# Using Evidence-Centered Design to Develop Automated Noticing of Students' Engineering Design Goals

James P. Bywater

Dissertation Defense

Paper Two

## Abstract

Engaging students in engineering design has been shown to lead to improvements in both students' mathematics and science understanding, as well as their self-efficacy in these subjects. Research suggests that engineering design may be particularly engaging for students typically underrepresented in science or mathematics. However, implementing learning activities that incorporate engineering design practices presents challenges because in order for teachers to provide effective guidance to students about their designing practice, teachers need to understand and be skilled at noticing the design processes that each student takes. This study leveraged the log data collected while students used Energy3D—a computer-aided design (CAD) and simulation software specifically designed for K-12 students to construct buildings and analyze their energy efficiency. Drawing upon the Evidence-Centered Design framework to make inferences about student performance, this study (1) used machine learning data analytic techniques to analyze the Energy3D log data for patterns of student behavior; (2) qualitatively analyzed student think-aloud descriptions of their design behavior to find emergent patterns; and (3) investigated the alignment between the two analyses to explore the validity of inferences made by the machine learning methods about student design behavior. Our results identified four clusters of goal-oriented student design behavior that were validated with students' think-aloud data. Results provide validated evidence for developing automated ways to capture students' design goals that may eventually be used to help provide guidance to students. In addition, the method used in this study has the potential to be applied to other settings that generate sequences of fine grain-size actions such as mouse-click level log data such game-based learning contexts.

*Keywords:* evidence-centered design, noticing, automated, engineering design

## Introduction

Engineering design involves applying mathematics and science to solve real-world problems. Engaging students in engineering design has been shown to lead to improvements in students' mathematics (Chiu et al., 2013; Burghardt, Hecht, Russo, Lauckhardt, & Hacker, 2010) and science understanding (Klahr, Triona, & Williams, 2007; Kolodner et al., 2003; McBride, Vitale, & Linn, 2018; Schnittka & Bell, 2011), as well as students' self-efficacy about mathematics and science (e.g., Cantrell, Pekcan, Itani, & Velasquez-Bryant, 2006; Plant, Baylor, Doerr, & Rosenberg-Kima, 2009). Research suggests that engineering design may be particularly engaging for students typically underrepresented in science or mathematics (e.g., Han, Capraro, & Capraro, 2014). Engineering design concepts and practices such as defining problems, optimization, and developing solutions have also been included alongside scientific practices within the Next Generation Science Standards (NGSS; NGSS Lead States, 2013).

However, implementing learning activities that incorporate engineering design practices presents challenges (e.g., Kolodner et al., 2003; Moore et al., 2014). In order for teachers to provide effective guidance to students about their designing practice, teachers need to understand and be skilled at noticing the design strategies that each student takes (e.g., Crismond & Adams, 2012). For example, teachers looking to help students troubleshoot or revise designs need to understand students' individual designs, help them plan and conduct systematic tests, and use data from tests to inform how the design should change. Moreover, each student will likely have a unique solution instead of one "right" answer, and the design goals that lead to that solution are likely to differ. Noticing the design behaviors (i.e. the design practices, strategies, and goals) of each student in a classroom is complex and challenging (Purzer, Moore, Baker, & Berland, 2014), especially as the majority of precollege teachers currently have little experience with engineering design.

Research identifies various ways to document or capture design behaviors in the moment such as think-alouds (e.g., Gero & Tang, 2001) or reflections (e.g., Chen et al., 2005). These methods are typically labor intensive and are usually implemented in undergraduate or professional settings (e.g., Cross, 2011), and are thus not as applicable in precollege classrooms. Consequently, without tools that help capture students' design behaviors, teachers are more likely to focus on the features of the final solutions rather than the designing practices (Wang, Moore, Roehrig, & Park, 2011).

Computer-aided design (CAD) environments often used in engineering projects provide opportunities for students to engage in design behaviors, while also enabling the collection of rich data about student actions. This data has the potential to be leveraged to identify engineering design patterns for teachers (e.g., Worsley & Blikstein, 2014). However, typical CAD software environments are not themselves designed for learning; that is, CAD software tools are primarily developed for engineering functionality, not educational functionality.

This study leverages log data collected while students are using Energy3D, a computer-aided design (CAD) program specifically designed for and implemented in K-12 educational settings (Xie, Schimpf, Chao, Nourian, & Massicotte, 2018). Energy3D enables students to construct energy efficient solutions with a variety of structures such as buildings, homes, and solar farms. Students can analyze a variety of performance variables, and learn and reflect upon earth and physical science principles through tutorials and reflective notes. In addition to serving as a CAD and simulation engine, Energy3D collects fine-grained information on student actions.

Leveraging the Evidence-Centered Design framework (ECD; Mislevy, Almond, & Lukas, 2003) to draw inferences about student performance, this study aims to: (1) use machine learning data analytic techniques to analyze the Energy3D log data for patterns of student action; (2) qualitatively analyze student think-aloud descriptions of their design behavior to find emergent patterns; and (3) investigate the alignment between the two analyses to explore the validity of inferences made by the machine learning methods about student design behavior. By exploring the validity of the machine learning analysis in this context, we work towards developing automated ways to capture students' design behaviors in the moment that can eventually be used to help provide guidance to students.

## Background

### Evidence-Centered Design

To frame the characterization of students' design practices, this study draws upon the Evidence-Centered Design framework (ECD; Mislevy, Almond, & Lukas, 2003). ECD views assessment as the means of making inferences about student abilities from students' demonstrated performance on tasks. The ECD framework has been used to facilitate assessment of a broad set of abilities from traditional assessments that focus on student knowledge to performance assessments that focus on a variety of student skills and practices. For example, the ECD framework has been used in open-ended simulations and games to create valid measurements of constructs such as problem solving, causal reasoning, and systems thinking (Shute, Ventura, & Ke, 2015; Shute & Kim, 2011; Shute, Masduki, & Donmez, 2010).

At the core of the ECD framework are the student, evidence, task, and presentation models (see Figure 1). The *student model* answers the question "What are we measuring?" and defines the variables or constructs we wish to measure, how different variables may relate to each other, and the values of the variables for a given student. For example, in this study we are looking to characterize students' design behaviors such as generating, testing, and revising solutions. The *evidence model* answers the question "How do we measure it?" and describes how we should update the values of the variables we are measuring based on what is observed within a task, or what behaviors reveal the constructs targeted within the student model. For example, students using analysis tools in Energy3D to predict the energy performance of buildings would constitute evidence of students' testing and evaluating design behavior. The *task model* answers the question "Where do we measure it?" and describes the task including the necessary features that are necessary for a user to interact with. In this study, students are challenged to build a house that uses zero net energy over a year. Finally, the *presentation model* answers the question "How does it look?" and describes the interaction interface and the tools available during the assessment. In this study, the presentation model is the computer-based Energy3D program. These models can each be developed iteratively but need to be aligned and calibrated for valid inferences to be made (Mislevy et al., 2003).

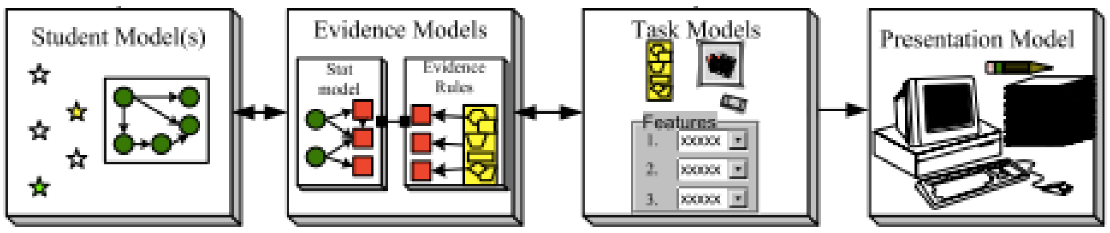


*Figure 1.* The core Evidence-Centered Design (ECD) models (taken from Mislevy, Almond, & Lukas, 2003).

**The student model: Engineering design behaviors**

Since we are seeking to measure *engineering design behaviors*, our student model will draw upon the Informed Design Matrix, which describes the different strategies of beginning and informed designers (Crismond & Adams, 2012) and which overlap with the Next Generation Science Standards (NGSS) engineering practices (NGSS Lead States, 2013). For example, in Informed Engineering Design, students are given opportunities to engage in problem definition by understanding the specifications and constraints of a design challenge, to research and investigate knowledge that is needed for the challenge, to generate different solutions, to test these solutions, and to use these evaluations to inform subsequent design cycles (see Figure 2).

| **Design Strategies** | **Beginner** | **Informed** | **Example behaviors indicative of design strategies** |
|---|---|---|---|
| **Problem definition** | Treat design task as a well- defined, straightforward problem that they prematurely attempt to solve. | Exploration of the problem space, defining criteria, specifications, constraints. | Asking questions about project criteria, specifications, or constraints within the net-zero home energy challenge (e.g., what does net-zero energy mean?) |
| **Build knowledge** | Skip doing research and instead pose or build solutions immediately. | Do investigations and research to learn about the problem, how the system works, relevant cases, and prior solutions. | Conducting investigations within Energy3D on solar energy, incident angle, solar panel efficiency, etc. |
| **Generate Ideas** | Work with few or just one idea, which they can get fixated or stuck on, and may not want to change or discard. | Practice idea fluency in order to work with lots of ideas by doing divergent thinking, brainstorming, etc. | Building multiple homes with different numbers of windows, different styles of roofs, different foundation/interior layouts. |

| | | | |
|---|---|---|---|
| **Testing Designs** | Do few or no tests on prototypes, or run confounded tests by changing multiple variables in a single experiment. | Conduct valid experiments to learn about materials, key design variables and the system work. | Test effect of window placement on energy performance by changing one window to different sides of house while keeping all other aspects constant.<br>Test effect of solar panel placement by changing one solar panel to different sides of the roof while keeping all other aspects constant. |
| **Revising Designs** | Design in haphazard ways where little learning gets done, or do design steps once in linear order. | Do design in a managed way, where ideas are improved iteratively via feedback, and strategies are used multiple times as needed, in any order. | Immediately following window test, design is revised to put windows in optimal place.<br><br>After running energy performance analysis, designs are revised to optimize a particular criteria |

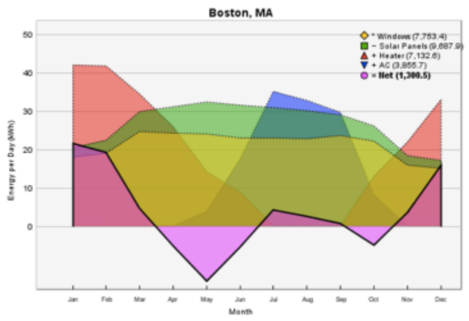
*Figure 2*. Design strategies that comprise the student model for this study (adapted from Crismond & Adams, 2012) and description of behaviors indicative of those design strategies within Energy3D (similar to an evidence model).

