice of a community. They are used in this realtime casting of anticipation, and with time incorporated as part of the organisation - internal or external to the communal body.

the R&D-model

In this section I will present the Realtime & Development model as text and diagram - and describe each part and their interrelations. The objective for the R&D-model is that it will be a relevant and actionable conceptual tool for knowledge production in practice - particularly within architectural design computation. I have therefore formulated the model terminology to be comprehensible on several levels,

with or without awareness of the supporting literature, and with or without the additional background of the thesis. This way, the model can start to make practical sense just through the diagram.

Realtime > < Anticipation/Surprise > < Stance/Organisation > <Development

The model diagram comprises a *Body* (orange) entangled with the *Realtime* mode (R), and the *Development* mode (D) of a *Situation* (white). *Anticipation* is placed with an arrow pointing towards realtime. From realtime, another arrow points at *Surprise*. In the direction of that arrow is *Organisation* - in turn pointing towards development. From development an arrow points back towards *Stance* that in turn points towards anticipation.

An embodied agent continuously frames a situation for itself. It cannot sense all available environmental information, and it would only be cognitively over-burdened from being informed by all of reality. The situation is then sculpted as a terrain of affordances by casting *anticipation* towards it. Anticipation is cast from a stance that the agent takes in response to this sculpted situation. The stance is drawn from the organisation of the agent and constantly modulated in order to minimise surprise - the part of perception that was not anticipated. The stance can also leverage non-human capacities by making use of materials accessible within the development mode of the situation. This is how development informs realtime.

Enduring surprise triggers reorganisations in cycles further and further away from realtime. A learning agent adapts to better anticipate situations. The learning cycle the farthest from real-time is a reorganisation of the development mode of the situation. The agent externalise parts of its organisation as knowledge materials. This is how realtime informs development.

DISCUSSION

To conclude this thesis I will discuss the findings from my experience in practice, and the Realtime & Development model. This will be formatted by the underlying research question *what difference does computation make in architectural practice?* and the series of sharper questions set in the context of design computation in architectural practice.

What new architectural tasks can we identify?

The task of designing design workflows is, in a general sense, not new. It is what every project principal does: establish a team with complementing roles and responsibilities to make sure the design process proceeds. The difference design computation makes is that it requires these efforts to be made explicit; novel infrastructures, tools, competencies and cultures not only bring a need for organisation of the new - they force an articulation of the old. Computation requires new techniques on every level of design activity, to avoid pushing the architectural mindset towards quantification. Accommodating this change without loss of disciplinary competency and integrity, requires developing a new disciplinary sensibility for managing qualities in quantitative systems. To achieve this, the design of design workflows must build on a kind of systems thinking where all parts - human and non-human - inform the whole. A counterpart to systems creation is analysis. The critical delimitation of where the designed artifact ends and the designing system begins is a new task for architects engaging in design computation.

What new architectural knowledge can we identify?

An epistemological finding is that architectural practice engages in a certain *kind* of knowledge. Pragmatist knowing in action is combined with the ability to embed knowledge in materials. Architectural competency also includes a capacity to iteratively use these materials for knowing in action. Design computation sharpens the contrast of this duality in that it brings a new range of powerful materials to architectural knowing in action, as well as the need to develop those materials.

One example of a new material is the visual design script, where knowledge about the design artifact is encoded alongside knowledge of the project conditions. This juxtaposition enables the designer to bring in relevant aspects at will into the design process. Conversely it allows for capture of thought, triggered by the design activity, without disruption of the process. This can come in the form of critical reflection and creative ideas. The visual script can function as an infrastructure that captures Design computation knowledge material has a greater potential for reuse than traditional architectural media. It is easier to extract, copy, store, share, search, modify and comment code than paper. Knowledge material is therefore abundant within the design computation community. In repositories, projects are scrapped for useful parts, that are made available outside of the original project context.

What new organisations of architectural practice can we identify?

The result of applying systems thinking when planning and executing project engagements and developments at White, is a heterogeneous assemblage of people, technology, methodology, knowledge and design project. Design computation allows the assemblage a life after the project. Since the morphological process is coded into a script, this design logic can easily be reused, with or without adjustments, in a new project. As an inverse of the repository, the assemblage can move between project engagements, and maintain identity while continuously replacing its component people, code, design concepts, etc.

There are also new professional roles within the assemblage. Architects can develop design workflows for other architects. This roles entails new capabilities and responsibilities - towards team members, colleagues, and clients. Also the user of the workflow experiences the difference computation makes. New skills and attitudes are demanded, new capacities for knowing architectural practice are won. Computation introduces organisational structures focusing on development and design computation as a specialism. This forges new relationships between project architect and specialist, and more precariously - between project architect and specialist architect. This new environment needs strategic facilitation - establishing communication and expectations around responsibility and scope, for each project engagement.

Further R&D

One of my aims for this licentiate thesis is that it can function as a platform for further research. In the context of Dsearch and White, this will happen naturally for me individually and as a member of an established network. I also intend to use its practice-based approach as one of the drivers in Dsearch's continued presence in the exchange between practice and academia. Involving theory from cognitive science, as well as from AI and software development, has proved to be highly illuminating for traditionally very disciplinary issues - such as architectural representation and design methodology. Describing myself as extra-disciplinary in relation to cognitive science, I will pursue the opportunity to revert back to an intra-disciplinary designer - in transdisciplinary collaborative research with other specialists.

Answering the question *what difference does computation make in architectural practice?* is probably a never ending quest that is futile to shoulder alone. For this purpose, I see a direction for further practice-based research on this topic in current practice at Dsearch. William R. Torbert would today describe Dsearch as a social network* in the sense that it is "motivated by a sense of mission and values that transcend financial value" (Torbert 2004, ch10). The next step of organisational development Torbert calls *collaborative inquiry*. This entails a shared criticality towards the methods and strategies employed by the network and a stronger collaborative drive to pursue the mission. Dsearch are taking such steps, strengthening the network as a community of practice. Etienne Wenger argues that developing "a practice require the formation of a community whose members can engage with one another and thus acknowledge each other as participants" (Wenger 1998, p149). He states that learning, for communities, is an issue of refining their practice and ensuring new generations of members and cautions too strict control:

> Communities of practice are about content – about learning as a living experience of negotiating meaning – not about form. In this sense, they cannot be legislated into existence or defined by decree. They can be recognized, supported, encouraged, and nurtured, but they are not reified, designable units. Practice itself is not amenable to design. (Wenger 1998, p229)

Dsearch are moving towards a practice-based collective inquiry through a gradual and prototype-driven development of an infrastructure for realtime critical reflection. Technical, cultural and social aspects of how a community of practice can know design computing is addressed through the activities described in this thesis. Leveraging this knowledge in further development, will empower its members without clashing against the constraints of practice. This is vital, considering my background concerns regarding quantification and automation.

As for the research question *how do I find out what difference computation makes in architectural practice?*, my intention is to trial the Realtime & Development model in the field. I plan to test its validity in 1st-, 2nd-, and 3rd-person practice at White, in anticipation of consonance or dissonance. The model would also benefit from further theoretical articulation. Here, each mode of the model can serve as a research theme. To provide one example: the *Situation* mode could be expanded by integrating the notion of professional vision (Goodwin 1994) that education researcher Gustav Lymer has brought to bear on architecture. Lymer frames professional vision as a kind of perception that can be trained and encultured - "seeing as social practice" (Lymer 2009, p145). Architectural professional vision allows the perceiver to *see through* architectural representations and mentally inhabit the virtual spaces behind. Lymer has studied this as a collective practice at critical seminars in architectural education. This theory could potentially articulate the R&D-model in terms of how the enculturation into a discipline such as architecture impacts how an agent can expand the frame of its situation by attending to culturally specific territories of experience.

Closing Thoughts

In closing, it is time to address the main title of this thesis. It is equal parts pun and statement. I believe that there exists a potential within architectural practice-based development to perform academically valid research. Conditions for this must be in place: research competency and resources, as well as an understanding of what kind of knowledge is accessible from a designerly insider perspective. It is also a statement directed at academia. Substituting realtime for research in the *Realtime & Development* title is a critical reminder that the realtime mode of knowing is often missing from academic research.

I set this thesis against a bleak background of automation. This phenomenon is hard to critically assess. On the one hand it instinctively raises a will in me to defend professions and their knowledge. On the other hand automation can potentially free us from the cognitive equivalent of back-breaking labour that no one will miss. What I do know is that as a researcher and developer of design computation, I am an active part in this development. My worry can be summed up in the trope that if a machine can do it, it is trivial - and as soon as a machine can do it, it becomes trivial. The rise of the robots is not really about terminators, but about capital. Automation does not inherently increase the risk of war between species, the lurking danger that it does present is the devaluation of human labour. Political, societal and economic solutions are beyond me, but I hope to have shown a path forward for an intellectual engagement with computation that can empower professional architects and augment their design capacities in realtime and development. I am in no way capable to conduct a stringent academic discourse on the relative strengths of labour and capital, but as a professional - I am worried.

REFERENCES

Figures

Design Computation Practice
All figures in this part: © White arkitekter AB unless other sources cited.

Frit Pattern
© Bert Leandersson

Bridge Facade
© Bert Leandersson

Typical rational schema
(Gedenryd 1998, p25)

The waterfall model of software engineering
(Gedenryd 1998, p24)

The basic predictive processing schema
As found in (Clark 2015, p 30), adapted from Lupyan and Clark (2014). Downloaded from (2017-11-22):

http://philosophyofbrains.com/2015/12/14/surfing-uncertainty-prediction-action-and-the-embodied-mind.aspx

Realtime & Development model
All figures by author.

Bibliography

Alexander, Christopher. 1964. *Notes on the Synthesis of Form*. Vol. 5. Harvard University Press.

Alvesson, Mats. 2003. "Methodology for Close up Studies – Struggling with Closeness and Closure." *Higher Education* 46 (2): 167–93. doi:10.1023/A:1024716513774.

Argyris, Chris. 1993. *Knowledge for Action: A Guide to Overcoming Barriers to Organizational Change*. Jossey-Bass Management Series. Jossey-Bass.

Bradbury-Huang, Hilary. 2010. "What Is Good Action Research? Why the Resurgent Interest?" *Action Research* 8 (1). Sage Publications Sage UK: London, England: 93–109.

Brannick, Teresa, and David Coghlan. 2007. “In Defense of Being ‘Native’: The Case for Insider Academic Research.” *Organizational Research Methods* 10 (1): 59–74. doi:10.1177/1094428106289253.

Bryant, Levi R. 2014. *Onto-Cartography: An Ontology of Machines and Media*. Edinburgh University Press Series. Edinburgh University Press.

———. 2016. “For an Ethics of the Fold.” *Multitudes* 65 (4). Assoc. Multitudes: 90–96.

Brynjolfsson, Erik, and Andrew McAfee. 2014. *The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies*. W. W. Norton.

Burry, Mark. 2011. *Scripting Cultures: Architectural Design and Programming*. John Wiley & Sons.

Clark, Andy. 2013. “Whatever Next? Predictive Brains, Situated Agents, and the Future of Cognitive Science.” *Behavioral and Brain Sciences* 36 (3). Cambridge University Press: 181–204.

———. 2015. *Surfing Uncertainty: Prediction, Action, and the Embodied Mind*. Oxford University Press.

Clark, Andy, and David Chalmers. 1998. “The Extended Mind.” *Analysis* 58 (1). JSTOR: 7–19.

Coates, Paul. 2010. *Programming. Architecture*. Routledge.

Coghlan, David., and Teresa. Brannick. 2014. *Doing Action Research in Your Own Organization*. SAGE Publications.

Davis, Daniel. 2013. “Modelled on Software Engineering: Flexible Parametric Models in the Practice of Architecture.” PhD thesis, RMIT University.

———. 2016. “Evaluating Buildings with Computation and Machine Learning.” In *Proceedings of the 36th Annual Conference of the Association for Computer Aided Design in Architecture, Ann Arbor*.

De Kestelier, Xavier. 2013. “Recent Developments at Foster + Partners’ Specialist Modelling Group.” *Architectural Design* 83 (2). Wiley Online Library: 22–27.

De Rycke, Klaas, Christoph Gengnagel, Olivier Baverel, Jane Burry, Caitlin Mueller, Minh Man Nguyen, Philippe Rahm, and Mette

Ramsgaard Thomsen. 2017. *Humanizing Digital Reality: Design Modelling Symposium Paris 2017*. Springer.

DeLanda, Manuel. 2011. *Philosophy and Simulation: The Emergence of Synthetic Reason*. Bloomsbury Academic.

———. 2016. *Assemblage Theory*. Edinburgh University Press Edinburgh.

Dennett, Daniel C. 2013. *Intuition Pumps and Other Tools for Thinking*. W. W. Norton.

Derix, Christian. 2009. "In-Between Architecture Computation." *International Journal of Architectural Computing* 7 (4). Multi Science Publishing: 565–86.

Eagleman, D. 2015. *The Brain: The Story of You*. Canongate Books.

