AUTODESK UNIVERSITY

CS122807

Talking with Robots About Architecture

Jeffrey McGrew
Because We Can

Learning Objectives

- Gain a better understanding of how to be empowered by automation
- Gain a framework in which to judge how to best automate something
- Gain a realistic view of what's possible today and in the near future via robots and automation
- Discover up-to-date efforts using robots and automation in construction

Description

This class will be a deep dive into the future of building: automation, communication, and whether "robots" will change everything. Come for an informed, useful, and realistic overview of how architects, engineers, and builders use automation today, and how they may use it tomorrow. Learn better ways to think about automation, how to apply it within your work, and how you both can use and will be likely disrupted by it in the near future. See many real-world examples of automation in use today—both on the design side via Building Information Modeling (BIM) and on the physical side via robotics. Gain a better understanding of how to empower yourself through automation, and how to avoid wasting lots of time on dead ends—or worse, getting replaced altogether.

Speaker(s)

Jeffrey McGrew

My goal is to make the world a more interesting place. For over 20 years I've done this by making great things. Along the way, I've become a licensed architect, designed and built one of the world's finest bars, turned shipping containers into playrooms, revitalized defunct commercial spaces, and helped built one of the largest digitally-fabricated structures in history. I've founded companies, advised startups and industry, built software, and lately been teaching industrial robots how to dance. Since 2006, I've been the co-founder of Because We Can, an award-winning design-build architecture studio in Oakland, California. I've helped build a brilliant and dynamic team and fostered the deep client, community, and industry relationships a small creative firm needs to succeed. As co-founder and lead architect, my responsibilities for design direction and digital fabrication strategies are balanced with the more quotidian business demands of sales, marketing, and project management.

AUTODESK UNIVERSITY

Intro

This handout is to accompany my talk for AU 2017. It's all about automation, which is both an awesome, empowering force, as well as a disruptive, wasteful one sometimes. This is for both people doing the hands-on implementation of automation within their companies and for those who are managing those people and/or businesses.

It's got four parts; starts with a general overview of the pluses and minuses of automation in general, then gets into ways of thinking about automation and tips to follow when automating things, and finishes with a big-picture take on how to make yourself more 'automation proof'.

My talk will include more case studies and discussion about the different types of automation than this handout, while this handout will go deeper in some of the larger picture ideas. Enjoy!

Automation is *both* Awesome & Terrible

Automation can be an empowering, fantastic, and profitable technology to leverage both personally in your day to day work and within the larger context of your company. But automation, like any other technology, when misapplied and / or misunderstood can turn from an empowering force that will help your efforts to a huge waste of time and resources.

Automation is no more a 'magic bullet' for things than any other technology. Robots only do exactly what you tell them to do, mostly. While it may seem that robots are taking everything over, until 'strong AI' comes along (which I personally doubt will ever happen) there is still much they are, and probably always will be, terrible at as well.

Automation is Awesome

It's Empowering

By automating difficult and/or repetitive tasks, we can empower ourselves and coworkers to be able to do bigger and better things. We can free ourselves from grunt work to focus on the more fun and interesting tasks.

It's Interesting

Automating things is an awesome test of problem solving skills, systems thinking, and detailed work. It's a really fun thing for any designer, engineer, or builder to get into.

It's Exciting

While all-software automation is somewhat exciting, physical robots are way cool. And it's way more fun to automate some boring task than to sit in drudgery brute-forcing things.

It's Profitable

You can create a whole lot of value via automation, which can then lead to more money. And if you can automate things, you can typically demand better positions and jobs. And if you own your own business, it can make you a lot of money.

Automation is Terrible

It's Expensive

Automating things takes a lot of time, money, and upkeep, so it better be worth it. And it's hard to know when it's worth it.

It's Difficult

Programming and robots are already hard, figuring out the right things to automate is even harder sometimes.

It's Fragile

Terminator this ain't. Most automation only works if everything is exactly the way it's supposed to be. A single thing out of place and it breaks. Or worse, when the client changes something, and breaks your whole automated workflow.

It's Disruptive

Sometimes fully leveraging automation requires you to change the whole way you do business, which can be really challenging.

