CES501766

A Case Study for Generative Design in Horizontal Infrastructure

Stacey Morykin
Gannett Fleming, Inc.

Mark Jaffee, E.I.T
PGH Wong, Inc.

Learning Objectives

- Discover operational efficiencies by combining computational design and optimization.
- Learn about how generative design could benefit project teams.
- Learn how to establish a Generative Design process for your next project.
- Discover ways to accelerate your design schedule.

Description

Have you ever spent time iterating conceptual designs of a project to realize it's not the preferred layout, or, even worse, it's not the optimal design? Ambitious targets for high-quality, resilient, sustainable infrastructure require accelerated digital transformation. Join our case study that examines the use of generative design in the horizontal infrastructure space, specifically applying it to the layout and design of a rail maintenance facility. By combining the use of Civil 3D software for optimization and Dynamo for computational design, we'll explore solutions to complex design problems—helping to make a more informed final design decision closer to the beginning of the project. By the end of the session, you'll see that, combined with setting parameters, design automation tools reduce repetitive manual work and demonstrate operational efficiencies, allowing more time to innovate and focus on design accuracy and quality while improving business growth and profitability.

Speaker(s)

Stacey Morykin works with AEC professionals to enhance their workflow through innovative technology. After spending nearly two decades in design, development, implementation, support and management of computer and non-computer-based technologies, Stacey truly understands the importance of communicating product design intent and constructability. Stacey is currently a Certified Professional in the current version of Civil 3D, an Autodesk Expert Elite in Civil Infrastructure and holds an Associate Degree in Computer Science. Her most recent accomplishment was "Top Speaker" at AU 2019 for her hands on Lab on Dynamo for Civil 3D.

Mark Jaffee is an entry level engineer with three years of experience in rail, utilities, and site civil design. He has an extensive knowledge of Dynamo and enjoys creating automations that streamline the design process. Mark currently works for PGH Wong modeling utilities and providing constructability reviews on the California High Speed Rail project. Mark graduated with a Bachelor’s Degree in Civil and Environmental Engineering from the University of California, Berkeley in 2019.

Introduction

This case study is intended to provide a story of how we used computational design to generate multiple outcomes. It is meant to focus on our thoughts and theories throughout the process and provide a little insight into where we are currently at with the project. We'll be learning what worked well for us during this process and what we could do better moving forward.

The Challenge

Site layout is a complex problem. Most times it feels much like fitting a square peg in a round hole. We as engineers and designers spend hours moving and rotating geometrical features like buildings, parking lots, and basins to provide our clients with multiple options for the final design of their project. This iterative process does help the engineer and client understand what might be possible on a given site, but these iterations are time and resource-intensive. Add any additional complexity to the site layout like rail constraints and the iterative process becomes a daunting task. One radius or slope of the rail alignment needs to change and it could have a significant negative impact on the project budget. We at Gannett Fleming started to consider the possibility of automating this process through the use of computational and generative design. We could visualize a number of outcomes including a modular framework for using generative design on other projects, preventing wasting of time during generation of these layouts, and maybe even hint at a final layout that would work closer to the beginning of the project allowing us to spend more time understanding the design criteria and our clients needs.

What is Generative Design?

"A generative design approach involves several steps with distinct types of algorithms that encode logic in each step. The methods used in each step are very different from each other, so we generally categorize them in the following way:

Step		Algorithm Type
Generation of new design studies	>	Generators
Evaluation of each study	>	Evaluators
Ranking and solution achieves goal	>	Solvers

The major principles are:
G – Generate
A – Analyze
R – Rank
E – Evolve
E – Explore
I – Integrate

From the Generative Design Primer, published by Autodesk.

Where Do We Start, How Do We Start?

Designing a rail yard has many different parameters, so how do we eat this elephant? We needed to start by analyzing how the problem is currently solved by engineers. With this understanding, we could break down the design process task by task and identify opportunities for automation on an individual task level. The thought was that some of these tasks could be good candidates for a generative design program.

Business Process Analysis

Once we were done breaking down the design process into tasks, we organized them into swim lanes. As each task was placed into a lane, we considered if the task could be automated and if the automation would improve the current business process. Breaking down the problem into this structure would help us to:

1. Communicate the approach to other members of the team.
2. Identify pain points in our process
3. Determine which steps in our process we could automate.