Findeli, Alain. 2001. "Rethinking Design Education for the 21st Century: Theoretical, Methodological, and Ethical Discussion." *Design Issues* 17 (1). MIT Press: 5–17.

Fioravanti, Antonio. 2017. "ShoCK! – Sharing of Computable Knowledge!" In *Proceedings of the 35th ECAADe Conference - Volume 1, Sapienza University of Rome, Rome*.

Ford, Martin. 2015. *Rise of the Robots: Technology and the Threat of a Jobless Future*. Basic Books.

Gedenryd, Henrik. 1998. "How Designers Work - Making Sense of Authentic Cognitive Activities." Lund University Cognitive Studies. PhD thesis, Lund University; Cognitive Science.

Goodwin, Charles. 1994. "Professional Vision." *American Anthropologist* 96 (3). Wiley Online Library: 606–33.

Hensel, Michael U. 2013. *Design Innovation for the Built Environment: Research by Design and the Renovation of Practice*. Routledge.

Heron, J. 1996. *Co-Operative Inquiry: Research into the Human Condition*. SAGE Publications.

Latour, Bruno. 2005. *Reassembling the Social: An Introduction to Actor-Network-Theory*. Clarendon Lectures in Management Studies. OUP Oxford.

Lawson, Bryan. 2004. *What Designers Know*. Elsevier/Architectural Press.

———. 2006. *How Designers Think*. Taylor & Francis.

Leach, Neil. 2016. "Digital Tool Thinking: Object-Oriented Ontology Versus New Materialism." In *Proceedings of the 36th Annual Conference of the Association for Computer Aided Design in Architecture, Ann Arbor*.

Lienhardt, Julian, and Jonas Runberger. 2017. "Collaborative Models for Design Computation and Form Finding – New Workflows in Versioning Design Processes." In *Humanizing Digital Reality: Design Modelling Symposium Paris 2017*, edited by Klaas De Rycke, Christoph Gengnagel, Olivier Baverel, Jane Burry, Caitlin Mueller, Minh Man Nguyen, Philippe Rahm, and Mette Ramsgaard Thomsen, 463–78. Springer.

Lymer, Gustav. 2009. "Demonstrating Professional Vision: The Work of Critique in Architectural Education." *Mind, Culture, and Activity* 16 (2). Taylor & Francis: 145–71.

Magnusson, Frans, and Jonas Runberger. 2017. "Design System Assemblages: The Continuous Curation of Design Computation Processes in Architectural Practice." In *Professional Practices in the Built Environment*, edited by Rowena Hay and Flora Samuel, 194–204. University of Reading, UK.

Magnusson, Frans, Jonas Runberger, Malgorzata A. Zboinska, and Vladimir Ondejcik. 2017. "Morphology & Development - Knowledge Management in Architectural Design Computation Practice." In *Proceedings of the 35th ECAADe Conference - Volume 2, Sapienza University of Rome, Rome*.

Menges, Achim, and Sean Ahlquist. 2011. *Computational Design Thinking: Computation Design Thinking*. John Wiley & Sons.

Negroponte, Nicholas. 1970. *The Architecture Machine*. MIT press.

Nilsson, F, and Marja Lundgren. 2016. "The Changing Shape of Practice: Integrating Research and Design in Architecture." In, edited by Michael U Hensel and Fredrik Nilsson. Routledge.

Runberger, Jonas. 2012. “Architectural Prototypes Ii: Reformations, Speculations and Strategies in the Digital Design Field.” PhD thesis, KTH Royal Institute of Technology.

Samuel, Flora, and Rowena Hay. 2017. “Professional Practices in the Built Environment: Supporting the Development of Collaborative Research.” In *Professional Practices in the Built Environment*, edited by Rowena Hay and Flora Samuel, 1–5. University of Reading, UK.

Schön, Donald A. 1983. *The Reflective Practitioner: How Professionals Think in Action*. Basic Books.

Schumacher, Patrik. 2011. *The Autopoiesis of Architecture: A New Framework for Architecture*. Vol. 1. John Wiley & Sons.

Shin, Kang G, and Parameswaran Ramanathan. 1994. “Real-Time Computing: A New Discipline of Computer Science and Engineering.” *Proceedings of the IEEE* 82 (1). IEEE: 6–24.

Star, Susan Leigh, and James R Griesemer. 1989. “Institutional Ecology,translations’ and Boundary Objects: Amateurs and Professionals in Berkeley’s Museum of Vertebrate Zoology, 1907-39.” *Social Studies of Science* 19 (3). Sage Publications: 387–420.

Terzidis, Kostas. 2006. *Algorithmic Architecture*. Routledge.

Torbert, William R. 2004. *Action Inquiry: The Secret of Timely and Transforming Leadership*. Berrett-Koehler Publishers.

Torbert, William R., and Steven S Taylor. 2008. “Action Inquiry: Interweaving Multiple Qualities of Attention for Timely Action.” *The SAGE Handbook of Action Research: Participative Inquiry and Practice* 2.

Wenger, Etienne. 1998. *Communities of Practice: Learning, Meaning, and Identity*. Cambridge university press.

Woodbury, Robert.F., and others. 2010. *Elements of Parametric Design*. ROUTLEDGE CHAPMAN & HALL.

Zuber-Skerritt, Ortrun, and Chad Perry. 2002. “Action Research Within Organisations and University Thesis Writing.” *The Learning Organization* 9 (4). MCB UP Ltd: 171–79.

Paper1

Harnessing the Informal Processes Around the Computational Design Model

Runberger, Jonas, and Frans Magnusson. 2015. “Harnessing the Informal Processes Around the Computational Design Model.” In *Modelling Behaviour: Design Modelling Symposium 2015*, edited by Mette Ramsgard Thomsen, Martin Tamke, Christoph Gengnagel, Billie Faircloth, and Fabian Scheurer, 329–39. Springer International Publishing Switzerland.

Harnessing the Informal Processes Around the Computational Design Model

Jonas Runberger and Frans Magnusson

Abstract

This paper presents a strategic framework that facilitates the introduction of computational design techniques into architectural practice. The presented architectural design case, and the strategic framework itself, were developed within the Dsearch, a computational development team part of the R&D at White arkitekter AB. An important aspect of the work within the team is to support the integration of computational processes new to the practice, and promote organisational learning that enables a continuous development. The strategic framework therefore is related to certain concepts within the fields of Sociology and Knowledge Management, such as the notion of boundary objects as first defined by Susan Leigh Star and James R. Griesemer, then later developed by Etienne Wenger as an important factor for collaboration within communities of practice. The strategic framework—an assembly of a number of boundary objects, helps elevate the design model to a design system—a project specific set-up that facilitates design versioning, quality control of processes, and organisational learning. Examples are provided through a case project—the development of a 60 m public bench for Forumtorget (Uppsala, Sweden).

Introduction

This paper is written from the vantage-point of a team of computational designers, Dsearch, founded with the explicit purpose of introducing digital and computational strategies and techniques to White arkitekter AB, a large Scandinavian architectural firm. The chosen approach has been to overlay existing projects with computational development, rather than forming a strictly back-office tooling department. The shift from experimental computational practice, to computational design implemented in conventional practice, has highlighted a need for an office infrastructure, mediating between new and old methods and models. The paper details a strategic

J. Runberger (✉)
White arkitekter AB/KTH, Stockholm, Sweden
e-mail: jonas.runberger@white.se

F. Magnusson
White arkitekter AB/Chalmers, Gothenburg, Sweden

M.R. Thomsen et al. (eds.), *Modelling Behaviour*, DOI 10.1007/978-3-319-24208-8_28

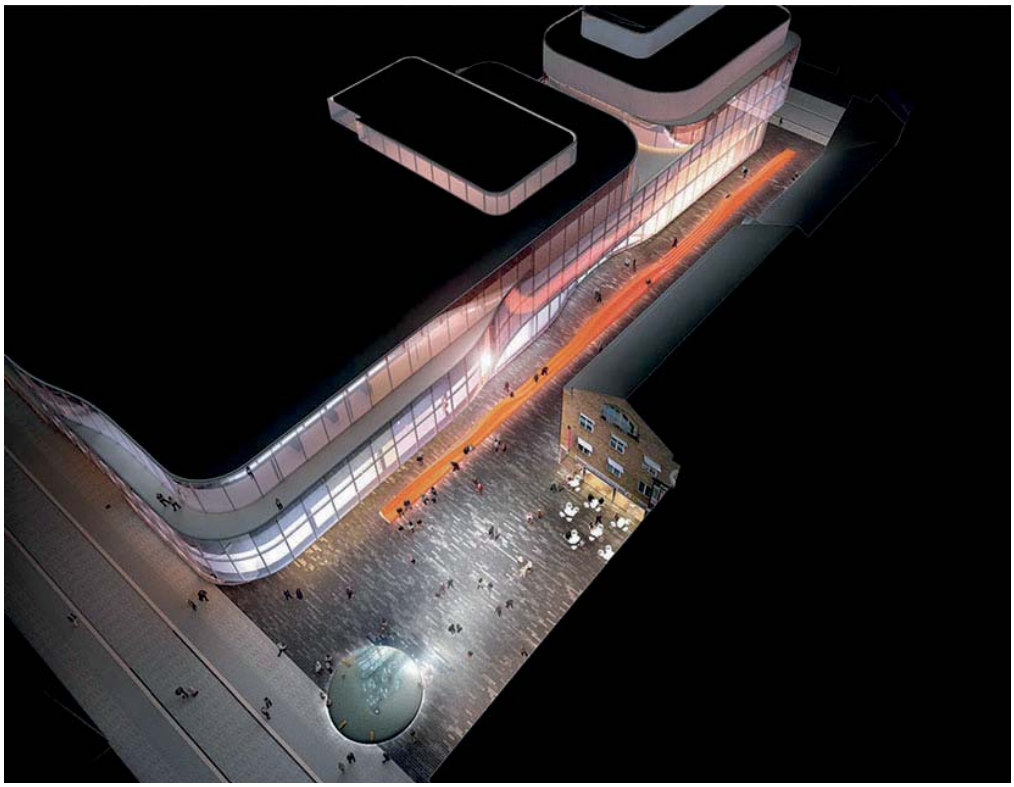


Fig. 1 Representation from the competition entry of the Forumtorget case project—the square and urban furniture

framework, informed by current computational design thinking with a theoretical underpinning from the fields of Sociology and Knowledge Management, and presents how it has been deployed by Dsearch to establish this infrastructure. The framework comprises a collection of so-called boundary objects, where the Design System, expanding the scope of a Design Model, plays a vital role (Fig. 1).

Context

The introduction of computational design into general architectural practice calls for an explicit attention to process and methodology, which in itself can be a provocation to an implicit consensus on methodology—Dsearch has not been preceded by a corresponding "analogue methods team". Together with the approach of active project participation by Dsearch in both design and methodological development, within a context having little previous experience of computational design, this may induce an anxious form of organisational learning—one where the stability of architectural practice is shaken.

The most transformative aspect of this change, is the notion of second order development—to borrow the term introduced within cybernetics, i.e. the process of developing a system that in turn generates, regulates and/or evaluates design (Heylighen and Joslyn 2001). Second order development creates models that are associated with first order design processes and procedures—generated automatically or through the interactions with a designer. In a very literal sense design processes are codified into the model alongside actual proposals. The models

produced are at once more powerful, demanding and fragile than traditional design media; they are more open to the receiving context and have embedded agency as long as they are housed within a specific technical and managerial infrastructure.

Second order development also introduces a new specialized culture within architectural practice. Where different participants before could share a drawing, a physical model etc., the design models today also consists of code that requires a new kind of knowledge and technical infrastructure to create, read and use. While graphical programming environments have to some extents lowered the learning curve, there are still distinct boundaries between architects who understand computational design, and those who don't. This is also true for the relation to recently established workflows where BIM software introduces a configuration mode of design. The change in agency of the model media introduced through computational design, together with the cultural and technical implications, often has to be reflected in the mode of operations of second order computational design.

A Strategic Framework of Boundary Objects

Organizational learning—how teams and overall organizations can learn and adapt beyond individual skills and talents, is of great importance to the context of this paper. Most management routines target efficiency and quality in regards to time and delivery, and can thus be related to the idea of single-loop learning—where goals, values and strategies are taken for granted, as opposed to double-loop learning that requires reflection and innovation (Argyris 1999, pp. 68–75). In single-loop learning, each problem is addressed within an existing framework of solutions, with smaller deviations or corrections of errors. Personal experience or already set procedures endure, and a defensive position is established against the unfamiliar. Double-loop learning entails looking at the task at hand with an open mind, and making theories explicit and clear in order to find new solutions. It is a dialogical, or associative approach—employing new modes of operation to achieve new goals. With this in mind, situating experimental computational design processes in conventional practice requires a strategic framework that allows for double-loop as well as single-loop learning.