## The presentation model: Energy3D

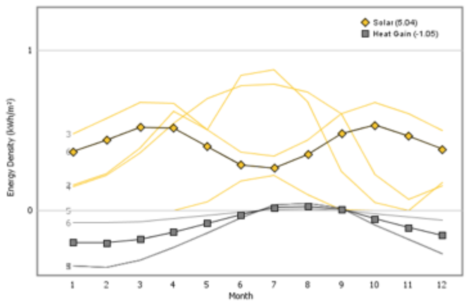
Energy3D provides students with a CAD environment that allows students to both build and test different designs (Xie, Schimpf, Chao, Nourian, & Massicotte, 2018). With Energy3D, students can design buildings that incorporate solar panels in order to meet design challenge specifications and constraints. Students can use the embedded simulation tools to examine energy gains and losses under various conditions to help students understand concepts such as energy transfer and solar radiation as part of design (see Figure 3).

House design

Solar irradiance heat map for June



Energy use graph



Sensor graph

*Figure 3*. Sample screenshots from Energy3D (Xie & Nourian, n.d.).

When engineering design projects are implemented in pre-college settings, students rarely iterate on their designs, and have little opportunity to visualize, test, and redesign due to time constraints (Katehi, Pearson, & Feder, 2009). Energy3D addresses this challenge by providing students with a CAD environment where students can rapidly iterate on their designs, test the performance of their designs with a click of a mouse, and visualize the performance of their design with multiple representations. Energy3D enables students to visualize and analyze their designs throughout different times the day (sun path in the sky), throughout the year (sun height in the sky), at different locations throughout the world by using real data gathered from the National Oceanic and Atmospheric Administration. As such, Energy3D has been used by a variety of researchers, teachers, and even industry members.

## The task model: Net-zero (or less) Energy House Challenge

This study is situated within a project where students were challenged to design a house within Energy3D that would use less energy to heat and cool the house than would be generated from solar panels. The students were given constraints that were chosen so that the challenge was neither too easy to meet nor too difficult. The constraints were:

- **Energy efficient:** Consume no net energy over a year (try to make the Annual Net Energy as negative as possible).
- **Cost:** The house should not cost more than $150,000.
- **Size:** The house should comfortably fit a four-person family: Building area 100-200m$^2$, height to top of roof 6-10 meters.
- **Curb appeal:** Each side of the house must have at least one window on each wall. Solar panels cannot hang over roof edges.

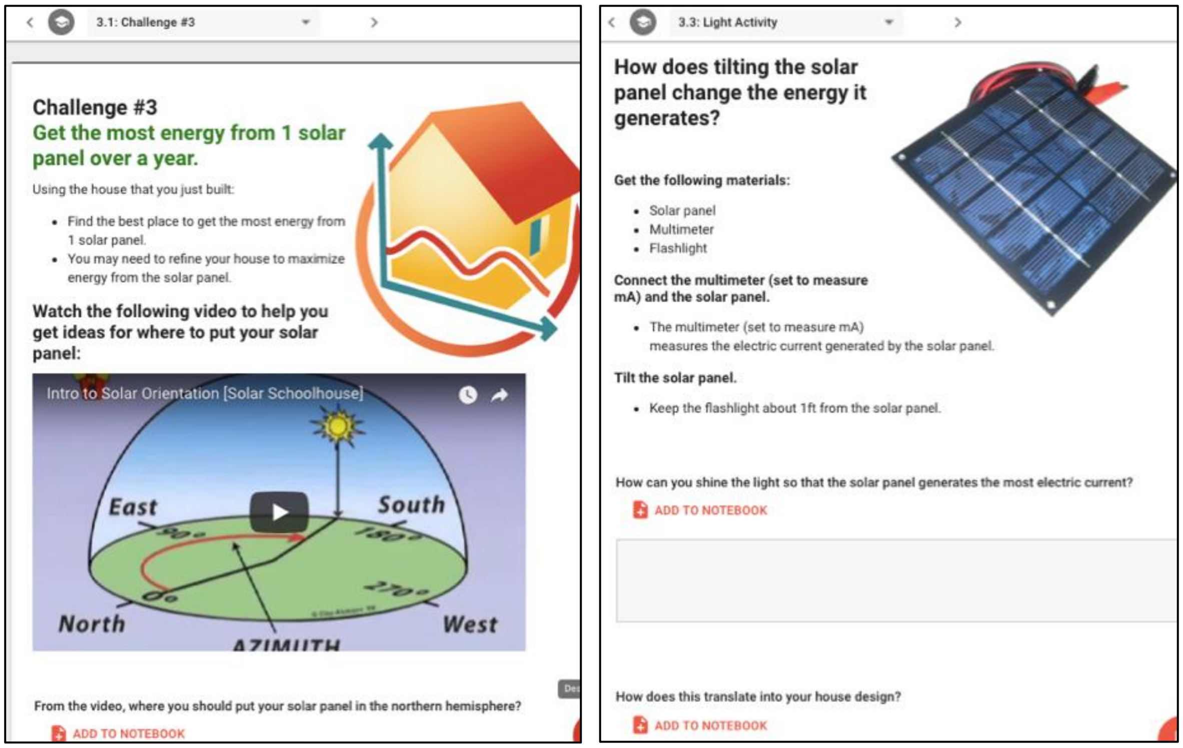
The constraints were chosen so that while many different design solutions were possible, it was also necessary for students to consider trade-offs, make design revisions, and consider solar science concepts to successfully find solutions. For example, with an unlimited budget it is a relatively straightforward task to design a house that satisfies the other constraints. However, the budget constraint limits the number of solar panels a student can use in their design, which motivates the need to know where to put the panels to generate the most energy, and therefore the need to know the solar science concepts that underpin these observations.

The net-zero energy house challenge was scaffolded for students with mini-challenges to give students opportunities to be successful throughout the project, as well as help students to become familiar with Energy3D in stages. For example, the students were given an initial challenge to build a house within budget, without needing to meet the constraint that asked for the house to use net-zero (or less) energy. The initial challenge aimed to help students become familiar with how to use Energy3D to add, resize, and remove items such as walls and windows, and how to find information about their designs such as cost and size. Subsequent challenges ask students to run the energy analyses with Energy3D and revise their designs in order to improve the energy efficiency of their design. The students are then asked to design a net-zero energy house that meets all four constraints (see Figure 4).

| **Challenge 1** | **Challenge 2** | **Challenge 3** | **Challenge 4** |
|---|---|---|---|
| Constraints:<br>☒ Energy efficient<br>☑ Cost<br>☑ Size<br>☑ Curb appeal | Constraints:<br>☒ Energy efficient<br>☑ Cost<br>☑ Size<br>☑ Curb appeal | Constraints:<br>☒ Energy efficient<br>☑ Cost<br>☑ Size<br>☑ Curb appeal | Constraints:<br>☑ Energy efficient<br>☑ Cost<br>☑ Size<br>☑ Curb appeal |
| Goal:<br>Design a house that meets all but the energy efficient constraint. | Goal:<br>Adjust the position and size of windows and trees to improve energy efficiency. | Goal:<br>Add one solar panel and adjust position and size to improve energy efficiency of design. | Goal:<br>Design a house that meets all the constraints. Make annual net energy negative. |

*Figure 4.* An overview of the scaffolded design challenges.

In addition, the students were provided with hands-on activities and outside information such as videos about solar energy concepts to help them make design decisions. Hands-on activities and information were provided to students with a just-in-time approach within the project itself as well as from teachers. Instead of lecturing on solar science concepts to the whole group before the project, teachers recommended or explained concepts that would help students understand their designs as needed. For example, when the students were deciding which side of their roof to place a solar panel, teachers engaged students in a short hands-on activity illustrating the rotation of the tilted earth around the sun to help students understand different positions of the sun at different times during different seasons. Simulations within Energy3D also allowed students to watch the sun move during different times and dates, and where the shadow of the sun falls for their design in each case (see Figure 5).



Challenge #3

Get the most energy from 1 solar panel over a year.

Using the house that you just built:

- Find the best place to get the most energy from 1 solar panel.
- You may need to refine your house to maximize energy from the solar panel.

Watch the following video to help you get ideas for where to put your solar panel:

From the video, where you should put your solar panel in the northern hemisphere?

How does tilting the solar panel change the energy it generates?

Get the following materials:

- Solar panel
- Multimeter
- Flashlight

Connect the multimeter (set to measure mA) and the solar panel.

- The multimeter (set to measure mA) measures the electric current generated by the solar panel.

Tilt the solar panel.

- Keep the flashlight about 1ft from the solar panel.

How can you shine the light so that the solar panel generates the most electric current?

How does this translate into your house design?

*Figure 5.* Screenshots of one of the scaffolded design challenges (left) and a hands-on activity (right).

