Data mining a design repository to generate linear functional chains: a step toward automating functional modeling

Katherine Edmonds, Alex Mikes, Bryony DuPont, Robert B. Stone

Oregon State University - Design Engineering Laboratory
School of Mechanical, Industrial & Manufacturing Engineering
Corvallis, Oregon 97333
Rob.Stone@oregonstate.edu

Abstract

Populating the different types of data for a design repository is a difficult and time-consuming task. In this work, we report on techniques to automate the population of data related to product function. We explore a preliminary method to automate the generation of the functional chains of components from new products based on hierarchical data from an existing design repository. We use datasets of various scale and specificity to find correlations between functions and flows for components of products in the Design Repository. We use the results to predict the most likely functions and flows for a component, and then verify the accuracy of our algorithm by cross-validating a subsection of the data against the automation results. We apply existing grammar rules to order the functions and flows in a linear functional chain. Ultimately, these findings suggest methods for further automating the process of generating functional models.

1 INTRODUCTION

Product design in engineering is a well-studied process [1], yet many aspects remain difficult and hard to define after decades of research, especially the early stages of concept generation. However, that concept generation phase is the one part of the design process where there is the most room for creativity and innovation [2]. Additionally, the concept generation phase is the least costly time of the design process to integrate major changes[3] and exploration during this phase should be encouraged. We use the term designers broadly to refer to those who are working in their field to develop new concepts or products, as well as iterating on existing

concepts and products. During concept generation in product design, designers focus on gathering accurate customer needs, determining engineering specifics, deriving the functionality of the intended product, and ideating potential form solutions.

Functional decomposition is a well-known abstraction technique that allows designers to develop a graphical representation of a product's functionality as a functional model [3] [4]. There has been extensive work done to develop consistency in the nomenclature, beginning with the development of the Functional Basis terms [5] [6]. However, as Nagel et al. [7] point out, there is still inconsistency in user input in the structure of functional models. Novice and experienced designers all have difficulty building functional models because of the complicated process, and this can lead to the functional modeling process being entirely omitted from the concept generation phase. Yet we know that concept generation is more complete when function is considered [3]. Figure 1 shows how incorporating functional modeling early into the design phase can shift the majority of resources in the project lifecycle earlier in the design process when the cost of making changes is low, but the impact of those changes is high. While the above is true, designers are often more comfortable with component-based solutions, and tend to focus more on components rather than the functionality of a sub-assembly or product. This type of design often benchmarks existing products during the concept generation phase [8].
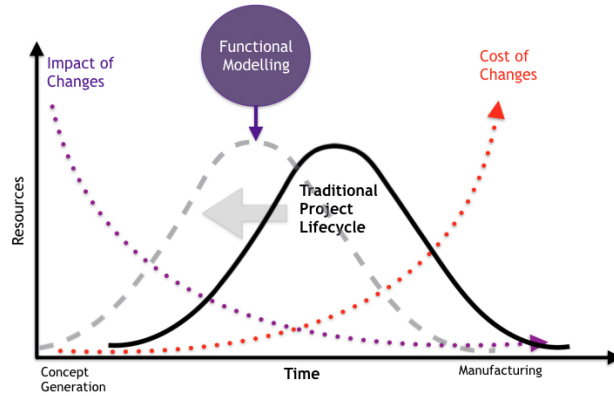

Figure 1: Cost Analysis Over Time During Concept Generation

Eckert and Stacey [9] designate the term "source of inspiration" for the conscious use of previous designs in the design process. Design repositories can provide designers with data at multiple levels of abstraction, such as components, functional representations (e.g. functions and flows), or high level customer needs responses,

offering a central location for "source of inspiration" products. Our research uses data from an existing design repository, know as the Design Repository, to support this type of reuse in design. Our data-driven design (DDD) approach leverages research from the multi-decade long project developing a design repository [10] [11] [12] [13] [14]. For our purposes, we define data-driven design as methodologies for extracting information and insights from data and existing research to improve design processes [15].

