

chapter 1

Sketching for Conceptual Design

Introduction

Good design is easy to recognize but difficult to describe. When it is missing, you immediately notice even if you can't say precisely why. Design has consistently defied all attempts at logical rationalization since before the Enlightenment, though, not for lack of trying by generations of design theorists. We observe in the world examples of effective design every day, but the rules and standards that led to that design's realization are notoriously tricky to decode.

Armed with a rigorous knowledge of building science that has been informed by thousands of years of historical precedent, architects can engage in a uniquely iterative process of design and reflection that produces constructible, functional, and at the same time, inspiring, and delightful buildings. The good ones can do it on demand, skillfully dropping themselves into a flow state where sketches pour out in a flood of work. Once in that state, they can reflect-in-action with consideration for hundreds of individual requirements, including unstated but inferred ones and all their various opportunistic affordances. By accommodating both objective reality and ethereal poetics, architects can bring a rigor of thought to any set of inputs, framing and reframing the original problem as necessary and iterating collaboratively until a path forward has been reached.

Creative Rigor in Design

The design work of a professional architect is rigorous, in-depth, and (to outsiders, anyway) more than a little mysterious. Building construction projects are notoriously complex, with little opportunity to rehearse solutions before committing fully to them in construction. Once construction has begun, the cost of making changes to the design can be astronomical. Every detail must be considered before construction begins.

Architects are trained to start sketching from very little concrete information, perhaps only a brief list of requirements, a site, and some local building codes and ordinances. The project may have exciting opportunities that are evident or at least implied by the available information, but it may also include unrealistic expectations and hidden, self-contradictory dependencies. It might be challenging to make many formal propositions about the building at first, but architects are trained to dive right in regardless. In time, through a process of sketching, reflection, and reframing of the problem, they know how to figure it out. As professionals, architects are great at just "winging it," figuring out both the real issues (including unstated requirements that only later become evident) and their ideal solutions.

Some architects, particularly those who fancy themselves a bit closer to art than science, may actively resist any external attempt to rationalize their process. Design, to them, is a mystical art open only to those properly initiated. In truth, they may not really know how they do what they do, though few of them will readily admit this. But there's nothing to be afraid of if you are one of these kinds of designers. The magic of your work is not diminished because you can't rationalize it for others. Most designers, even the most rational of them, cannot describe how they do what they do or how they know what they know about the design. The intuitive leaps you make while working are not diminished because you can't entirely explain how you reached them. Your ability to make those leaps, correctly and effectively, is what defines you as a designer.

If they are honest with themselves, all designers have trouble describing how they reach the conclusions they reach while designing. They may talk about how individual decisions simply "feel" better than others, how one design direction is inherently better than another one because of where it leads the design next in its evolution. When pressed, they may be able to post-rationalize their past right decisions, even citing evidence and prior art to lend a sense of rigor to their overall design process. Skillful action almost seems defined by the fact that the professionals who engage in it know much more than they can say. And they can act based on more than they know, as well. This seems contradictory, irrational; perhaps even to the most rationally minded, it may seem irresponsible. If you can't say how you know something to be accurate, how can it be?

It happens as well outside of the design professions that the most skilled professionals, even among medical doctors, lawyers, and business managers, are actually at their best and most productive when they are working in ways they can least describe to others. A lawyer making an impassioned plea on

behalf of a client in front of a jury, while certainly well prepared and rehearsed, may improvise new arguments on the spot as they watch the faces of their jurors. A trader on Wall Street may "feel the pull" of a barely visible market dynamic. Expert accountants can see through a wall of numbers in a spreadsheet and make accurate conclusions about the health of the organization it represents or the opportunities ahead of it in the market. Even baseball players "know" intuitively what they need to do to win a game . . . though they have no way to explain how they will pull it off. None of this makes any rational sense; we think of it as the "art" in their practice and (rightly) respect it deeply.

Since the Enlightenment, our culture has relentlessly applied the structures and norms of scientific, rational thought to all professional practice. "*Ars sine Scientia nihil est.*"¹ wrote Jean Mignot, fourteenth-century architect of the Milan Cathedral. Without a basis in some kind of rational theory, the practice of any art is without worth. That theory must be logical, consistent, and reducible to first principles that are universal enough to provide a common foundation beneath all similar work. In professional practice, trained architects should be able to apply the theory to real-world situations and come up with logically consistent design results.

Contemporary architectural theory isn't consistent in the same way that the underpinning theories in physics or philosophy are. Where the traditional work of architects could be rationalized, that rationalization has often led to the identification of a new profession that specialized in just that more rational act. For example, Structural Engineering. Or Construction Management. Perhaps even Interior Design, with its increased dependency on premanufactured lighting and furnishings, could be thought of in this way. In time, as more and more of the rationalizable components of the

¹"Art without science is nothing."

traditional matrix of responsibilities of an architect have migrated to new professions, only the mysterious, intangible design work of the architect has remained behind.

Once everything that can be rationalized and specialized has been accounted for by others, it is tempting to assume everything of merit has been covered. What a heartless and cold world we would build for ourselves if this were, in fact, the final solution. The sum of all rational processes in architecture will not give you a complete design, not even close. It is intuitively obvious to people when the spaces in which they are living their lives do not delight them, even if they can't say why or how. They, of course, also know if some obvious box wasn't checked in the design if the roof leaks, the garage has collapsed, or the driveway washed out in the last storm. But it is easy to tell when the poetic concerns haven't been met, too. Few people will care that the roof doesn't leak, because they won't care about the building at all.

The problems faced by architects may be different from those faced by doctors or lawyers. They may be more ambiguous and less consistent. Every building inevitably presents a different set of problems, a different set of requirements, and a different set of opportunities. There are undoubtedly universal principles that apply. The building must provide light and air to its occupants, must protect them from weather, and must support them against the pull of gravity, for example. But there is little guidance for architects seeking to add truly delightful, life-changing space into the world. It is tough to rationalize this part of the work.

How Do You "Design"?

So how is it done? How does great design happen? It is certainly wrong to assume that design

practice can in no way be defined. Even the purest of abstract expressionist painters acknowledge there is significant science underpinning their practice. The physics of light, the chemistry of pigment, and the biology of seeing all have roles to play; without them, there could be no poetry. They also know how to get themselves into the mental space where the work can happen. Maybe it is about waking up early or maybe staying up late. Or perhaps it is about just the right music on the stereo. Somehow, it is repeatable for them. And no matter what you think of architectural designers as professionals, anyone who has been trained through an architectural education intuitively "gets" the processes and practices that lead to being able to accomplish a great design. Design thinking is reproducible and persistently observable in human behavior. There must be some way to describe it.

Architects speak of their work as being part of their "professional practice," and I think we can find the answer we're looking for in that language. There are norms and standards that all architects must know – from building codes that you must meet to pass inspection before occupancy to the universal principles of gravitation that you must know so you design buildings that won't fall on their occupant's heads. Architects have to become the people who can answer questions about the building that nobody has thought to ask yet, and they must be able to do so consistently, decisively, and (even) defensibly. The best architects have fully internalized these and many more physical and philosophical truths. But to work exclusively within their mandates is to be blocked before you've even begun.

I think the *practice* we refer to when speaking about architectural work is more akin to the practice of a professional musician. Like an architect, a professional musician must know some physics, some historical precedent for their chosen music, and many other things. They must (probably) memorize

the music they plan to perform – or at least internalize it enough that they don't need to waste time in performance reminding themselves what notes to play next. They must do all of these things in preparation for their performance, and they must reinforce them and drive them into instinctual behavior through relentless repetition. They "practice" endlessly, obsessively, repeating the same themes over and over again, testing variations, exploring newly discovered nuance and expression. They learn their subject so profoundly that they "become" it, physically embodying the music they will soon perform in public.

Malcolm Gladwell claims you need 10,000 hours of practice before you can be any good at something new (Gladwell 2008). I think most professional architects would say that they are just beginning to get warmed up by then. John Hejduk, who was dean of the Cooper Union School of Architecture when I studied in there in the early 1990s, taught that few architects were able to accomplish works of any real merit before their 50th birthday. There is just too much to learn, too many experiences to be earned in the field, and too few opportunities to try something genuinely new. The knowledge of a professional architect must be driven so deeply into their beings that it is instantly accessible, "... as easy as riding a bike." And it takes time and experience to do that. It takes lots of practice.

A well-practiced architect knows a collection of artistic themes that have worked well for them in the past, and they can synthesize new interpretations of them that respond to differences in programmatic conditions in a profoundly intuitive manner. They are practiced at recognizing new opportunities in the design as they are uncovered and know well how to "listen" to the design while they are working when it starts to "talk back" to them. They know the basic universal theory, where it exists, so well as to be able to take intuitive leaps beyond it into new ideas, new territory for exploration.

In his work *The Reflective Practitioner*, (Schön 1983) described the work of architectural design as closer to jazz improvisation than to engineering. The architect is like one of a group of musicians who are working together to improvise new music based on a shared theme, a jazz standard. The musicians all know the tune and their own instrument deeply. They have confidence in their necessary skills that permits them to explore the space around the melody, responding to new interpretations from their fellow musicians as they appear. We love listening to jazz music precisely because of the tension between the standard and the musician's interpretations. Everyone in both the band and the audience probably knows the melody. There is a basic musical structure (play the tune together, take some individual solos, then play the song together again) that is also understood by all. The music is exciting because of its interpretation, the musician's exploration, and the unique expression in the moment of this particular performance.

A skilled jazz musician is unlikely to be able to describe precisely why a particular sequence of notes was chosen in their last solo. And it is equally unlikely that they planned their solo out much in advance of the performance. Upon reflection, after the performance, they might be able to name a few of the events that lead to their decision in the moment, perhaps something they heard from the preceding soloist, or maybe just a memory of something from farther back than that. If forced to fully post-rationalize, a logical and consistent story could emerge based on history, physics, and prior practice. But in the moment of the performance, none of this would be in the musician's mind. They would be too busy playing.

In a more rational problem-solving profession, a practitioner might consult standard practices or industry norms to model the problem, then optimize a set of variables until a fixed set of performance criteria are met. The basic requirements

(e.g., “the building must not fall down”) can be modeled using real physical simulations and adjusted until the criteria have been objectively met. There is some room for interpretation, of course. You might choose steel over concrete for some complicated set of reasons. A unique soil condition might be discovered on-site during excavation that inspires a different solution for the foundation. Skilled engineers, well-practiced in their art, will know how to reflect-in-action just like a jazz musician.