This graphs needed to;

1. Identify tasks appropriate for automation
2. Understand the flow of information needed for the scripts: inputs and outputs
3. Clearly identify the ‘low hanging fruit’

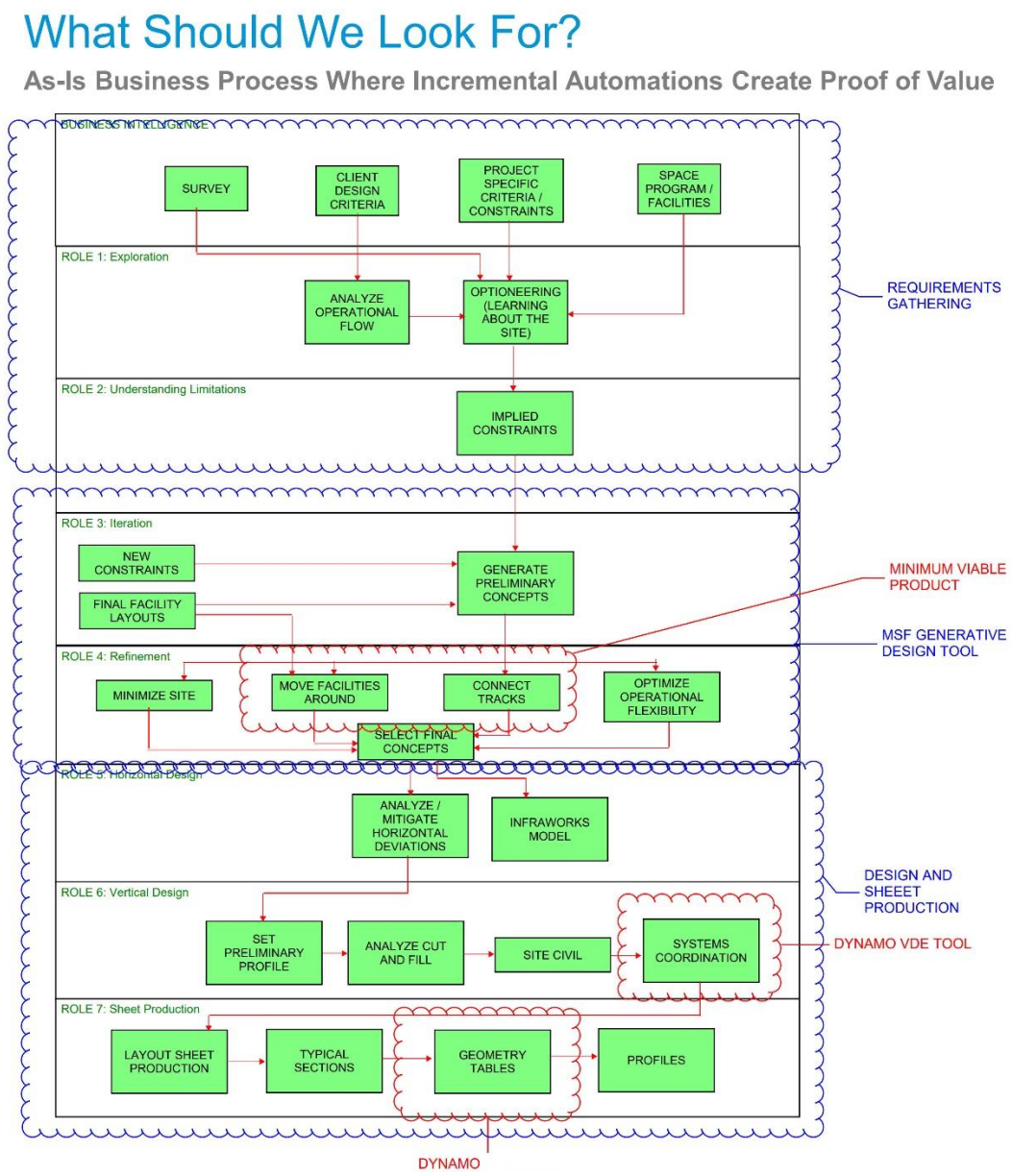

Figure 1: Design Process for Rail Yard Layout

Identifying tasks appropriate for automation

From our business process graph above, we saw that Role 4: Refinement took the most amount of time in our process. Every time a facility was moved, the tracks needed to be re-connected. If the tracks did not connect, the facility needed to be adjusted. This process is repeated until a feasible design is created. Doing this manually was the most tedious and repetitive process in our design. Engineers could spend hours exploring a concept only to find that the geometry does not work with the site constraints. This consumed a large part of the design effort and engineer's time in the early stages of the conceptual design.

When we looked at our design process, we saw a variety of automation challenges. Some tasks, such as gathering design requirements from design standards or producing sheets for deliverables required input from the engineer and just seemed more challenging to automate. We also identified other areas of our design process where we could use Dynamo to help automate mundane tasks such as creating Geometry Tables and vehicle dynamic envelopes. However, Moving facilities and Connecting Tracks were both iterative and time consuming; these are the real meat and potatoes of the generative design tool and computational design.

Computational design is a design method that uses a combination of algorithms and parameters to solve design problems with advanced computer processing. Every step of a designer's process is translated into coded computer language. The software program uses this information alongside project-specific parameters to create algorithms that generate design models or complete design analyses. Once the initial programming is completed, design becomes a dynamic and repeatable process.