Susan Leigh Star and James R. Griesemer's work to understand the communication and cooperation in heterogeneous groups of actors has good bearing on the task set for Dsearch. They developed the concept of *Boundary Objects* to explain the successful cooperation of these actors without explicit consensus regarding the aim of their activities. The basis was the use of common "objects which both inhabit several intersecting social worlds… …and satisfy the informational requirements of each of them" (Star and Griesemer 1989, p. 393). Flexibility of interpretation combined with their identity and capacity to structure work and flow of information throughout various social groups are key to boundary objects. Star and Griesemer also identify several different types or categories of boundary objects, such as repositories, ideal types (maps or diagrams) or standardized forms.

Etienne Wenger adds another important aspect to this concept; reification—to make the abstract concrete and legible, a process that could be embodied also in artefacts (Wenger 1998, p. 58). Similar to the way that architectural representations act as boundary objects that reify abstract ideas into important steps in a design process (such as sketching and model-making), the reification of boundary objects in a computational design environment may help both internal development and communication to general design teams. This may be even truer for second order design teams with no formal training, i.e. architects shifting from design to computational design.

The way design steps are formalized and made explicit within second order design follows in itself a process of reification, and in this way the computational design model becomes a boundary object that can support collaboration between several specialists. This notion of process as boundary object is a distinct difference to

the conventional practice of end-results (proposals, deliveries) as the boundary objects and clarifies the relation between second and first order of design. In order to facilitate collaboration between first and second order design teams, auxiliary boundary objects such as standards, logs or repository content, are necessary. In this way, the design model can be expanded to a design system—a project specific assembly of boundary objects and actors that supports communication and workflow.

As part of general process development the full set of boundary objects are continuously refined constituents of the strategic framework, in essence a boundary infrastructure (Star 2010, p. 602) that employs specific constellations of boundary objects to establish flows of information and structure work across various environments within White arkitekter, including the internal environment of Dsearch (Fig. 2).

Deployed Boundary Objects

The *Service Matrix* is a two-dimensional diagram targeting a need to specify and communicate the conditions for and expected benefits of [group] participation in a project to potential collaborators—internal and external to White arkitekter (Fig. 3). Specification is carried out as a continuous dialogue with the project principal to establish an awareness of mutual expectations, as well as a reduction of uncertainty in terms of development time and deliveries. In this sense the matrix fits the description of a boundary object of the Ideal type:

> It is abstracted from all domains, and may be fairly vague. However, it is adaptable to a local site precisely because it is fairly vague; it serves as a means of communicating and cooperating symbolically- a 'good enough' road map for all parties. (Star and Griesemer 1989, p. 410)

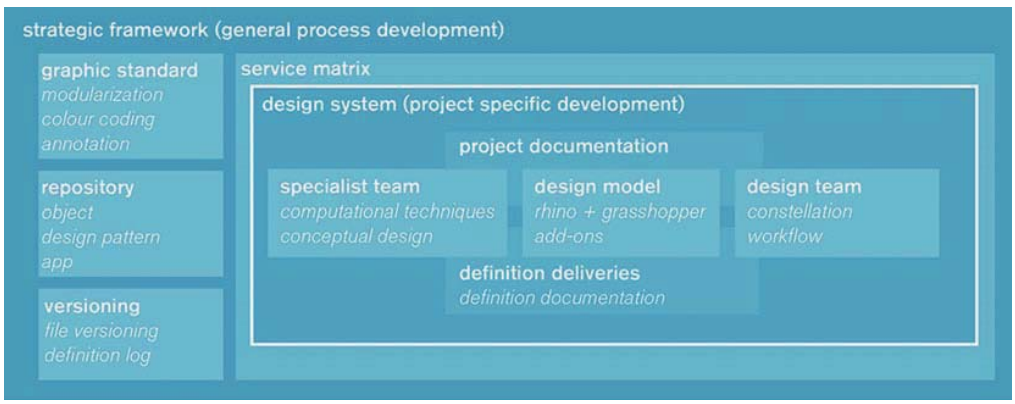


Fig. 2 The strategic framework and deployed boundary objects

Fig. 3 The service matrix, with the status of the case Forumtorget marked

On the matrix design axis, representing the expected level of design development and architectural innovation, the *Inspiration and Information* level means that the project is discussed in relation to design references and computational methods. In cases where a design concept is well developed and explicitly articulated, a post factum second order implementation is regarded as a *parametrisation*. When Dsearch uses a developed model to deliver a design proposal, it is labelled as *design development* whereas the *concept development* level implies Dsearch taking part in shaping the design concept—introducing computational thinking. *Innovative design* starts with the core value of innovation and is benchmarked against the international design field.

On the methodology axis, representing the level of complexity in computational development, Dsearch can provide *guidance* to the design team regarding relevant methods for the specific project. Design patterns from the repository can be *repurposed* to fit the project needs, but if the necessary modifications are plentiful, complicated or involves new methodology this is considered *system development*. The next level applies when general and computational design methods are intertwined with other specialist methods (energy, daylight, structure etc.) into an *integrated development*. *Innovative methodology* is needed to address problems where no preceding methods are identified.

The *Design Model* is a first and a second order model, coupled; in our case, this currently means a Rhino/Grasshopper, or a Revit/Dynamo model, plus associated data files. When applied in projects, the design model is expanded to a *Design System*, a curated assemblage continuously formed by the models, but also by the design problem at hand and the human actors of the project in addition to methods and tools—bespoke and conventional. The design system is explicitly situated on a specific coordinate in the service matrix. In this way the matrix mediates the formation of each design system, aiding the specification of what resources from the strategic framework and the larger practice will have to be included in the design system in terms of engagement and expected outcome.

The continuous specification of the design system establishes explicit outer borders for the project—design problem, concept, client intentions, etc.—for both second order development and first order design teams. In that way the strategic framework regulates collaboration and supports the flow of information within and outside of the design system. Star and Griesemer terms this a *Coincident Boundaries Object* where the "…result is that work in different sites and with different perspectives can be conducted autonomously while cooperating parties share a common referent." (Star and Griesemer 1989, p. 410).

Re-usable sections of definitions are routinely extracted, stored and indexed in one of three formats—Objects, Apps or Design Patterns—at a central *Repository*, along with a documentation of key specifications (Fig. 4). The act of editing and documenting parts for re-use also doubles as an explicit annotation to the original definition. Star and Griesemer mean that the index and modularity of a Repository allows that people "…from different worlds can use or borrow from the 'pile' for their own purposes without having directly to negotiate differences in purpose". One intention behind the repository is just that: hopefully objects developed by Dsearch will find their way to unexpected uses within the larger practice (Star and Griesemer 1989, p. 410). *Objects* (User Objects in Grasshopper nomenclature) wrap a section of the definition into a component with the same affordances as standard components (such as search, legibility and commenting) with the difference that the underlying section still is available for later revision. This format is used for generic processes with an expectedly wide use; different projects or multiple instances within the same definition. *Apps* (in the everyday sense) are complete and freestanding definitions with a specific purpose, as close as Dsearch gets to software development. These repository formats can be said to act as Standardized Form boundary objects in the sense that they strongly regulate the flow of information

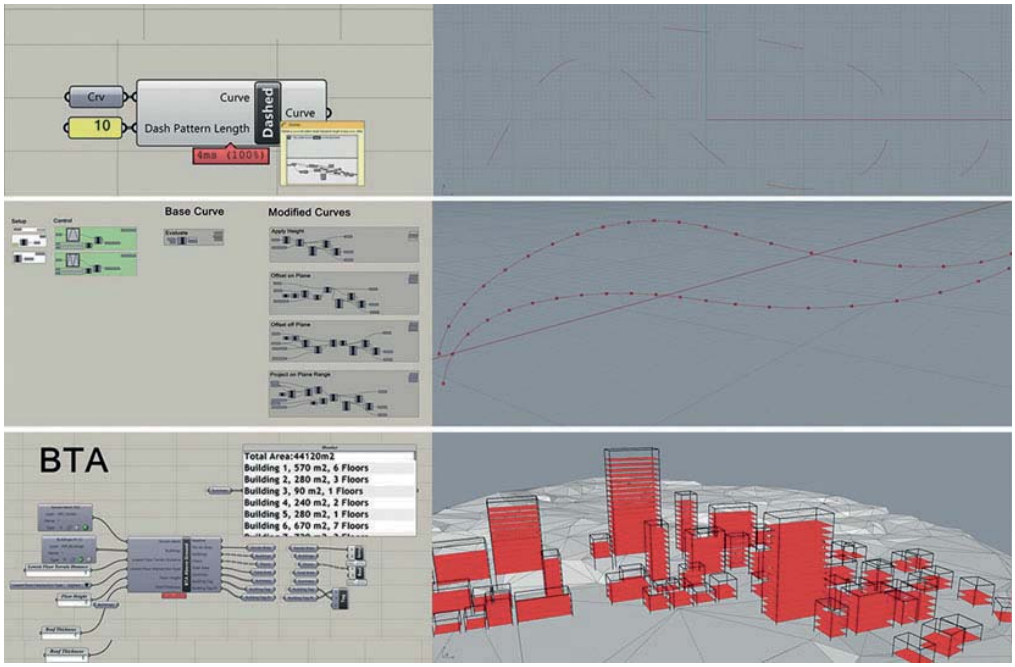


Fig. 4 Examples of objects, design patterns and apps

within a design model in a way that is consistent across all contexts. "The advantages of such objects are that local uncertainties … are deleted" (Star and Griesemer 1989, p. 411).

Design Patterns, "above nodes but below designs" (Woodbury 2010, p. 187), on the other hand, are more complex, specific and often need modification when applied to a new definition context. They may or may not have a distinct geometrical output, but are seen as distinguishable parts of a definition that provides a particular outcome. Preserving ease of editing is prioritized before application, thus patterns are stored as snippets in the standard definition format.

Other prominent boundary objects of the framework are the *Graphic Standard* (Ideal Type, Fig. 5), stipulating annotation and modularization of the definitions for legibility and

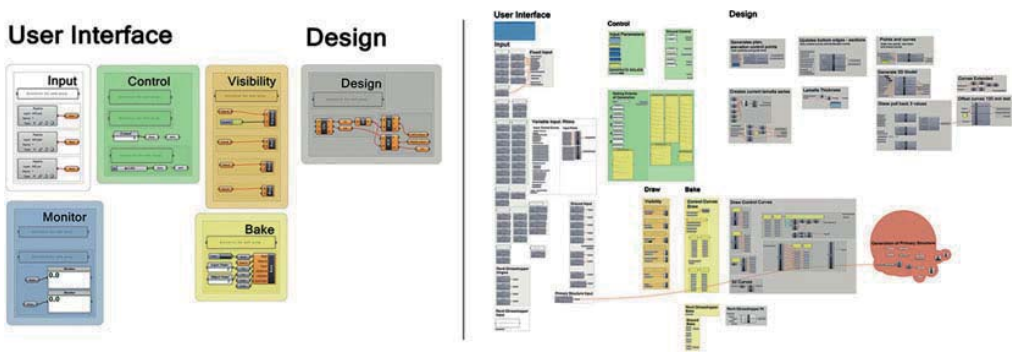


Fig. 5 The graphic standard for grasshopper, as implemented in Forumtorget

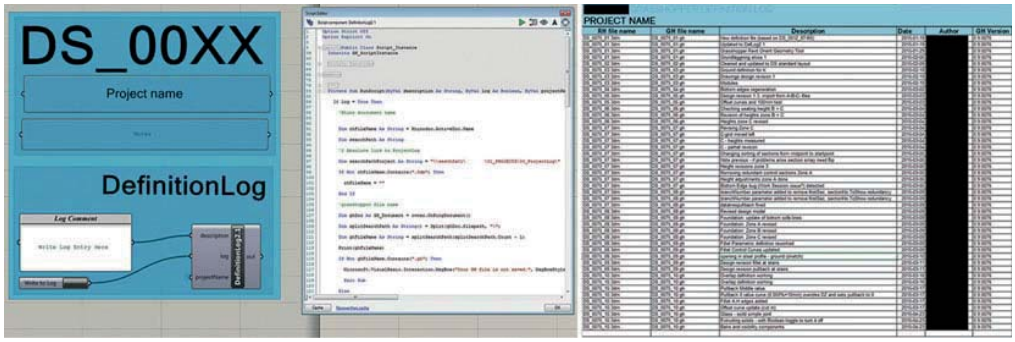


Fig. 6 The definition log object (with script) and definition log

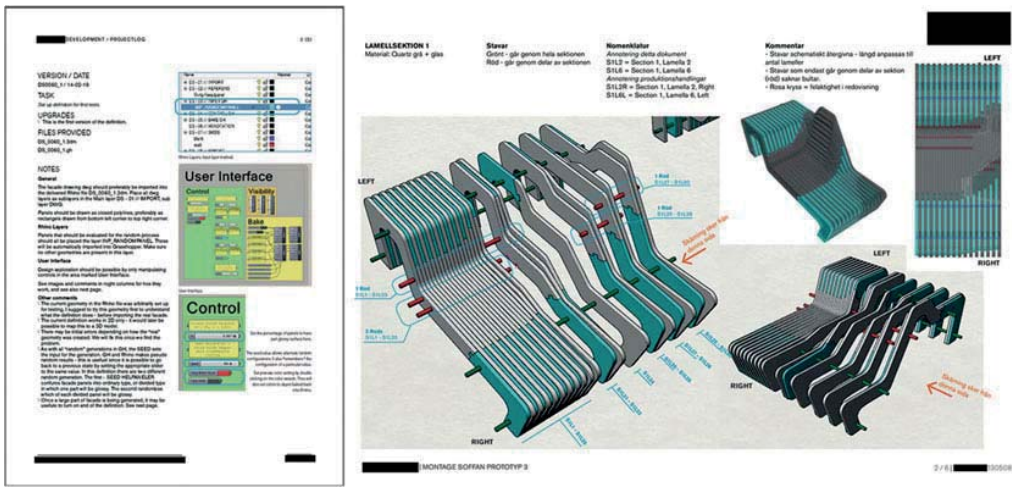


Fig. 7 The definition documentation and project documentation from Forumtorget

reusability (Davis et al. 2011), and different ways of regulating the versioning of design systems—the *Definition Log* object documents brief technical development to a spreadsheet archive, a *Definition Documentation* template is used to communicate changes in functionality between key stages and for deliveries (Fig. 6) and a Project Documentation to communicate design outcome (Fig. 7). As boundary objects they belong to the Standardized Forms type—methods for common communication across dispersed work groups (Star and Griesemer 1989, p. 410). The graphic standard and definition log facilitates easier collaboration within the computational design team, while the definition documentation documents advanced use and provides relevant information for a general practitioner to use a delivered design system. A separate *Project Documentation* template is used to present design outcome to external parties.