Schön found in his research that all professions – even those most commonly assumed to be driven wholly by the most rational of scientific standards, have a component of intuitive, creative, design-like thinking in their practice. A doctor, for example, when confronted by a patient presenting symptoms for some illness, must intuit a diagnosis quickly and accurately. The majority of symptomatic conditions are unique, likely something the doctor has never herself quite seen before. In the moment, relying on years of practice to inform an intuitive response, she can respond with a plan of action. A life may depend on the accuracy and speed of response, so it has to be right.

Medical school prepares doctors by embedding in their minds an exhaustive collection of theoretical knowledge in biology, chemistry, anatomy, pharmacology, etc. Then, in grueling years of residency, they practice the application of that theory to real-world conditions under the guidance of a professional with many more years of experience than they have. Finally, they graduate into their professional practice, prepared for a career diagnosing and curing illness.

It is the “art” of medicine that is the most difficult to teach to medical students. Their primary knowledge may be profound, and their ability to recall scientific precedent may be without peer. But it is only through practice, in a professional setting on real patients with real problems and under the guidance of a doctor who has more experience

than they do, that students learn how to do the work they do. The best doctors are those who know the science inside and out but can synthesize unique solutions for unique problems immediately, intuitively, and creatively. “*Scientium sine Ars nihil est*”² may be an appropriately enduring counterpoint to the reforms of the Enlightenment that favor rational thought over intuition.

Architecture doesn't have anything approaching the body of scientific or philosophical theory that underpins medicine. Still, it does have an intellectual rigor of its own that blends art and science in a practice of design, which is both reproducible and trainable. Like doctors, skilled architects can make decisions for reasons that they cannot describe in the moment of their making. Unlike doctors, this ability is taught right from the beginning of their training. The education of an architect is distinct from that of other professionals in that it actively teaches both science and art right from the outset. A skilled architect is comfortable from an early age with the process of reflection-in-action that allows them to work pragmatically as both scientist and artist.

Design Starts in the Studio

The key to this training is in the studio environment, where students work side by side for long hours on poorly defined problems in design. In the studio, young designers learn to think with their hands, to draw by seeing, and how to apply a unique intellectual rigor in framing and reframing problems as unanticipated opportunities present themselves. In a range of contexts from the most intimate of desk critiques to the most public of project reviews, young architects learn how to practice their profession. All architects have this experience in common, and for all of them, it is life-changing.

²“Science without art is nothing.”

From academic design studios, young designers graduate to professional practice, where the studio process begins again. Desk critiques with a professor are replaced by desk critiques with a partner, only this time with the added dimensions of time and money bearing down as well. It is easy to forget the theory and the art in professional practice, to double down on project delivery and efficiency, and to select out all the time spent on design exploration in favor of a faster deliverable. That is, after all, the most rational thing to do. It may not, however, be the best thing to do.

Most software is built to automate rationalized processes, based on an offset understanding of what is the most important thing a future user will want to do. Product managers in technology firms are trained to interview their customers and ask them pointedly about what it is they most find onerous about their work so that valuable tools can be built to assist them with those tasks. This works well for accounting packages, but it is difficult to create software that supports a process that the user cannot describe. Design thinking, particularly during its earliest, least definable sketching phases, is difficult to explain by even its most practiced professionals.

How to Sketch

"We see with the eyes, but we see with the brain as well. And seeing with the brain is often called imagination."

—Oliver Sacks (2009)

What is the first thing you decided about your last project? And what was the thing you decided before that? And before that? What was the decision you made before all the other decisions that you made? (See Figure 1.1) Can you remember what it was?

The composer John Cage advised, "If you don't know where to begin, begin anywhere."³ For architecture, everything starts with a sketch. It might be a sketch of an idea, a sketch of a proposal, or just a sketch to fill the emptiness and uncertainty. Nothing is more terrifying, more exciting, more filled with risk, and opportunity than a blank sheet of paper. In the beginning, anything is possible. Maybe, even everything is possible. If that moment isn't exhilarating, you might want to take up a different profession. For most creatives, this moment is pure, solid gold (Figure 1.2).

To be sure, no design ever starts from a completely blank piece of paper. Particularly in architecture, design always begins with a site and a set of other requirements and opportunities. Requirements are functional. They help to put some structure into that terrifying void; that is what your project looks like at the beginning. If you feel like your project has no constraints, you haven't looked closely enough. And if you absolutely can't find any, you're going to have to make some up on your own.

The moment you make the first mark on the project, whether a literal line on a piece of scrap paper or something else, it will be wrong. If it doesn't seem that way to you, you're probably deluding yourself about something. Maybe about a lot of things. Usually, the first idea you capture is the one that's been standing in the way of all the better ideas still to come. The best thing to do is to purge it out of your mind quickly. Clear the road and get ready to move on.

Your first sketch on a project might be a physical sketch (to be sure, I recommend that), or it might be a formula in a spreadsheet. Maybe it is a great conversation over coffee with your client. Or it might be a haiku. Or an interpretive dance.

³This is a broad theme for Cage, published in many places. For a canonical reference, begin with his "Diary: How to Improve the World (You Will Only Make Matters Worse)" (Cage 2019).

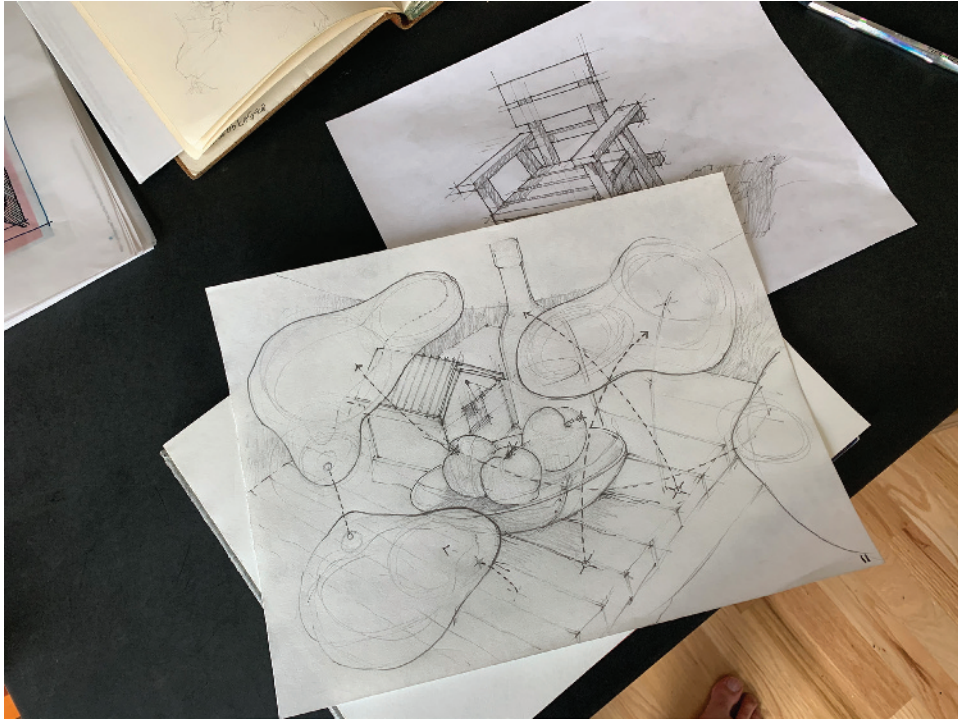


Figure 1.1: Sketching at my desk.



Figure 1.2: "If you don't know where to begin, begin anywhere."

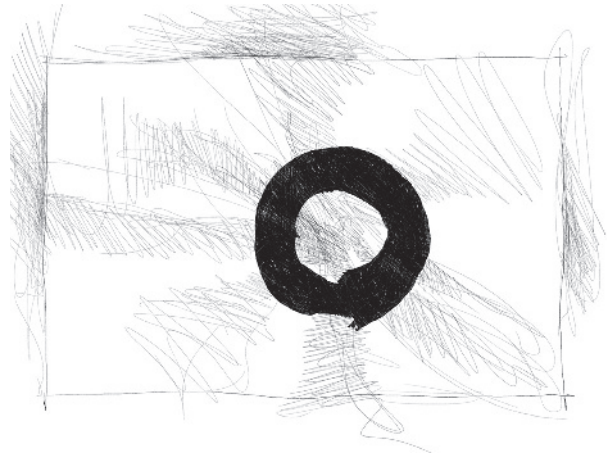


Figure 1.3: Sketching an ensō to check my status.

Whatever gets your blood moving conceptually, whatever purges the bullshit that was occupying your attention before you got started on this new thing. Keep it simple. Move fast (Figure 1.3). There's better stuff ahead.

If you are relatively inexperienced with your creativity, you are almost sure to hang on too tightly to your first sketch. Because it is hard to make things, it is hard to be creative. You're exposing yourself to critique the minute you pick up your pencil. What if the idea is terrible? What if the concept is OK, but the drawing doesn't communicate well? Or maybe the sketch looks like it is saying something completely different than what you meant it to say. What if your client looks at the sketch and sees that you didn't know what you were doing this whole time and you're nothing but an imposter⁴?

Experienced creatives know this moment well, and they recognize it for what it is. It is just something to be acknowledged and moved beyond quickly. Take that first sketch and throw it away. Crumple it up, toss it on the floor. Set it on fire, maybe. Poof, it is gone. Mourn its passing quickly and move on (Figure 1.4).

Now, make another sketch. And another. And another. Throw most of your work away – that helps you move past initial ideas and get faster to the good stuff. Keep your inner critic behind you as much as you can. Every sketch you make is going to be better than the last one was, but you're going to want to learn from the mistakes you've made. Every sketch has a purpose, even if it is only to prove to you that it represented a dumb idea (Figure 1.5).

Be prepared for unexpected consequences and happy accidents. They can come from anywhere, at

⁴"Imposter syndrome," the cognitive bias that afflicts creatives with the fear that they are about to be exposed as a fraud, is incredibly common among all high-performing professionals. While it is possible that you are an actual imposter, you might also consider giving yourself a break.

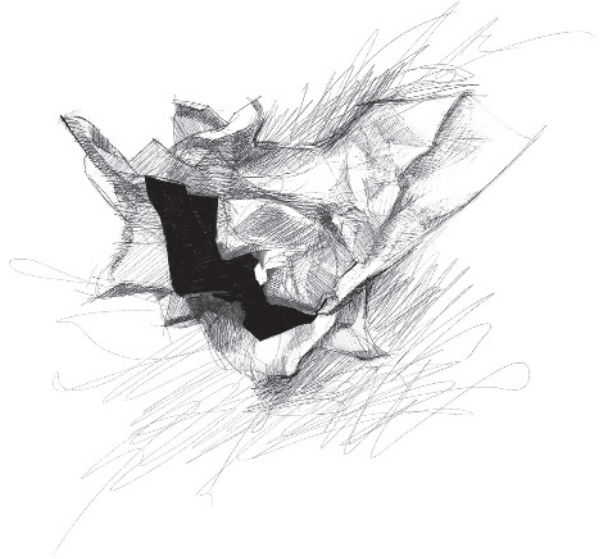


Figure 1.4: Letting it go.