**The evidence model**

The evidence model connects the net-zero energy house challenge and Energy3D with the student model, or engineering design behaviors we wish to measure. This paper builds on prior studies with students using Energy3D in classroom settings (e.g., Bywater et al., 2018; Chao et al., 2017; Seah, Vieira, Dasgupta, & Magana, 2016) and in particular, the studies that have used log data from Energy3D to assess design practices (e.g., Vieira, Magana, Purzer, 2017). For example, Vieira, Goldstein, Purzer, and Magana (2016) focused on characterizing student experimentation strategies within Energy3D. The authors determined specific sequences of actions within Energy3D that would indicate systematic experimentation, such as adding, moving, and resizing windows, conducting and analysis or seasonal change, then adding moving and resizing windows. The researchers then searched for these patterns within the log data and compared the number of experiments and found a relationship with post-test scores and design quality, with more experiments related to higher post-test scores and design quality. Similarly, Vieira, Seah, & Magana (2018) compared log data to think-aloud data of students' explanations during experimentation within Energy3D. The researchers found that while the process data identified relevant patterns, the think-aloud captured student reasoning for conducting experiments.

Although prior research has used log data to characterize design practices in Energy3D, the analysis typically starts with a researcher identifying patterns to look for based on their interpretation of the practice, and then searching for associated patterns in the data. Despite high theoretical alignment, this approach is labor intensive and may miss important patterns of student behavior that emerge from the data. Thus, this paper uses a ground-up approach to characterize behavior patterns in the log data, compares the patterns with student think-aloud data during the task, and investigates alignment between the log data and think-aloud data. Specifically, this paper aims to answer the following questions:

1. Using machine learning techniques, what patterns of student design behavior can be identified from the Energy3D log data?
2. What design behaviors do students describe during think-aloud interviews?
3. How valid are the machine learning techniques for identifying student design behaviors?

By answering these questions, this study aims to contribute to the validity of using log data to infer students' design behaviors in Energy3D. By complimenting past analytical techniques and leveraging an ECD framework, this study aims to provide guidance on whether machine learning techniques could and should be used in these settings.

## Methods

This study leverages a sequential mixed-methods approach (Creswell & Creswell, 2017), with the results from each phase used to inform the analysis of the subsequent phase. First, we used machine learning techniques to analyze the Energy3D log data for patterns of student actions. Second, we qualitatively coded student think-aloud descriptions at the grain-size of the log data patterns to find emergent patterns within student think-alouds. Third, we examined the results of each analysis together to see the extent to which there was agreement between the two approaches. In doing so we hope to examine how valid any inferences can be reached from the patterns that are found using the machine learning techniques, and what an evidence model for these inferences might be.

## Participants

The participants in this study were high school students (*n* = 75) enrolled in one of five environmental science classes at the same high school in the Eastern United States. Two of the classes were ‘honors’ classes and three ‘regular’ level classes. School demographics consisted of 34% Black, 9% Hispanic, and 45% White students with 45% of students receiving free or reduced lunch. The students participated by completing a series of scaffolded design challenges in Energy3D in their regular classroom setting (see Figure 4 above). Available to the students during this process were tutorial materials about the science of solar panels, and how passive solar heating occurs. Teachers and researchers supported students by answering questions or assisting with how to use Energy3D. After each design challenge, the students were asked to record their design, share it with their classmates, and answer questions about the rationale for their designs in writing. In addition, one researcher conducted informal interviews with students as they were designing by asking questions to probe their design reasoning such as, “Can you talk me through what you did with your design?” and “What are you thinking about your design?” The total project took five 90-minute periods for the regular level classes, and three 90-minute periods for the honors level classes. Most students worked by themselves, with some students working together in groups of two.

## Data Sources

**Energy3D log.** All the mouse-click level actions of each student group were logged. This gave detailed information about the time (to nearest second) that each action was performed. Examples of the types of mouse-click actions recorded were “add a solar panel”, “change the tilt of the solar panel”, “do annual energy analysis” and “animate the sun”.

**Student think-aloud descriptions.** While the students were working on their designs, a researcher asked students to describe their work and their approaches. Field notes about each conversation were made as well as an audio recording from which transcriptions were made. Field notes and audio recordings were aligned with the log data for each student group by comparing the timestamps for each data source.

## Log Data Analysis

**Data cleaning.** The Energy3D log data initially consisted of a total of 34941 records of individual mouse-click actions and their timestamp. Across the dataset there were 108 different types of action that were made. While most of these action types were related to designing the house, those that were not were removed from our analysis. Such actions included "open file", "zoom," "change camera angle." In addition, some of the actions that were functionally very similar were combined. For example, since "paste a solar panel" and "add a solar panel" performed the same function, both were considered "add a solar panel." Similarly, rather than maintaining the distinction between different types of roofs such as "hip roof," "shed roof," "gambrel roof," etc., we combined them all into one "roof" category. After these changes, our dataset consisted of a total of 14649 records of individual mouse-click actions of which there were 42 different types of action (see vertical axis in Figure 6).

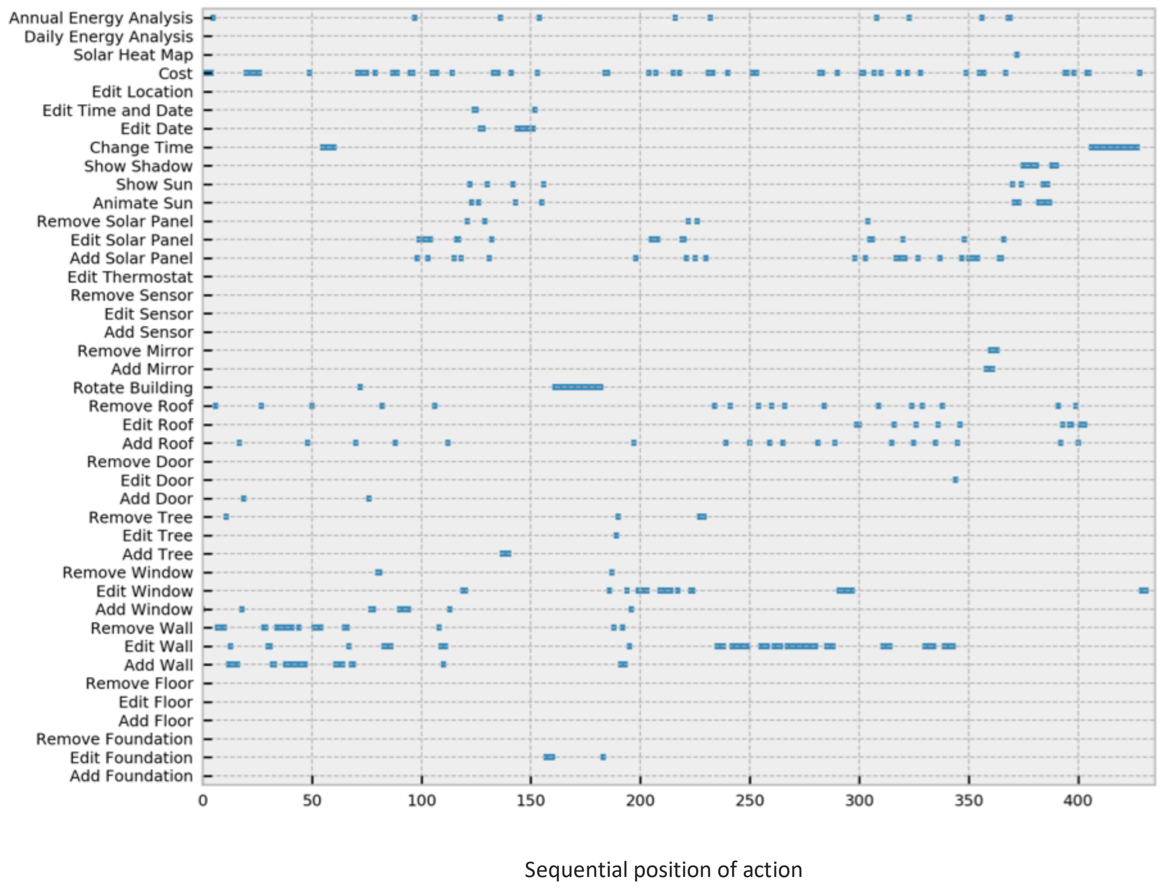


*Figure 6*. A plot for one student, indicating at what position in the full sequence of actions (horizontal axis) a particular action (vertical axis) was taken.

**Optimization functions.** We performed our analysis using the time-ordered sequence of actions for each student. A plot of the 432 actions recorded for one student is shown in Figure 6. We split the full sequence of actions for each student into contiguous action blocks such that the split positions maximized the difference between the distributions of the proportions of actions within adjacent action blocks. The optimization function used paired *z*-tests to compared the proportion of each action in an action block with the proportion of each action in an adjacent action block (1), and all pairs of adjacent action blocks were considered.

$$z_{s,s+1} = \frac{\Sigma_c\left(\frac{|p_{c,s}-p_{c,s+1}|}{\sqrt{p_{pooled}(1-p_{pooled})\left(\frac{1}{n_s}+\frac{1}{n_{s+1}}\right)}}\right)}{42} \tag{1}$$

Where $c$ = the set of 42 unique actions in the dataset

$s$ = the action block number, of length $n_s$

$p_{c,\,s}$ = the proportion of action $c$ in action block $s$

Having found the optimal split positions when a student's full sequence of actions is separated into two action blocks the process was repeated to find the optimal split positions with three actions blocks, then four etc., until optimal split positions for up to fifteen action blocks had been found. The best of these—the optimal number of action blocks for each student's full sequence of actions—was found by maximizing the average of the paired *z*-tests over all pairs of adjacent action blocks (2).

$$\underset{2\le i\le 15}{\operatorname{argmax}}\left(\frac{z_{1,2}+z_{2,3}+\cdots+z_{i-1,i}}{i-1}\right) \tag{2}$$

This averaging approach strongly favored small numbers of action blocks because initial splits tended to capture at the largest differences between action blocks and averaging in subsequent splits typically lowered the average. In order to adjust for this tendency, we introduced a smoothing variable, *t*, which had the effect of including an additional *t* zeros into the average (3).

$$\underset{2 \le i \le 15}{\operatorname{argmax}} \left( \frac{z_{1,2} + z_{2,3} + \cdots + z_{i-1,i}}{i - 1 + t} \right) \tag{3}$$

To select an appropriate value for *t*, we examined the impact of adjusting *t* on the optimal number of action block created. Figure 7 shows the distribution of the number of action blocks created for at different values of *t*, with *t* = 0 indicating the distribution that occurred before introducing *t*. From a visual inspection of this graph we chose a value of *t* = 4 because it was the most symmetric and had a broad spread.

