ABSTRACT
This paper provides an overview of computational and strategic approaches to three different contexts of collaboration in NBBJ’s architectural practice. First, in collaboration between designers, we demonstrate how custom computational workflow tools support training, best practices, and knowledge sharing, allowing us to share and have multiple designers constructing, utilizing, and modifying parametric models and tools in parallel. Second, we describe how computation facilitates collaboration with our clients, becoming an invaluable means to communicate and engage client agency in decision-making processes, through live design tools and direct data exchange. Finally, for the design of space itself, we integrate recent research in behavioral, spatial, and social science into analytical tools that allow us to maximize the potential for workplace design to encourage collaboration between occupants — the end users of architecture.
INTRODUCTION

Effective collaboration is of critical concern to any design practice. Streamlined collaboration must happen between designers, between designers and their clients, and between the end users of the spaces we design. To work effectively as designers in an architectural process, we must collaborate with each other, but increasing adoption of parametric design tools can actually impede rather than enhance that collaboration. Moreover, these tools go beyond our internal means of design conception to impact our business practice, changing the way we interact with clients and owners to engage their agency in critical decision-making. Finally, to achieve the architectural product we intend – spaces that serve and support our clients’ enterprise – we must ensure that our designs facilitate collaboration and human interaction, by applying analysis tools derived from scientific and social research to analyze designs as they evolve.

This paper seeks to illustrate a range of practical, real-world approaches to the problem of these three kinds of collaboration, through research, strategy, and above all, the deployment of custom computational tools and solutions.

COLLABORATION BETWEEN DESIGNERS

Digital tools have already radically changed how designers work together. With BIM adoption rates soaring over the last decade, most architects are now used to employing central models to collaboratively manage and modify designs in parallel, though certainly not without headache. While BIM is “parametric” in some sense, its collaboration tools are focused on the shared direct editing and documentation of geometry, rather than on a collaborative and iterative development of advanced associative and algorithmic logics.

At NBBJ, we are increasingly reliant on algorithmic tools and models for design development. As these models grow in importance and complexity, we are coming to face significant challenges. In his PhD thesis, Daniel Davis cites a report from the Aberdeen Group that identifies the “top four challenges to design reuse” – the recycling and sharing of digital design models:

1. Model modification requires expert CAD knowledge.
2. Models are inflexible and fail after changes.
3. Users cannot find models to reuse.
4. Only the original designer can change models successfully.
(Aberdeen Group 2009, cited in Davis 2013)

These four challenges present significant obstacles to effective collaboration. Through training, file management tools, the establishment of best practices, and platforms for knowledge sharing, we are attempting to confront them head on.

TRAINING

One obvious barrier to the integration of parametric tools into the design process is the wide range of exposure and skill level present on project teams. We have identified four general categories of experience with parametric tools, and employed a variety of training strategies to increase our ability to take advantage of algorithmic workflows (Figure 1). The first level, the “thinkers,” may not actually write code or build tools themselves, but understand the underlying concepts behind algorithmic workflows for propose tools to be built, or identify problems that may benefit from a computational approach. The second level, the “users,” have a base familiarity with the platforms employed (Rhino or Revit) and their associated scripting environments (Grasshopper or Dynamo) to successfully deploy and use tools constructed by others. The third tier, “hackers,” can modify and adapt tools built by others to new applications, and construct some simple workflows from scratch on their own. The final level, “experts,” are capable of constructing tools “from scratch,” and play an integral role in advancing our computational practice.

In order to build up each of these classes of involvement, we employ two primary approaches to training. The first is project-based, which we call an “ignition.” A week-long training session brings a whole project team together around a discussion of the capabilities of design computation, with a smaller group staying on to complete focused training and build up capabilities. The goal is to ensure that all on the team are at least “thinkers,” and that some
become “users” and “hackers.” The final days of an ignition involve an expert-led development effort to build a base of project-specific tools that can be used and modified by the project team to suit their needs. The advantage of tying training to projects is that students are forced to employ the training immediately to solve real problems for the project, which greatly enhances retention and future use.

To balance this just-in-time, focused approach, we also benefit from regular firm-wide and office-wide ad hoc user interest groups. A typical one of these, called “GHUG” for “Grasshopper User Group,” (though topics extend well beyond Grasshopper as a platform) meets twice a month, alternating between a “show and tell” format and hands-on, instructor-led training (Figure 2).