It's all about *Created Value*

Because automation is typically expensive (in time, or equipment, or both!), and the most expensive thing we can do is make something that no one winds up really using, we need to be careful about how we automate things, and what we automate. We need to make certain we're creating value by automating something, and that we're in turn capturing that value. By focusing on, and properly measuring, the value we create via automation we can make certain it's worth it, and that we're really making things better.

Four Types of Automation

So we've found that automation of tasks tends to fall into one or more of these four categories, each with it's own special challenges and issues. By understanding which type of automation you're dealing with, you can better tackle it, and make certain you're actually creating value.

Type One: Same Job, Only Cheaper

This is where we're replacing a task or even a whole job done by a person with automation, purely because it will be cheaper. The automated task won't be done that much better, or something really complex won't be made easy, but instead the automated task is just going to happen faster or take less people to do than the manual way of doing it. It's pure economics, so

with this sort of automation, you really have to be focused on making certain it *really* is cheaper in the long run.

With this type of automation, the mistake you can make is simple spending more automating the task than it would have cost to just pay someone to do it in the end.

Setup and ongoing maintenance costs will be very important to keep under control. It's easy for them to spiral out of control. You might have the best intentions, but get seduced away by what might be 'better' but really is just a waste of time or money. Or what you might think is easy turns out to be way harder to do. So doing whatever is the easiest and simplest thing you can do it probably best. Higher-level languages then, like Dynamo or built-in scripting within your apps, might be your best bet even tho they aren't as 'good' as 'real' applications written in lower-level Visual Studio or 'better' Cloud-based apps.

Beware of 'sunk costs' too, where a big investment today into automating something turns into tomorrow's albatros. Later we'll talk more about the concept of 'Technical Debt' but for this sort of automation, you really want to calculate your opportunity costs and try to keep getting locked into expensive proprietary solutions that are hard to migrate away from in the future.

Basic Dynamo scripting to renumber rooms or to renumber doors to match their rooms in Revit is a good example here. Or simple scripting in Fusion or automating part modeling with parametric iFeatures in Inventor. It's not that hard of a task, but it's not something that's going to be done that much better than a person doing it. It's just simply faster.

A good physical example are items like CNC-automated saw stops and fences for saws. Tigerstop is one of several companies that make one kind of these 'automatic fences'. You walk up to a chop saw or band saw equipped with one, punch a length into a connected tablet, and a stop for the saw automatically moves to that length. It's nothing a skilled fabricator couldn't do on their own, it's just faster. And you can still easily cut the material wrong.

And while you may think automation is always more error-free, typically that's not as much of a sure thing. There are still plenty of things that can go wrong, from operator-error, to GIGO, to misplaced items or data. So while automation may cut down on some of the errors, don't count on it eliminating them.

For Type One automation, ***Cheaper really has to be Cheaper.***

Type Two: Automation Does It Better

For certain sorts of tasks, automation simply can do a better job than a person sometimes. *Sometimes* being the key word here. While a CNC mill can certainly outperform a human machinist many times over, it only can do so on certain tasks. And only when they are making the right things, in the right way.

With this type of automation, the mistake you can make is automating the wrong thing, thinking you're making things better when you're actually not.

Automation can certainly be an incredibly powerful technology, empowering you to do something way better than ever before. Here it's more than just that the automated task is

faster, or cheaper done that way, it's genuinely better in some way. Higher quality, significantly less errors, or more precise, it where you're doing something that you could do, but that the automated solution is just simply constantly better at it.

However, it still holds true that automation is expensive. So for this type of automation, getting it right is more about fully and truly understanding the nature of the Task and Workflow itself than the technology itself. Automation can just as easily do a whole lot of the wrong things just as quickly as the right things. It's also easy to fall down 'rabbit holes' or waste time focusing on the wrong things, spending money to automate something that didn't really need it.

So when you're thinking of automating a task that falls into this type, there are three things you want to focus on.

First is to use some techniques from the lean manufacturing and agile software worlds, in that you want to really study the task to make certain you understand it. Process mapping, asking 'five whys', and actually timing the current task and it's upflow / downflow dependent task times can really help to solve if you're automating the right task. Because it's possible that making this specific task you're thinking of happen faster and better might just create bottlenecks, backlogs, or piles of waste waiting for someone else to have to get to. This specific task might be done better, but it's not making things really better.