We utilize the extensive previous work on the repository and expand the most recent work; the Form Follows Form approach [16]. The Form Follows Form (FFF) approach is based on the concept that most designers think in terms of components rather than function when in the concept generation phase. Utilizing this concept, we capture the underlying functionality of the chosen components using data from the design repository [16]. In the FFF approach, Bohm et al. calculated the frequency of function and flow associated with components *separately*. Our research continues to build on this concept by developing a *combined* association between component and function-flow of CFF, to predict the most likely functions and flows associated with each component. We choose to only consider the incoming flows to simplify our analysis, as we found in analyzing our datasets that less than 5% of the results have different inflow and outflow. With this data, we build linear functional chains for components. These linear functional chains will ultimately help us develop a method for automating the creation of functional models. As this research develops, machine learning from the *combined* CFF combinations is anticipated to help eliminate errors such as illogical or impossible CFF combinations in attempting to combine them later during functional modeling automation.

There is significant research on developing consistency in the grammar and syntax of functional models [5] [6][13]. The Design Repository has this consistency in language built into the data, for example, functions are entered using the Functional Basis terms [5], and components with the Component Basis terms [13]. This terminology allows us to create correlations that remain consistent throughout the datasets.

Our immediate research objectives are to 1) mine the design repository for datasets 2) calculate frequencies of CFF combinations and apply a classification threshold with an automation algorithm, 3) validate the accuracy of the automation algorithm, and 4) apply existing rules to develop linear functional chains based on our findings.
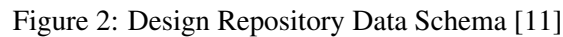
2 BACKGROUND

2.1 The Design Repository

A design repository is a product database where data can be searched and retrieved at different levels of abstraction to help improve design knowledge and data-driven design decisions[10]. A well-populated repository offers designers a wealth of information to aid in decision making. The Design Repository[1] we are using is comprised of 142 consumer-based electro-mechanical products and is housed online through the Design Engineering Lab at Oregon State University. Each product is divided into seven main categories of design information: artifact, function, failure, physical, performance, sensory, and media-related information types. A visual reference of the data schema (i.e., the connections between data) is shown in Figure 2 [11]. The repository database was created by reverse engineering products, and ultimately determining the existing connection between the component and function and flow. By using existing products, these CFF connections are also known to exist, however the limitation we are wrking with is there are connections that exist beyond the products in the database. As we grow the data in the repository we will also grow the CFF connections, as well.

While there is significant research and information available on how to build functional models [3] [4] [17], Sridharan and Campbell point out that even though there is a formal language, the Functional Basis [5] [6], repeatability is still challenging among students and researchers[18]. To help solve the consistency issue with building functional models, Nagel et al. [7] developed grammar rules. When applied correctly, these grammar rules help determine the appropriate order of the linear functional chains. Additional researchers, Sridharan and Campbell and Bohm and Stone, developed grammar rules and tested their rules with students building functional models. The students, given the grammar rules, created more consistent functional models, and had a better understanding of functional decomposition than the students without the grammar rules[14] [16]. Bohm and Stone developed rules associated with individual functions and dictate the allowed incoming and outgoing flows [16]. We found several of the grammar rules in the previous research applicable to our current research. We apply these grammar rules to the data returned from the automation algorithm. While there are some limitations, the functional basis language was used for its repeatability and openness at the time, in that the physics of the solution is not required to be known [17].

---

[1]The Design Repository is a database of design information. A basic web interface is available at ftest.mime.oregonstate.edu/repo/browse

Figure 2: Design Repository Data Schema [11]

2.2 Machine Learning and Data Mining

Data mining and machine learning are general terms that refer to many different techniques of using information to predict results. The work that we are doing is considered data mining because our algorithms extract knowledge from the data and do not alter it based on the findings, which would be considered machine learning. However, we borrow some of the terms and methods that are traditionally applied to machine learning problems to find patterns within our data.

A classifier is an algorithm learns from data, finds patterns within it, and then predicts whether something is or is not within a class. An example is a classifier that predicts whether or not an email is spam[19]. The machine learning algorithm looks at examples of emails that a person has labeled as *spam* or *not spam* and finds patterns within them to label any other email as *spam* or *not spam*[20]. The accuracy of this classifier is quantified by testing it against other emails labeled as *spam* or *not spam* and recording which predictions were correct and incorrect.

We are using data mining techniques to find the frequency of occurrence of CFF combinations in products in the design repository. We use that frequency information to predict what functions and flows a component will have in a new product.

2.2.1   Frequency

In our previous work [21], we used the Apriori algorithm to find associations between component and function-flow. During data analysis, we found that using association rules was excessive for the results we wanted to obtain. We simplified our calculations, focusing on the frequency of CFF combinations, which is numerically equivalent to the confidence metric from association rules. The Form Follows Form approach uses a similar method of calculating the frequency of the function and flows correlated with each component [16].