Understand the flow of information

One of the benefits of using a graphical structure like swim lanes is that we could see how information flowed from one task to another within the design process. If the generative design tool was capable of Moving Facilities and Connecting Tracks, then it would have to use the same sources of information that an engineer would use to solve the design problem. This would be a challenging issue. Providing this information to the computer was like explaining what makes a good site design to someone that speaks a different language.

Criteria for site layouts comes in a variety of different forms. Anything from geospatial boundaries to rail criteria could play an important role in a good design. Maybe the client prefers a building to face a certain direction or access along a street is limited. Perhaps the site must minimize the number of turnouts, site area, or linear feet of track. How do we make this information accessible to the generative design tool? We didn't really know yet how to translate some requirements into Dynamo logic and math. This was okay at this stage in the process. At least the business process diagram had exposed this part of the problem.

Identifying the "low hanging fruit"

Our business process analysis revealed just how daunting a challenge automating this design process would be and we were overwhelmed with what to do next. We knew that we had a lot to learn – using generative design in Dynamo and automating parts of our design process. Therefore, it was decided that development would begin with a simple automation. We wanted to pick a development path that could create value for the organization at different milestones while also avoiding pitfalls and dead ends.

To lower the risk of the first generative design automation, the task that was picked for development had parts that were already solved by the Autodesk Community in public forums. This platform is a great way to jumpstart development and not re-invent the wheel.

Problem 1: Moving Facilities

After the business process analysis was completed, our efforts shifted to creating a proof of concept that would teach us about generative design and solve an easy problem. Therefore, we had decided that we were going to begin development on the moving facilities around the site program. Buildings are the most cumbersome elements of the design, and their orientation has a big impact on the operational flow of a layout.

Moving facilities around the site would also be a great minimum viable product (MVP). An MVP is a half-baked concept that has the potential to create value for the company. The idea was that if moving facilities could not be developed enough for rail yards, another business line could benefit from implementing the technology on a similar project. Several AEC industries, such as industrial and systems engineering, have a need to find optimal layouts of facilities/equipment within an area.

Creating Our First Generative Design Script

At first, creating a generative design script was an up-hill battle. We had a lot to learn and not a whole lot of time for things to sink in. We began learning about generative design by reviewing what we know about Dynamo. How was generative design different? What kinds of things did we have to change for a script to work? First of all, generative design programs are not structured like typical dynamo scripts. Whereas Dynamo scripts can be organized in any way, generative design scripts follow a more rigid structure.

Figure 2: Typical Dynamo script processes

When analyzing the differences been Dynamo and Generative Design processes, we had to learn about a few new concepts:

1) Input Geometry and Generators
2) Data.Remember (node)
3) Define the Inputs / actions based on inputs
4) Metrics

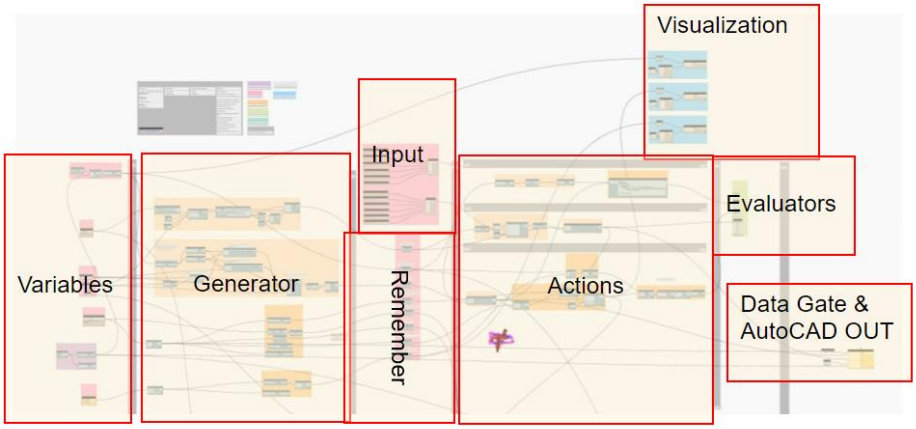

Figure 3: Generative Design Script Processes

Input Geometry and Generators (Dynamo Geometry)

Generative Design in Dynamo works by manipulating Dynamo DesignScript objects. For this reason, any autocad objects used in the script needed to be first translated into Dynamo objects. For example, architectural building footprint polylines were converted into objects like polycurves. We had to really think about what data we were providing to the generative design tools, and if it was really integral to solving the problem. Certain things like door frames and equipment layouts inside the buildings were omitted because they did not influence the overall site layout design. Certain areas such as waste disposal areas and outdoor work areas were included because they altered the footprint of the site and were important to the facility's functions.