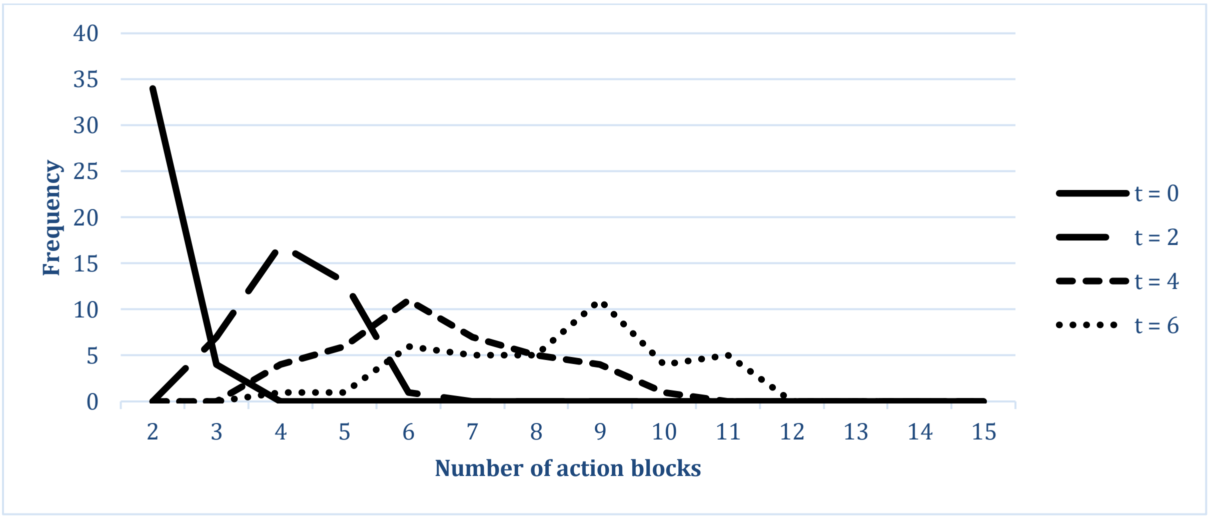


*Figure 7*. The impact of the smoothing variable, *t*, on the distribution of the number of action blocks created.

**Algorithm.** Using a brute force approach, that is, testing all the possible split positions within a full sequence of student actions of length *n* and split into *S* action blocks has the advantage of always finding the optimal split positions and the optimal number of splits, however the number of calculations required to do this is proportional to $n^S$. Figure 8 shows the time taken for the brute force approach to find optimal split positions for a sequence of length $n = 100$ in the limited cases where the number of action blocks is in the interval 2 through 5. In our study, with the average $n$ approximately 400 and with up to 15 action blocks considered, the brute force approach would take in the order of $10^{32}$ seconds, which is too long. Therefore, we considered other approaches that did not guarantee the optimal solution but could find near-optimal solutions quickly.

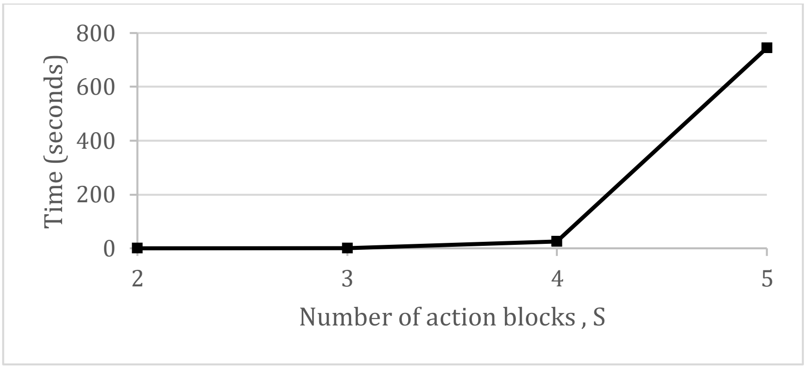


*Figure 8*. A comparison of the time to run a brute force algorithm to find optimal split positions for the cases where a sequence of length 100 is split into 2, 3, 4, and 5 action blocks.

We used both a genetic algorithm and a dynamic programing algorithm to find the optimal split positions. Each of these approaches have the advantage of finding split positions quickly. However, the each include the possibility of finding only local-maxima and may report non-optimal split positions. To investigate how problematic this might be with our data, we compared the maxima found by both of these approaches with maxima found using the guaranteed-optimal brute force algorithm when limiting the number of action blocks to at most three ($S \leq 3$). The genetic algorithm reported maxima that were on average 0.2% smaller than the brute force, and the dynamic algorithm reported maxima that were on average 0.4% smaller. While this indicates that both algorithms provide near optimal solutions, even better results are obtained when taking the best of the algorithms. In this case, the maxima were on average 0.03% smaller than the optimal brute force algorithm found. This indicates that while not optimal, using an approach that uses both a genetic algorithm and a dynamic algorithm, and takes the best of the two solutions generated provides very near optimal solutions, and can do so in a timely manner. Using this approach, we were able to split each student's full sequence of actions into action blocks where each block represents a common pattern of design behavior and a split indicates a change from one category of design behavior to another. Split positions for the example student shown above in Figure 6, are shown in Figure 9.

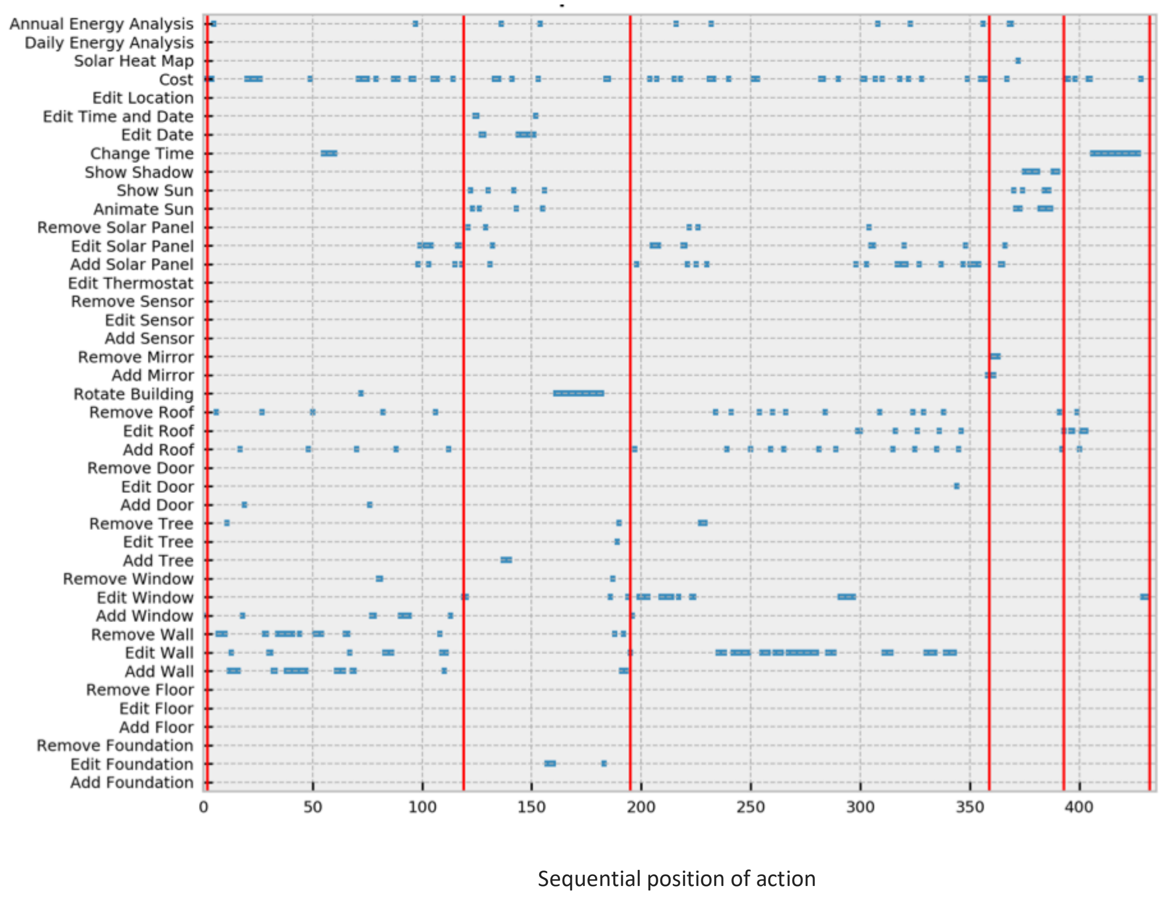


*Figure 9.* A plot for one student, with vertical red lines indicating the optimal split positions separating action blocks.

**Clustering.** After using the above approach to split each student's full sequence of actions into action blocks, we obtained 164 action blocks. Given that each action block represents a pattern of student design behavior and that we wished to look for similarities in the design behaviors across all students, we used a clustering approach to group similar action blocks. In addition to clustering similar action blocks into groups, this approach allowed us to look at the center of each action block and examine the proportion of actions that are typical of that cluster.

To do this, we first removed sixty action blocks that were small (with ten or less actions) because their size tended to produce proportions that were outliers and had excessive influence on our clustering outcomes. We grouped similar action blocks using a *k*-means clustering algorithm with the Euclidean distance between the proportions of actions within each action block used as the distance measure. To measure the validity of the clusters that we created we calculated the average silhouette width (Rousseeuw, 1987), and after repeating the *k*-means clustering 10,000 times, we selected the clusters with the largest silhouette value. We also attempted to use the silhouette value to find the optimal number of clusters, but found that the optimal number of clusters was always the largest number tested for. Instead, we examined the centroids of the clusters for increasing numbers of clusters until an additional centroid was created that appeared to be similar to one that already existed.