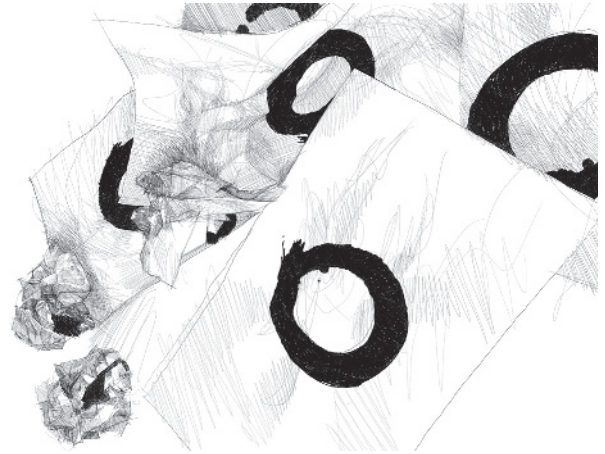


Figure 1.5: Iterate, iterate, iterate.

any time. It is your job to see them when they appear and to understand what they might mean to the project. The simple truth that ideas sometimes come from unexpected places is singularly frustrating to less-creative people. If the breakthrough approach for a project came from somewhere other than the

designer's mind, why do we need the designer? Let's just save that money for something else, right?

Wrong, of course. It is a delightful moment in every design when the project takes on a life of its own. Experienced designers observe that the project begins to "talk back" to them. That decisions made begin to naturally lead to one another as the project as a whole begins to fall into sharper focus. The window you placed above the foot of the master bed just happens to frame the client's favorite view of the mountains perfectly. You didn't plan that, but *bam*, there it is anyway. Embrace it and carry it forward.

You should also be prepared to see something great but not to be able to explain what makes it so. Great designs are found at the intersection of many smaller good ideas. Often, transcendently good designs can only be experienced holistically. Like art, the best architecture usually defies a simple description. Construction teams are frustrated by the fact that you cannot always explain your decisions rationally. If you describe your building to them without being able to explain everything you have designed, they will think you're nuts. If you have a great client (and the best projects come, as we say, from great clients), they might be willing to take your word for it, but you might be the only one on the team who sees it.

Are you feeling stuck? It happens to everyone at some point. Try changing media – if you were drawing with a pencil, pick up a marker. If you were nose-deep in a spreadsheet, go for a walk with your camera. If you were doing an interpretive dance, well, you're on your own with that one, Martha Graham. The important thing is to establish some movement in the flow of ideas, to get past that initial excitement and into the flow state of real, productive, creative design. You trained for this work. Your client hired you to do it. You are special, different, and unique on the job because you can do this work. Relish opportunities to do this work when you find them.

Once you find yourself in this state, you'll never want to leave it. Hours will go by like minutes. Before you know it, the streetlights are on outside the office, and everyone else will have gone home. And you'll feel like you're just getting rolling. Time to stoke up the espresso machine and get serious about this thing. Time to start asking the tough questions, time to get outside your comfort zone.

But then, there's a deadline. You need to come to some conclusions, prepare some concrete recommendations for your client. You're only getting paid for a tenth of the time you just put into this thing. It is time to wrap it up and move on to the next phase. Time is money, champ. Let's make some of that this time, shall we?

But it is so hard to put down the pencil, to come down from that magical flow state. You know that you can still improve the design, always make it better for your client, more comfortable to build, more magical for its future inhabitants. The end of conceptual design is going to make you feel some mourning, some melancholy, some doubt. Beware the inevitable bargaining phase, a pivotal moment in every grieving process. Maybe if you skip lunch today, you can squeeze in one more hour of work. Perhaps if you only explain how important this is, you can move the deadline. Just a little bit. And for the last time, you promise (Figure 1.6).

No creative ever admits to being done with a design. You're just out of time to work on it further. Design doesn't have an obvious single right answer, just a series of decisions, optimizations, and opportunities to be explored further down the line. This notion frustrates people who don't understand it. Your client has in mind (probably) that they would like to have a building to occupy. They would like it to be a magnificent building (they hired you, right?), but they would also like it to be done someday and for a price that won't put them in bankruptcy. And your contractor, guaranteed, just wants to, "... get this project done."



Figure 1.6: Put. The pencil. Down. You're done. The charette is here for your work.

And so, you have to move your unfinished, incomplete, sketchy preliminary work on to the next phase, with all of its half-explored opportunities and practical concessions. If you're lucky, you'll get to keep working on it through the next phase; refining, detailing, and documenting the design for construction. Maybe if you're fortunate, you'll have a chance to track your design through the construction process to make sure it stays true to your intention and (maybe) to take advantage of unexpected opportunities that only appear after construction has begun. Beware, however, that construction teams universally hate change, even if the change you want to propose improves the project significantly.

All the rationalists on your project hate change, though, and will try to beat you down every time you propose one. Be strong, be professional. Stay agile; stay safe. Your business is to change things for the better, but most people hate all change passionately. Embrace that, hear it, and help move

people past their fear. Every architect who has proposed a change in the field thinks they are doing so because the outcome will be better. Every contractor believes this is just moving backward in the process, and if it was so damn important, why didn't you ask for it that way at the beginning.

And they are right. The most significant design decisions you are going to make on the project are made during the conceptual design phase. Before anyone else has woken up their teams to get started, you will have explored a hundred possible variations and navigated through a densely interconnected web of uncertainties, dependencies, and opportunities. You will know not just what the project will become, but also all the other things that it could have been instead. It is your responsibility to defend that until the job is done.

But the design is never done, and the building is a living organic thing that will continuously evolve throughout its decades-long lifecycle. It is a delusion to believe the building is ever complete. You know better. Maybe you can convince the team to give you one more day to work up a couple more sketches to show them how good the building could become. And so, the process begins again. To a designer, the work is never done, the design could always be improved, and there is still a next stage to consider. With time, energy, and the right practices, you can move the world in an ethical and positive direction. And at the heart of your work, no matter how you think about its purpose, you'll always find a sketch – sketch of a new plan, a further detail, a new way to reframe the original design problem. A sketch of the next thing to try.

Sketching Digitally

This is a book about sketching, particularly with digital assistance, and about drawing as a way of working. It is about sketching as a mindset and an

attitude during the early phases of work. Sketches, particularly those done in the beginning when ideas are fresh, and opportunity abounds, are where all the most impactful design decisions are made. It is critically important to figure out how to sketch effectively in the context of ever faster-paced project delivery schedules and tighter budgets. And I don't think we have really figured that out yet. I certainly don't believe we have figured out how to do it digitally. I wrote this book because I think that sketching is getting lost in the digital age, and I believe it needs to be recaptured.

Digital tools, in the end, are as good or bad as their users make them. In the remainder of this book, I will walk with you through a range of digital technologies common (and less common) in architectural practice today to explore their relative merits and usefulness to the problems a designer must face. Pencil and paper remain the most useful, durable, and scalable design technologies for most designers. No question about that – I love pencils. But there are more benefits to be had from digital tools than you might think, more value in particular in learning how to use them in a sketchy way. And the most significant benefits may come in surprising forms—not the ones you expect because they are what “everybody says.” Let’s take some time for introspection, some time for really thinking about what the right way to work might be.

Technologists like me are great at solving the problems they know how to answer. Like the man searching diligently for his wallet under a street-light far from where he probably lost it because “the light’s better here,”⁵ technology develops where it can be built more often than where it should be. Computers are great at solving problems that are rational and clearly defined. They can, with

bloody-minded efficiency, solve those problems over and over again, faster and more efficiently than you ever could on your own. And that is a great asset to your work. In what other contexts in your life do you have an assistant who never tires, which works fast and never questions your demands. Adequately considered and with the right expectations, digital sketching can take you and your work farther than you could ever go alone. But it would be best if you never abdicated responsibility for the original design to a machine. They just aren’t going to be as good at it as you are. Your intuition, your irrationality, and your ability to deal with abstraction cannot so easily be replaced. They cannot be replaced with any yet known technology, and in the end, they may be computationally irreducible.⁶ Not without some fundamental breakthrough in our understanding of how the mind works.

Because computers as “thinking machines” are designed from the particular knowledge and theories of cognition that crystallized during the Enlightenment. Computer “intelligence” is modeled on our own imperfect understanding of human intelligence. The core optimism in the design of computers is that human thinking is somehow rational, understandable, and therefore can be modeled, simulated, and made more efficient. But on an ontological level, we don’t really know how the mind works. There is good evidence, and theoretical interpretations abound, but at the root, there isn’t yet much consensus. Particularly when we consider the fluidly interpretive states of intuition and abstraction. The functioning of the human mind is still mostly irrational. Despite that, with training and experience, it becomes irrational

⁵There have been many variations on this joke, but my reference comes from Abraham Maslow’s “Motivation and Personality” (Maslow 1954).

⁶“Computational irreducibility” is a concept I learned from Stephen Wolfram in his book “A New Kind of Science” that refers to the fact that many computational systems, even quite simple ones, create output that cannot be predicted in any way other than simply running the system iteratively over time. (Wolfram 2002).

in the most practical, repeatable, and rigorously effective ways imaginable.

In May 1888, in a letter to his brother Theo,⁷ Vincent Van Gogh taught what has become for me the most critical lesson I know about sketching. Confounding the dominant pictorial paradigms of painting in his time with its ever-increasing photographic realism, he wrote, "I'm trying now to exaggerate the essence of things, and to deliberately leave vague what's commonplace."

This is a powerful statement about abstraction, about simplicity and about sketching. If I'm able to convince you of nothing else, I hope I teach you how to take the time to draw again – only this time, with computers. To sketch is to forgive yourself for having some bad, unformed, and ill-considered ideas. Ideas that are OK to explore, then OK to throw away to make way for the better ones to come. Sketchy ideas. You don't have to nail the whole design right away, and you don't have to do it as fast as you can. Take the time, do the work, and iterate, iterate, iterate. Learn how to recognize the right level of detail – the essential message you want to express. Your clients and the world they operate within will thank you for it in the long run.