Secondly, you should think if you really need for this particular task to be that much better. Sometimes better isn't really needed, and 'perfection is the enemy of good enough'. Just because you could automate something to be better doesn't mean that it's a good idea. Again, like with type one automation, you should do some cost benefit analysis before starting. But you should also think about how the task fits into the larger picture, and if there isn't something else you might be able to tackle that could be more meaningful.

Lastly, you can't automate away stupid. As humans will always be in the loop, don't just assume that automation will solve issues that actually have to do with training, personnel, company culture, or unrelated industry problems. Like any technology, it's not a magic bullet, and you'll want to carefully consider if you're actually automating the right things.

One good example of this type of automation are model-based, BIM-to-fabrication workflows. Going from design, to detailed model, to actual fabrication can make things faster, more accurate, and with better quality and tolerances. Structural steel, HVAC ductwork, millwork & cabinets, curtain walls, precast concrete, and more is all being designed with BIM tools, and then semi-automatically fabricated with automated tools. It's a very powerful way to work.

For Type Two automation, ***better has to actually be better.***

Type Three: Automation Makes It Easy

For some kinds of tasks, it's more about how automation can make what used to be extremely difficult easy. It's not just faster to do it this way, or results in a better end product, it's that it also makes it just much easier to produce overall.

Generative design is a great example of this type of automation. So while it may have been possible with just 'traditional' CAD apps to design something impossibly complex, it would have

also taken a frightening amount of time. By automating parts of the design itself, it can empower you to be able to make buildings far more complex than before, but in less time. Or have an app automatically easily generate thousands or millions of variations on a design to study it.

3D printing, while overhyped, is another good example of this type of automation. While it was possible to sculpt, mill, or cast materials into complex forms, the automated process of 3D printing makes producing things like this easy.

With this type of automation, the mistakes you can make are more in the design of the automation itself; if it's too hard to use, too confusing, too complex, then it's not really easier. It's more a problem of the user interface and user experience being good than it is about the underlying technology being as good as it can be.

With this type of automation the focus should be on 'Empowerment'. You want to keep people on what they are good at, and the automation on what it's good at. There's a newish term for this sort of thing, called a 'Centaur'. It's automation and a human, working together, each doing what it's best at. It's much more collaborative than the other types of automation.

For Type Three automation, ***Easy has to be Easy.***

Type Four: Automation Makes It Even Possible

Finally, for some sorts of things, automation makes it even possible in the first place. Without it, you simply couldn't work that way at all, or make the things you can by using it. This tends to be leveraging a newly available technology, or new combinations of technologies, and because of that, it tends to also require large business-wide shifts to make best use of it. It's not just faster, or better, or easier, it's a whole new way of doing something. And it can be awesome, or disruptive. Or both!

A great example of this type of technology is when CAD, and then BIM, first became commonly available to our industry. It automated a great deal of our 'drafting' production and gave everyone a whole new way to work, but required new ways of working, new roles within your company, and new costs and opportunities as well.

The mistake you can make with this type of automation is that it tends to require a systematic, 'holistic' approach, that might require great changes to your overall business or industry to really make use of it. Just because it's now possible, it might not be really useful, or not useful for your specific situation. Sometimes whole businesses have to be built off of this sort of automation for it to really work and create enough value to make it worth it.

For Type Four automation, ***Just because it's now possible doesn't mean it's worth it.***

Tips when Automating Things

So no matter what Type of automation we're looking at, and whether you are personally working on automating a task or managing someone who is, there are some general things you'll want to keep in mind.

Handle - Body - Tip

Automation is just a tool. And whenever you are designing a tool, it's good to break it down into three parts, and focus on how to make each part the best it's able to be. That is because these three parts have different goals, yet have to work together, to make for a good tool.

I think the first thing to consider is the 'Handle' of the tool, or rather the part that 'faces' the worker. What's the User Interface and User Experience supposed to be like? What goals do we have for it, and how to we measure if we're meeting those goals? Are we looking to make something very custom for a small group of experts, or something anyone can pick up and use?