Applied Strategies in Project Development

Based on a competition win for Forumtorget, Uppsala, in 2011 (Fig. 1), this urban furniture has been developed through series of computational design models, physical models and prototypes. The programmatic and formal concept includes

the manipulation and variation of seating configurations—the cross section is continuously shifting while providing a variety of seating opportunities on two sides over the 60 m length of the bench. Different ground levels and the integration of stairs into the design added further complexities handled through the design system, which was also set up for deliveries of production documentation. Due to the development and construction of an adjacent building, the design process was extended over time with intermittent activity. This allowed a development in distinct phases, dependent on the strategic framework to make sure that past design evaluations are not lost, and that changes in the development team over time does not impede on development.

The design system enabled that final decisions on form could be postponed to a very late stage, while bespoke technical solutions and general principles were resolved to the level that procurement could be handled. This follows the arguments made by Daniel Davis as he challenges the conventional model of high influence and low expenditure in early project phases, claiming that parametric models can "shift the cost of change in relation to design effort", "allowing designers to defer key decisions until later in the project" (Davis 2013, p. 208). In the experience of the authors, this is indeed possible, but in relation to overall project workflows it has required the strategic framework that expands the design model to a design system, and provides boundary objects that supports interactions over time.

The Forumtorget project has undergone four main development stages (Fig. 8). In the first competition stage, a simple design presented variation as a concept, developed through a basic computational model with visual representations as the main outcome. In the second stage, form development was conducted in order to set the boundaries in relation to design identity and site

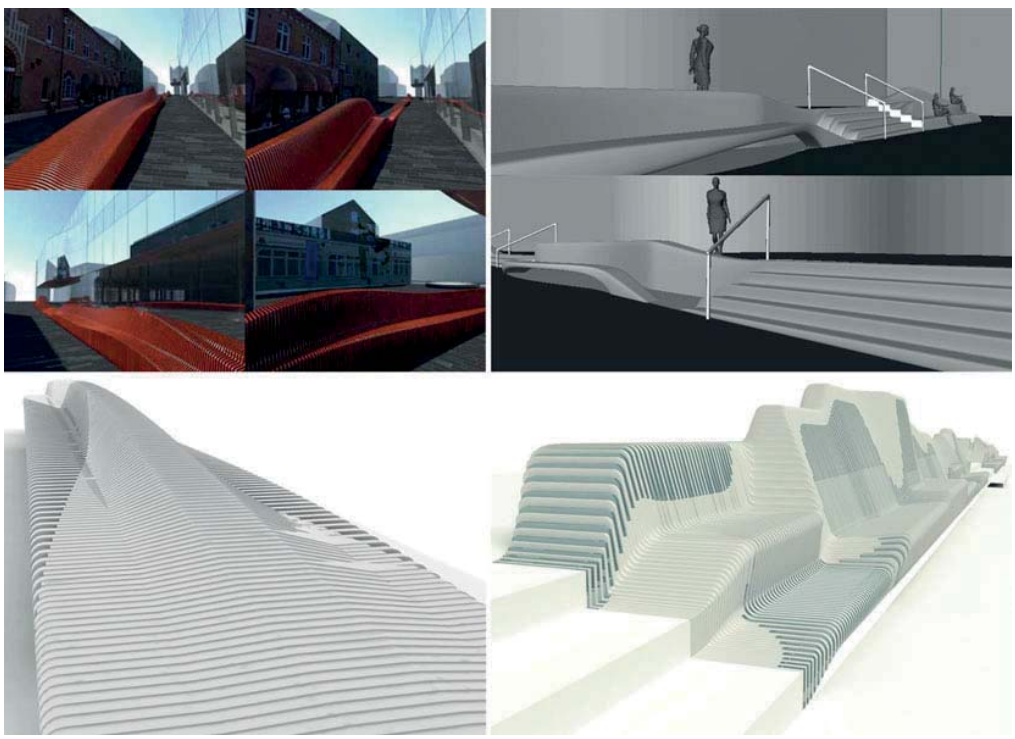


Fig. 8 Representations of Forumtorget from stages 1, 2, 3 and 4

specific conditions such as the two different levels of ground. The third design stage involved an iterative study of the form, structure, manufacture and assembly of the bench, through a continuously refined parametric model, with input from specialists as well as feedback from series of physical models and full-scale prototypes.

During the fourth stage, the technical details and structure were refined, the overall form revised, the design system consolidated to avoid errors in final production, and the generation of documentation for procurement and production initiated. From stage two to four, care was taken to keep exploring the overall spatial principles, yet not fixating any part completely.

A number of early key decisions allowed the continuous development based on the design system. A lamella-based form and fabrication principle that allowed rational fabrication of the "free-form concept" was selected. The cross-section was based on a seven segment polyline with individual control of all corners and their fillets, setting restrictions of possible formal variation, yet handling the zone-specific conditions and providing a formal continuity along the length of the bench as well as an adaptation to differentiated ground conditions. The use of control diagrams; the computational transformation of series of 2d curves to the design model, provided a control interface to overall form as well as detailing (Fig. 9).

Once the premises for the design system were in place, the project was developed through the established versioning approach facilitated through the strategic framework. Supported by a series of design reviews where alternate solutions could be discussed with different specialists, the second order development could operate through iterative loops, allowing technical solutions to be set while the specific form was continuously refined. The use of the definition log also provides a back-log of the development, where the following milestones can be identified:

- Versions 5–8: Setting up basic geometrical generation from control curves (Fig. 9).
- Versions 23: Generating production documentation for first prototype (Fig. 10).
- Version 38: Optimizing definition performance by introducing python scripts.
- Versions 41–58: Developing second prototype (Fig. 10).
- Version 88: Setting up relation to Revit for interface to overall project.
- Version 90: Setting up modularization for production and assembly.
- Version 95: Developing concrete foundations design model.
- Version 97: Developing joint between glass and solid materials.
- Version 100: Overall formal design revisions.
- Version 101: Developing primary steel structure.
- Version 102: Generating procurement documentation (Fig. 11).

The extended development time of the project made documentation very important—a number of different Dsearch members have entered and

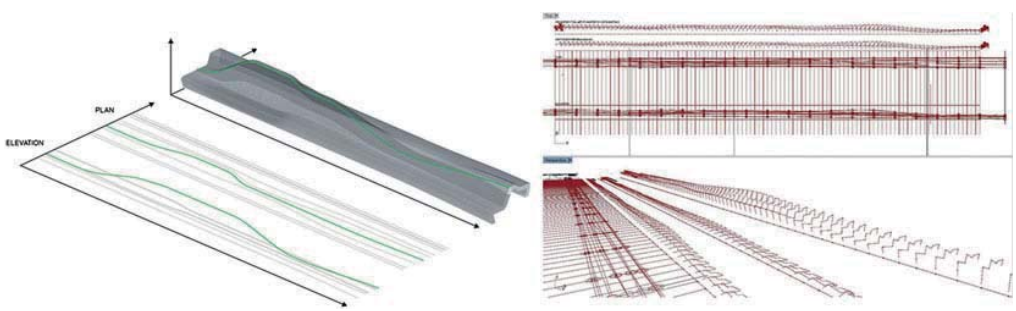


Fig. 9 Control curves representing plan and elevation as a diagram, and view from the design model

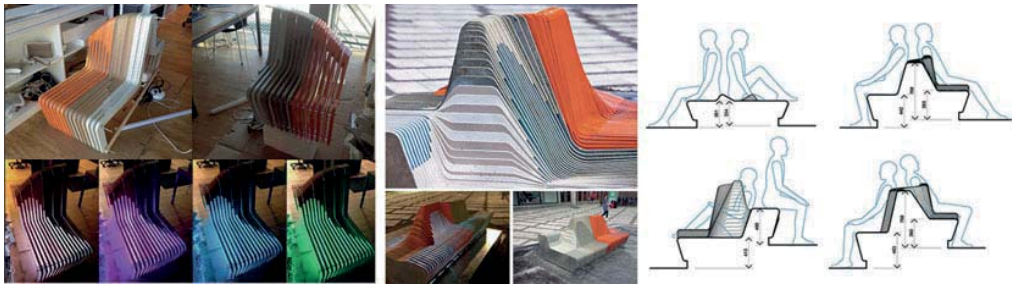


Fig. 10 First and second physical prototype, and representations from stage 3

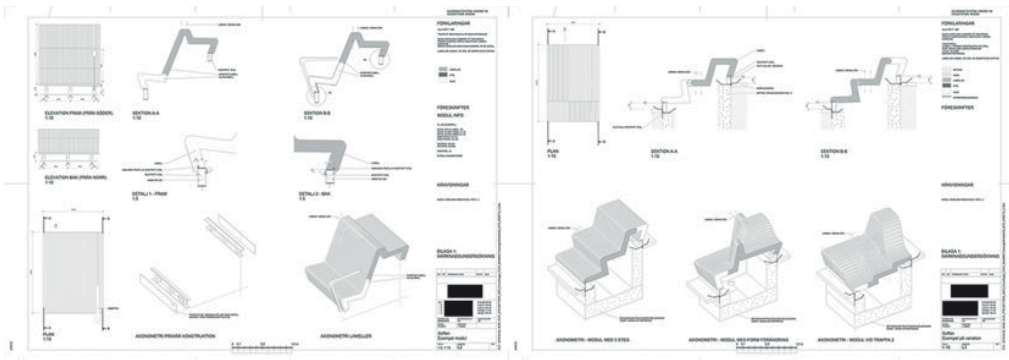


Fig. 11 Procurement documentation from stage 4

exited the project, and the definition log and definition documentation allowed an understanding of previous development. The close collaboration with other specialists and the client also required a continuous use of a project documentation, in particular the development and assembly of full scale prototypes. The overall design process in this way reflected an exploration of two related trajectories; the design development based on several specialisms, where issues such as comfort, identity, materials, structure and production informed the ongoing versioned computational development. In the fourth stage all these issues converged into a close to final design proposal, where also computational principles for procurement and production documents were part of the design system.

Concluding Remarks

Computational and second order design provides architectural practice with new assets, but the deployment in practice may face unprecedented challenges. This paper shows how a strategic framework can be applied in order to establish an infrastructure that in practice mitigates these challenges. Together with general design management, it affects design process and outcome, as well as organizational learning. In relation to this paper, single- and double-loop learning in computational design can be regarded in two opposing ways; one the one hand, computational design is highly procedural—with the proposed strategic framework that may seem even more

regulated—suggesting single-loop learning within a given framework; on the other hand, it can be regarded in contrast to existing management models within architectural practice, where the procedures taken are very familiar—we know the modes of representation, (we believe) we know what a client wants, and we understand how the construction industry operates. Given that computational design provides new opportunities in conventional practice, but depend on special skills and a rapid technological advancement—the purpose is to use the strategic framework to enable a continuous exchange between first and second order design, providing double loop learning within both contexts. As the field of computational design continues to develop, the framework enables additional aspects to be introduced, such as iterative analysis, material performance and additive manufacturing.

The strategic framework and its constituent bounding objects is to be seen as a bottom-up approach to managing design processes and flows of information across a heterogeneous environment within a larger firm. Grounded in computational design thinking, it provides an alternative to current trends of explicit process modelling and comprehensive BIM standardization. The associated terminology from Sociology and Knowledge Management has in the development of Dsearch provided useful concepts bridging between first and second order of design and addressing issues beyond computation and design—the interactions between designers of different backgrounds. In essence, it provides a deeper understanding of elements that previously made sense in an intuitive way and articulates them. The terminology opens up these interactions for a wider discussion—by aligning ourselves with recent research on practice cultures elsewhere, we are able to distinguish and evaluate potential assets for future practice and research.