Data.Remember:

Special nodes called Data.Remember nodes tell the Generative Design tool what to keep in mind when running a generative design study. If geometry needed to be moved, or visualized in the generative design tool, it needed to pass through a data.remember node.

Define the inputs / action based on inputs

To understand the necessary inputs for the generative design tools, engineers first considered the ways in which building layouts are manipulated on a site. An engineer may mirror, move, or rotate a structure footprint in any number of ways to get the design just right. These actions could be accomplished by using a boring geometry.rotate or geometry.translate node. Other engineers may want to do something more complicated, such as toggle a dynamic block to explore different design possibilities. This would require a more complicated consideration of what geometry to remember and what to specify as an input. The hard part of this exercise became thinking about how a computer can be provided with a means to do these same manipulations in the generative design tool.

Evaluator (Metrics)

How do we know what a good design looks like? How can we measure it? We needed to code a way for the computer to refine and optimize the possibilities to provide meaningful results. We called this part of the code a metric, as in code that would quantify and measure an observable property of the design. Metrics could be anything important to the final site design. Maybe the client wanted a certain area of lay-down space, a certain number of parking spaces, etc. Perhaps the engineer wanted to minimize curves or maintain clearances to other facilities. We started with a few metrics that would be able to sift out undesirable site layouts such as:

Requirement	Evaluator
All buildings fit on site	# intersections with site boundary = 0
Buildings can't overlap with each other	#intersections with building boundary = 0
Rail Specific	
Buildings should be on the same bearing / angle	# of unique items in a list of building rotations

Table 1: Sample Evaluators

Picking A Solution Method

Dynamo's generative design tool uses the NSGA II algorithm to solve studies. The different methods available for solutions include optimize, cross product, randomize, and like this.

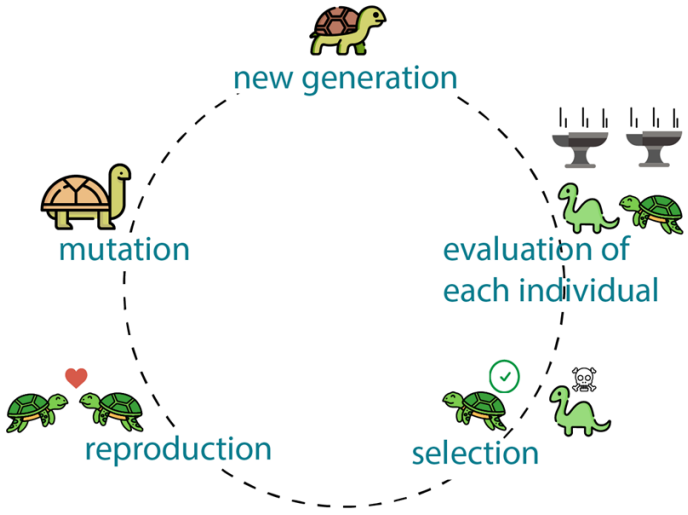

Figure 4: NSGA II Algorithm Solution Processes
Credit: Generative Design Primer by Autodesk

For our proof-of-concept analysis, we chose randomize as the solution method as we were interested in all the possible arrangements of buildings on a site. If our design had more site constraints such that we knew where some buildings needed to be placed, optimize may have been a better choice. The solution method we chose depended largely on the current stage of the design process for the site.

Solution Method	Description
Optimize	Target specific inputs and attempt to converge to a solution specified by the user
Cross Product	Shows all possibilities (all combinations of all parameters)
Ramdomize	Vary the inputs and present all solutions
Like This	Explore variations of a solution that you already have.

Table 2: Generative Design Solution Methods

Failure

Initial results from the generative design tool were a total fluke. Just getting everything to work properly and validate had been a big achievement, but we had new problems to contend with. One issue was that all the results seemed identical. It was difficult to identify how the designs were different and if there were any meaningful unique possibilities. This was caused by input ranges that were too large. For example, rotation could be an integer between 0 and 360. We don't need generative design to show us a rotation of 50 degrees vs 51 degrees, just a general idea of what works on site is really what we wanted. The same issue came up when it came to displacements in the X and Y directions for facilities.