Be the Designer

Among building construction professionals, there is an understandable desire to compress construction schedules and make them more predictable. Faster, cheaper, better is ever the mantra of industrialization. Craft often suffers in the name of progress, and not always inappropriately so. To adequately house the world's ever-urbanizing population, we have to build more efficiently. To

build faster, we need to optimize project delivery schedules. Construction schedules compress everything that can be shortened, leaving behind only the truly irreducible elements of the overall work. "Design," identified as a distinct phase, given a name, and defined with fixed deliverables, appears to rational schedule-optimizers like a reducible component of the overall project. The ephemeral, never-ending work of the designer must be put in a box and quantified. You can't change how long it takes for concrete to cure on a job site, but you can tell the architect to spend less time picking out the wallpaper.

If you understand design as a process, a continuous one that is never completed, shortening the amount of time spent on it does not lead to better buildings. You can surely spend less time navel-gazing about wallpaper, but there is so much more to the process than that. By reducing design to trivial things, construction teams are missing the big stuff, too. The absence of design is bad design, not no design or less design.⁸ Time spent on design is fundamentally about making the project better than anyone might have expected it could be at the outset. What are we saying about the future if we reduce the time we're willing to spend making it better?

In the construction industry, the word *design* is overloaded, misused, and poorly understood. At its best, design satisfies both functional, known requirements ("keep the roof from leaking, please") and the more intangible, human, emotional needs that make you feel like a better person when you're wrapped up in them. Somewhere, however, *design* picked up pejorative connotations that are wholly unflattering. Design became associated with the "pretty" things in a project; the colors, the furniture,

⁷In a personal letter from Vincent to his brother Theo Van Gogh, Arles, Saturday, 26 May 1888. Reprinted in (Jansen 2014).

⁸Restated from Douglas Martin, "The alternative to good design is bad design, not no design at all" (Martin 1991).

the finishes. Practically oriented teams began offering design as a “value-add” above and beyond the cost of just making a building. Something you might add to the project if you have a little extra money. And of course, in this context, it is an easy thing cut from the construction budget.

But if instead, you think of design as the activity of figuring out what we will build before we build it, you can't avoid doing it anymore. There is no such thing as a building that has not been designed . . . but there are plenty of buildings that were built by teams who didn't spend enough time during a design phase to really work things out well. In many cases, real design decisions are made passively by professionals with no practical training in the art. Design decisions may be made by people who don't even really know that they are doing design work at all, leaving critical decisions up to ad-hoc memories of best practice or just going with “what we did last time.” There is nothing common about common sense – it always carries with it with unanticipated baggage. Not acknowledging that “design” happens on every project isn't a way to avoid doing it altogether. It is, instead, a way to ensure that it is done poorly and not being given enough time and attention.

I don't much care how you go about the essential practice of design. There are as many design methodologies as there are designers in the world. I'll, of course, have some suggestions in this book, but in the end, I think you want to come up with your own way of doing it. Architecture schools are good at training people to do “design thinking.” But if you think there is a single best practice for doing it, a unique “right way” (and by association then also a much longer list of “wrong ways”), you are doomed forever to a life of doing derivative works. Your work might end up looking like the work of whoever convinced you to follow their process. I want to persuade you to just keep doing it. Whatever it is, by any means necessary.

Design can be thought of a something that happens in a phase at the beginning of the project, but that isn't really the complete answer. If you take the broad definition I'm promoting, the design isn't done until the project is complete. Not, that is, so long as someone is still making decisions about what the building is. Maybe, even, the design is never done – not as long as the building is still standing. Someone somewhere will always be thinking about what to change next.

Architects are the people who know what the building is. Their particular knowledge comes from an understanding of the project's unique design requirements, the wants and needs of stakeholders, the affordances and limitations of the site, and the thousands of other things that all go together into the designer's process. And that process is what really matters. No design is hatched fully complete from the brow of the architect. Great designs take time, mainly time spent making bad decisions, culling through loose ideas, and iterating to a final decision about what should be done next.

You are responsible to your client to make the building as right as it can possibly be. You may be alone at times, maybe most of the time, in truly understanding what the building wants to become. Be quiet, be confident, and, when necessary, be persistently immovable on the decisions that really matter. Because you may be the only one who understands just how much of a difference it will make to do it the right way.

But as soon as you can, as often as you can, be the expert in a room full of people who care about the same great outcome for the project that you do, but who are experts at other things that you don't know anything about and who represent points of view that complete the project in ways you can't. No project is built by a singular vision, nor should it ever be. Even the smallest of plans have a team of people working on it. Learn how to take feedback

as easily as you give it. Even harsh critique is a gift. Someone thought enough of you and your work to tell you it should be better. Shut up and listen to that every chance you get.

You get better for your next job by learning from the experts around you on this job. Learn how to present your ideas, even if they are only poorly formed sketches so far and remain open to feedback. Critique is a way of finding the problems you are blind to yourself. Challenges lead to reframing the question and to unblocking the flow of improvement and advancing the design.

In the end, the design isn't yours. It belongs to the project, and if you have done your early work well, the rest of your team will pick it up and run with it. All those hours you spent getting the conceptual foundations laid will pay off when the next level of detail comes into focus, right between the lines you laid down in your first sketch. It is a beautiful moment when it happens. The design has, in a sense, a life of its own. It knows by itself what it will become, and your job is just to let it go.

It (Really) Isn't about the Tools

Most designers at some early point in their career are confronted with the feeling that they aren't going to get hired unless they know how to use some digital design tool. Maybe it is AutoCAD or Revit, or maybe SketchUp; perhaps it is the latest Hollywood rendering engine, or maybe it is something else. Before computers, young designers had to prove their worth with pencils, gauche, and portfolios. But now, internet forums are crowded with questions about which tool is the "best" one, which tool is the "industry standard" for their particular chosen field. Students are clearly worried that they won't be able to break into the profession without knowing some tool that they have heard

". . . everyone uses." And, if you look, you'll quickly find people who will feed those worries.

To be sure, employers aren't happy about paying to train newbies on the job. This is true in really any profession. Architects are no different, though perhaps they run on thinner profit margins than other professions. Every professional has some quantity of drudge work to get done, and most are only too happy to foist that off on someone they hire just out of school. Maybe it was manual drafting once. Now it may be CAD work or architectural visualization. Or organizing the sample library and just keeping the coffee pot warm.

True virtuosity with any skilled task comes only after many hours of practice doing that skill. We become experts at what we practice, and so choosing the right thing to practice does make a real difference. The tools you choose to help you accomplish the skill do matter, but maybe not as much as most people think. A virtuoso violinist will be able to make any instrument sound great, even a broken-down fiddle from a garage sale. Surely, they can make even more beautiful music with a Stradivarius violin if they have one, but their skill is in them, not the instrument. Conversely, an amateur playing that same Stradivarius will not automatically become a master through playing it.

Many architects suffer a kind of tool-oriented identity crisis at some point. They might lose an important job to someone who uses a different tool than they do. Clearly, that other guy is no smarter, so it must be the tool that made the difference. Or they might find that some celebrity whose work they admire in the profession has adopted something new. It is easier to believe that it was the tool that made the difference, not the designer who used it. Classically, all that was really needed to express an architectural idea to someone who will help to build it is a piece of paper and a pencil. You don't need software to make architecture. You certainly

don't need someone else telling you what software you need.

Technologists build software tools that address an identifiable need among customers in a market willing to spend money to improve their work. For architects in the 1980s, it was the manual drudgery of drafting that seemed most in need of improvement. It takes a lot of concentration to put together an effective, complete, and well-coordinated set of construction documents by hand. This is especially true when the design that is being documented is still being refined every day. If what you really care about is the design of the thing you are building, you naturally come into conflict with the static documentation of that design. You really need a lot of your decisions made before documentation starts... and that lends a stiff formality to the ambiguous certainty of your conceptual work. Wouldn't it be great if there were a tool that magically solved all those problems for you? Heck yeah, it would.

The software industry responded by designing and launching computer-aided drafting tools like AutoCAD. In time, these more primitive tools were followed by whole-building simulations that even automated the basic drawing layout to some degree. Computers are really great at sweating the details of your design for you. It seems like a perfect match, even. Let the human do the squishy organic conceptual thinking and leave the computer to handle the details. Unsurprisingly, many software companies have recognized this opportunity and launched products that aim to support it. And architects have correctly embraced them for the quantifiable performance improvement they bring.

But none of these tools, even those that cost thousands of dollars per year and promise a single all-inclusive solution, can by itself turn a marginally worked out design into a masterpiece that changes the lives of its inhabitants. Nor should you want them too. If you are trying to learn to sing

opera, these tools are like the autotune for your quivering beginner's voice. They'll make you look better at a glance, but they can't really make you a better designer all by themselves. And those who claim expertise in these tools are often masking their underlying inexperience with design. I'm sure you've seen drawing sets that look great but represent designs that can't actually be built.

Ditto for rendering tools, though, as a class of applications, they are even more commonly associated with the design process. Buying the latest rendering engine, the most expensive Hollywood animation program, or the hottest new hardware to run it all will not make you automatically into a great artist. Again, a genuinely great architectural illustrator can make a compelling rendering with only a stick of vine charcoal and a piece of paper. You'll find no shortage of people who claim that the success they have had in their practice has come because they bought some new rendering tools. Always look closer when people tell you that. What are they not telling you?

Most software is sold on the promise that it is simpler to learn, easier to use, faster, and more powerful than its competitors on the market. This is true for free software as well, which has the advantage of costing you nothing but the time it takes you to learn how to use it. These messages are honed to resonate with an architect's deepest set professional fears – the fear of obsolescence, the fear of missing out on future success, and the fear of being exposed as an intellectual imposter. Every professional, particularly the creative ones, has these fears and needs to come to terms with managing them. But new software tools won't help you overcome these fears automatically.

The way to overcome your professional fear is found in work. Real, hard, constant practice to become better at what you do. The tools you choose should support your practice, not define it. Examine everything in your toolbox regularly and honestly

appraise its position in your process. Be quicker to reject new things than you are to add them. Always refine your process to fit your needs. And commit to the tools you can't think of a reason to reject in the end fully, thoroughly, and enthusiastically.

Take input from others you respect and let them mentor you where they can. This includes the experts within software companies. Put aside your natural paranoia and fear of exploitation by people who are making a living by selling you something. Inside every software development team are people who genuinely care about making your practice a better one. They have your back, always. Regardless of how it may seem to you from the outside at times. But don't ever listen to, or even worse, take direction from the mob on the internet.