As Coren Sharples of SHoP Architects has noted, project teams can “split into technology staff — typically entry level, with a high degree of software acumen but very little of any practical design experience, and design staff, who have practical experience in the design and detailing of buildings but are probably not as skilled with the latest software” (Sharples 2010). To avoid this split between skilled designers’ lacking computational knowledge and computational experts’ lacking design acumen, NBBJ’s approach to computation has always been one of studio integration: specialists integrated into studios and design teams rather than treated as a separate, internal consulting group. The effort put into firmwide training has paid dividends in enabling productive partnerships between experienced designers and more junior designers with computational expertise (Figure 3). Our best computational tools have emerged from this kind of partnership.

FILE MANAGEMENT AND BEST PRACTICES

As we amass a library of reusable computational tools and project-specific models, we encounter the problem that everyone builds their models differently, and that visual scripting platforms do not have well-established standards for organizing and structuring models. Moreover, with the panoply of available add-ons and plug-ins, and range of software versions in use, each coder’s
design environment is slightly different; in practice it is rare to simply be able to open a Grasshopper definition from another user without dependency or version mismatch errors arising.

In an effort to address some of these issues, we have developed both best practices and technical tools to facilitate consistency across computational tool development. NBBJ utilizes a Grasshopper new file template designed to present useful information to a designer that encourages clear, well-structured definitions (Figure 4). Embedded in the template are text panels, where the designer can describe the purpose and general use of the tool. An embedded script tracks authorship, date of last modification, file location, and all external plug-in dependencies. Another script in the template will auto-color groups according to a prefix in the group name, allowing for an easy color coding by inputs, outputs, actions, and warnings, minimizing user effort to remain organized.

In order to address the inconsistency of individual users’ development environments, we are deploying a custom plug-in for Grasshopper which can automatically install libraries and package functions onto users’ computers (Figure 5). This way, when a broadly useful tool or script is developed, it can quickly and automatically be made available to all other users in the firm. Moreover, when new versions of plug-ins are released, all users can be simultaneously upgraded, assuring consistency so that definitions can be easily shared between users without worry of missing dependencies or mismatched versions.
KNOWLEDGE SHARING

Over the last year, we have focused significant energy toward increasing knowledge sharing around design computation. As a firm with eight offices worldwide, it is a challenge to make sure that a technical advance in one office is quickly made available to the others. A new internal web portal provides a user-friendly space to document experiments, projects, work-in-progress, and tools (Figure 6). Tied in to the site is a custom metadata parser for Grasshopper that scrapes our database of computational tools and automatically documents them, with authorship information, a list of external dependencies, visual thumbnails, and description text from within the definition itself, provided it is structured with the appropriate template.

FUTURE WORK

While strategies for standardizing the documentation and sharing of computational tools through templates, synchronization, and metadata harvesting all mark improvements in the way our practice manages parametric collaboration, we are aware that there remains much to be done in this arena. There is no substitute for a true source control solution, akin to Subversion, Git, or Mercurial, as employed by software engineers. Shireen et al sketched some of the critical requirements of such a system, notably the need to be able to “fork” parametric models into multiple directions, edit them in parallel, and “merge” them back together, but to our knowledge no system truly supporting this in a visual scripting environment has yet been developed (Shireen et al 2012). We fully concur with Daniel Davis’s assertion that software engineering is “an important precedent for architects using parametric models” with critical implications for how these models are used in practice (Davis 2013).

COLLABORATION BETWEEN DESIGNERS AND CLIENTS

While effective cooperation between designers remains a critical component of the architectural process, designers themselves are far from the only agents involved in conceiving and realizing architecture. Consultants, contractors, specialists, and clients all exercise agency on the development of design, and the structure of a project process must facilitate communication and collective decision making. There is already much written about digital tools as a means to enhance exchange of information with consultants and contractors (Ashcroft 2010); this section will focus on the way digital and algorithmic tools can be employed to transform the relationship between designers and clients.
The nature of the designer-client relationship can be the pivotal factor in the success or failure of a design project. When it is at its best, clients feel that their expectations are being fulfilled, their goals and desires realized, their concerns heard, and their risks mitigated. Designers feel informed, empowered to advance the design, and safe to challenge assumptions. However, these relationships may devolve into antagonistic tension or conflict, with both clients and designers feeling unheard or unable to arrive at satisfactory outcomes.