## Student think-aloud data analysis

Codes were initially developed by reviewing the transcripts of the recorded audio and identifying intentional, goal-oriented student activity. To establish reliability in coding the transcripts, two members of the research team independently coded a randomly selected subset of 20% of the transcripts. Following a discussion of coding discrepancies, the code descriptions were clarified and a further, randomly selected 20% of the transcripts were independently coded by the same two researchers. After achieving agreement greater than 90%, consensus was reached for any discrepancies and one member of the research team then coded the remaining transcripts.

## Aligning clusters and codes analysis

We used the timestamps from the think-aloud data and the timestamps from the Energy3D log data for each student to find at what position in a student's full sequence of actions the think-aloud occurred. We then observed which think-aloud codes were associated with which clusters by looking at which action blocks the codes were found within, and which clusters those action blocks were associated with (see Figure 10).

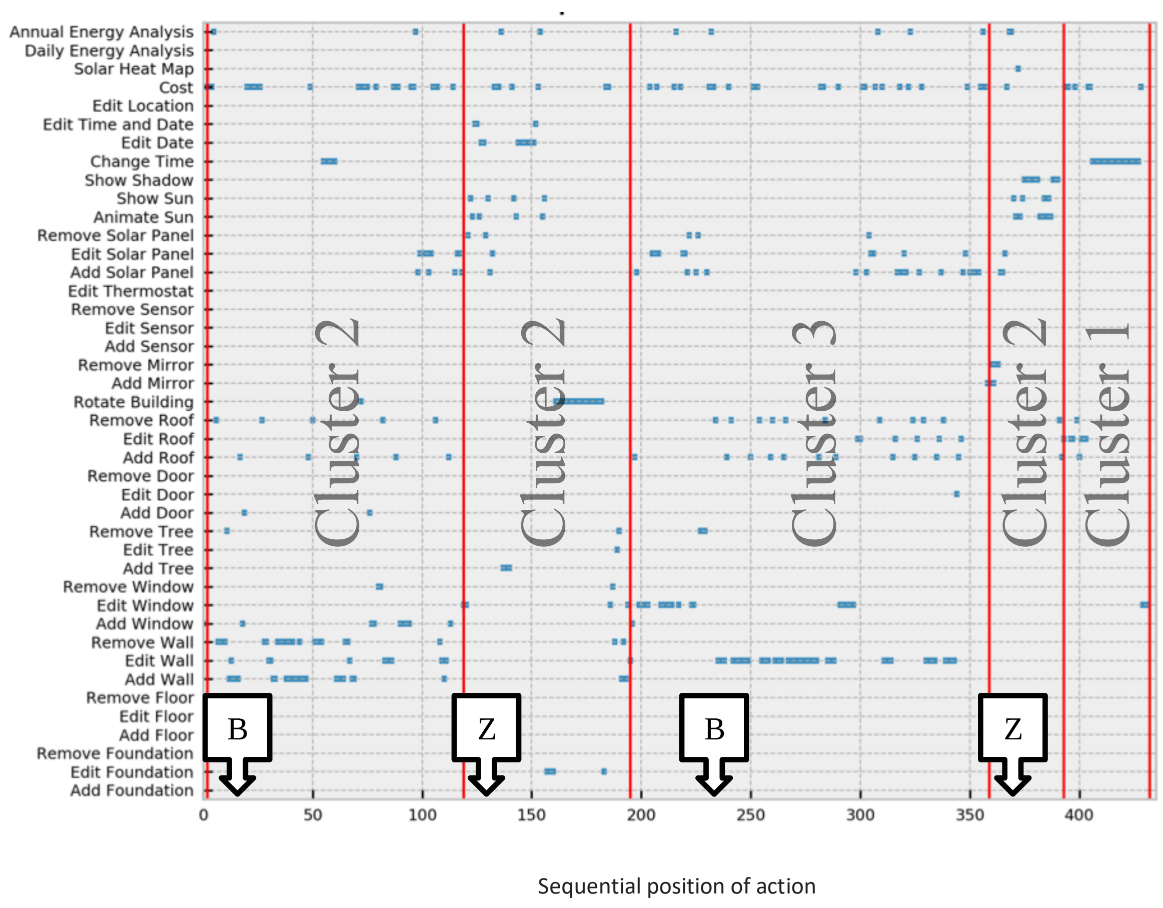


*Figure 10.* A plot for one student showing the cluster that each action blocks was assigned to and the position at which think-aloud descriptions led to the goal codes B and Z.

If an action block was removed from the clustering process because it was too small, any codes that were positioned within those action blocks were not included in our analysis. This occurred for 13 codes. Furthermore, since one cluster only had 2 codes in it, we dropped this cluster from our comparison of cluster and codes. To test the alignment of the log data clusters and the think-aloud codes, we performed a Chi-squared test for independence to test the null hypothesis that goal codes and cluster were independent. In addition, because the expected values of many of the cells in our table are smaller than required to meet the conditions for this test (cell count > 5), we also performed a two-sided Fisher's exact test with the same data.

## Results

### RQ1: Using machine learning techniques, what patterns of student design behavior can be identified from the Energy3D log data?

The analysis found five distinct clusters of student design behavior. The distribution of the proportions of the Energy3D actions for the centroids of each cluster are shown in Figure 11 along with the number of action blocks in each cluster. The cluster numbers were arbitrarily chosen. Each cluster show particular features:

Cluster 0 was dominated by actions that effected the position of the sun (e.g. editing the date and the time of day, animating the sun), alongside actions that impacted the position and presence of trees (e.g. add, edit and remove trees), shadows (e.g. show shadow), how the building was oriented (e.g. rotate building), and occasionally solar panels.

Cluster 1 was similar to Cluster 0 in that it is also dominated by actions that effected the position of the sun, alongside actions that impact the position and presence of trees. However, there were fewer actions that animated the sun and showed shadows and there was an absence of actions that rotated the building and edited the solar panels.

Cluster 2 showed a broader spread of actions with an elevated proportion of analysis actions (e.g. find the building cost of the design and calculate the annual net energy used by the design) in addition to actions related to solar panels, and actions related to the house such as walls, windows and the roof.

Cluster 3 was dominated by actions that relate to the house, especially actions that relate to the walls and the roof. The proportion of actions that find the building cost was also elevated but in contrast to Cluster 2, there were few actions to analyze the annual net energy of the design. This cluster also has an absence of actions related to the sun, the trees and the solar panels.

Cluster 4 was very heavily dominated by the action that changed the latitude of the location of the house design. Within Energy3D, this had the effect of altering the position of the sun in the sky. Some other actions that impacted the sun position (e.g. edit date, animate sun) were also present within this cluster, but most of the other actions found in other actions were absent.

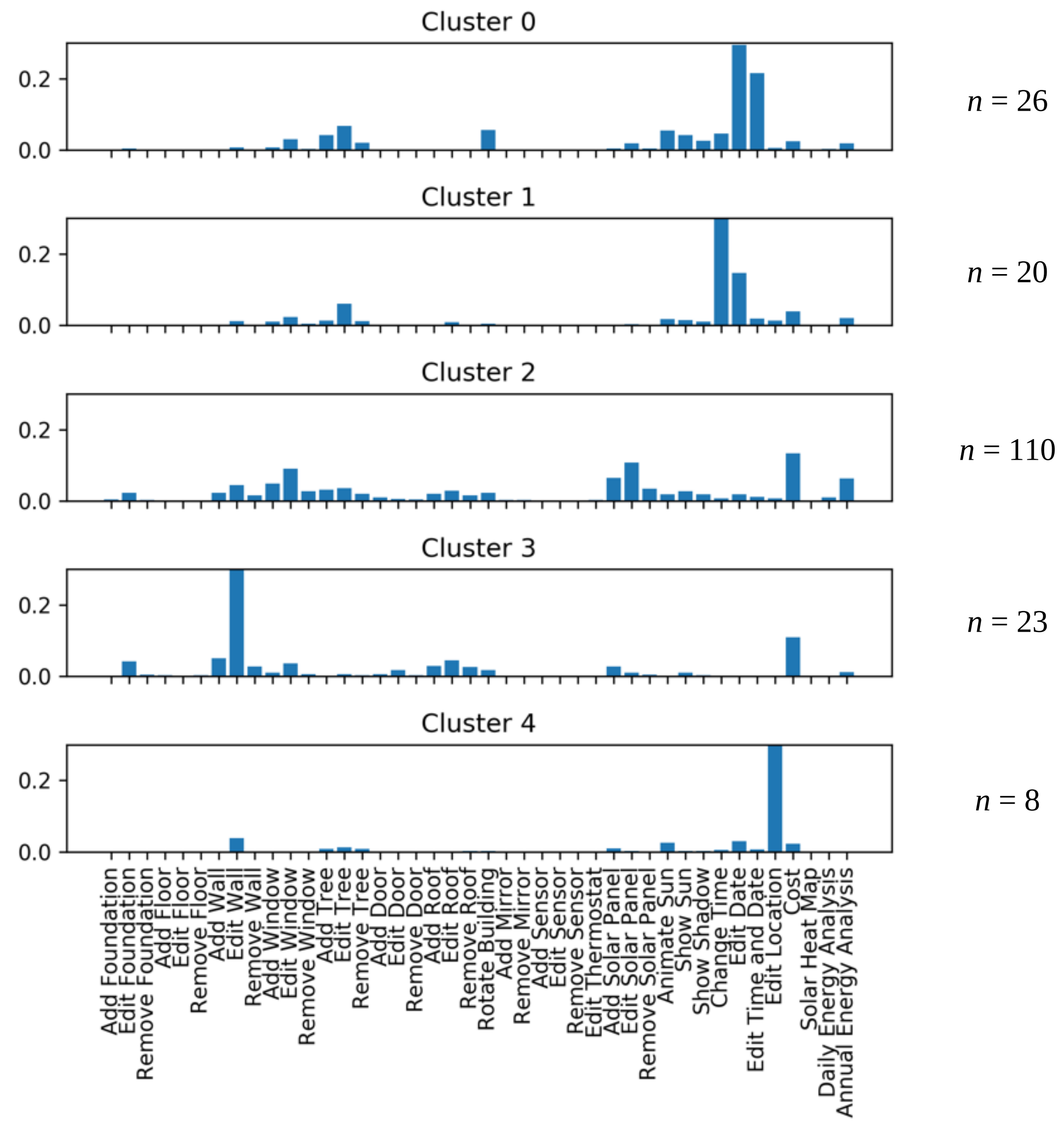


*Figure 11.* The distribution of the proportions of each action for the centroids of each cluster, along with the number of action blocks in each cluster. Proportions greater than 0.3 are clipped.