Acknowledgements Forumtorget design team: Gustav Jarlöv, Sam Keshavarz, Jens Modin, Andreas Milsta, Torbjörn Eliasson, Michael Malmborg, Jonas Wannfors. Dsearch team: Jonas Runberger, Frans Magnusson, Hamia Aghaiemeybodi, Pedram Seddighzadeh Yazdi, Vladimir Ondejcik. All images © White arkitekter AB.

References

Argyris C (1999) On organizational learning, 2nd edn. Blackwell, London

Davis D (2013) Modelled on software engineering: flexible parametric models in the practice of architecture. PhD dissertation. RMIT University, Melbourne

Davis D, Burry J, Burry M (2011) Understanding visual scripts: improving collaboration through modular programming. Int J Architectural Comput 9(4):361–376

Heylighen F, Joslyn C (2001) Cybernetics and second-order cybernetics. In: Meyers RA (ed) Encyclopedia of physical science and technology, 3rd edn. Academic Press, New York

Star SL (2010) This is not a boundary object: reflections on the origin of a concept. Sci Technol Hum Values 35:601

Star SL, Griesemer JR (1989) Institutional ecology, translations' and boundary objects: amateurs and professionals in Berkeley's Museum of Vertebrate Zoology 1907–39. Soc Stud Sci 19(3):387–420

Wenger E (1998) Communities of practice, learning meaning and identity. Cambridge University Press, Cambridge

Woodbury R (2010) Elements of parametric design. Routledge, London

Paper2

Design System Assemblages: The Continuous Curation of Design Computation Processes in Architectural Practice

Magnusson, Frans, and Jonas Runberger. 2017. “Design System Assemblages: The Continuous Curation of Design Computation Processes in Architectural Practice.” In *Professional Practices in the Built Environment*, edited by Rowena Hay and Flora Samuel, 194–204. University of Reading, UK. https://www.reading.ac.uk/architecture/soa-professional-practices-conference.aspx.

Design System Assemblages – the continuous curation of Design Computation Processes in Architectural Practice

Frans Magnusson[1], Jonas Runberger[2]

[1], *White arkitekter AB*
SRE In the making
Chalmers University of Technology, Sweden
frans.magnusson@white.se

[2]*White arkitekter AB*
Chalmers University of Technology, Sweden
jonas.runberger@white.se

Abstract

This paper maps design systems, a mode of operations formulated by Dsearch, a design computation R&D unit at White arkitekter AB. The authors also discuss the organisational learning resulting from facilitation of architectural design with computational methods and development of bespoke workflows. Two design system cases are described using assemblage theory, as developed by Manuel DeLanda. This materialist ontology is found useful, both in terms of research reflexivity and descriptive clarity. The authors critically assess their position as insider action researchers; rather than perceiving academic knowledge as necessarily distinct from practical, the paper shows that knowledge produced in design practice, research and development in practice, and academic research, differs in degree - not in kind. Design computation management is considered an emergent mode of architectural practice, beyond the specific aspects of form making - bridging project, development and research dynamics. The research and design methodologies laid out here should be read as steps towards an epistemological foundation for prototype driven organisational learning with respect to design computation in architectural practice.

Key words: architectural practice, design computation, assemblage theory, action research, prototype

Introduction

This paper maps the assemblages of professionals, communities, tools, methods, design concepts, code, algorithms and systems emerging in projects and feeding back into methodology within architectural design computation practice. Investigating design systems, a mode of operations formulated by Dsearch (Runberger and Magnusson, 2015), an in-house design computation R&D unit at the large architecture firm White arkitekter AB; the authors account for the organisational learning involved in facilitating architectural design with computational methods and the development of bespoke workflows for these methods. Two design systems are described in functional terms and in chronological sequence, so as to capture the historically contingent nature of these entities. SOfFTA is a design driven development mapping neatly onto one architectural commission, albeit with changing conditions over time. Urban Values is developed as a support system distributed over many interested parties.

As practitioners within Dsearch, the authors will critically assess their position as insider action researchers (Brannick and Coghlan, 2007) as well as what kinds of knowledge can be produced in design practice (Hensel and Nilsson, 2016). In this paper, case material is discussed using assemblage theory, as developed by Manuel DeLanda (2006). Rather than using idealist notions such as categories and essential qualities, morphogenetic processes define and explain populations of individuals. Other materialist thinkers such as Andy Clark (2008), Bruno Latour (2005) and Levi Bryant (2014) let us conceive architectural practice as a social process mobilising non-human entities alongside humans (Callon, 1984), allowing the researcher to attribute humans, artefacts, and environment distributed agency. The authors find this world view valuable for describing and managing the messy reality of daily practice.

Design computation management is here considered an emergent mode of architectural practice, beyond the specific aspects of form making, performative design or process optimisation - bridging project, development and research dynamics. This discussion is connected to a perceived potential of the design system assemblages as vehicles for new forms of organisational learning. One aim for Dsearch, is the continuous development of new knowledge - in addition to methods and procedures. Rather than perceiving academic knowledge as necessarily distinct from practical, the authors propose to place practice-based academic knowledge on a continuum together with several other types of data, information and knowledge. The ambition of this paper is to exemplify how this approach plays out in practice.

Insider Action Research and Prototypical Processes

Organisational research scholars Teresa Brannick and David Coghlan mount a defence of researchers being native to the situation under study. They argue that "as researchers through a process of reflexive awareness, we are able to articulate tacit knowledge that has become deeply segmented because of socialisation in an organisational system and reframe it as theoretical knowledge and that because we are close to something or know it well, that we can research it." (Brannick and Coghlan, 2007, p60). The approach behind this paper aligns well with Brannick and Coghlan's definition of insider action research as epistemologically subjectivist, but within an objectivist ontology. This implies that any object of study is independent of the researcher, but what she can know in a specific situation is determined by her vantage point and interaction with the object.

An insider position obviously brings several advantages in terms of access to and pre-understanding of the studied situation, but also

requires heightened reflexive sensibilities towards personal and cultural biases of the researcher and her community. One method to achieve this reflexivity is formulated by management scholar Mats Alvesson; he suggests the use of "theories which challenges common sense, not only for the direct application but also for encouraging perspective on one's own lived reality and thus facilitating looking upon things in a more all-sided way…" (Alvesson, 2003, p186). For this research, as well as for design and development at Dsearch, DeLanda's assemblage theory is one such challenge.

The parallel engagement in both design process and method development requires an iterative mode of operation, which requires different modes of development, and enables different modes of learning. Direct learning by doing, using already established computational design methods and applying them in a specific project can be regarded as single-loop learning, as defined by Chris Argyris [Argyris (1999), pp68-71). When operating both directly and at a method development level, double-loop learning is important, in the way that design systems as assemblages also effects the guidelines and systems directing work within a practice. This in turn requires a continuous development and testing of prototypes, where final objectives are still blurred, and cannot be specified – reflected in the general prototypical nature of operation adopted by Dsearch, as opposed to a specification driven development process (Schrage, 1999). The research and design methodologies laid out for this paper should be read as steps towards an epistemological foundation for prototype driven organisational learning with respect to design computation in architectural practice.

Design System Assemblages

Design systems comprise physical and computational models and drawings; the human actors of the project and the groups they form - design team, r&d team, specialists, clients, external consultants, etc; but also design issues, policies and contracts; in addition to methods and tools – bespoke and conventional. Of course all aspects above are present in any project; it is the use of scripting to embed intellectual work in code, making it explicit, durable and operational, that gives meaning to the curation of these system assemblages as a design task transcending project engagements. This task requires a sensitivity towards existing as well as emerging communities of practice (Wenger, 1999) so that the embedded intellectual work is not lost in standardisation. The nexus of this curation lies at the meta level of tailoring the assemblage properties for each specific project - designing design workflows.

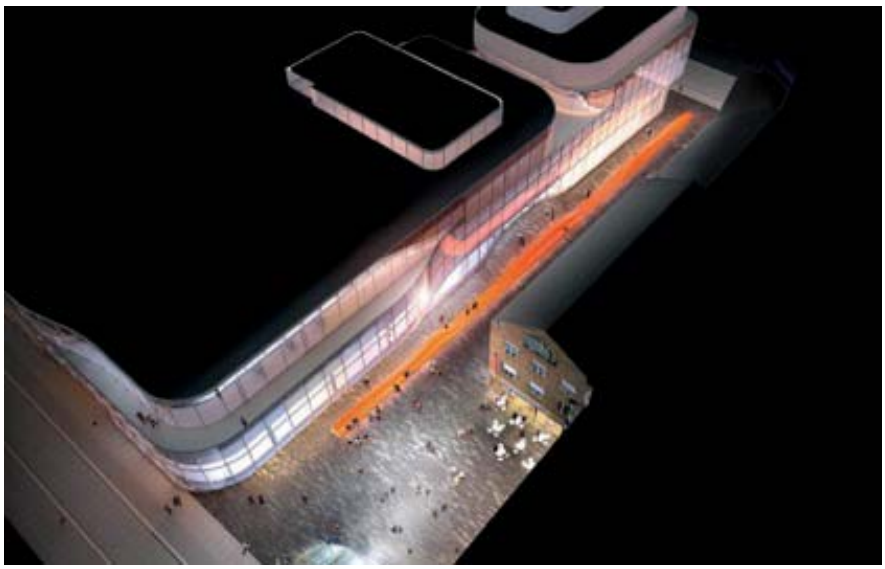
Manuel DeLanda defines assemblages as "wholes that are irreducible and decomposable" (2011, p185). This means that wholes have an agency that cannot be fully explained by the agency of their parts, and conversely: each part has agency that is not fully explained by its relations within the assemblage. DeLanda states that these relations of exteriority let assemblages accumulate or lose parts without automatic loss of identity; neither at part or whole level. Such contingent historical processes are how assemblages are formed and maintained: a population of components (themselves assemblages at a lower level of complexity), come together as an individual in a stable arrangement with emergent qualities and agencies.

DeLanda gives us two parameters to analyse the identity of assemblages; territorialization and coding. "The more homogeneous the internal composition of an assemblage and the better defined its outer boundaries the more territorialized its identity may be said to be" (2011, p187). A territorialized system can for instance display sharp boundaries by requiring specialist knowledge of its users. Coding then determines the potential for

change in territorialization, highlighting the power of design computation to decode the previously very change resistant world of cad software.

the SOfFTA Design System

Developed over a number of generations, this design system provides the workflow for a complex and site-specific piece of urban furniture – part of a winning competition (titled SOfFTA) for the Forumtorget square in Uppsala, Sweden. The extended development time over several years reflect conditions external to the project, such as the ongoing construction of an adjacent building. This has affected the assemblage in several ways, facing updates of software, new design team constellations, new client representatives and an extended tendering process. The urban furniture – a 65 meter long bench with a continuously shifting section, negotiates a border between different ground levels. Currently at the start of final production and assembly, the bench will be constructed from planar cut sections of quartz composite and glass. With an important objective to allow for formal design decisions very late in the process, producer final design revision is underway integrating producer specific constraints. This implies that organisational logic is more strongly coded than form in the morphogenetic process. On the level of the design artefact, this is a highly territorial assemblage made of relatively similar components with an intricate but clearly defined boundary.

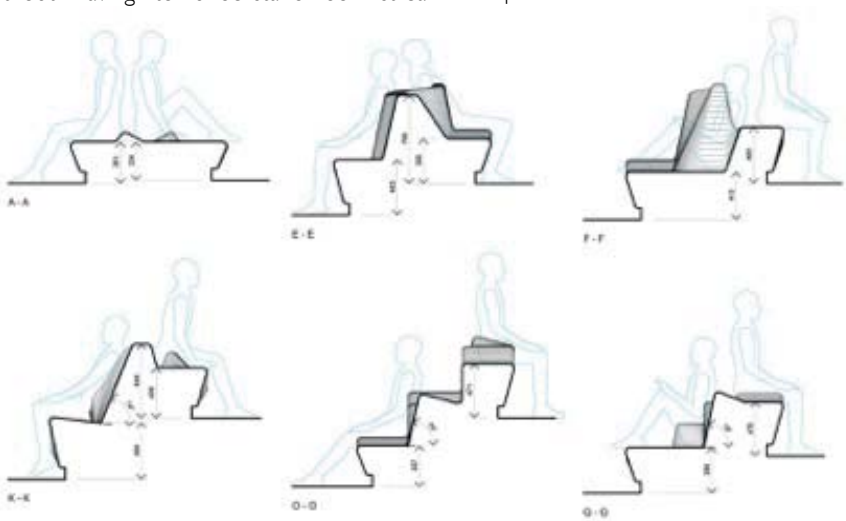


SOfFTA

Workflow

The current design workflow is based on a number of two-dimensional polyline sections and control curves manually manipulated in the geometric modelling application Rhino , processed through a Grasshopper script to generate fabrication outlines, allowing overall control of the form as well as detailed control of all seating configurations. This allows for a slight deterritorialization of the workflow where a designer can focus on producing different ergonomic seating section curves without having to understand downstream processes in the script. Here, the section curve is strongly coded, so that it stays geometrically intact in the interactions with other design elements.