On recent projects, we have sought to use digital tools as a means to address (or at least reduce) the challenges associated with our client relationships. We have found parametric tools in particular to be an effective way to both communicate ideas and involve clients in decision-making processes.

THE SPRING DISTRICT MASTER PLAN

For the design and development of the Spring District, a 10-year master plan in Bellevue, Washington, NBBJ developed a parametric model in Grasshopper to manage the complexity of the twenty-nine-parcel, multi-phase project. Master plan design variations could be quickly evaluated against floor area requirements, setback information, parking counts, use group ratios, and height limits.

As the design of the master plan solidified, and the types of decisions being made changed, the model evolved accordingly. The planning process shifted from the general form, disposition, and program arrangements on site to more fine-grained design decisions, such as the location of key amenities and tenants intended to attract people to the district. The tool was extended to study these questions as well, enabling on-the-fly evaluation of retail visibility, live-scale comparisons to precedent projects, and anchor program adjacencies (Figure 7).

This model proved critical not only as a design tool, but also as a means to involve the client, developer Wright Runstad, directly in the design process. Live parametric geometry could be adjusted and modified in project meetings with the client’s input, dynamically updating both graphical and numerical (spreadsheet) output. “What-if” scenarios could be quickly validated against a host of complex requirements and restrictions, ensuring that time was not wasted pursuing options that were unfeasible or undesirable. The client’s involvement in adjusting, assessing, and validating design options represented a radical tightening of the typical feedback loop of a designer-client process.

A traditional client meeting may involve the presentation of several design alternatives, to which the client responds and gives feedback. Designers take that feedback, evolve the design accordingly, develop presentation graphics, and repeat the process. Eventually a satisfactory design is arrived at, and the client may take the documentation of the design for communication with other stakeholders in the process.

With a live model in the meeting, this process is streamlined considerably. The design model can be updated to respond to client feedback on the fly, and immediately validated against the set of restrictions, requirements, and aims of the project, which are algorithmically codified in the project model itself. Presentation-quality graphics and formatted Excel spreadsheets of associated metrics are updated live, so that the client can take the design documentation away from the meeting reflecting revisions made in the same meeting. The speed, extensibility, and fluidity of Rhino and Grasshopper as a platform make this kind of live adjustment, analysis, and documentation possible.

2ND AND PIKE

The design of commercial office buildings is always a balancing act. Developer clients are highly sensitive to project metrics: rentable areas, layout efficiency, and project cost must be closely monitored to ensure a profitable investment. At the same time, they want the properties they develop to be unique and attractive to tenants. Developers typically rely on a pro forma, a numerical cash flow projection, which ties together all costs associated with the project against projected income (Schmidt 2013). Standard constructions of a pro forma evaluate costs in great detail, but importantly, only reflect the value of design itself in the assumed rental rates the developer will charge to tenants. Design quality and building features are assessed against market rents to establish a realistic number. As design evolves, the pro forma remains a central tool for a developer to assess viability and evaluate the design. However, this presents a challenge: metrics like project cost translate directly into the pro forma, while the inherent subjectivity of design value resists concurrent quantification.

In order to address this dilemma, NBBJ sought to rethink the way early-stage design development is communicated to developer clients. NBBJ selected “2nd and Pike,” a mid-rise developer office building project in downtown Seattle, to test these ideas. Urban Visions, the developer, provided us with a copy of their pro forma for the project. We constructed a parametric building model in Grasshopper, and linked it directly to the pro forma spreadsheet, outputting building square footage and other relevant metrics like curtain wall area and estimated structural costs. As design param-
eters changed, they could be evaluated, not only in terms of cost, but also in terms of the ultimate rent per square foot that the client would need to charge. This tool was utilized from the earliest stages of design, when even the overall program area and number of floors was still in flux, and proved useful as a way to “right-size” programs (Figure 8).

To address the question of value, other algorithmic tools were tied into the model. A landmark view analysis tool in particular proved highly impactful to the design outcome, due to particular site constraints and conditions. One of the major potential values of the project site was its view potential, due to its proximity to Puget Sound. Despite the site’s generous height limit, its limited footprint meant that even built out to maximum FAR, only the building’s top floors had access to the prime views. The design team opted to study ways to extend the building’s height in order to take greater advantage of the view potential. Many ideas, including double-height spaces, base-level residential, winter gardens, and staggered half-floors were tested as strategies to achieve this. A live heat map color-coded access to particular views, including Puget Sound, the Space Needle, and Lake Union to the north. With this visual feedback, the building mass could be adjusted to take maximum advantage of view opportunities. Other live analysis overlays included another heat map coded by hours of direct sun hitting the building, a feature that informed the location of interior “winter gardens” as a solar buffer.