Virtuosity with pencil and paper, with charcoal and newsprint, with pen and trace, comes with practice to most people. The cost of tooling up is modest, well within reach of anyone interested in learning to use them. Digital tools are different in that they usually cost real money. Even free-to-buy tools will end up costing you money in the end. You still need to buy a computer to run them. You still need to spend time learning them. You still need to put in the hours of practice required to reach that reflection-in-action state where you get the real design work done.

Additionally, digital design tools come with fixed costs of obsolescence. Eventually, you are going to have to buy a new computer to upgrade your software to the latest version. You might get frustrated at how quickly you run through paper and pencils, but digital tools don't last forever, either. In the software industry, we refer to this as "bit rot."⁹ Basically,

this is an acknowledgment that digital technology is still in its infancy in many respects – that it still makes radical leaps in performance and capability from year to year. And those leaps into the future often come at the expense of last year's technology. Like a shark who needs to move forward to keep water flowing through its gills, technology must continue to advance to stay alive.

Logically, you might think you can just hop off this constant upgrade train at any time you want. Just freeze your computer, your operating system and all the applications that are installed on it right where they are, right where you like them today. And to be fair, some people do try to do this. In the end, though, I wouldn't recommend this strategy to anyone serious about their tools. Digital tools are far from maturity. Unlike a pencil, whose underlying technology has remained unchanged for centuries, digital tools still have a long way to go before they stabilize. This is especially true with software from companies that stay in business by selling upgrades. Every year, they must add some new feature that captures enough attention to drive customers to buy again. Can you imagine how your pencil might look if you kept adding new features to it every year?

True virtuosity with any tool comes only after lots of practice. Good habits are usually earned by kicking bad ones. It is easy to conclude that a piece of software that changes out from under you should never be upgraded again. There's a terrible loss of agency that comes from a change to a tool that is inflicted on you for reasons you don't understand by a development team that doesn't know you, doesn't appreciate you, and doesn't listen to your needs. How can you really hope to gain virtuoso-level skills with a tool that changes all the time? Stradivarius-playing violinists never had to deal with that kind of change, right?

It is both a problem and an advantage that digital tools are changing all the time. New hardware

⁹"Bit rot" refers to the fact that software, if it isn't updated for compatibility as the hardware and operating systems required to run it are improved over time, will eventually fall into obsolescence and become impossible to run any more. Documented in *The Hacker's Dictionary* (Raymond 1996).

advancement affords new opportunities (features, performance, stability) to the developers of software tools that run only on the new hardware. Good software developers are always looking both forward (the latest, hottest, newest opportunities) and backward (the features our users already know how to use) at the same time. But there's no responsible way to stem the flow of technological change. While you may well rage against it, it is always best to embrace the constant cycle of improvement (even when it doesn't feel like an improvement to you) and keep moving forward.

So how can you be a virtuoso on an instrument that is always changing? By keeping your attention focused on the simple bits that still matter the most. Details in the implementation of a particular capability may change from release to release, but the core paradigms change only very slowly. Consider Photoshop, for example, which I have been using on and off since its initial launch in 1990. It has always had a conceptual ability to modify the brightness/contrast levels in a photograph. The build of Photoshop I'm using today is running on hardware almost unimaginably more capable than the Macintosh SE I first ran it on. It has hundreds of additional features but is still rooted in its core can change the brightness and contrast levels in a photograph. My monitor is better, my computer faster, the camera that took the photo is profoundly different, too. But if I knew how to judge the right brightness and contrast for a photo then, I still remember that now. But I can do it with much more control, much more precision and much more interactively than I could back in the 1990s.

If I kept a frozen system, with the original hardware, software, and operating system to run that original version of Photoshop, I could still use it. But think of all the other things I would be missing out on. No internet, no digital cameras. No color, even. If I move forward with the technology, I have more time to compare versions, more time to try different shots

from the same photography session. More time to iterate and interactively find just the right solution. More time to collaborate with others and come to a consensus on the final product. Quite a lot has improved about Photoshop since that first version.

We are 50 years into the digital revolution, give or take. Fifty years ago, the first TCP/IP packet traveled across the internet, the first CAD system was demoed, and the first human being walked on the moon. AutoCAD, the first CAD system of merit to run on personal computers, grew to popularity in architecture in only the mid-1980s – just 30 years or so ago. Desktop publishing, popularized with Apple Macintosh computers, the Postscript language, and cheap laser printing systems, really came into its own only 20–25 years ago. And 3D modeling, available widely to ordinary people on their personal computers (notwithstanding the meteoric rise . . . and subsequent fall of Silicon Graphics), really only came to life in the late 1990s with the advent of cheap GPU-carrying graphics cards from companies like NVIDIA. SketchUp, born in 2000, rode that wave side-by-side with first-person 3D shooter games like Quake. Even now, as I'm tapping away at a state-of-the-art 2018 Mac mini, this is all still very new. Architects in their fifties all still remember drafting by hand. And they are the ones who will be doing most of the great works of the next two decades.

You might be thinking that you can just build your own software, just perfect for you and your needs without having to compromise in any way. Software development projects are expensive to maintain. You need a talented team of developers, product managers, UX experts, and quality assurance testers – not to mention the sales, support, and marketing folks who all work together to keep a company's lights on every day. As an individual designer, you can't afford their work by yourself. For purely economic reasons, your needs are going to be bundled together with the similar needs of a bunch of other folks like you. Together, if you all like

what you see and are willing to pay for it, vital and capable digital tools can be brought to life.

Outside the insulation of a small number of super-titanic companies like Adobe, Microsoft, and Autodesk (and of course others), a software company's lifespan may be shorter than you imagine. I was reminded of this during the 2006 opening ceremonies for Denver's newest art museum, a crystalline structure designed by Daniel Liebeskind's office. In the building's opening exhibition, curators had added a display of the technology used in the building's construction.

They showcased an (excellent) 3D modeling application called FormZ. Like many architectural designers in the late 1990s, I had been using FormZ in my professional work, and at one time I knew it quite well. When the Denver Art Museum opened its new galleries to the public in 2006, SketchUp usage had become quite common inside design studios around the world. One of the curators asked me, upon discovering my association with SketchUp, if it had also been used on the Denver Art Museum. I confessed that it had not even really existed when that project was in its conceptual design phase . . . six years previous to the opening ceremony. Now, 10 years or so later, it is more likely that an architectural designer would have used SketchUp than FormZ. This is no slight to FormZ, which remains an excellent tool for architecture, substantially better than it was when I used it, even. But companies rise and fall in the technology world. Faster, in many cases, than the large construction projects on which it might be employed.

I maintain a personal archive of old 3D modeling software, especially their manuals, that I find enormously useful. The first software on any new market sets norms and practices that all others must either follow or reestablish on their own. The basics that we all rely on today are all in there, right at the beginning. In fact, if you look farther back, to Ivan

Sutherland and his "Sketchpad" demo,¹⁰ you'll see even more precedent-establishing basics outlined for others to follow.

Learning to be a digital sketching virtuoso relies on a few basic principles. Simplify your practice such that your basic tool needs are simple and universal. But don't ever let yourself become dependent on any single feature on a particular tool or any individual company's services. Never let your practice become so brittle that something as simple as moving a button or changing the name of a menu item will break your flow. Change is inevitable in software development, and you don't really have much practical agency over it. But you can manage the way you work, your practice, and your expertise. With an appropriate focus on the result you really want – which is presumably (as an architect) a way to design and build better buildings – you can build mastery, even virtuosity, over any collection of tools you may be confronted with.

Some aspects to digital practice stand a better chance of lasting than others. The internet, for example, is likely to outlast us all. It was built for resilience and designed to last. The internet has grown so large since its birth 50 years ago that it almost defies imagination. There is something new on the web every second of every hour of every day. And yet, the core principles behind its conception remain available to all as well. I am a strong proponent in the reliability of the internet. Hardware failures and local disasters have become, for me, only annoyances for my digital archives. My digital work, I believe, is going to outlast everything else I have done.

¹⁰While the idea for similar systems existed in science fiction prior to his work, Ivan Sutherland, of MIT's Lincoln Lab, is generally acknowledged to have demonstrated the first functional Computer-Aided Design (CAD) system. His demonstration application "Sketchpad ('the Robot Draftsman')" set the standard for all subsequent CAD systems (Peddie 2013).

Stay Agile

At the beginning of any design project, you will know almost nothing about what the design will ultimately become. Even the best and most carefully researched projects will have a minuscule fraction of even the most critical requirements defined. You may think you know a lot about the project, but you will have only begun to discover what the project will eventually become.

This is not unique to architecture. In fact, all design projects share this problem. When you are making something truly and profoundly new in the world, you don't have much precedent to fall back onto for guidance. For example, writers working on an original work of fiction commonly admit that they don't know how their story will end until they get there. Certainly, they may have a rough outline to guide their work, but the best writers remain open to make changes in response to new things they discover about the story during the process of writing.

In mathematics, the phenomenon of emergence in complex systems is well known and has been widely studied. Stephen Wolfram describes entire classes of mathematical problems that are computationally irreducible in his book *A New Kind of Science* (Wolfram 2002). They can only reveal their future states through iterative development over time. Even the simplest of computational systems, for example, the one-dimensional cellular automata most often associated with Wolfram's work on complex systems, are capable of showing results over time, which are wholly unpredictable from their initial conditions (Figure 1.7).

Designers of a more engineering mindset often question this point, believing that if the requirements of a project can only be defined completely enough at the outset, then there is only one possible path through the design process. Leading,

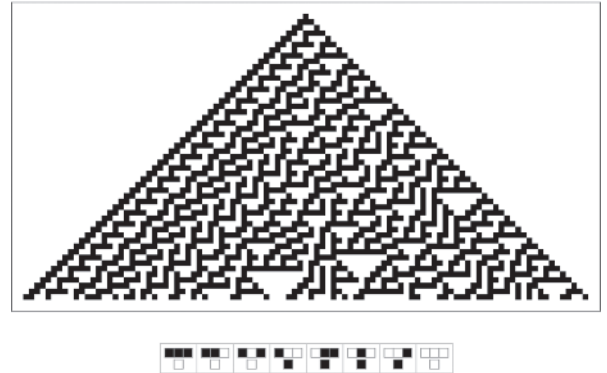


Figure 1.7: Rule 30, an elementary 1D cellular automaton, evolving from a single seed.

eventually, to a single best solution as a highly predictable outcome. To be sure, a complete understanding of all known requirements at the start of a project will help everyone on the team considerably.

But you shouldn't despair at your inability to meet your engineering colleagues' high expectations. Design, like it or not, is a wicked problem. As defined by H.W. Rittel (1973), wicked problems are those without a clear best answer. Their solution seldom follows a predetermined path, and there is no deterministic conclusion to the process of their solving. Every wicked problem is unique, and the formal definition of the problem may only be apparent after a solution has been proposed.