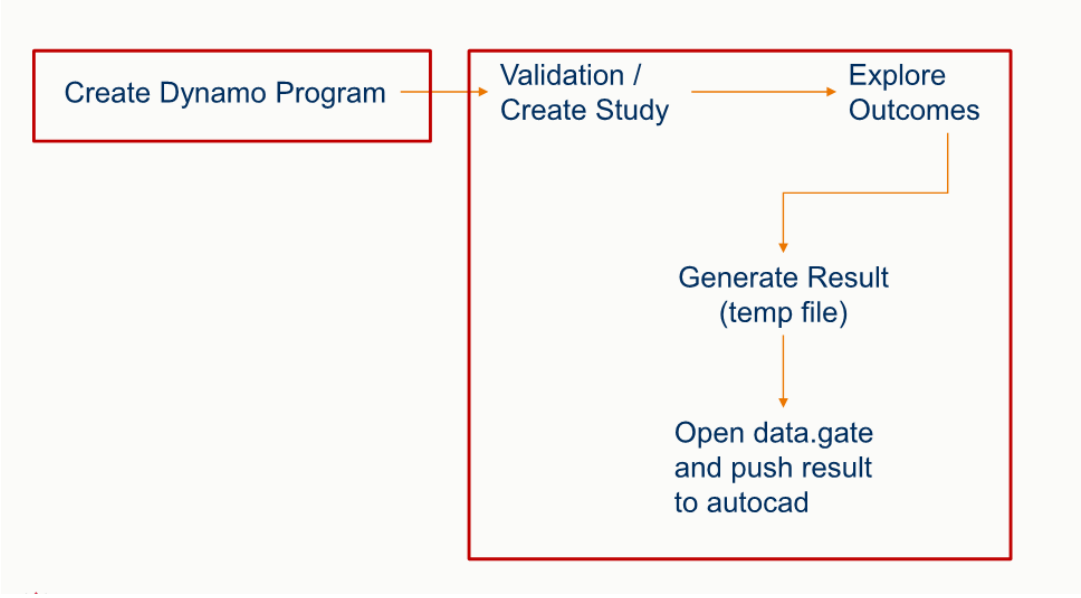

Figure 5: Generative Design Workflow

Refinement: The Fix

We needed to take some generative design power away from the tool for better outcomes and at the same time not over constrain the sketch. Before reworking the script we took some time to think about our ultimate goal and the program's use in rail design. Facilities

are usually arranged on rail sites on a common bearing or angles that are compatible with track turnouts. This helps eliminate any unnecessary curves and complications in the track design. This angle constraint also creates an angled grid of intersecting lines. These intersections could be possible facility locations. The next challenge would be translating these constraints into inputs for the generative design tool.

To constrain the amount of possible rotations, engineers imposed a mapping function on the rotation input. Any number between 0 and 1 would map to multiples of the turnout angle. In this configuration, small changes in the input value would translate to larger rotation changes. For example, 0 maps to 15 and 1 maps to 30. To constrain the displacement of facilities on the site layout, a mapping similar to the rotation mapping would work. Any number between 0 and 1 would map to a list of points where the angled grid lines intersect. After these changes, the initial results seemed promising. The differences in generative design results were evident and the filters were working as promised.