A digital solid model is generated solely for the purpose of design evaluation in renders and 3D prints. Overall the workflow reflects team organisation and design task with boundaries defined by the Rhino/Grasshopper design model, but deterritorialized and decoded just right to invite specialists and host project members to interact with the design at specified interfaces.



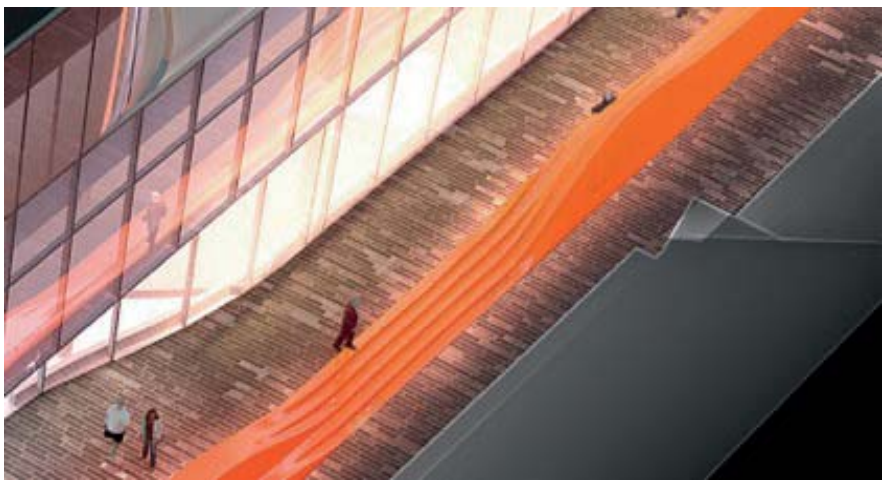
Seating configurations

Digital and physical models

History

The 1st generation of the design system assemblage was set up for fast conceptual development and representations to be included in the competition proposal. A directly manipulated Rhino geometry was transformed into sectioned lamellas through the associated Grasshopper script. A highly territorial setup, the definition was developed by Dsearch, and used by a Dsearch specialist as part of the competition team consisting of architects, landscape architects and artists.

After the competition win, a designer with advanced modelling skills but limited computation experience was introduced into the design team, deterritorializing the 2nd generation of the assemblage. Early form experimentation through the use of Rhino with the T-Splines plugin allowed for easy formal variation, but this approach was abandoned due to lack of parallel global and local control.

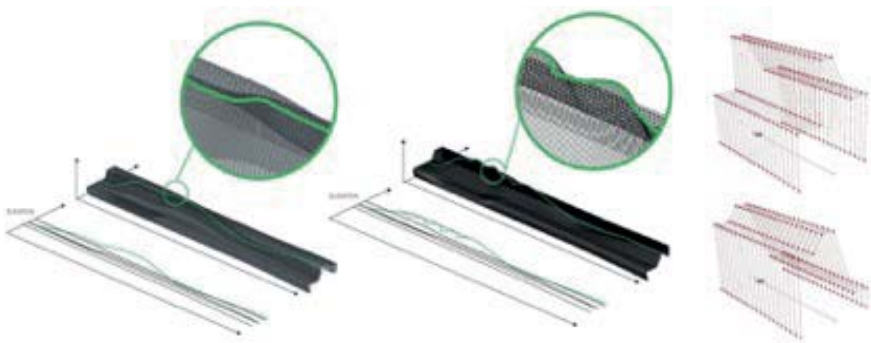


Competition proposal

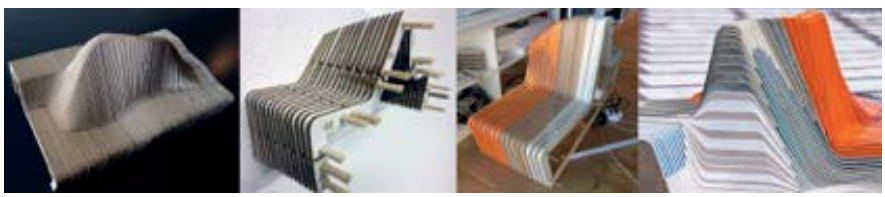
Early experimentation

Once geometrical principles were established, a new design model was set up, using two-dimensional Rhino splines for elevation and plan control of the eight control points of each lamella section polyline, generated and organised in the Grasshopper script. In order to achieve further control of specific seating configurations, a number of manually set Rhino spline control sections were introduced, and the earlier splines were used to control the transition between different sections. This coding of geometries in the morphogenetic process was traded for a deterritorialization of the workflow. As the need for parallel design and computation development intensified, in particular due to contextual considerations and the integration of structure, the design development was from this point re-territorialized within Dsearch.

In generation 3 the assemblage was again opened up for the integration of a furniture designer and a furniture producer, enabling advanced development of production aspects as well as seating configuration for comfort. This process was carried out through a series of five design reviews, each based on incremental development of the design system. During these sessions, issues such as form, materials, lighting, detailing and production aspects were coded by decisions based on digital representations, physical models and full-scale mock-ups.



Control splines



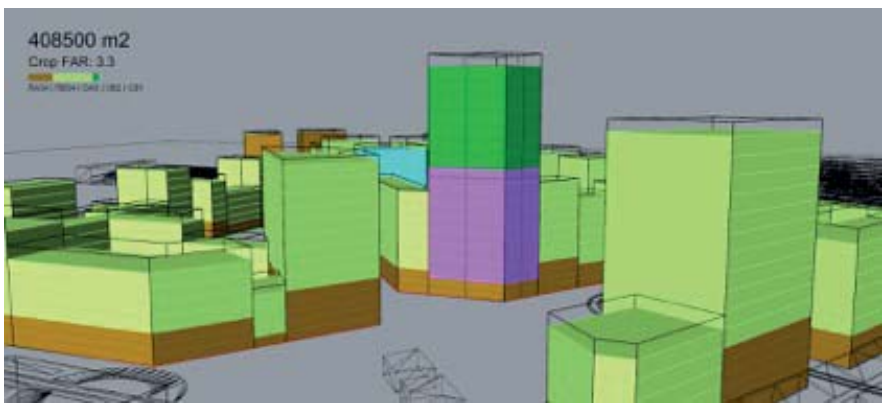
Models & prototypes for design reviews

In generation 4, client side project leaders for the overall construction process of the square joined the assemblage, and automated procedures for tender and construction documentation were developed. This stage ended with the completion of the tendering process. The currently initiating 5th generation entails the production. This includes final adjustments to the design systems due to feedback from production tests with the actual machines to be employed, as well as a final design revision of the overall form.

the Urban Values Design System

While supporting a relatively simple task, to quantify and display information such as floor areas and ownership in realtime while sketching building volumes; this system exhibits complexity in that it presents several parallel workflows and interfaces. The contingent nature of its development is highlighted by the number of commissions and internal development instances at White involved (so far 12).

Automated documentation



Realtime display

Workflow

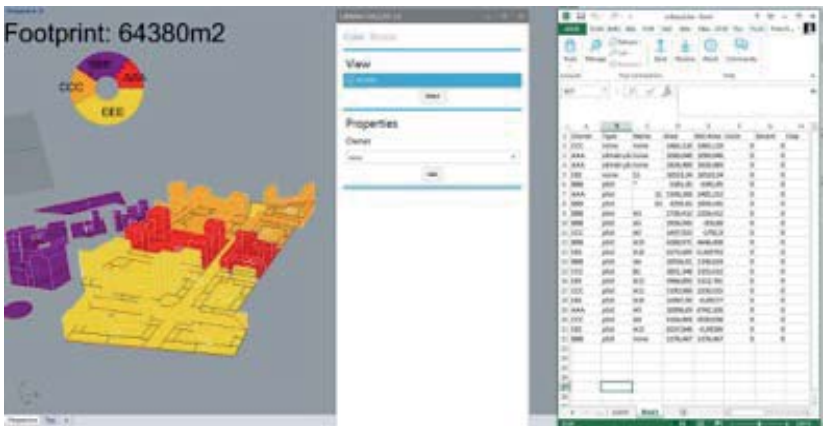
The current workflow is based on direct manipulation modelling of building volumes in Rhino, together with structured input of parameters from a spreadsheet. This information is then processed in a Grasshopper script which executes a change in the model visualisation and the data report. As one example, floor use data determines the building volumes to be sectioned into sequence of floors with specific area types. On screen the floor volumes are colour coded by use type and the area data is summarised in a heads-up display. Textual data is sent to Excel for further aggregation.

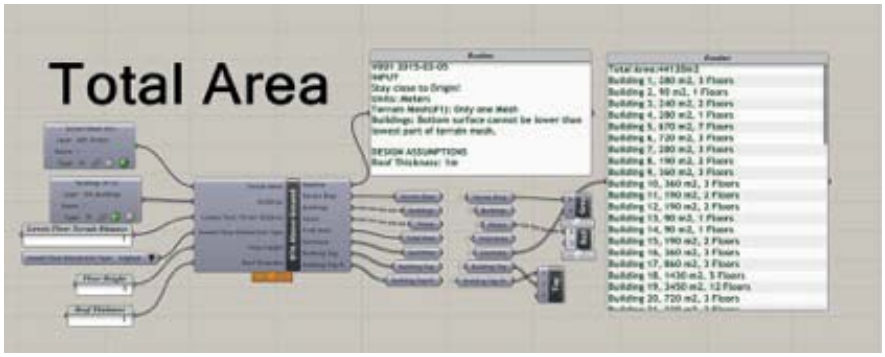
Workflow

Grasshopper is in this case fully controlled by an external user interface, removing the necessity to be familiar with GH scripting. Dividing control between Excel, Rhino and the user interface creates a somewhat heterogeneous workflow; but because the reliance on software that are standard at White, the design system is still accessible for any colleague after a short introduction. To set up the system on the other hand, requires the installation of many individual software modules per design team member.

History

The system history starts with a colleague developing a script for early stage urban studies and shortly thereafter leaving White. The principal of commission A contacted Dsearch to assist in the use of the script. Having seen it in use before, the principal had a good conceptual understanding of what it could deliver, but no means to operate it in terms of documentation, user interface or personal Grasshopper experience. Considering the unstructured state of the Grasshopper script the decision was made to develop a new script from scratch. The assemblage thus survived scrapping what at first glance could be regarded as 'the object' by maintaining intent, user and project context. At this point floors would only be distributed above ground; this rather specific feature was present in the system from day one because of the demanding topography in which its original host project was set.





Commission B, also set in difficult terrain, had the need to combine the system with other workflows in Grasshopper. To simplify the use of the Grasshopper script it was packaged into a cluster, essentially a black-box where users have less understanding of its internal workings. Clustering could be described as territorializing the script by giving it sharper boundaries, making it easier to instantiate in a project context, but at the same time it is also re-inserted into a larger more heterogeneous project assemblage. Black-boxing is also a form of coding, making the script more resistant to change.

Cluster

Commission C, set on flat land, demanded the ability to handle complex building programs and produce spreadsheet reports via a web service for further downstream aggregation. The introduction of an external user interface turned the visual aspects of the Grasshopper script into back-end eliminating the need for a cluster and the script was again deterritorialized. A head-up display was developed to get instant visualisation of the model data. This development phase produced a workflow that with its strongly coded realtime feedback on geometry could facilitate a more exploratory, decoded, approach to the building modelling.

Commission D, set on a slight incline, had specific modelling intentions regarding ground levels and basements. This required for the first time in the system history an option to not subtract the terrain from the buildings; an example of the contingent and historically determined nature of this kind of development. In contrast with commission C, where the host project implementation was carried out by a colleague involved in the system development and also seated at a desk within visible range from the lead developer; commission D, set in another country, involved a more complex project organisation and implementation coordinated over the phone. Greater local complexity at longer distance from the development team led to risks being assessed differently. Subsequently, data exchange between Rhino and Excel was rolled back to an earlier version using a plug-in, removing the necessity to register team members at a web service.

Discussion

In addition to provoke reflexivity in research approach by challenging common sense, assemblage theory also explains composite arrangements and messy situations. This attained ontological sensitivity could be brought to bear on design activities at all levels: where does the boundaries of any given object really lie? Applying this theory to the case material gave rise to the insight that the analysis of an object is also always an analysis of its immediate situation. Despite being housed

in a materialist ontology, the way to know an assemblage is to put it into action, something for which designers are well prepared.

Brannick and Coghlan point to the sentiment of Mats Alvesson that observing participant is a better term to use than participant observer. "Participation comes first and only occasionally is complemented with observation in a research-focus sense" (Brannick and Coghlan, 2007 p66). This term strongly resonates with Schön's reflective practitioner (Schön, 1983); arguably both roles are approaching the same stance towards knowledge from the positions of academia and practice respectively. The authors hope to have shown that knowledge produced in design practice, research and development in practice, and academic research, differs in degree - not in kind. The key difference lies in specificity; not all knowledge is relevant for all audiences. If the responsibility of an observing participant is to react to a situation where knowledge of wider use is produced and ensure that documentation is carried out in a way that allows for further curation - a reflective designer could also pro actively engage in arranging such situations towards specific knowledge outcomes.