Then the 'Body' of the tool, or rather the part that's the actual code or physical robotics. What systems are you going to standardize on? What trade-offs and compromises are you making by doing that? How are we going to generate 'unit tests' so we can test the important parts of our solution to be certain it's meeting our goals for production? Is there another platform or system that we should test as well?

Then finally, the 'Tip', or rather the part that actually touches the Work. For code, this might be the specific API or file formats you'll be working with, for physical robotics you might be looking at the tooling, cutters, or actuators; either way it's the bit that touches what you're working on. So how does it interface with the work? Does it introduce errors? And how do we measure that? Is there a better solution available off-the-shelf, or do you really need to custom design something?

By defining these things before you start you can save a lot of time effort, and in the end make much better tools.

Measuring & Metrics

So again, if automation is expensive and complex, we need to be focused on the actual value created by it. One good way to do that is by smartly measuring things along the way.

One way to do this is by focusing on just a few important metrics to measure before, during, and after something has been automated. And it's best to couple these together into what are sometimes called 'Cohort Metrics' where you measure two or more different things, and compare them vs. each other over time, instead of just looking at one or two measurements in isolation.

So for a Type 1 automation, you might focus on 'Takt Time' or the overall time it took to complete the task prior to automation and then after automation. However, you'd also want to keep track of machine downtime, total units produced per year, and maintenance costs over the same timeframe, and compare all three numbers side-by-side, so you get a real picture of what's going on. Just because the time to complete that task might have gone way down, if it's also actually producing the same or less than before because the machine is down all the time then it's vital to understand for really measuring the value produced.

A really great book that digs into this concept well is 'High Output Management' by Andy Grove, which I highly recommend everyone read. Several times! ;-)

Technical Debt

This is a wonderful term from the software industry. It reflects the implied cost of additional rework caused by choosing an easy solution now instead of using a better approach that would take longer or cost more. It's a great way to put an actual dollar amount, even if it's 'fuzzy', on the compromises you may be making as you design your automated system.

So, for example, if I'm looking at buying two different CNC routers to automate production within a woodshop I could buy a cheap slower entry-level one or a much more expensive fast fancy one. If I go with the cheap one, with the assumption I might have to upgrade later when things get busier, I've incurred a 'technical debt' equal to the difference in price between those two tools (and the cost of their setup). A debt I'll have to 'pay off' in order for my automation plan to move forward once production hits a certain demand.

Or another example is that you've decided to stick with your largely AutoCAD-based automated scripting for shop drawing production. In order to 'upgrade' in the future, you'd not only have to buy Revit or Inventor or whatever, you'd also have to hire someone to rewrite everything for the whole new platform. So that decision is 'borrowing from the future' and creating possible large expenses in the near-term that could sink you when a competitor shows up that doesn't have that debt.

So carefully consider your choices, and the future costs they may have. Technical Debt, like any other debt, it's inherently 'bad'. Debt can be VERY helpful for a business to grow faster or jump on other opportunities. But it has to be managed properly, otherwise it can quickly destroy any gains you might have produced with your automated solution.

Larger Issues around Automation

There are some fundamental issues around Automation, and some that are specific to AECO, that are important to keep in mind.

Automation is Fragile

Terminator this ain't. Trading redundancy for efficiency is a great deal until a single small part stops working, and you have no other way of producing the same work until it's fixed. Software and Hardware break all the time, either due to internal or external forces. So thinking that automated solutions are robust and stable is foolish, for even fully approved software updates can completely sink your systems, and just simply time is always working against you as companies stop supporting those systems or even just as important parts wear out.

Automation is Stupid

Robots only do exactly what you tell them to. Until there is some form of 'Strong AI' (which I personally don't think will ever happen, or at least not in some illegence run amouk SciFi way) your automated solution can't also invent a better way of doing something later on. Which a person totally could. So while Automation can add a lot of value once, people can add value constantly.