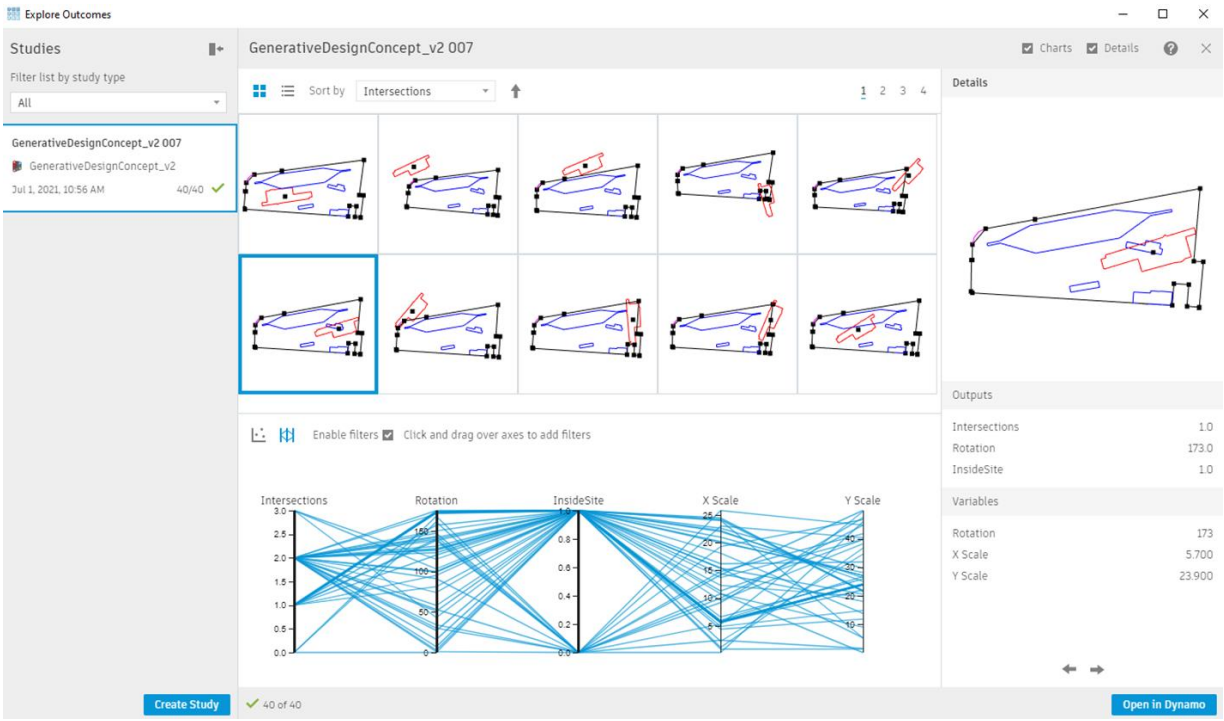

Figure 6: Generative Design Study Outcomes

Problem 2: Connecting Tracks

While the generative design results improved, their convergence to a final conceptual design was coming up short. In the design of rail facilities, the connection from one site facility to another site facility describes the operational flow of the site. Some transit agencies include an operational segment in their design criteria, which explains the typical operations of transit vehicles moving in and out of the site for service. While the generative design tool could move buildings around the site, it had no way of associating a layout with an operational flow. Options created by the generative design tool could meet the metric requirements, and even still-it could not operate as a functioning facility. Where do we go from here?

In the business process analysis, moving facilities around on the site and connecting tracks seemed like tedious and mundane tasks ripe for automation. Generative design could move around facilities but connecting tracks seemed like a much harder challenge. We knew that engineers spent a considerable amount of time reconnecting all the tracks between facilities with the proper radius, track spacing, and clearances. How do we teach a computer to understand how facilities should be connected and do the same?

We did not really understand how to approach this problem from a computational design standpoint. There were too many things to consider such as translating design criteria into logic, creating site geometry, converting entities to generative design, and displaying results. We decided to go with a different approach for this part of the challenge. Engineers would create a Dynamo script (not generative) that would work as a design aide for other engineers. Once the inner workings were better understood, the design aide would be integrated into generative design. In creating the design aide, two distinct problems were identified: understanding operational flow and creating desirable geometry.

Operational Flow

Rail facilities typically have a sequence of operations that need to be performed; for example, dispatch from service, arriving from service, as well as maintenance and inspection. A good site design allows these movements without interfering with other site operations. But, how would a computer understand how facilities should be connected together on a site? To solve this problem, two different approaches seemed possible:

CONNECTIVITY MATRIX	CAR WASH	CLEANING PLATFORM	STORAGE	MAINTENANCE	INSPECTION 1
CAR WASH		X			X
CLEANING PLATFORM					
STORAGE				X	
MAINTENANCE	X		X		
INSPECTION 1					

Figure 7: Connectivity Matrix

Connectivity Matrix

The connectivity matrix was an excel sheet that could be interpreted by Dynamo. Facility names would populate the first row / column to create a square matrix of origins/destinations. If a cell was marked with an X, the corresponding origin/destination pair would tell Dynamo which facilities to connect. This seemed like a great option because engineers were already familiar with Excel sheets. On the other hand, large sites resulted in excel tables that were large and confusing. If a connection needed to be changed, it was easy for the engineer to populate the wrong cell and end up with the wrong outcomes.

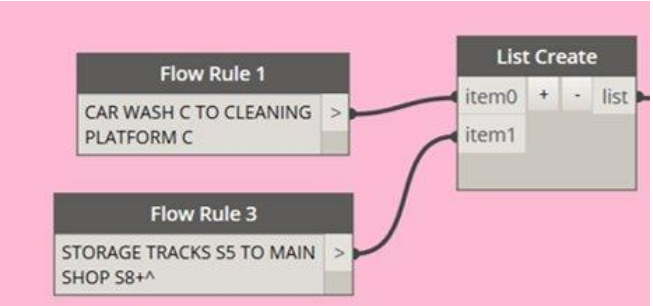

Figure 8: Connectivity Rules

Connectivity Rules

Connectivity rules used string sentences to describe how the facility should be connected. Dynamo would parse out the origin / destination facility names and attempt to connect the buildings together. While mistakes were less common, writing out all the possible connections was time intensive and the coding was much more difficult than the Connectivity Matrix approach.

Connectivity rules were chosen as it most closely mirrored how the information was presented in design criteria manuals. We thought that this would be a sound strategy as this would make the tool easier for engineers to understand and therefore help-in others using the tool. A tool can be really cool, but a bad user experience can make it that much harder for others to use and adopt a new approach.