References

Alvesson, M. (2003). 'Methodology for close up studies – struggling with closeness and closure.' *Higher Education*, 46(2), pp. 167–193.

Argyris, C. (1999). *On organizational learning.* Wiley.

Brannick, T., & Coghlan, D. (2007). 'In defense of being "native": The case for insider academic research.' *Organizational Research Methods*, 10(1), pp. 59–74.

Bryant, L. (2014). *Onto-cartography: An ontology of machines and media.* Edinburgh University Press.

Callon, M. (1984). 'Some elements of a sociology of translation: Domestication of the scallops and the fishermen of st brieuc bay.' *The Sociological Review*, 32, pp. 196–233.

Clark, A. (2008). *Supersizing the mind: Embodiment, action, and cognitive extension.* Oxford University Press.

DeLanda, M. (2006). *A new philosophy of society: Assemblage theory and social complexity.* A&C Black.

DeLanda, M. (2011). *Philosophy and simulation: The emergence of synthetic reason.* Bloomsbury Academic.

Hensel, M. U., & Nilsson, F. (2016). *The changing shape of practice: Integrating research and design in architecture.* Routledge.

Latour, B. (2005). *Reassembling the social: An introduction to actor-network-theory.* OUP Oxford.

Runberger, J., & Magnusson, F. (2015). 'Harnessing the informal processes around the computational design model.' In *Modelling behaviour* (pp. 329–339). Springer.

Schön, D. (1983). *The reflective practitioner: How professionals think in action.* Basic Books.

Schrage, M. (1999). *Serious play: How the world's best companies simulate to innovate.* Harvard Business Review Press.

Wenger, E. (1999). *Communities of practice: Learning, meaning, and identity.* Cambridge university press.

Paper3

Morphology & Development - Knowledge Management in Architectural Design Computation Practice

Magnusson, Frans, Jonas Runberger, Malgorzata A. Zboinska, and Vladimir Ondejcik. 2017. “Morphology & Development - Knowledge Management in Architectural Design Computation Practice.” In Proceedings of the 35th ECAADe Conference - Volume 2, Sapienza University of Rome, Rome.

Morphology & Development

knowledge management in architectural design computation practice

Frans Magnusson[1], Jonas Runberger[2], Malgorzata A. Zboinska[3], Vladimir Ondejcik[4]
[1,2,3] White arkitekter AB / Chalmers University of Technology [4] White arkitekter AB
[1] frans.magnusson@white.se [2,3,4] {jonas.runberger|malgorzata.zboinska|vladimir.ondejcik}@white.es

In this paper we address the problem of knowledge management in architectural design computation practice, reflecting on our practice at Dsearch - a design computation network within White arkitekter. As a means to investigate relevant aspects of visual scripting, we introduce the notions of code, algorithm and note. We also introduce two different modes of operation within architectural practice: morphology and development - which help us distinguish the diverse knowledge types typically occurring in the structure of visual scripts. We describe two sets of tools developed by Dsearch to continuously integrate planning and documentation with design development work. The main conclusion from our practical experience of this approach is that it allows critical reflection into an efficient workflow. This constitutes a new kind of practice based and action oriented knowledge that can be curated in the form of design narratives.

Keywords: *design computation, architectural practice, knowledge management, visual scripting, Grasshopper*

INTRODUCTION

This paper presents a vocabulary and a set of conventions supporting knowledge management in architectural design computation practice. The conventions comprise a set of tools and routines for Rhinoceros (RH) & Grasshopper (GH), a common 3d-modelling and visual scripting environment. These are developed by Dsearch, a network of architects specialised in design computation within White arkitekter - a large architectural practice. An earlier version of this work, in particular the Dsearch Graphic Standard, have been used external to White during the Textile Hybrids workshop at Hafen City University to facilitate collaborative development between architecture and engineering students (Lienhardt and Runberger 2017).

Grasshopper, affords the designer a high degree of freedom to organise the visual aspects of the script to be more legible, without changing the underlying graph - the specific logic that executes the script. GH also allows the graph to rearrange itself. This can for instance be utilised in a powerful way by using genetic programming (Harding 2016) to evolve new versions of scripts. Dsearch has instead chosen a nim-

ble approach, deploying small, and in some cases disposable, auxiliary objects to rearrange or gather information from the script. This paper states the case for considering the layout of a visual script as a design task in its own right - vital for knowledge management. Examples of how this kind of design thinking relates to learning and communication are drawn from the practice of Dsearch.

Such knowledge often remains the internal concern of an organisation, but the purpose of this dissemination to a wider audience is to continue an established development within the discipline of architecture, relating computation to design thinking and management (Derix 2009; Hudson 2010; J. Runberger 2012; Davis 2013; J. Runberger and Magnusson 2015; Magnusson and Runberger 2017).

METHOD

The research behind this paper has been carried out through design and through practice. The authors are themselves members of Dsearch and active in the development work being described and discussed here. To make the voice of the paper clear, the arguments and experience of the authors as researchers are consistently expressed using the pronoun 'we'. When discussing work or established conventions at White, we refer to Dsearch and its network.

Dsearch is positioned to address questions and identify potentials in ongoing projects within White, and address them through design of new design workflows or organisational changes. The authors, being part of this network and its larger context, have collected and analysed feedback from these interventions; through direct experience as practitioners complemented by interviews of individual peers and focus groups. Expressions such as "in our experience" or "we find that" are used throughout the text; such statements should be read as originating in several years of prototyping development, facing the full complexity of architectural practice (Schrage 1999; Capjon 2004).

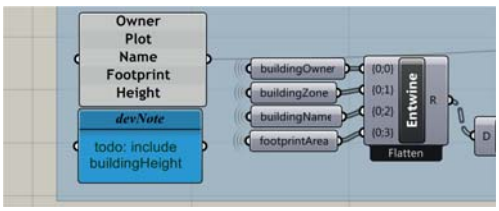
We define this practice based approach to research as a designerly form of insider action research (Magnusson and Runberger 2017; Brannick and Coghlan 2007); seeing practice oriented and academic knowledge as a continuous spectrum - rather than different in kind. That approach aligns with the definition of *design research* as placing "...specific focus on the creation of new insights and knowledge through the actual design work or professional practice" (Hensel and Nilsson 2016, pXVI) or *science of design*, in which design strategies from everyday practice are carefully studied and used as valid sources to generate new knowledge and understanding of that practice (Cross 2006). It is also placed within the field of research by design. This field has on the one hand identified digital technology as a common theme in many practises (Hensel and Nilsson 2016) , but also pointed to the importance of the projective aspects of research in architectural practice, that go beyond internal instrumentalism, addressing additional questions (Leatherbarrow 2012). This parallels a contemporary ambition to reform architectural practice as a project of knowledge production, separate from or overarching individual building project needs. One example here could be UN Studio where computational techniques are seen as one of several important platforms (Berkel and Bos 2016).

VOCABULARY

Three ubiquitous concepts are appropriated and used with specific definitions within this paper: *code* is defined as 'instructional notation, determining computation'; *algorithm* as 'descriptive notation, representing code agency'; and *note* as 'operative notation, facilitating project development and use'. These concepts are not mutually exclusive properties of an object, but should rather be seen as different aspects of a script. They are introduced here as part of a terminology aiming to establish an integrated understanding of computation within design practice.

The notion that one single object can express itself in several ways can be exemplified in GH by describing a component - one node of the data-flow graph, the code layer underpinning the visual , algorithmic, layer in GH. As code, each component de-

termines one stage of the data-flow, one node in a graph, turning input data into output. As algorithm, it notates this process with name, icon, and a mouseover text description. There is also more indirect information arising as a result of the process: hints of data types and structures at input and output parameters. Together with the algorithmic aspects, such information allows the designer to construct a mental representation of not only the computation performed at that specific stage, but its significance in relation to the overall design project. Grasshopper also offers components that can display visual or textual information without affecting the data-flow. This is analogous to comments in text-based programming, but the visual aspects allow for a more intricate relationship between code, algorithm and note. In Figure 1 the textual algorithmic information provided by the GH component is for instance complemented by the inclusion into a colour-coded group and the placement next to a text panel with notes relating to project development aspects.

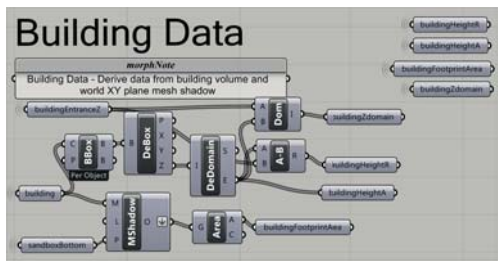


The authors have also found it useful to distinguish between design and development processes on the one hand, and the computational and morphological processes performed by a script on the other. While the former develop sequentially over time, the latter execute momentarily according to a predefined logic. The deterministic morphological process is thus not to be confused with the open ended design process; the morphological process is rather subjected to modification through the iterations of a design process.

Dsearch has developed sets of tools and conventions for both modes of operation, supporting designers with the planning and documentation of script development. The **Morphology Set** deals with various ways to create and collate algorithmic information, aiming to represent the data-flow processing as a process generating architectural form. The **Development Set** conversely supports quality assurance in planning and documenting project issues, beyond form generation, as well as the capture of ideas and issues of relevance beyond the specific project. These modes are strongly intertwined with each other and with the underlying code; for instance when morphological aspects are linked to design decisions taken at a project meeting.

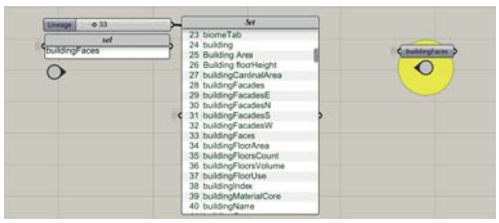
MORPHOLOGY

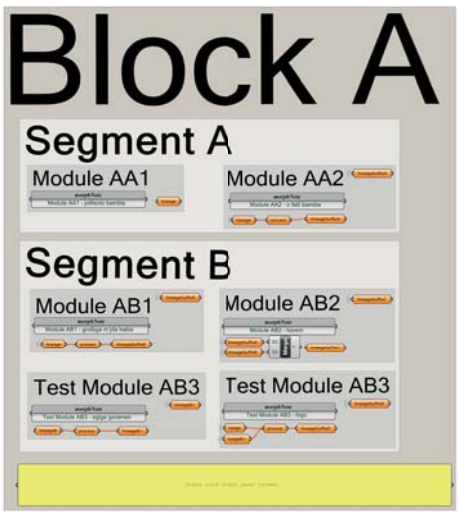
In our experience, stringent naming of parameters facilitates mental representation by linking algorithmic description to the code. Following Woodbury's credo of *clear names* (Woodbury and Gün 2010), Dsearch has put forward the convention of *Lineages* in order to represent parameters as they are processed throughout the data flow (Figure 2). This is done by relating the name of 'parent' parameters to their downstream 'offspring'. If a module output parameter is considered the downstream offspring of an input, the input name is suffixed, captioning what difference the intermediate process brought to the Lineage.



The **Lineage Set** object lists all named parameter components and moves the grasshopper window to their most upstream instance (Figure 3). This provides a descriptive overview as well as access for

deeper investigation or maintenance. The **Renamer** object facilitates reorganisation of lineages by batch-finding and replacing parameter names. Lineages paint a rich picture of how parameters are processed in the script. Some parameters have been observed to form long lineages, using other lineages for their process without losing identity; a phenomenon correlated with significance within the design process.

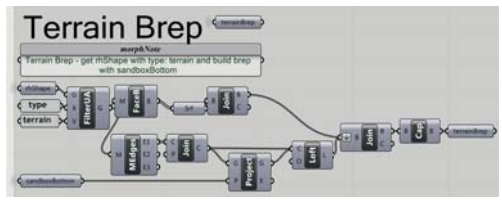




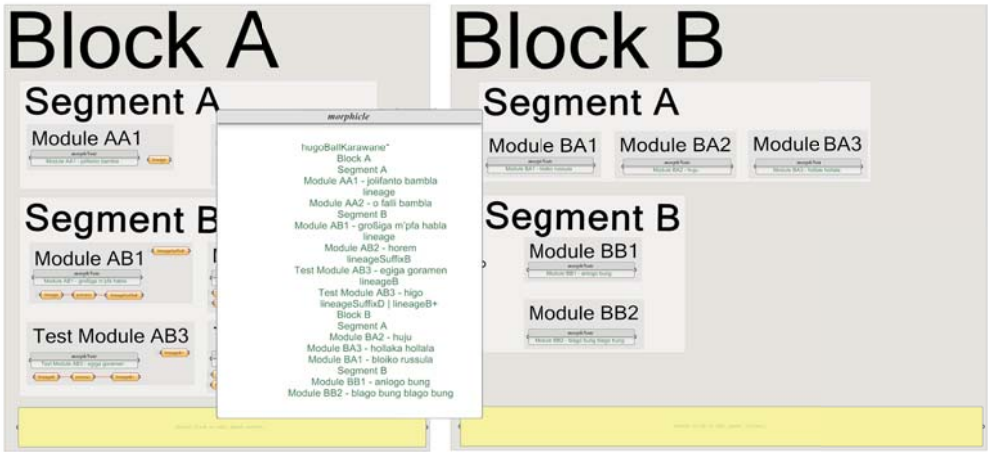
Davis, Burry & Burry agree with Woodbury on the importance of clear parameter names and add that also modularisation is critical for "...the legibility of the script, and therefore the ease with which the script can be shared, reused and modified" (2011, p374). If naming conceptualises the relation of parameters to the overall morphological process, it is also helpful to visually and conceptually distinguish parts of the script - establishing sub-processes with explicit input and output parameters. Dsearch proposes to use the GH functionality to group components and then nest these groups within larger groups - in Dsearch terminology: components within *Modules* within *Segments* within *Blocks* (Figure 4).