Rittel and his colleagues found wicked problems in domains as diverse as politics, economics, and environmental science. Natural disasters and disease epidemics pose wicked problems to be solved. On closer inspection, there are wicked problems everywhere you look. But design is undoubtedly among the most wicked of them all.

While wicked problems resist most attempts at rational systematization, some practices can help

you and your team to cope. In software engineering (which, despite the name and many outward appearances, has much in common with other design professions), a tactical methodology for managing the wickedness of software development has become widely used. Called *agile development*, it is a system by which development teams can deal iteratively with changes that become apparent only over time. Design decisions made early in the development of a piece of software can be reworked or replaced in time without driving the entire team insane.

I think there is much to learn from Agile development in software, and there are enough shared problems between software and construction that I feel a little “agile” thinking during design is quite valuable. There have been many books written about agile as a software development methodology, but there’s little application of that thinking in architecture and design. Think of this as a proposal for something that might work for you, not a silver bullet that guarantees your success.

Like all great social movements, Agile started with a manifesto. The core principles were initially published in the Manifesto for Agile Software Development: “Our highest priority is to satisfy the customer through early and continuous delivery of valuable software” (Beck, Beedle, and van Bennekum 2001). Agile favors individual interactions over the adoption of tools, working software over documentation, customer collaboration over contract negotiation, and the ability to respond to change over blind adherence to a predefined plan. If that sounds familiar to you, I think it should. You may already be practicing according to at least some of these principles. Agile software development adds some ideas that will seem a little more radical, though, as well.

What might an “agile” sketching practice look like for architecture? Let’s imagine what an agile practice,

based on the original Manifesto, might look like when you’re sketching on the design of a construction project. The following sections are adapted from the Manifesto (Beck, Beedle, and van Bennekum 2001).

Client Satisfaction by Early and Continuous Delivery of Valuable Design

Every project begins with a single line in a sketch and, if all goes well, ends with the construction of a building that meets and exceeds the owner’s expectations. But no design is hatched fully formed and perfect from the mind of a designer. It takes time and iteration to reach the best solution for construction. Traditional construction practices follow “waterfall” project management schedules, where each discrete task is assigned a duration, with dependencies, and then is tracked through to completion. This takes too much time, and it is way too rigid for effective design practice.

Instead, you should plan on delivering a little work all the time. Don’t collect all your decisions into one giant presentation that hits both your client and the rest of your team with hundreds of new pieces of information all at once. Instead, think in a continuous delivery process. Every day, every minute, you should have the best current representation of the project ready for review. There will always be unsolved problems, and there will always be unfinished parts. But there should always be a design “build” ready to share, at any time.

Welcome Changing Requirements, Even When They Come Late in the Design

“Agile processes harness change for the customer’s competitive advantage” (Beck, Beedle, and van Bennekum 2001). Design work requires iterative reframing of the problem as new requirements emerge.

You know the value of this without even having to acknowledge it, though it is a natural part of every project to be angry when things change. Especially if the changes cause you to rethink decisions you already thought were complete. Or, even worse, if they cause you to throw away something big that you worked hard to complete. An agile practice recognizes that every design begins with imperfect requirements and that, inevitably, conditions change from day to day. Embrace that and become a champion of changes as they appear in the project.

Because, sometimes, you will be the one who wants to change something. If you have been open and flexible about things that are changing out from under your work, you set an example for others on the team to follow. Changing requirements indicate the team is learning and understanding the problem they have set out to solve more clearly and in greater detail.

Be responsible, however, and always be professional when things don't change in the direction you want them to. You are always working to make the best of situations as they manifest; don't complain about things that can't be helped by your complaining. Sometimes, especially in the latter stages of a project, it is appropriately hard to make the changes that you might want to make.

Deliver Complete Design Proposals Frequently

From Beck, Beedle, and van Bennekum (2001): "Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale." Agile software teams usually set up automated "build" servers that continuously rebuild the software every time someone checks in a change. Once a day, all the day's changes are collected up, and a "daily" build is made that might be ready for closer testing. And

maybe once a month, a real ready-to-release to the user community build might be done that is carefully tested and documented, then sent out into the world for people to use.

In your design practice, you can think in similar terms. If your clients happens to drop by your office, you should always have something to show them; today's best and most complete version of their design. If you are genuinely integrated with other design consultants, you might have a daily meeting where the latest design can be shared, and issues coordinated. You'll need a "daily" version of your work for that. And maybe you have a weekly meeting with your team as well, where you have some real-time to dig into deeper issues. You need a "build" for that as well.

You should be thinking at all times about the complete project. Don't get hung up in excessive detail on any one part of the project before bringing everything else to a similar level of development. You should always know what the best current information is on any particular question about your design. Forgive yourself for not having detailed answers to every question, though. There will always be some aspects of the design that are not complete, even that might be self-conflicting or clearly out-of-date. By definition, some part of the design is not complete until your design work as a whole is complete. And given the "wicked" nature of your work, there will always be some new thing you could work on a little bit more.

Maintain Close, Daily Cooperation between Stakeholders and Other Designers

"Business people and developers must work together daily throughout the project" (Beck, Beedle, and van Bennekum 2001). The more often you communicate with stakeholders on your design, the less likely you are to uncover a huge catastrophic

problem that you drop onto the team like a bomb. Frequent communication about incremental issues and the details of their development reduces the stress of everyone on the team. Ideally, you should be talking to your client, your contractor, and the other consultants and trades on your job daily. You want them to think of you as the best source of information at all times. Don't be annoyed when they contact you for help – that is exactly what you want them to do. And it is evidence that they respect you and your design work.

Projects Are Built around Motivated Designers Who Should Be Trusted

"Build projects around motivated individuals," Beck, Beedle, and van Bennekum (2001) write. "Give them the environment and support they need, and trust them to get the job done." Great design projects are built around great designers. With no fixedly defined ideal design solution in mind at the outset, your client is really depending on the professional skill of the team of people they assembled to get the design work done. Fundamentally, they are paying for a design team to work on the design for their project until the clock runs out.

It is commonly believed that design is a fixed deliverable, an asset to be delivered to the construction team fully detailed, rehearsed, and perfected before construction begins. I prefer to think of design as an activity that is performed for as long as the client's budget will allow. Among the many assets created during design will be "final" documentation of the design for construction.

In an agile mindset, the final set of construction documents is only the last in a continuous series of design documents created as the design evolves through iteration over time. This is the service you are providing, and if you are trusted to do it well, your team can, together, work with much more velocity.

The Best Design Teams Are Integrated and Co-Located

The concept of a co-located design team is not foreign in the construction industry. Integrated project delivery (IPD) contracts often stipulate that teams must be co-located for maximum efficiency. Wherever possible, you should seek out opportunities to share physical studio space with everyone on the construction team. "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation" (Beck, Beedle, and van Bennekum 2001).

If you accept the Agile notion that the best project communication is continuous, then it makes sense to work in a situation where constant contact is possible. Rather than spending 10 minutes word-smithing an email, what if you could just pop over to your neighbor's desk and have a quick chat? The benefits to overall design quality and the velocity of its delivery are apparent.

Think back to your design education and to the friendships that you formed in the studio environment. Studio environments are at the center of every functional design team, though they may be mysterious and (occasionally) frustrating in how they work. If you can create a studio environment across the team responsible for your project's design, you will have caused an enormously positive change to traditionally isolated and suspicious design team dynamics.

Credible Design Proposals Are the Best Measure of Progress

There is nothing quite like an automated software build for a design project in the construction industry. However, you can still have a continuously updated current best design available to the team at all times.

The essence of this point in the Agile Manifesto is that delivered outcomes are a more robust measure of success than anything else you might spend your time doing. Nobody is impressed if you met a delivery deadline with a design that stinks. What matters, universally, is delivering excellent design work over and over again.

I rewrote this point to include the term *widely credible*. If you are the only person on the team who thinks your design is a great one, then you really are in trouble. To be truly useful on your team, you need people to think of your design contributions (in particular) and of the current best state of the design (broadly) to be credibly great. Credibility means different things to different people. To an engineer on your team, it might mean that you have correctly anticipated how long you will really be able to span structurally between columns. Or, maybe, that the story you are telling about environmental sustainability is groundable in fact.

If the design proposals you champion aren't regarded as constructible, efficient, or in any way diverging from norms and standards of practice, you will need to do a better job selling them to the team. Or, maybe, you just need to do a better job with your design.

Agile Processes Promote Sustainable Practice

Find a pace you can sustain. I use the term *sustainable* here to refer not to green building practices but to give a name to the *sustainability* of your design practice. At no point should you be expected to cause yourself physical harm to meet a deadline or a delivery target. Beck, Beedle, and van Bennekum (2001) write, "Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely."

Agile development is designed to be something sustainable as a practice for as long as you want to keep doing it. By switching from a rigid waterfall management style with big deliverables (and big catastrophes if the deliverable falls short) to one of continuous gradual improvement, agile designers can learn how to manage their time more humanely.

Presumably, you got into professional design because it is something you want to do for a long time. And you get better at doing it the more you practice, so there's a distinct value in sticking with it for a long time. If you can't learn how to sustain yourself physically and emotionally, not to mention financially, you won't be able to do that. You're either going to burn out, exhaust yourself, or just plain run out of money to stay in business. You have to make your practice a sustainable one, one that you do every day for weeks, months, and years at a time.

Continuous Attention to Technical Excellence and Good Design Enhances Agility

The construction industry is distressingly full of distrust and self-preservation behavior. And yet, with very few exceptions, every single person on every part of the extended team is likely motivated more by a desire to do great work that they can be proud to say they did. Nobody willfully derails the process, and nobody intentionally sabotages the project. If they did, they wouldn't be in business for very long.

Your animal brain, the lower order primate thinking that kicks in when you are afraid, might lead you to think someone on the team is out to get you. It is highly unlikely that this is true. Instead, they are just like you, trying to do the best job they can and make the most meaningful contribution possible

in the time available. Some people may be more naturally motivated to avoid effort than others, but laziness is easily overcome by the motivation to work together to deliver great work.

Look for the best in your team, and in return, give your best as well. A team continuously committed to excellence works faster, more efficiently, and is a joy to collaborate with. You can contribute to the positive team dynamic by committing to bring your best effort, holding nothing back, at all times. Every single project you work on should be the best one you have ever done.

Good design makes space for more good design. Excellence in your practice inspires others to give their best as well.