Automation is Boring

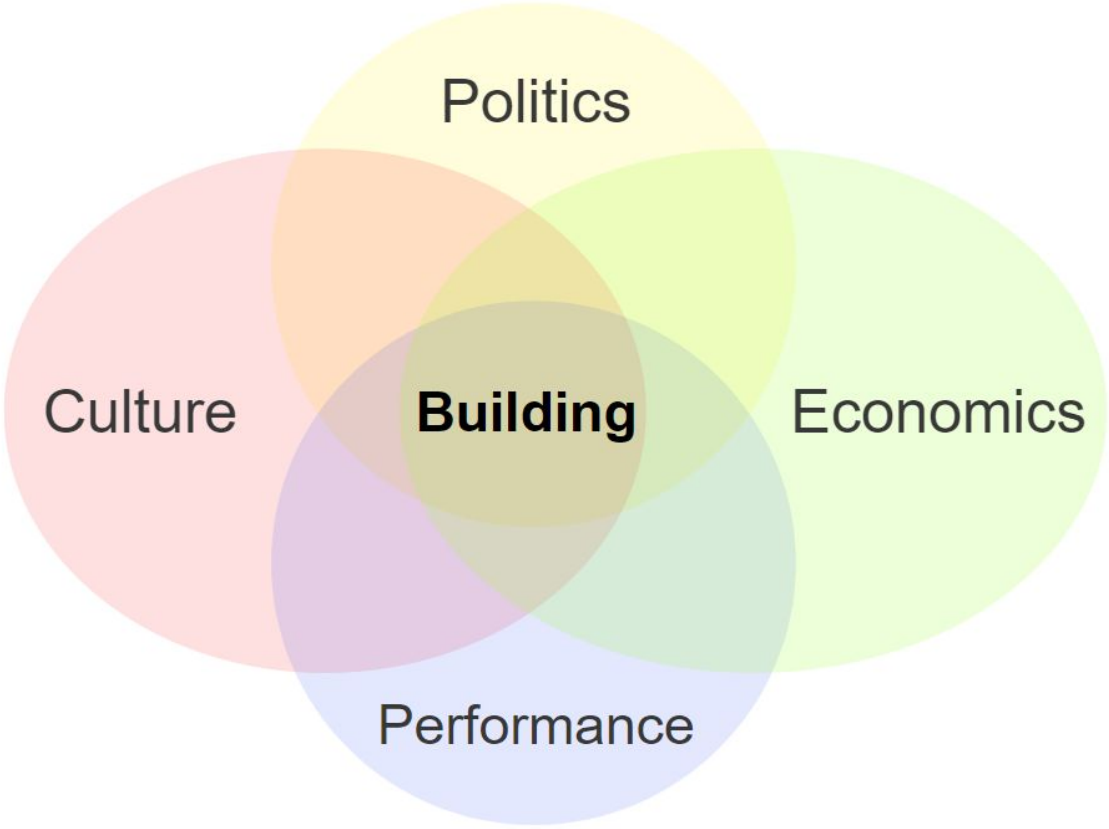
People honestly only care about the things other people do. A great example here is that Mozart, the famous composer, made a game where you roll dice, follow some rules, and the game writes a Waltz a quartet could play. Even a beginning programmer can take this, write it as a program, and produce more Waltzes than you can listen to in a lifetime. And no one cares about any of those songs. Because we as people just care about what other people have done. So even if you could fully automate something as complex as producing a building, those buildings would be totally boring and no one would really care about them very much. And even if you could fully automate the production of something like apartment complexes, it would be the people living in them (and thus customizing them!) that would actually give them any real value. Or make anyone care very much about them at all.

When we started our business we made a big deal about how we used automation to produce all the creative elements our company makes. And not one of our clients cared AT ALL about that fact. It's mildly interesting for a moment, but they honestly care that we're great to work with, are going to deliver as promised, and that they like what we do.

Buildings Aren't Cars

So why can't we build buildings the way that Tesla makes cars? Well cars are mass-produced items made of mostly totally custom parts, while buildings are totally custom items made of mostly mass-produced parts. Automation for AECO is going to look very different than it does for other industries.

Also buildings are a result of four sometimes-conflicting, hyper-local forces:



Culture is the design itself and how it relates to it's location, users, context, aesthetics, etc. There are some things here that can be generally automated, such as form generation or generative design, but it tends to be very project-specific, and what is useful now might be 'dated' in the near future.