Frequency determines the probability of two items appearing in the same itemset. In our datasets, we find the frequency by calculating how often a component and function and flow appeared together. The frequency values for all CFF combinations for each component sum to 100%, regardless of the number of functions and flows per component. For example, the CFF combination *screw* and *couple solid* appears in the consumer products dataset 589 times out of 647 total CFF combinations for the component screw, so the frequency of that combination is 589/647 or 91%.

Sometimes a CFF combination may only appear once if it is an unlikely combination or is a specialized component or function only appearing in one product in the repository, such as *pressure gauge* and *indicate mechanical*. In these cases, the frequency that the CFF combination occurs is 100%.

2.2.2   Threshold

In our work, we are predicting the functions and flows for components, and our threshold is a cutoff that predicts that the top 70% of functions and flows would be likely for future components. This 70% threshold was developed based on previous research by Bohm, who found that 70% of functions and flows are realized within the first 30% of unique instances of a particular component, which he credited to the Pareto optimal gaming theory [16]. Additional optimization research has been done on the threshold values finding the optimum range to start at 55%, however in this work we chose to stick with the more conservative value of 70% [22]. Our automation algorithm orders the frequencies from largest to smallest, sums the frequencies of each CFF combination, and applies a 70% threshold to each component. This algorithm is different from a traditional classifier that would discretely label a class based on individual probability. We found in our data analysis that the 70% threshold is often the point where adding additional functions and flows for a component contributed a negligible delta to the sum of frequencies and decreased the accuracy of the automation results.

2.2.3   Cross-Validation

A common method to find the accuracy in a machine learning classifier is known as cross-validation, which withholds a subset of data from the initial set, so the machine learning algorithm does not learn from this subset. This subset is then used to find the accuracy of how well the classifier performed at predicting results [23]. Testing with data from which the classifier did not learn is essential for reducing bias in the results [24]. The subset of withheld data is known as the *testing set* and the rest of the data that the machine learning algorithm processes is known as the *training set*.

Due to the variability in data, cross-validation is often performed multiple times with different testing and training sets and averaged over all iterations. Kohavi found that 10-fold cross-validation produces the best results for most applications even when additional computational power is available [25], so we use this method to determine the accuracy of our automation algorithm using the metrics of precision, recall, and the F1 Score. The general method is known as *k-fold cross validation*[26].

In our previous work, we used a single product (a Delta jigsaw) for our *testing set*, and the Black and Decker dataset was our *training set* [21]. This initial exploration helped us gain valuable insight into the initial stages of this process, but cross-validation is a more robust method.

2.2.4   Precision, Recall, and the F1 Score

The method and effectiveness of calculating accuracy varies based on the type of data being used. Simple accuracy is calculated as a ratio of correct responses to total responses. In our case, a correct response is when the data mining algorithm finds a function-flow combination for a component that matches the *testing set*. Simply counting correct responses misses some of the additional ways in which the automation can be wrong. Precision, recall, and the F1 score account for these cases by using the confusion matrix shown in Table 1 to calculate ratios of the true positives, false positives, and false negatives [27]. Note that true negatives are not included in the calculation for metrics because, for many systems, including ours, most results are true negatives, and including these in our accuracy calculations would highly increase our results and make the classifier appear to be performing better than it is.

Precision is the ratio of correct CFF combinations to all CFF combinations identified by the automation algorithm (Equation 1). This number is the ratio of CFF combinations that were identified as being in the product that are actually in the product.

Recall is the ratio of correct CFF combinations to all CFF combinations found

Table 1: Accuracy Confusion Matrix

		Predicted CFF?	
		Yes	No
Actual CFF?	Yes	True Positive	False Negative
	No	False Positive	True Negative

in the automation algorithm (Equation 2). This number is the ratio of the actual CFF combinations that were correctly predicted.