**RQ2: What design behaviors do students describe during think-aloud interviews?**

Given that results from the first research question identified large grain-size design behaviors, we decided to code the think-aloud data for goal-oriented activity rather than the smaller grain-size design strategies (Crismond & Adams, 2011). For example, rather than coding for strategies related to "defining the problem" or "testing the design", instead, we coded for the specific design goals associated with the design behaviors of the students. The final codes are described in Table 1. In cases where no design goals were evident during a think-aloud interview, no codes were assigned. For example, when students were logging into the project, or stated that they were unsure about what they were doing, no code was attributed to the data.

Table 1

*Descriptions of each of the codes generated from the think-aloud data*

| **Goal code*** | **Description** |
|---|---|
| B: Building | Making/editing the building *with the goal* that it meets the building constraints of 6m < height <10m, $100m^2$ < area < $200m^2$, and cost < $150,000. |
| T: Trees | Changing the position of the trees *with the goal* that the annual net energy for the building is reduced (and building constraints maintained). |
| W: Windows | Changing the position of the windows *with the goal* that the annual net energy for the building is reduced (and building constraints maintained). |
| P: Panels | Changing the position of the solar panels *with the goal* that the Annual net energy for the building is reduced (and building constraints maintained). |
| Z: Zero net energy | Changing the design (building, trees, windows, OR panels) *with the goal* that the annual net energy is zero or negative (and building constraints maintained). |

*Multiple codes per think-aloud interview are possible.

A total of 139 student think-aloud interviews collected during the study that were coded with at least one goal code. As an illustrative example, the following transcript was coded as ‘Building.’

| | |
|---|---|
| Student: | I was just trying to look at the area [of the house] but… if I can figure it out… [mumbles]…it should be a little bigger, I think, in order to fit the requirements because its 100 for area I think. |
| Researcher: | So, you’re trying to make the area be 100? |
| Student: | I think that’s what it’s supposed to be, yeah... I’m just going to go ahead and delete that wall and then redo this one. [Long pause]. It’s so close! |

In total 164 codes were assigned to the transcripts of these interviews with the break-down by code shown in Table 2.

Table 2

*Total frequencies of each goal code within the think-aloud transcripts*

| | | Goal code | | |
|---|---|---|---|---|
| Building | Trees | Windows | Panels | Zero net energy |
| 56 | 38 | 27 | 20 | 23 |

**RQ3: How valid are the machine learning techniques for identifying student design behaviors?**

Table 3 displays the goal codes found within action blocks for each cluster. There were large differences in the number of goal codes associated with each cluster in part because there were more action blocks in some clusters than others but also because some action blocks had multiple codes and other action blocks had none. For example, the number of goal codes associated with Cluster 2 was much larger than for the other clusters. Therefore, while the associations between clusters and goal codes are described in more detail below after examining the adjusted standardized residuals which account for the differences in counts, it is apparent that some clusters were associated with just one goal (e.g. Cluster 3 and the "Building" goal) but that other clusters were associated with multiple goal codes (e.g. Cluster 2).

Table 3

*Goal codes found within action blocks for each cluster*

| Cluster | Goal code | | | | | |
|---|---|---|---|---|---|---|
| | Building | Trees | Windows | Panels | Zero net energy | Total |
| 0 | 3 | 3 | 1 | 4 | 1 | 12 |
| 1 | 0 | 3 | 2 | 0 | 1 | 6 |
| 2 | 25 | 31 | 22 | 16 | 17 | 111 |
| 3 | 18 | 0 | 1 | 0 | 1 | 20 |
| 4 | 1 | 0 | 0 | 0 | 1 | 2 |
| Unclustered | 9 | 1 | 1 | 0 | 2 | 13 |
| Total | 56 | 38 | 27 | 20 | 23 | 164 |

*Note.* Cluster 4 codes and the unclustered codes were dropped from our analysis.

The chi-squared test for independence rejected the null hypothesis ($\chi^2 = 46.387$, $df = 12$, $p = 5.95\times10^{-6}$), as did the two-sided Fisher's exact test ($p = 1.846\times10^{-6}$). This indicates that goal code and cluster are not independent. To examine the nature of their dependency, we calculated the adjusted standardized residuals for each cell (see Table 4).

Table 4

*Adjusted standardized residuals showing which goal codes occurred higher (positive values) and lower (negative values) than expected.*

| Cluster | Goal code | | | | |
|---|---|---|---|---|---|
| | Building | Trees | Windows | Panels | Zero net energy |
| 0 | -0.46 | 0.01 | -0.87 | 2.11* | -0.54 |
| 1 | -1.67 | 1.46$^{††}$ | 1.05$^{†}$ | -0.98 | 0.24 |
| 2 | -3.77 | 1.49$^{††}$ | 1.30$^{††}$ | 0.61 | 1.16$^{†}$ |
| 3 | 6.15*** | -2.76 | -1.58 | -1.89 | -1.19 |

*Note*. For goal codes that occurred higher than expected $^{†}p < 0.15$; $^{††}p < 0.10$; $^{*}p < 0.05$; $^{**}p < 0.01$; $^{***}p < 0.001$.

Examining the standardized residuals in Table 4, we see evidence that:

Cluster 0 is associated with changing the position of the solar panels with the goal that the annual net energy for the building is reduced (and building constraints maintained);

Cluster 1 is associated with changing the trees and windows to reduce the annual net energy;

Cluster 2 is associated with both changing trees and window to reduce the annual net energy but also changing the overall design (building, trees, windows, or panels) with the goal that the annual net energy is zero or negative and the building constraints are maintained; and

Cluster 3 is associated with making or editing the building with the goal that it meets the building constraints.

These results are similar to the descriptions of the cluster distributions from the first research question analysis. However, these cluster descriptions are based on what the students explained they were doing and present much stronger validation of the cluster descriptions that we had earlier. One discrepancy is that based on an examination of the distributions alone, Cluster 1 appeared to be similar to Cluster 0. However, when considering the students think-aloud descriptions, Cluster 1 appears similar to Cluster 2. This difference may be due to the small number of goal codes that were linked to Cluster 1 (*n* = 6), or may be that small differences in the proportion of some of the actions are markers of different goals. For example, Cluster 0 has a larger proportion of “rotate building” actions than other clusters, and this appears be a strong indicator that students are examining solar panels to improve energy efficiency.

Comparing the clusters, that were found using the ground-up machine learning approach, and their associated goal codes with the task model design challenges we find strong alignment (see Figure 12).

| | | | | |
|---|---|---|---|---|
| **TASK MODEL (Figure 4)** | **Challenge 1**<br>Constraints:<br>☒ Energy efficient<br>☑ Cost<br>☑ Size<br>☑ Curb appeal<br><br>Goal:<br>Design a house that meets all but the energy efficient constraint. | **Challenge 2**<br>Constraints:<br>☒ Energy efficient<br>☑ Cost<br>☑ Size<br>☑ Curb appeal<br><br>Goal:<br>Adjust the position and size of windows and trees to improve energy efficiency. | **Challenge 3**<br>Constraints:<br>☒ Energy efficient<br>☑ Cost<br>☑ Size<br>☑ Curb appeal<br><br>Goal:<br>Add one solar panel and adjust position and size to improve energy efficiency of design. | **Challenge 4**<br>Constraints:<br>☑ Energy efficient<br>☑ Cost<br>☑ Size<br>☑ Curb appeal<br><br>Goal:<br>Design a house that meets all the constraints. Make annual net energy negative. |
| **GOAL CODES** | **B: Building** | **T: Trees**<br>**W: Windows** | **P: Panels** | **Z: Zero net energy**[a] |
| **CLUSTER** | **Cluster 3** | **Cluster 1** | **Cluster 0** | **Cluster 2** |

[a]*Note.* Cluster 2 was also associated with goal codes for Trees and Windows, but these are not shown here because they can be considered sub-goals within goal Zero net energy.

*Figure 12*. Alignment between task design challenge, goal codes, and cluster.

## Discussion and Implications

This paper aimed to identify and validate evidence from the Energy 3D log data for student design behaviors. Our ground-up, machine learning approach identified five clusters of student design behavior, four of which we were able to validate with students' think-aloud data. The design behaviors identified were the larger grain-size goal orientations of students using Energy3D rather than the smaller grain-size design strategies (e.g. Crismond & Adams, 2011). For each distribution of the proportions of Energy3D actions we identified a distinct goal orientation (see Figure 11). These goal orientations aligned with the four design challenges presented to the students (see Figure 12). Together, these distributions constitute the ECD evidence model that we set out to identify. These distributions connect the log data collected while students participate in the design challenges (task model) using Energy3D (presentation model), with the goal orientation of the student design behaviors (student model).

Identifying goal-oriented student design behavior from their mouse-click actions can benefit students and teachers. A large body of research highlights the importance of students identifying goals and using various learning strategies to attain those goals (e.g. Zimmerman, 2000). Within design contexts, working to meet design goals is essential to the process of creating solutions. Students need to be able to set design goals for themselves, monitor and evaluate their progress towards those goals, and utilize different design strategies or practices based upon whether they are meeting their goals. Being able to capture the goals that students are working towards can help automatically provide more specific and targeted feedback to students.