By pairing live analysis of environmental and human factors with the pro forma link, our design process brought value and cost together, enabling the design to push the envelope of a typical office building without extending beyond the realm of feasibility. Moreover, the use of the tool to communicate with the client was critical in building a relationship of trust, which empowered the design team to push beyond the typical.

COLLABORATION BETWEEN END USERS

So far, the focus of this paper has been the way technology itself can facilitate designers working together and with their clients. This development is of course in no way limited to the architectural profession; work of all kinds is subject to radical transformation by technological means. Even the most mundane communications tools like Microsoft Outlook can alter and enhance the way workers of all kinds share information and build professional relationships. This is all the more evident in the working environments of our tech clients, who rely heavily on cloud-based collaboration tools like Google Drive and even fully-fledged source control solutions like Git to share in the collective development of products or delivery of services.
As architectural designers concerned first and foremost with the production of the built environment, we are not content to let the problem of productive collaborative partnerships be solved through digital means alone. NBBJ has long understood that physical space can enable new ways of working, improve employee happiness and productivity, and enhance organizational performance. Designing these environments has been an integral component of our practice, and clients now see them as a means to remain significant in a competitive world. While historically our belief was rooted in intuition born of our collective experience, recently we have begun to understand how well it is supported by research in neuroscience and social science. We are currently collaborating with researchers to transform primary research into applied outcomes: John Medina, Professor of Bioengineering at University of Washington is our 2014 NBBJ Fellow. Moreover, we have begun to take advantage of algorithmic tools informed by this research as a means to explore and validate our designs, to ensure that the physical workplaces we design facilitate end user productivity and collaboration.

VISIBILITY AND VIEW

The visual experience of space has a clear impact on how teams work together. The ability to survey an entire floor has obvious benefits: a sense of connectedness, an understanding of where team members are located, and the ability to observe what they may be working on. From the perspective of workplace interaction, separation by more than 100 feet is functionally equivalent to being in different buildings, if not in different geographical locations (Heerwagen 2004). However, a maximally open floor plan may be neither practical nor desirable; zones of privacy and seclusion are critical to enable ad hoc work sessions and focused collaborative sessions.

Our thinking on both the analysis and impact of visual access in the workplace has been strongly influenced by the work of Kerstin Sailer of The Bartlett on the “Generative Office Building” (Sailer 2012). She studies the effect of visual connectivity and spatial integration on the propensity for a space to allow “new relationships, new ideas, new products and new knowledge to emerge” (Ibid). Her methodology of analysis – the space syntax concept of a visibility graph analysis – is something we have implemented in Grasshopper and applied regularly to our workplace designs (Figure 9). Our implementation is multi-threaded to function quickly at high resolution, providing actionable visual feedback tied into Sailer’s own recommendations: encouraging high levels of global integration and local visibility, providing a moderate distribution of integration levels, locating building circulation routes to run past workstations, and concentrating attractors in highly integrated areas.
Interior views within the workplace are critical, but so too are views out. Developers have long used this as a means to sell a project and achieve larger returns. Our studies, in collaboration with Peter Kahn of the University of Washington, have found that while what one sees is important, view diversity is a greater contributor to productivity and brain stimulation. Non-rhythmic sensory stimuli can improve attentiveness and interest, while minimizing feelings of fear, anger, aggression and sadness (Ulrich et al. 1991). All contribute to a greater desire to interact and engage colleagues. When that view is of natural subjects, its positive effect is compounded. We have developed computational view diversity analysis tools, enabling us to organize building siting and orientation to maximize view diversity from interior workstations.

CONNECTIVITY AND SERENDIPITY

The layout of circulation networks is typically evaluated from the standpoint of efficiency. Generally, an effective circulation network is regarded as one that does not take up undue space, and facilitates getting from point A to point B in as little time as possible.