Simplicity, the Art of Maximizing the Amount of Work Not Done, Is Essential

In principle, you should seek to maximize your working efficiency at all times. Don't get distracted by shiny things that aren't related to the project at hand. You know what they are like – an idea that you have always wanted to try, even though it really doesn't have anything to do with the project your client wants you to make for them. You have to show discipline, and restraint, in all these situations. The best design work is concise, clear, and refined. Throw away more from the design than you commit to keep. The simplest ideas are the easiest to explain, the easiest to share and (ultimately) the most powerful when they finally make their way all the way through the construction process.

In design, there is another meaning to this rule. Simplicity in design is more than just a way to maximize the efficiency of your work. You are also responsible to simplify the work of all those who

SKETCHING FOR CONCEPTUAL DESIGN

will follow your design through execution. And usually, it takes a little extra work during design to make a project simpler to execute. Practice the art of simplification every chance you get. You will find that simplicity usually comes only after exploring more complex ideas and throwing out everything about them that isn't directly aligned with your main idea. As Antoine de Saint-Exupère famously advised, “*. . . perfection is finally attained not when there is no longer anything to add, but when there is no longer anything to take away.*”¹¹

The Best Requirements and Design Emerge from Self-Organizing Teams

The most motivated people are the ones who feel a sense of agency over the work they are doing. When you ask someone, or worse when you tell them what they should work on next, you reduce their agency and diminish their motivation. Just a little bit, maybe, but it is there, nonetheless. The most motivated and productive designers are those who love what they are working on and who are so committed to the overall idea that they can make their own decisions about the things they are working on without much oversight or review. They make the right decisions because they understand why, profoundly and personally, those are the right decisions to make.

You won't always have a say about who you are working with, but you can always make an effort to allow teams to self-assemble. And when a team on your project self-assembles for the right reasons, you can recognize it, celebrate it, and then get the hell out of their way.

¹¹“*Il semble que la perfection soit atteinte non quand il n'y a plus rien à ajouter, mais quand il n'y a plus rien à retrancher.*” (de Saint-Exupéry 1967).

At Regular Intervals, the Team Should Engage in a Constructive Design Critique

Every design process includes a process of reflection and critique. In your own design process, you are likely reflecting-in-action continuously every time you pick up a pencil. Your team needs an opportunity to review progress as well. "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly" (Beck, Beedle, and van Bennekum 2001).

The importance of critique and process review is likely the one point in the Agile Manifesto that needs no explanation to an architect. It is an unusual way of working for most software developers, but it is built into the experience of every designer during their design education. Peer critiques over a drafting board, desk critiques with a partner, or even large-scale pin-up critiques with a group. You know that it is vital to get outside of your own head and get some feedback from others as often as you can.

Critique is a skill that needs to be learned for many people who have not had a design education. It is far too easy to take a critique of work you have done as personal criticism; that you are a terrible person because of the design work you presented. You can set the tone for critique by always keeping it positive and upbeat, even when you have devastating comments about the design proposal at hand. I always look for a little laughter in a fierce critique. I let the designer who is presenting know that I think they are just fantastic as a person, even though their design really stinks.

There are some straightforward guidelines for a successful, productive, constructive critique that I like to follow. When I am working with folks who aren't accustomed to either giving useful critique

or to receiving it themselves from others, I will often lay out some ground rules before we begin.

Critique Is Not the Same as Criticism

Critique is delivered after careful consideration of the design proposal as a whole. It is intended to suggest ways that it could be made better or more appropriate to respond to the stated project requirements. A practical critique is one that is recognized by the designer as being correct, even insightful, based on a unique perspective that the designer did not have themselves.

Criticism, on the other hand, is only aimed at finding fault with the proposal. Criticism seeks to degrade the validity of a proposal rather than finding a way to build it up or improve it. Criticism has little value in a critique and should be avoided or ignored.

Critique the Design, Not the Designer

While the design work under critique was certainly created, championed, or at least pitched by a designer, any comments should be targeted at the design, not at the designer. Save comments about the skills of the designer for another context, or just keep them to yourself.

Be Specific, Be Relevant

There's not much value in general statements like, "This design is really nice. I like it." Your general statement of support may be welcome to the designer, but it won't help them advance the design to its next level. Specific comments about what works, and why that is relevant to the project are useful and may be expanded on by others. Equally, possibly even more valuable are comments pointing out something that doesn't work as well, and why that matters.

Always Ask the Hard, Obvious Questions

Just because it is obvious to you, it may not be obvious to everyone else in the room. Never keep something important to yourself. Everyone else may have just missed it.

Critique Takes Practice

It is hard to give useful, actionable, and appreciated critique. You shouldn't be frustrated if you are bad at it at first. Try to remember that you are critiquing the efforts of another human being if you can remember nothing else. The person who is sharing their work is exposing themselves in ways that are usually pretty hard. Acknowledge their bravery and give them the help they need.

The Design Team Owns the Design, Not an Individual Designer

While a single person is, of course, responsible for each individual design asset that you are reviewing, you should always claim collective ownership of the design as a whole. By entering into a real critique of the work, you are exposing yourself and your design ideas almost as much as the person who pinned their work up on the wall in the first place. You are all in this together, so you should act like that.

Practice Your Agility

Many aspects of an agile methodology may seem familiar, and maybe some are a little uncomfortable. Agile practices demand a level of transparency that makes some people quite uncomfortable. You are going to have many more of your flaws exposed, along with many more of your successes. The most important thing to remember, though, is

that both the risks and rewards are more frequent, but they are also much smaller. You will find yourself subject to major course-correction much less frequently. Instead, you'll receive more frequent little taps and minor corrections. In the long run, you'll find you and the rest of your team are much more aligned and resilient to external changes.

Agile design practices are an excellent match for wicked problems. With Agile, you always have the ability to adapt to something new that is discovered only after you've completed some prior work. Every day, in fact, you can tear up something you did the day before in favor of something better that emerged overnight. Since you haven't had time to do much work since the last time you shared, you aren't going to have to reverse much of your work most of the time. And that is a much more sustainable way to work.

Stay Safe

Even with the most efficient of agile practices, the conceptual design phase is a chaotic time for most designers. Sketches are flying around all over the place, with the ideas they embody rising and falling in favor with dramatic speed. As the hours melt away in your reflection-in-action flow state, it is easy to lose track of progress. It is also easy to forget the work that you have left behind.

Finding that one great sketch that you think you remember doing last Thursday, but you're not sure can be quite demoralizing. In physical sketching and modeling, all of this stuff is usually left lying around the studio in a happy (but eminently discoverable) pile of clutter. But digital processes can quickly feel like they leave you at arm's length from your stuff making it incredibly easy to lose things. Sometimes irretrievably and forever. You can't just shuffle through that stack of discarded sketches until you find the one you remembered.

There are excellent desktop search technologies you can bring to task when you feel like you've lost something, and I encourage you to take the time to really learn how to use them. For design work, however, they have limited capability. Much of the digital design work you're going to do will be stored in data formats that are hard to index and, as a consequence, hard to rediscover once lost. This is especially true for 3D model assets. You might only be able to find them if you remember the filename and the approximate date on which you made them. Search technologies, including the ones from Google that claim to be able to "find and make useful all the world's information" are really only good at searching for strings of text, not images or 3D models.

There are specialized search engines available to architecture firms today that do promise to improve on the search-ability of nontextual digital assets. Probably by the time this book is in your hands for reading, there will be a dozen architecture, engineering, and construction-specific collaboration platforms on the market that you might consider using to store your project data. In their own ways, they are all excellent and up to the task of managing your project data. Most importantly, they are beginning to learn how to index building model information. And indexing is the first step toward making assets re-discoverable. There isn't a single best solution for this problem yet.

Back Up Everything

If you have been using computers for anything at all (even just sending and receiving email), you have probably also encountered crashes with data loss. And if you haven't lost data to hard drive failure, you eventually will. Digital data can feel much more ephemeral than "real" drawings on pieces of paper. If you store it only in one place, on a single fragile spinning hard drive platter, it is. Properly

managed, however, digital data is far more reliable, far more secure, far more discoverable, and (ultimately) far more durable than a piece of paper in a flat file in your office. You just have to get it off that one old hard drive under your desk.

Now that the internet (and cloud computing) has matured, there's really no excuse for losing data ever again. Once you give your data over to the cloud, you'll never want to go back to the old way you stored things. It is really that much better. Cloud backups are easy to set up, cheap to manage, and absolutely bombproof in their reliability. The internet was literally designed to withstand nuclear attacks.

Let me emphasize this point. You should be backing up everything you do to the cloud. This is safe, reliable, and performs like a champ. It doesn't matter which service you choose, and I'll leave it to others to argue the relative merits of Google Drive over Amazon or Apple's iCloud. If you don't have a backup right now using one (or, better, several, redundantly) of these cloud data storage providers, you should put this book down right now and go set it up. It's OK, I'll wait. Come back when your backup has started, because you'll have a bit of time on your hands while you're waiting to read more of this book.

...

There are two kinds of digital creatives in the world today – those who have lost data to hard drive failure and those who will lose data to a hard drive failure in the future. Which one are you going to be?

...

OK, now that you have that out of the way, let's talk about why storing your data in the cloud is the better choice than keeping it on your local computer's hard drive. The benefits are considerable, though

you may have some lingering doubts. Lots of people do, though fewer and fewer outside of the AEC industry. It is OK. I'm here to help you get over your cloud aversion.

To begin with, you never want to keep all of your data in one place. Especially not on a single device. What happens if that device fails? Crisis. At one time, setting up a backup system that included some offsite storage was an expensive and complicated process. I still remember rotating magnetic tape backups for my office that had to be shuttled offsite to a bank safe deposit box once a week. That was enough of a chore that it probably only really got done once a month. A device failure in this context was still a catastrophe, just a modestly smaller one. Today, you can set and forget an offsite backup with a simple system extension and a monthly subscription. I pay about five dollars per month to keep unlimited backups of each of my personal computers. It is the most comfortable bill I pay every month.

If anything happened to my computer, like if my house burned to a cinder in a wildfire, I would go down to the computer store, buy a new computer, connect it to the internet, turn it on, restore from my backup, and I would be up and running in an hour or so. Quick, simple, and very safe. I have used this capability dozens of times and thanked the gods of the internet every single time.

The simplest possible way to store the physical sketches you produce in your design work is probably just to dump them all in a big box. I'd wager that all creatives have some version of this hiding in a closet, garage, or studio. I call mine "The Boneyard." Minimally, it is waterproof, and I've done some basic things to keep mice from nesting in it. Loosely organized by project, or at least according to some broadly thematic project-like structure, my boneyard has everything in it I ever thought

was valuable in even the most minor way. I am, admittedly, a bit of a hoarder in this respect.