Performance is how well the building 'works'; it's energy consumption, it's efficiency of materials, how it performs during events like fires or earthquakes. Lots can be automated here, both in the design and operation of the building.

Economics is not only how much did it cost to build, it's also it's operating costs, rents it's able to collect, cost to alter in the future, etc. While some automation might help reduce the construction or operating costs, there are large factors here (such as rents you can charge) that just aren't problems you can address via automation.

Politics is all the regulatory issues that always surround buildings. While some permit drawing production can be automated, things can vary so much from city-to-city that such automation can quickly become so complex as to be not worth it.

AUTODESK UNIVERSITY

Strategies for Surviving Automation

Want to not be replaced by robots? Here's some general ideas and tips from someone who's spent the last year working alongside robots in AECO.

Empowerment vs. Disempowerment

Do you tell the robots what to do, or do they tell you what to do? And, more importantly, are those robots working for you, or are you working for them?

For example, as a small business owner, if I automate some task, even if that automation replaces something I currently do, even if *it's now telling me what to do*, I'm still being empowered by that automation. For the extra value it creates I'll capture in some way.

Whereas if I'm working for a large company, and whatever I'm doing, or whatever my boss is doing, can easily be replaced by automation I'm much more likely to be disempowered by that. I mean, while Uber and Lyft drivers are 'empowered' to work their own hours (which is way cool) they also have little control over Uber or Lyft itself, who could easily decide they don't need that particular driver anymore for reasons that drive has no control over.

So putting yourself into positions where more automation just empowers you more is where you want to be. How can we do that?

Wicked Problems

One way to do that is by focusing on 'Wicked Problems'. These are problems that are complex, dynamic, with lots of interrelated parts, near-infinite possible solutions, and/or fuzzy 'victory conditions'. Turns out computers are just BAD at solving these sorts of problems usually, while humans are awesome at them.

A great example here I think is the game of Go. It's a complex, interrelated game of near-infinite solutions. But it's got a well-defined 'victory state' in that there are rules for the game that you can't change. So thus how 'good' a game was is something that can be measured. And thus automated. While Go remained a 'hard' problem for AI researchers, a clever machine learning team just made a Go robot that is now at grandmaster status, and it even plays in a way no human ever would. They did this by having it play countless millions of games against a copy of itself until it 'learned' to play really well. It wasn't really a 'Wicked Problem', just a really really hard one to solve until recently. And even with it 'solved' it's not like anyone is going to stop playing Go, even professionally, as Go itself is interesting enough of a game for people to feel it has value. The super-smart Go robots will likely just get their own 'league' and everyone will keep playing Go.

But if Go was Calvinball, and the rules *could* change as the game was played, this approach to automating it simply wouldn't work, for there wouldn't be a way to measure who 'won' as easily. Or if you made a Go board that was twice as big, exponentially expanding the number of possible moves beyond what can be easily computed currently. Or if you added some more social rules to Go, like bidding in Poker, where you have to tell if someone is lying or not. Or if

Go was more complex in how the pieces interrelated, in that some pieces change in relation to how other pieces are played.

So focusing on ‘wicked problems’ is one way to stay relevant no matter what happens with automation. Design, engineering, and construction are FULL of such problems. Become an expert in one, such as local building codes (and the politics within), or software optimization, or cost-efficient structural design for a particular complex building type, or in existing facade renovation for mid or high-rise buildings (which are ALL going to need it in the next 20-50 years) and even if a robot comes along, it won’t be able to keep up...

Soft Skills

The other way to ‘future proof’ yourself is to learn some ‘soft skills’.

People like to hire their friends. Even when all else isn’t equal. So gaining some skills in sales or other customer-facing area is a sure way to stay relevant.

People like to work with their friends. So learning some great management skills to get the most out of the people you work with, and who work for you, is another great way to stay relevant.

And robots really don’t buy anything. So understanding how to get people’s attention, communicate a story, and some marketing basics is another great soft skill to have.

Robots are terrible business people. Learning how to be a business-focused, value-creating, creative problem solver is another way to stay relevant.

Finally, robots are bad at coming up with things the world’s never seen before. A/B testing would have never designed the first iPhone, for example. So gaining some design skills, learning what people really would want, and how to deliver on that will make you irreplaceable.