For example, if students are spending a significant amount of time in one cluster, say “trees and windows”, then automated feedback can help students with specific prompts such as, “It looks like you are trying to understand how trees and windows affect the performance of your design. Perhaps you should try these activities on how trees can be used to make homes more energy efficient.”

Similarly, identifying students’ design goals may help teachers provide targeted guidance to students. Providing information to teachers about the goal orientation of each of their students may help teachers monitor students’ progress in a classroom full of students working on different designs at different paces. For example, if students have been working on a specific goal for a long time, teachers can use that information to target that specific student and provide help or guidance. If students are not working on a specific goal, the teacher can use that to either help the student get back on track or diagnose what they are trying to do.

Furthermore, the automatic detection of design goals may provide the necessary context for understanding and giving feedback to students about smaller grain-size design behaviors such as their design strategies. For example, to give feedback about troubleshooting or conducting experiments it is important to know to what end the students used those strategies. Currently, the relation of specific strategies to design goals are only implicit in the informed design matrix. The relation of specific engineering practices to particular design goals in the NGSS is also missing in the NGSS framework. This highlights a tension between the need to articulate and define specific strategies to help students understand and engage in design practices, and the need to not lose sight of the overall context and goals under which those practices should be used, which is central to design. Identifying design goals may be a necessary component for being able to capture student design knowledge. Results imply that design goals may be a necessary and crucial component to both the student and evidence models in the ECD framework, as well as a crucial component of informed design.

The machine learning analysis used in this study did not make any prior assumptions about the goals of the students. As such, the analysis has the potential to be applied to other settings that generate sequences of fine grain-size actions such as mouse-click level log data. Such setting may include learning simulations or game-based learning contexts in which students might pursue a variety of goals. Automatically identifying the goals for the students in these contexts could then be used to support feedback to students both automatically within the setting but also via their teachers who could be provided with additional insight into their students' goals.

## Limitations

The generalizability of the results from this study are limited by the fact that we examined a small sample of students in one instructional context. Further research is needed to assess whether the evidence model developed in this study would be applicable in other schools or learning settings, and with other groups of students. In addition, from this study alone it is unclear how robust the evidence model is to changes in the task model. We suspect that small changes to the design challenges that were posed to students would have little impact on the evidence model, but that larger changes may miss particular goal orientations or require the evidence model to be recalibrated. However, the impact of changes to the task model remain unclear and a potential topic of future study.

This study developed an evidence model to support teachers and students to identify student design goals. This is an important prerequisite to providing feedback to students about their goals. However, this study did not investigate the impact of identifying student design goals on the effectiveness of feedback, or on how this supports student meta cognition and learning. While research suggests that identifying student design goals can play an important role in these ways, an examination of this crucial question, in this context, is an important next step.

Unfortunately, the machine learning approaches used in this study were not able to identify smaller grain-size design strategies. Articulating the evidence model for such strategies would complement the results and support a student model that includes both design goals and design strategies. In order to identify the design strategies, future work might consider revising the approaches used in this study. For example, goals may have been identified in this study because the analysis focused on finding action blocks of log data with proportions of actions that were as distinct as possible from adjacent action blocks. This approach focusses on the types of actions undertaken within a given action block rather than the order of these actions within the action block. To identify smaller grain-size design strategies it might be necessary to consider the order of individual actions in more detail (e.g., Vieira, Seah, & Magana, 2018) as well as consider the impact of the individual actions on the quality of the design such as how the action of editing a window changed the window size, the cost to build, or the energy used (e.g., Magana et al., 2019). Alternatively, assigning weights to underrepresented actions of interest may highlight particular design strategies within the log data. These alternative approaches may be useful for any future work that intends to use machine learning to identify student design strategies.

## Conclusion

This study demonstrates that for sequential log data, it is possible to automatically capture the design goals of students during design projects in CAD environments. This could be used to augment guidance and feedback in these settings as well as inform teachers about the goal orientation of the students during their design activity.

**References**

Beal, C. R., Mitra, S., & Cohen, P. R. (2007). Modeling learning patterns of students with a tutoring system using Hidden Markov Models. In *Proceedings of the 13th International Conference on Artificial Intelligence in Education (AIED)*, (July), 238–245.

Boyer, K. E., Ha, E. Y., Wallis, M. D., Phillips, R., Vouk, M. A., & Lester, J. C. (2009). Discovering tutorial dialogue strategies with hidden Markov models. *Frontiers in Artificial Intelligence and Applications*, *200*(1), 141–148.

Burghardt, M. D. & Hacker, M. (2004). Informed design: A contemporary approach to design pedagogy as the core process in technology. *Technology teacher*, *64*(1), 6-8.

Burghardt, M. D., Hecht, D., Russo, M., Lauckhardt, J., & Hacker, M. (2010). A Study of Mathematics Infusion in Middle School Technology Education Classes. *Journal of Technology Education*, *22*(1), 58-74.

Bywater, J. P., Chiu, J. L., Floryan, M., Chao, J., Schimpf, C., Xie, C., Vieira, C., Magana, A., and Dasgupta, C. (2018). Using Machine Learning Techniques to Capture Engineering Design Behaviors. *Proceedings of the International Conference of the Learning Sciences – Volume 3* (pp. 1359-1360). June 23-27. London: International Society of the Learning Sciences, Inc.

Cantrell, P., Pekcan, G., Itani, A., & Velasquez-Bryant, N. (2006). The Effects of Engineering Modules on Student Learning in Middle School Science Classrooms. *Journal Of Engineering Education*, *95*(4), 301-309.

Chao, J., Xie, C., Nourian, S., Chen, G., Bailey, S., Goldstein, M. H., ... & Tutwiler, M. S. (2017). Bridging the design-science gap with tools: Science learning and design behaviors in a simulated environment for engineering design. *Journal of Research in Science Teaching*, *54*(8), 1049-1096.

Chen, H. L., Cannon, D., Gabrio, J., Leifer, L., Toye, G., & Bailey, T. (2005). Using wikis and weblogs to support reflective learning in an introductory engineering design course. *Human behaviour in design*, *5*, 95-105.

Chiu, J. L., Malcolm, P. T., Hecht, D., DeJaegher, C. J., Pan, E. A., Bradley, M., & Burghardt, M. D. (2013). WISEngineering: Supporting precollege engineering design and mathematical understanding. Computers & Education, 67, 142-155.

Creswell, J. W., & Creswell, J. D. (2017). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications.

Crismond, D. P., & Adams, R. S. (2012). The informed design teaching and learning matrix. *Journal of Engineering Education*, *101*(4), 738-797.

Cross, N. (2001). Design cognition: results from protocol and other empirical studies of design activity. In C. Eastman, W. Newstatter, & M. McCracken (Eds.), *Design knowing and learning: Cognition in design education* (pp. 79–103). Oxford, UK: Elsevier.

Gero J. S. & Tang H.-H. (2001). The differences between retrospective and concurrent protocols in revealing the process-oriented aspects of the design process. *Design Studies, 22*(3), 283–295.

Han, S., Capraro, R., & Capraro, M. M. (2014). How science, technology, engineering, and mathematics (STEM) project-based learning (PBL) affects high, middle, and low achievers differently: The impact of student factors on achievement. *International Journal of Science and Mathematics Education*, *13*(5), 1089-1113.

Katehi, L., Pearson, G., & Feder, M. (2009). National Academy of Engineering and National Research Council report: Engineering in K-12 education.

Klahr, D., Triona, L. M., & Williams, C. (2007). Hands on what? The relative effectiveness of physical versus virtual materials in an engineering design project by middle school children. *Journal of Research in Science teaching*, *44*(1), 183-203.

Kolodner, J. L., Camp, P. J., Crismond, D. P., Fasse, B., Gray, J., Holbrook, J., ... Ryan, M. (2003). Problem-based learning meets case-based reasoning in the middle-school science classroom: Putting learning by design into practice. *Journal of the Learning Sciences, 12*(4), 495–547.

Magana, A.J., Elluri, S., Dasgupta, C., Seah, Y.Y., Madamanchi, A. and Boutin, M. (2019). The role of computer-based and teacher feedback on middle school students' self-generated heuristics in the context of a design challenge. *Journal of Science Education and Technology*.

McBride, E., Vitale, J., & Linn, M. (2018). Learning Design Through Science vs. Science Through Design. *Proceedings of the International Conference of the Learning Sciences – Volume 3* (pp. 17-24). June 23-27. London: International Society of the Learning Sciences, Inc.

Mislevy, R. J., Steinberg, L. S., & Almond, R. G. (2003). Focus article: On the structure of educational assessments. *Measurement: Interdisciplinary Research and Perspectives, 1*(1), 3–62

Moore, T. J., Stohlmann, M. S., Wang, H. H., Tank, K. M., Glancy, A. W., & Roehrig, G. H. (2014). Implementation and integration of engineering in K-12 STEM education.

In *Engineering in Pre-College Settings: Synthesizing Research, Policy, and Practices* (pp. 35-60). Purdue University Press.

NGSS Lead States. (2013). Next Generation Science Standards: For States, By States.

Plant, E. A., Baylor, A. L., Doerr, C. E., & Rosenberg-Kima, R. B. (2009). Changing middle-school students' attitudes and performance regarding engineering with computer-based social models. *Computers & Education*, *53*(2), 209-215.

Purzer, S., Moore, T., Baker, D., & Berland, L. (2014). *Supporting the implementation of the Next Generation Science Standards (NGSS) through research: Engineering.* Retrieved from https://narst.org/ngsspapers/engineering.cfm

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, *20*, 53-65.

Schnittka, C. & Bell, R. (2011). Engineering design and conceptual change in science: Addressing thermal energy and heat transfer in eighth grade. *International Journal of Science Education*, *33*(13), 1861-1887.

Seah, Y.Y. & Magana, A.J. (in press). Exploring students' experimentation strategies in engineering design using an educational CAD tool. *Journal of Science Education and Technology.*

Seah, Y. Y., Vieira, C., Dasgupta, C., & Magana, A. J. (2016). Exploring Students' Experimentation Strategies in Engineering Design using an Educational CAD Tool. *Proceedings of the 46th Annual Frontiers in Education (FIE) Conference*. Erie, PA. October 12-14, 2016.