Creating Facilities

Dynamo needed to somehow understand that lines in modelspace represented a rail facility with track connections. We realized that lines and layers alone were not enough to give Dynamo everything it needed. What we could do is use AutoCAD'S block architecture to provide Dynamo with the information it needed to connect facilities within the site. Dynamo would need to know the following about a given site:

1) What tracks connect to the site
→ Set tracks on a “RAIL-CL” layer
2) What is the outline of the facility
→ Only include the building outline, omit all else
3) What is the rotation of the facility
→ Define the block relative to the UCS
4) Where should the track begin
→ Use a line vector with a start and end point
5) What direction should rail stock travel on the site
→ Use a line vector with start and end point

Automations are only so flexibile with their inputs, and strange input is likely to confuse the Dynamo program. As we prepared more projects for use with this connecting tracks tool it became evident that junk in = junk out. We needed to take the time to groom and clean our files before attempting to use them with an automation. This

ended up improving the quality of our finished product and drew attention to some very bad habbits.

Creating Track Geometry

How do we begin to think of a computational approach to connecting track geometry together? We couldn't know how each building would be oriented in generative design, so we needed to solve 5 different possible connections for each facility (start to start, start to end, end to end, end to start). The path that resulted in the shortest length and fewest intersections with site geometry was chosen. Even then, we couldn't get the tool to solve for all possible conditions. Sometimes the desired connection was not the shortest path, but rather a connection that aides in the overall site flow.

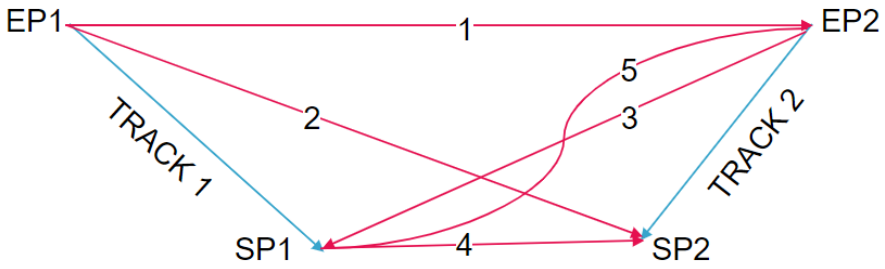

Figure 9: 5 Possible Track Connections

As we started to learn more about the problem, we realized how complex of a challenge connecting tracks would be. It is not all about geometry. We could code something that connected facilities together, but it would be difficult for the program to understand connections that also help with the bigger picture of overall site flow. To give engineers the possibility to choose which solutions they wanted, delimiters (such as *, + , &, etc.) were added to the connectivity rules. These delimiters would drive the program to solve for a connection some other way than the shortest path. We hoped that by studying how engineers utilize the delimiters, a computational approach would be revealed.

Solving each curve connection five times is a very computationally intensive approach toward this problem. Small changes in site geometry would cause a latency of 5-10 seconds every time a change was made. While this technology could assist engineers now with the mundane task of connecting facilities, it had a long way to go before being ready for Generative Design. Therefore, it was decided that Connecting Tracks would be deployed as a design aide first and Generative Design ad-in later.

Lessons Learned from the Design Aide

Connecting tracks became a design aide instead of building the program in Generative Design. Even as a design aide, it was tremendously valuable. An engineer didn't have to spend all day filleting curves. No more wasting time on a design that will never work with the site. So much wasted time and effort is saved. It didn't matter if you rotated or moved one

building...even the whole site. Maybe the engineer wanted to set a turnout, or optimize a simple ladder. The tool would maintain the connections to all facilities by running Dynamo on an "Automatic" basis. Beyond its functionality, connecting tracks also facilitated important discussions that would impact future generative design development. We had to consider things like the user interface, file preparation for automation, and ease of use. We started to think more about how development now can positively influence adoption elsewhere within our organization later.