This enhances legibility and comprehension in that it structures the definition into levels of scale. The visual scale cues should also aid the navigation of an *abstraction gradient*, ranging from comprehensive understanding of script agency to in-depth technical detail of the code (Zboinska 2015). The script is modularised on the code level by grouping components so as to form a coherent process. By adding specific input and output parameter components the interior of a module can be reprogrammed without losing input and output connections.

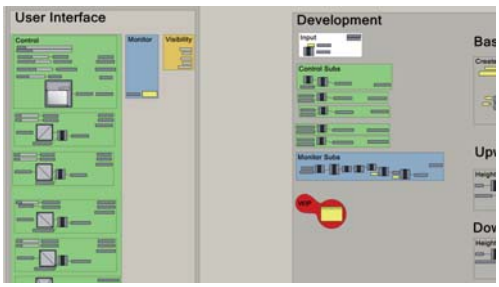
Algorithmic aspects include colour coding of the group and annotation that is legible in various scales. A concise heading is written in a big font for large scale overview. Close up, input and output parameters are arranged so that their linages can be browsed in conjunction with a comprehensive textual description of the module, entered into the **Morphology Note** (Figure 5). This arrangement is complemented with small panels of annotation highlighting vital and/or unintuitive parts of the process.



The **Morphicle**, a portmanteau of the words morphology and chronicle, automatically creates a textual and visual summary of the script, arranged according to its modularisation (Figure 6). All Morphology Notes and headings are grouped with input and output Lineage names to form a description of each



Module, and its relation to Segments and Blocks. A visual sequence is created from selected Lineage geometry and data, and displayed via the Human UI interface.

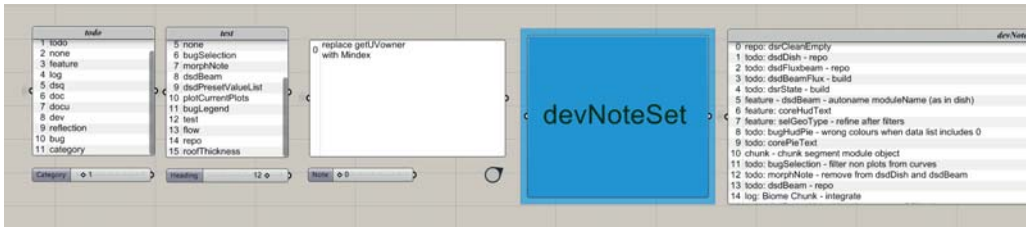


This object is inspired by the *literate programming* approach of Knuth, urging programmers to prioritise the human reader over the computer. Knuth developed a language and a set of programs he called the WEB system where any program serves as a source language for two processes. "One line of processing is called *weaving* the web; it produces a document that describes the program clearly and that facilitates program maintenance. The other line of processing is called *tangling* the web; it produces a machine-executable program" (Knuth 1992, p101). Grasshopper, being a visual data flow based environment, is continuously 'tangling' the code while a script is being developed. What the Morphicle object adds is a script that 'weaves' various textual descriptions together, mediating a visual layout into a kind of literature.

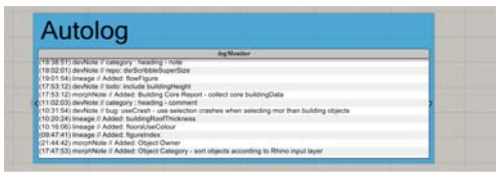
The Dsearch *Graphic Standard* stipulates a colour for the group, according to functionality (Figure 7). Examples include: referencing and baking Rhino geometry, manual control or input of parameters, executing advanced script modules, etc. One primary use of this colouring is to highlight the *User Interface* part of the script where parameter controls are located for use by non-specialist collaborators. This is then distinguished from the back-end *Development* section where the actual code is found (Figure 8).

Development

The note aspect is supported by the **Development Note** - a pre-configured panel component for noting down issues relating to script and project development (Figure 9). The designer enters text in the syntax of "category: heading - note". Categories include: log, todo, feature, bug, etc. Headings can ei-



ther reference parts of the script such as Lineages or modules, or they can come from the project in the form of design concepts, evaluation criteria or contractual agreements. In this way project discussions can be coupled with specific aspects of the morphology. Conversely, project issues can be raised by the designer with minimal disruption of the programming workflow. Tangentially related ideas and concerns inevitably crop up, triggered by thinking on the problem at hand. While important and in need to be noted down, they threaten to disrupt the designers train of thought. Our experience shows that the possibility to record a thought in its relevant context minimises the time needed to describe relevant background in order to remember, or explain it to a team member - thus minimising the disruption.



The **Development Note Set**, collects all notes from one GH script and sorts them by category and heading. This allows the designer to see for instance a list of all tasks on the todo list, or all notes regarding a specific bug. Project management, such as time planning and task allocation, is also here supported by rich, context-specific information by zooming the GH window to a specific note and review it in the context of the script. One way to tick off an item on the todo list during a review would then be to simply delete the note.

The **Autolog** object continuously records project development events along with a time stamp (Figure 10). This entails changes and additions to the Lineage Set, Morphology and Development Notes as well as quality assurance data, such as filenames and plugin versions. The designer can enter information by using the log category in a Development Note. Events such as key development steps of the definition functionality and the iterative exports of definitions for the use by non-specialists are recorded in this way.

A potential development for the Developer Notes and the Autolog is to integrate them with other project communication planning and documentation tools, such as mail and meeting minutes, or a task management system, such as Trello. The key consideration here is how to establish a filter for curation of this data, so that it forms a relevant description of the project history - beyond log data. Such a **Chronicle** would complement the **Morphicle**. We believe that such cross referencing of continuously collected information, can result in new insights regarding project specific issues, methodological and organisational strategies and wider disciplinary questions. Runberger proposes to format this kind of knowledge as *design narratives*; rich histories of project decision-making, cross referenced with morphological motivations (2012, p82-83).

DISCUSSION

On a general practice level we can state that these conventions facilitate breaking down the dichotomy between management (as in planning, task allocation and documentation) and actual design work in projects. We argue that by integrating management into the development workflow, conditions for designer agency and increased efficiency are established. This ethos is in line with contemporary software development culture, for instance expressed by the agile programming movement: "The best architectures, requirements, and designs emerge from self-organizing teams." For the presented conventions, Dsearch makes heavy use of developments distributed throughout the Grasshopper community - especially the Metahopper and Human UI plugins developed by Heumann, Syp and Holland. What is offered back to the community is a long experience of applying and contextualising this development in a way that provides quality assurance and knowledge management within design computation practice.

The time pressures of practice risk leading to cultures where reflection is delegated to the lowly prioritised task of documentation - separated from the tasks and deliveries at hand. We would like to stress the importance of integrating this reflection on practice with the daily design work in projects. Only the practising architect that has access to project specific knowledge, can communicate this first-hand knowledge from a critical perspective - to a wider audience. This knowledge brokering is vital for individual organisational learning, as well as the development of the architectural community of practice (Wenger 1999). We recognise the necessity to carry out this kind of knowledge production as intuitively as possible, so as not to disrupt daily practice. The design narratives, sketched out above, can hopefully strike a realistic balance between learning and production - layering quality assurance driven documentation with critical reflection.

For the wider academic audience the relevance of this research lies in the conceptual framing of the content. We see a need for a vocabulary concerning visual scripting in architectural practice, beyond software specific nomenclature. Though computer science can provide technical definitions for a directed acyclic graph based data flow programming language such as Grasshopper, much of that terminology is out of reach for the design disciplines. For architectural practice, terms must be defined that relate visual programming to issues of design development, project management and strategic knowledge management. This paper presents Dsearch terminology that to a varying degree has stood the test of time. More importantly the underlying distinctions and conceptualisations have already proved to be valuable in the internal discourse at White.

CONCLUSION

The graph, defining a computational process, is a genuinely new kind of representation for architecture. Forcing the designer to express design decisions explicitly, the representation of form is no longer constrained by a final static geometric description, or tied to a historical narrative of its design process. This notion of morphological process as distinct from chronological design process, and the corresponding graph based representation, are regarded as novel additions to design methodology with great potential for further theory development.

Zboinska proposes to annotate visual scripts with a supplementary algorithm in order to boost comprehension, knowledge sharing and collaboration (2015). This is done by the use of standard GH components placed in parallel to the script. The main argument of this paper is that it is possible to expand on this idea, by relying even further on the visual aspects of Grasshopper. Dsearch allows the algorithm to be scattered and partial during development, with its constituents carefully placed in the context of the script. Only post factum is it then curated into a coherent form.

The use of notes to relate issues regarding programming, design, project management and strategic development to the script context, is a minor feature with major implications. This cross referencing

between morphology and development concerns, creates opportunities for new insights. Continuously collected along with the algorithm, this information can be synthesised as a design narrative - a new kind of practice based and action oriented architectural knowledge.

REFERENCES

van Berkel, B and Bos, C 2016, *Knowledge Matters*, Frame Publishers

Brannick, T and Coghlan, D 2007, 'In Defense of Being "Native": The Case for Insider Academic Research', *Organizational Research Methods*, 10(1), pp. 59-74

Capjon, J 2004, *Trial-and-error-based innovation: Catalysing shared engagement in design conceptualisation*, Ph.D. Thesis, AHO

Cross, N 2006, *Designerly ways of knowing*, Springer

Davis, D 2013, *Modelled on software engineering: Flexible parametric models in the practice of architecture*, Ph.D. Thesis, RMIT University

Davis, D, Burry, J and Burry, M 2011, 'Understanding visual scripts: Improving collaboration through modular programming', *International Journal of Architectural Computing*, 9(4), pp. 361-375

Derix, C 2009, 'In-between architecture computation', *International journal of architectural computing*, 7(4), pp. 565-586

Harding, J 2016 'Evolving Parametric Models using Genetic Programming with Artificial Selection', *Proceedings of the 34th eCAADe Conference - Volume 1, University of Oulu, Oulu*

Hensel, MU and Nilsson, F 2016, *The Changing Shape of Practice: Integrating Research and Design in Architecture*, Routledge

Hudson, R 2010, *Strategies for parametric design in architecture: an application of practice led research*, Ph.D. Thesis, University of Bath

Knuth, DE 1992, *Literate Programming*, Cambridge University Press

Leatherbarrow, D 2012, 'The project of design research', in Hensel, MU (eds) 2012, *Design Innovation for the Built Environment: Research by Design and the Renovation of Practice*, Routledge

Lienhardt, J and Runberger, J 2017 'Collaborative Models for Design Computation and Form Finding: New Workflows in Versioning Design Processes', *Design Modelling Symposium 2017*, Paris, p. Pending

Magnusson, F and Runberger, J 2017 'Design System Assemblages: the continuous curation of Design Computation Processes in Architectural Practice', *Proceedings of Professional Practices in the Built Environment*, Reading, pp. 194-204

Runberger, J 2012, *Architectural prototypes II: Reformations, speculations and strategies in the digital design field*, Ph.D. Thesis, KTH Royal Institute of Technology

Runberger, J and Magnusson, F 2015 'Harnessing the Informal Processes Around the Computational Design Model', *Modelling Behaviour: Design Modelling Symposium 2015*, Copenhagen, pp. 329-339

Schrage, M 1999, *Serious Play: How the World's Best Companies Simulate to Innovate*, Harvard Business Review Press

Wenger, E 1999, *Communities of practice: Learning, meaning, and identity*, Cambridge university press

Woodbury, R and Gün, O 2010, *Elements of Parametric Design*, ROUTLEDGE CHAPMAN \& HALL

Zboinska, MA 2015 'Boosting the efficiency of architectural visual scripts: A visual script structuring strategy based on a set of paradigms from programming and software engineering', *Modelling Behaviour: Design Modelling Symposium 2015*, Copenhagen, pp. 479-490

[1] http://www.grasshopper3d.com/group/human-ui

[2] https://trello.com

[3] https://www.grasshopper3d.com/group/metahopper

[4] http://www.agilemanifesto.org