Shute, V. J. (2011). Stealth assessment in computer-based games to support learning. In S. Tobias & J. D. Fletcher (Eds.), *Computer games and instruction* (pp. 503-524). Charlotte, NC: Information Age Publishers.

Stevens, R., Johnson, D. F., & Soller, A. (2005). Probabilities and Predictions: Modeling the Development of Scientific Problem-Solving Skills. *Cell Biology Education, 4*(1), 42–57. http://doi.org/10.1187/cbe.04-03-0036

Vieira, C., Goldstein, M. H., Purzer, Ş., & Magana, A. J. (2016). Using Learning Analytics to Characterize Student Experimentation Strategies in the Context of Engineering Design. *Journal of Learning Analytics, 3*(3), 291-317

Vieira, C., Magana, A. J., & Purzer, S. (2017). Identifying Engineering Students' Design Practices Using Process Data. In *Proceedings of 2017 research in engineering education symposium (REES). Bogotá-Colombia*.

Vieira, C., Seah, Y. Y., and Magana, A. J. (2018) Students' Experimentation Strategies in Design: Is process data enough? *Computer Applications in Engineering Education*. 26(5), 1903-1914.

Wang, H. H., Moore, T. J., Roehrig, G. H., & Park, M. S. (2011). STEM integration: Teacher perceptions and practice. *Journal of Pre-College Engineering Education Research (J-PEER)*, *1*(2), 2.

Worsley, M. & Blikstein, P. (2014). Analyzing engineering design through the lens of computation. *Journal of Learning Analytics*, *1*(2), 151-186.

Xie, C. & Nourian, S. (n.d.). Energy3D: Learning to build a sustainable future. Retrieved from https://energy.concord.org/energy3d/

Xie, C., Schimpf, C., Chao, J., Nourian, S., & Massicotte, J. (2018). *Learning and Teaching Engineering Design through Modeling and Simulation on a CAD Platform*, Computer Applications in Engineering Education, 26(4), pp. 824-840.

Zimmerman, B. J. (2000). Attaining self-regulation: A social cognitive perspective. In *Handbook of self-regulation* (pp. 13-39). Academic Press.

## Appendix A

## Dynamic Algorithm

```
def find_sections_mf(data, max_sections=5):
    print("STARTING DYNAMIC ALGORITHM")
    n = len(data)
    sparse_array, items = list2sparse(data)
    counts = make_counts_array(n, sparse_array)

    # initialize arrays
    # A stores best value of the optimization function for [first d rows, with c cuts]
    A = np.zeros((n, max_sections+1), dtype=object)
    # P stores previous cut position that A used (e.g. 0,1,2 v 3,4 => 3)
    P = np.zeros((n, max_sections+1), dtype=object)

    #first two columns are zero/empty since 0 cuts/sections is meaningless and 1 cut/section has no comparison.
    for c in range(2, max_sections+1):
        for d in range(n):
            if d+1>=c:
                tmpf = []
                tmpi = []
                for i in range(c-2, d):
                    array1=counts[P[i, c-1], i+1]
                    array2=counts[i+1, d+1]
                    g=optim(array1, array2)
                    #print(array1, array2, g)
                    f = ((A[i, c-1] * (c-1)) + g) / c
                    tmpf.append(f)
                    tmpi.append(i)
                A[d,c] = np.array(tmpf).max()
                P[d,c] = tmpi[np.array(tmpf).argmax()]+1
    best_f = A[n-1,].max()
    cut_positions=[n]
    row=n-1
    section = A[row,].argmax()
    while section>0:
        cut_pos = P[row, section]
        cut_positions.append(cut_pos)
        row = cut_pos-1
        section = section -1
    cut_positions_asc = list(reversed(cut_positions))
    sections = []
    for x in range (len(cut_positions_asc)-1):
        start=cut_positions_asc[x]
        end=cut_positions_asc[x+1]
        sections.append(data[start:end])
    return cut_positions_asc, best_f
```

## Brute Force Algorithm

```
def find_sections_bf(data, max_sections=10):
    print("STARTING BRUTE FORCE ALGORITHM")
    n = len(data)
    sparse_array, items = list2sparse(data)
    #print("unique items are:", items)
    counts = make_counts_array(n, sparse_array)
    ft = []
    cpt = []
    for c in range(2, max_sections+1):
        print("working on", c, "sections...")
        if n>=c:
            fs = []
            cps = []
            for combos in itertools.combinations(list(range(1, n)), c - 1):
                cut_positions = [0] + list(combos) + [n]
                f=0
                for i in range (len(cut_positions)-2):
                    array1=counts[cut_positions[i], cut_positions[i+1]]
                    array2=counts[cut_positions[i+1], cut_positions[i+2]]
                    g = optim(array1, array2)
                    f = ((f*(i+1))+g)/(i+2)
                fs.append(f)
                cps.append(cut_positions)
            ft.append(np.array(fs).max())
            cpt.append(cps[np.array(fs).argmax()])
    best_f = np.array(ft).max()
    best_cuts = cpt[np.array(ft).argmax()]
    sections = []
    for x in range (len(best_cuts)-1):
        start=best_cuts[x]
        end=best_cuts[x+1]
        sections.append(data[start:end])
    return best_cuts, best_f
```

## Genetic Algorithm

```
def find_sections_gen(data, max_sections=10, mate_prob = 0.5, mut_prob = 0.3):
    print("STARTING GENETIC ALGORITHM")
    n = len(data)
    sparse_array, items = list2sparse(data)
    counts = make_counts_array(n, sparse_array)
    best_cuts = []
    for c in range(2, max_sections+1):
        if n>=c:
            ## MAKE ORIGINAL POPULATION
            #set population size (reduce if small data set)
            pop_size = 5000
            cut_combos = special.comb(n - 1, c - 1, exact=True)
            if round(cut_combos/2) < pop_size:
                pop_size = round(cut_combos/2)

            #create original parents
```

```
population_cuts = []
while len(population_cuts) < pop_size:
    cut_positions = [0] + sorted(random.sample(range(1,n), k=c-1)) + [n]
    if cut_positions not in population_cuts:
        population_cuts.append(cut_positions)
population = []
for cuts in population_cuts:
    f = get_fitness(cuts, counts)
    population.append([cuts,f])
population = sorted(population, key=itemgetter(1), reverse=True)

for generation in range(5000):
    ## IDENTIFY MATES
    mates = []
    i=0
    while len(mates) < 2:
        if random.random() < mate_prob:
            mates.append(population[i])
            if i < len(population) - 1:
                i = i + 1
        else:
            if i < len(population) - 1:
                i = i + 1
            else:
                mates.append(population[i])

    ## MAKE OFFSPRINGS
    #combine mates
    number_of_offspring = 2
    offspring = []
    while len(offspring) < number_of_offspring:
        splice_position = random.randint(2, c)
        #need to figure out while parent to come first based on which has lower number at splice point
        if mates[0][0][splice_position] < mates[1][0][splice_position]:
            cuts = mates[0][0][:splice_position] + mates[1][0][splice_position:]
```

```
                else:
                    cuts = mates[1][0][:splice_position] + mates[0][0][splice_position:]
                offspring.append(cuts)

            #mutate offspring cuts
            for cuts in offspring:
                if random.random() < mut_prob:
                    #print("mutation occured!")
                    for i in range(1, c):
                        nudge_down_max = int((cuts[i] - cuts[i-1]) / 2)
                        nudge_up_max = int((cuts[i+1] - cuts[i]) / 2)
                        cuts[i]=cuts[i] + random.randint(-nudge_down_max, nudge_up_max)

            #check offspring not already in population
            population_cuts = []
            for member in population:
                population_cuts.append(member[0])
            offspring_to_add = []
            for cuts in offspring:
                if cuts not in population_cuts:  ##FIX THIS WITH TUPLE TRICK
                    offspring_to_add.append(cuts)

            ## REPLACE UNFIT POPULATION WITH OFFSPRING
            population = population[:pop_size-len(offspring_to_add)]
            for cuts in offspring_to_add:
                f = get_fitness(cuts, counts)
                population.append([cuts, f])
            population = sorted(population, key=itemgetter(1), reverse=True)
        best_cuts.append(population[0]+[c])

bestist = sorted(best_cuts, key=itemgetter(1), reverse=True)[0]
sections = []
for x in range (len(bestist[0])-1):
    start=bestist[0][x]
    end=bestist[0][x+1]
```

```
        sections.append(data[start:end])
    return bestist[0], bestist[1]
```

## Miscellaneous related functions

```
def optim(array1, array2, fn="two_props"):
    if fn == "sum_sq":
        diff=array1 - array2
        return np.square(diff).sum()
    if fn == "sum_abs":
        diff = array1 - array2
        return np.absolute(diff).sum()
    if fn == "chi2_homo":
        table = np.vstack((array1, array2))
        chi2, p, dof, ex = stats.chi2_contingency(table+1)
        return chi2
    if fn == "two_props":
        array1 = array1 + 1
        array2 = array2 + 1
        props1 = array1 / array1.sum()
        props2 = array2 / array2.sum()
        pooledp = (array1 + array2) / (array1.sum() + array2.sum())
        variance = pooledp * (1 - pooledp) * ((1 / array1.sum()) + (1 / array2.sum()))
        se = np.sqrt(variance)
        abs_zscore = abs(props1 - props2) / se
        stat = abs_zscore.sum() / len(abs_zscore)
        return stat


def get_fitness(cuts, counts):
    f = 0
    for i in range(len(cuts) - 2):
        # print("i", i, "c", c)
        array1 = counts[cuts[i], cuts[i + 1]]
        # print("ar1", array1)
        array2 = counts[cuts[i + 1], cuts[i + 2]]
        # print("ar2", array2)
        # print(cuts)
        g = optim(array1, array2)
        f = ((f * (i + 1)) + g) / (i + 2)
    return f
```