There are several advantages to my physical boneyard. I know I probably really only have one place to look for an old drawing. If I remember something, but can't find it in the boneyard, then I have to get on with the grief of knowing that it is probably lost forever to me. It wouldn't have been stored anywhere else. Sorting through a bunch of old drawings can actually be a pretty rewarding experience, even if you don't find what you were looking for initially. I often find not what I was looking for, but instead something serendipitously close to it that I wasn't looking for . . . but actually gets the job done much better. Every old project is full of ideas that didn't make it into the final project. Maybe they can find a new home in my next project?

You have to adjust your thinking some when you add digital sketches into your practice. Of course, if my garage burned down, as in the same wildfire that melted my computer in the previous example, this work would be lost forever. My digital backups, dispersed redundantly across the internet, would likely survive a nuclear holocaust. The internet was designed for redundancy, for reliability, and for resilience to disaster. It is much more reliable than a box full of paper in my garage.

Separate from my cloud backup solution, which just keeps a snapshot of my entire system tucked away safely in the cloud, I also keep a digital equivalent of my physical boneyard. Like the box full of paper in my garage, my digital boneyard contains digital assets from projects I have done going back to the very first computer I ever owned. Every time I get a new computer, I just roll it forward. Every new project gets its own folder, and I just keep piling them in. Luckily, persistent storage capacity is still advancing faster than my ability to make new digital data. Every new hard

drive I buy has at least double the capacity of the one it replaces.

This folder gets a little bit extra backup. I simultaneously sync it to several different internet cloud storage providers, which gives me additional redundancy and protection from the rising (and falling) fortunes of Silicon Valley. Today, I use Google Drive, Apple's iCloud, Dropbox, and Adobe Creative Cloud. I also use Trimble Connect, which provides me unlimited cloud storage on top of a collection of other more exciting things specific to the AEC industry.

The contents of my digital boneyard may include files in formats that are unique to editing products long since disappeared from the market. There's no perfect way to prevent that. In general, I try to choose digital tools that I think will be around for a long time, or that have formats I know I'll be able to open somewhere else in the future. I've made good choices about this and bad ones.

And there's an important lesson learned. Beware the specter of data lock-in. Be suspicious of any software ecosystem that has an easy entrance, but no apparent exit. Every digital content creation tool has its own proprietary file formats, for good and responsible reasons. If you want to be able to do something unique with data, some special modeling techniques, or whatever, you will have to adapt to a specialized representation of that data. This is particularly true with 3D modeling tools. No two modeling applications are precisely alike, unfortunately. As a consequence, you shouldn't expect to be able to open 3D models from one in another without going through a lossy translation process.

Luckily, there are open standards that make it easier to go through that lossy translation process. With SketchUp, we have always been careful to support both import and export for the most critical exchange

formats. Standards vary by industry, but for us, it is Autodesk's DWG¹² (for 2D CAD data, primarily), IFC¹³ (for BIM data), COLLADA¹⁴ (for 3D rendering models), and STL¹⁵ (for 3D use with 3D printers). Both imports and exports are inherently lossy, so you need to adjust your expectations appropriately. 3D model data is tough to standardize.

But most other data types you will find yourself working with are much more settled into commonly accepted standards by now. With some exceptions, any image editor can open any raster image file. Any text editor can open any text file. And so on. And every application worth your attention can open files from older versions of itself without any hassle. Photoshop v.20 (2019) can open files created in the very first release of Photoshop almost 30 years ago. SketchUp 2019 can open files created right back at the first release as well (Figure 1.8).

I have seldom lost data irrecoverably to obsolete software, but it has happened. For a year or two in the early 1990s, I did a lot of writing on a NeXT computer. I loved that machine – just a beautiful piece of tech. I used a simple little word processor called WriteNow, and I stored my files on a shared network drive. I still have all of those files today, rolled forward dozens of times to new storage systems and

¹²Autodesk's DWG ("DraWinG") format is a proprietary binary data format invented by Mike Riddle and licensed for use as the native file format for AutoCAD since 1982 (Walker 1989).

¹³IFC (Industry Foundation Classes) is a platform neutral, open file format developed by building SMART to aid in the exchange of canonical building design data in the construction industry (buildingSMART International 2020).

¹⁴COLLADA ("COLLABorative Design Activity") is a 3D data interchange format managed by the Khronos Group designed to facilitate asset exchange in the media and entertainment industry. (Khronos Group 2020).

¹⁵STL ("STereo-Lithography") is a low-level 3D CAD data interchange format created by 3D Systems to facilitate exchange of CAD data for rapid prototyping and 3D printing. (3D Systems, Inc. 1989).

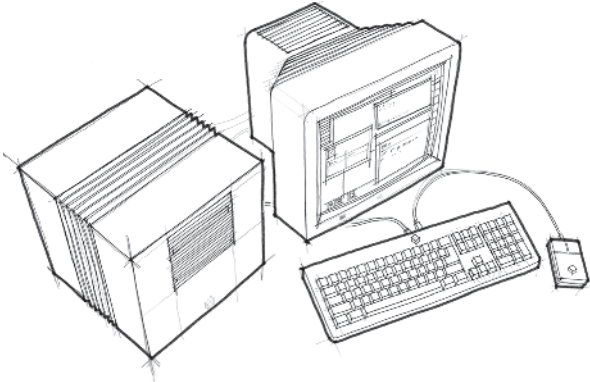


Figure 1.8: My once-beloved NeXTCUBE computer.

backups. WriteNow, on the other hand, shipped its last release in 1992. I have the documents I created, but I'll never be able to open them again. Unfortunately, there is little you can do to recover from this. Mourn the loss and move on, as I have done.

Don't Overorganize

Of course, in time, your personal boneyard of digital data will become large enough that you will start to have trouble finding what you are looking for inside it. Time to get organized, right? Every office I have ever worked for or with has pushed a highly formalized project directory where every single piece of data, real or imagined in the future, has a place to be stored. Usually, the work of a committee (at least in the larger offices where there are enough people to organize committees) is expected to be able to solve the problem of finding data again in the future. If everything on a project has a place where it should be stored, only irresponsible people will store it anywhere else, right?

I have to be honest. I hate this kind of thinking. Templates pre-suppose that every project is the same. Hierarchical systems (like a rigid folder structure) almost always fail to accommodate unexpected

design redirection. "A place for everything and everything in its place" won't help your design mojo during the conceptual design phase of your project – whatever it may be. You don't know what the "everything" you're making is yet. How on earth could you be expected to know where it should be put?

Don't feel ashamed that your previous attempts at a priori classification of the design stuff you haven't yet started to make have failed. The problem is that it is almost impossible to classify drawings, models, and digital data before you have made them for the first time. Eventually, your design will settle down into a set of assets that you can tuck away neatly into a taxonomy of your choosing. Still, in the beginning, I recommend just giving in to the chaos. Just dump it all in one folder and figure it out later if you have to. And if you have a good enough search tool at your disposal, you may never really have to.

Over time, as your key design patterns for the project start to emerge, you'll be much better equipped to assert some organizational structure. Ultimately, you're going to need some system that helps you reference things you've made later. By the time your work is ready for construction, it will be classified richly, logged into formal version control, and changes will be formally handled in team meetings. But when you're just getting started, there's no need for all of that. It will only get in your way, and there is plenty of natural attribute data on most modern computer storage systems to get you started. You don't need an a priori taxonomy in addition to that. Just the usual folders in your computer's filesystem.

If you were active on the internet during the early years of the World Wide Web, you probably remember finding information through extensive human-curated indexes like Yahoo! Sites had to be carefully added to these indexes by a human being, much like a traditional reference librarian,

who would have to be able to classify everything according to some rigid preordained taxonomy. Once organized, users of the system could browse the taxonomy, progressively clicking headings and subheadings to drill down to the place where, eventually, they could find the website they were looking for.

Now, it is not easy to find browsable indices of the internet. Can you imagine a browsable index that contained over a billion individual sites? That just isn't practical anymore. Search engines have replaced top-down taxonomies, and it isn't hard to see why. Search is much more direct than it was to surf an index, hoping to find what you needed.

Version Control Systems

The simplest of systems are unquestionably the best ones when it comes to organizing your data. You can and should (right now if you still haven't; how many times do I have to tell you?) have regular "set and forget" backups running for all your digital data, and you don't need to have a right folder structure in place before you get your backups running. But sometimes a backup alone isn't enough. Design projects aren't particularly linear in their development, despite what project managers might hope for them. Sometimes, a change you made must be rolled back to a previous state and taken in a different direction. Sometimes you need to consider two different variations of a design in parallel. For anything like this, you need a version control system.

Luckily, architects aren't alone in needing to be able to manage versions during design. Software development projects have very similar needs. And for once, as an architect, you can take advantage of another industry's hard work more or less for free. Thanks to the community development of large

software projects like the Linux kernel, reliable and straightforward version control is readily available for any computer. My favorite system is Git,¹⁶ created by Linus Torvalds to help him manage the Linux kernel project.

Version Control Systems like Git operate on a simple principle. Every time you save a file (any file) in a traditional filesystem, you overwrite the previously saved version of the file. That previous state of the file is then lost forever, replaced by your latest version. If you replace your traditional file system with a version control system, you can keep both the previous version and the new version, two separate snapshots of the file from two different times. If you decide later on that an older version was better than the one you're working on now, you can revert back to that earlier version. Or, if you want to keep both the older version and your newer one, you can keep them both, forking the design into two separate version trees. Once a version has been committed to the system, it can be preserved forever, always available if needed.

Every designer in any industry should learn how to use version control systems to support their work. They are easy to set up, easy to operate, and very secure. You can easily integrate them into your existing backup system without any special setup. They also make it easy to share your design responsibility with others. Version control systems allow you to invite multiple contributors to make their own versions of a project, offering up improvements that can either be reviewed and discarded or integrated into the rest of the project's design work.

¹⁶Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency" (Software Freedom Conservancy 2020).

Version control systems are very accepting of changes in the way you store data in them over time, making them ideal systems in which to allow a project organization to emerge gradually. You can commit changes to your project's folder structure as quickly as you can check in any other change. Versions of the project organization are snapshotted in time like everything else. And a simple system like Git is likely going to last a really long time. If it is used to manage a project as vast

and meaningful as the Linux kernel, it is probably going to outlast your needs.

You owe it to yourself to learn more about using version control to keep your sketches organized. It isn't hard, it doesn't cost a lot of money, and it is enormously powerful in the long run. You can be confident that the basic premise of version control is here to stay. Give it a try, and you'll never return to your previous way of doing things.