Our perspective is that circulation paths can be designed with intersections in mind. While we can’t force interaction, we can encourage it by overlapping circulation routes – between teams, programs, floors, campuses – and then augment these relationships through various attractors or amenities. To this end, we utilize graph analysis algorithms to locate the most likely routes through a circulation network, weighted by circulation type and network centrality. Agent-based simulation can provide further detail, tracking the movement of simulated office workers over the course of a day, between focused working tasks, team meetings, and other scheduled and unscheduled events like lunch, coffee, and bathroom breaks. This additional information allows us to test how layout configuration changes impact the number of crossings. Observational studies identify the critical importance of these crossings in movement patterns to raise the perception of availability of others for informal “recruitment” (Backhouse et al 1992) into conversations which may in turn lead to productive relationships (Heerwagen 1999). These studies show that as many as 80 per cent of interactions in a creative workplace are unplanned (Heerwagen 2004). The design of circulation networks and workplace visibility can have a major impact on these kinds of unplanned interactions.

Finally, in the large-scale planning of circulation networks, we have employed self-organizing structures to establish minimal path networks. We rely on force-directed edge bundling (Holten et al 2009) – the so-called “wooly thread” algorithm – to produce networks that automatically consolidate similar paths to simplify the straightforward network of direct linear routes between all points (Figure 10). These consolidated networks have the benefit of getting people from place to place in a non-optimal but reasonably efficient manner, while encouraging the kinds of face-to-face encounters that build organizational cohesion.

CONCLUSION

Computation has proved its value to NBBJ’s practice well beyond the production of novel forms or as a rapid iteration tool. As a means for designers to work together, studying design at multiple scales at once, it allows us to more nimbly develop design alternatives and keep ideas moving forward. However,
this is only possible with the right technical frameworks, tools and practices in place, so that parametric models do not devolve into “walled gardens” of individual endeavor. As a way to communicate with clients, it drastically increases the speed of design feedback, and improves relationships by putting a nimble process of design revision on the table, accessible to stakeholders. Finally, as a method by which to analyze and refine building designs themselves, it allows us to predict and improve the experience of building occupants, even encouraging interaction and collaborative work.

The examples given in this paper are indicative of a growing diffusion of computational thinking into our design practice and culture. The critical issues that emerge in our work – collaboration as a key among them – are increasingly tackled through strategically deployed technological and computational solutions.

REFERENCES

ANDREW HEUMANN leads NBBJ’s Design Computation team, overseeing strategy, development, and implementation of computational tools for diverse projects and applications. He has developed a suite of tools for NBBJ’s corporate and commercial practice, which aid in the management of project metrics, environmental and urban analysis, and façade design. Andrew is trained in both architecture and computer science, and has lectured and taught seminars at Cornell University, Yale University, California College of the Arts, and the University of Washington. His work has been published in Wallpaper* magazine, CLOG journal, and presented at conferences including SIMAUD, the AEC Technology Symposium, and Facades+. He has also been a member of the NBBJ’s “Digital Practice” Online knowledge sharing portal.

RYAN MULLENIX, Partner at NBBJ, has led the design of numerous award-winning projects, and is currently the lead designer for Google’s new Bay View Campus. His multidisciplinary collaborations with academia have helped to draw ideas from leading centers of energy conservation, advanced systems, and material science. In an evolving digital world, he is committed to exploring how technology enables greater creativity faster. Ryan has published articles in Tall Buildings Magazine, and Columbus CEO, and has been an invited critic at various design schools including the Knowlton School of Architecture and the Taubman College of Architecture and Urban Design.

IMAGE CREDITS
Figure 1. Heumann, Andrew (2014) Levels of Computational Expertise. Diagram.
Figure 2. Heumann, Andrew (2013) Photograph.
Figure 3. Holland, Nate (2014). Grasshopper Definition
Figure 4. Heumann, Andrew (2013). Grasshopper Template File.
Figure 5. Heumann, Andrew (2014). Diagram of custom Grasshopper plug-in.
Figure 6. NBBJ (2014) “NBBJ Digital Practice” Online knowledge sharing portal.
Figure 7. NBBJ (2013). Custom Grasshopper Script
Figure 8. NBBJ (2014). Custom Grasshopper Script
Figure 9. Heumann, Andrew (2014). Custom Grasshopper implementation of VGA algorithm.
Figure 10. Heumann, Andrew (2014). Custom Grasshopper implementation of force-directed edge bundling algorithm.