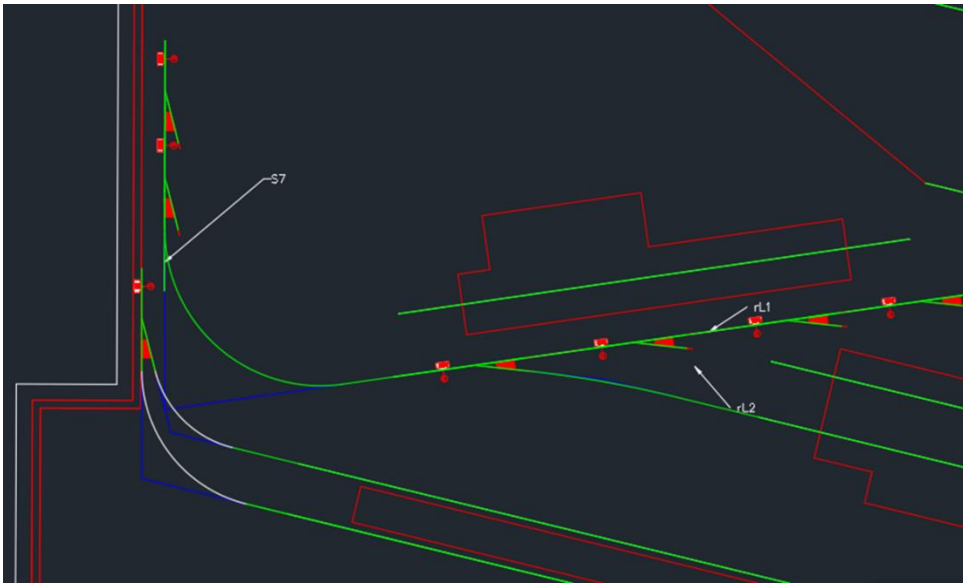

Figure 10: Connecting Tracks Run On A Rail Yard Layout

When the design aide was ready, it was implemented in a few projects and some issues would have to be resolved before advancing the program to generative design. Depending on the inputs, generative design studies take a while to process. Future scripts could use Dynamo geometry and vectors, which are easier for Dynamo and generative design to process. The overall solving approach would also have to be re-worked before implementing into generative design.

Another issue faced was preparing the project drawings to work with the design aide. While engineers placed a lot of effort in making the user interface for the tool simple and easy to use, detailed and time-consuming work was needed to prepare the data in a way that the program expects. Engineers created a series of Dynamo scripts to assist users in setting up the blocks and tracks necessary to operate the design aide. Training and hands-on instruction with the design aide also helped in educating the user on the limits and possibilities of the design aide. Future development could mitigate this issue by having programmers and users coordinate more closely at early stages of the program's development.

Future Development

Where We Are At and Where We Are Going

Currently we have two scripts that work well separately:

- Move facilities: Utilizes the generative design tool to move building footprints around on a site.
- Connect Tracks: A Dynamo design aide that connects tracks in a layout drawing.

These scripts work great by themselves, but there are some technical hurtles and challenges to overcome before they can be integrated into one program. Eventually we wanted a script that can move buildings and facilities together that inspires feasible layouts. This generative design script would reveal all the possible layouts within a site and show how facilities could be connected.

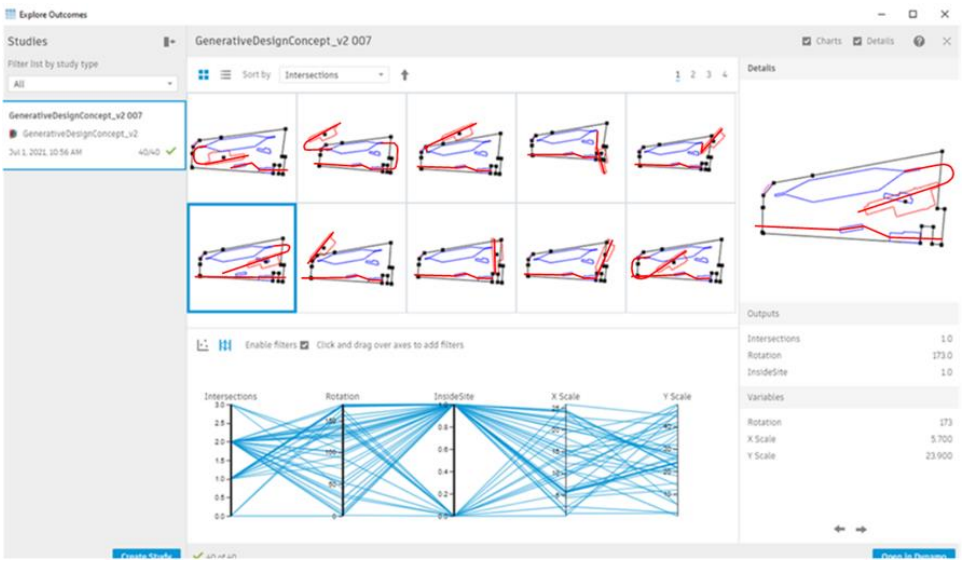

Figure 11: Generative Design Tool with Connecting Tracks and Moving Facilities

It is important to let findings speak for themselves. Creating a dynamo script for horizontal design in the conceptual design phase was dooable. However, getting something to be closer to the 5%-10% design phase would be very difficult due to all the complexities that would have to be coded into Dynamo for a rail yard site design. We can do is show engineers some possibilities without having them spend hours and hours of time figuring it out.

We had envisioned a tool that would automate the entire iterative process. What we found was that generative design would complement some parts of that iterative process. Without significant time and resource investment, there are some problems so complex that they are better left to the engineer. We now know where generative design can impact our design process and have the scripts to support it.

This could just be the start for using generative design in other areas of our business. Connecting tracks and moving facilities could be packaged into custom Dynamo nodes. These nodes could be shared company-wide to encourage the use of generative design and achieve a modular approach to computational design problems. This is just the start of an exciting disruptive technology.