The F1 Score is the harmonic mean of precision and recall that equally balances the importance of the two metrics and punishes extremes (Equation 3). F1 is a more powerful metric than simple accuracy and provides a better analysis of the ability of an automation algorithm to predict results.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1 = \frac{2 * precision * recall}{precision + recall} \tag{3}$$

3   METHODS

In this work, we mine the design repository for data to find the most likely functions and flows correlated with each component for several datasets. We refer to this correlation as component-function-flow (CFF). We are building on previous work using association rules, where we found associations between component-function-flow on a single dataset (12 Black and Decker products) [21]. Here, we expand our learning datasets as well as our validation methods. We chose three data subsets from the repository driven by component: 23 products with the component **heating element**, 32 products with the component **blade**, and 44 products with the components **container/reservoir**. We applied our automation algorithm (described later) on each dataset separately and calculated the accuracy of its ability to predict function and flows for an input component. We then compared the accuracy results of each of the three data subsets to a dataset containing all 142 **consumer products** from the design repository and an additional subset containing 12 products that were all made by **Black and Decker**.

We chose products with the **heating element, blade**, and **container/reservoir** components in an attempt to single out products with similar functionality. We

chose the Black and Decker products because this was the most extensive dataset available to provide a company product portfolio, which offers a subset of products based on construction rather than functionality. *We hypothesize that narrowing the dataset to functionality based on component will yield more accurate function and flow prediction results.* The Black and Decker and consumer product datasets serve as reference datasets to test our theory. We developed an automation algorithm in Python, which calculates and sums the frequency of each CFF appearing in each dataset. The algorithm applies a classification threshold to the top 70% sum of frequencies for each component. The correlations found within the 70% threshold in our data mining process can then be used to predict the linear functional chain of a component.

Step 1. Retrieve Datasets. To extract information, we query the repository to create five test datasets: 1) all **consumer products**; 2) all **Black and Decker** consumer products to represent a general product family by the same manufacturer; and three subsets of consumer products with 3) **heating element**, 4) **blade** and 5) **container/reservoir** as a component in the assembly to represent products with a similar component and functionality. We chose to combine reservoir and container into one dataset because of the similarity of functionality, and combining them allowed us to have a similar size dataset as the other two component-based datasets.

Step 2. Automated frequency and thresholding Next, we apply an automation algorithm to each of the five datasets, implemented in Python v3.7. First, the algorithm calculates the frequency of the functions and flows for each component in the input dataset; then, those values are sorted from largest to smallest, summing to 100%. The threshold is applied to capture the top 70% of the sum of frequency values for each component in each dataset, based on the Pareto Frontier from the Form Follows Form method [16]. The results from the electric cord component from the blade dataset provide a simple example in Table 2.

Table 2: Example dataset to illustrate threshold automation for the component Electric Cord

Electric cord	Frequency delta	Running sum of frequency	Threshold
Import electrical	0.35	0.35	Keep
Transfer electrical	0.3	0.65	Keep
Export electrical	0.15	0.8	Keep
Position solid	0.12	0.92	Reject
Couple solid	0.08	1	Reject

For the electric cord example, the frequency of the first two functions and flows sums to 65%, so the third is added to the list to reach the 70% threshold, which

brings the sum to 80%. Our method assumes that capturing approximately 70% of the total frequencies will begin to give an accurate representation of the functions and flows that a component usually performs. Additionally, this Pareto optimal threshold is the point where adding additional functions and flows for a component usually contributed a negligible delta to the sum of frequencies and increased the error in the automation results.

Step 3. Cross-Validation. As a means of verifying the accuracy of the automation algorithm, we use a 10-fold cross-validation method to find the precision, recall, and F1 score of each iteration. The design repository categorizes products by an identification number, which we randomize and separate into ten folds.

For example, the blade product dataset contains 32 products. This number is not divisible by ten without a remainder, and our data requires each product to remain intact, so we actually have eleven folds. Ten folds have three products each, and the eleventh fold has the remaining two products.

We apply the frequency calculation and thresholding algorithm three times for each dataset that we queried. The first validation is a traditional cross-validation and finds the accuracy of the automation algorithm when the training set comes from the component-specific dataset (blade, heating element, reservoir/container), and the testing set is also the component-specific dataset. The second validation finds the accuracy when the training set is the consumer products dataset, and the testing set is the component-specific dataset. The third validation uses the Black and Decker dataset as the training set and the component-specific dataset as the testing set.

We stray from traditional cross-validation in two of three of these validation tests by selecting the folds for the testing set and training set from different datasets. This method gives us a cross-reference for accuracy between datasets and allows us to see if one dataset is better at predicting results for itself or for another dataset. The three variations of accuracy testing are shown in Table 3. These three validations were performed for each of the three component-specific datasets, resulting in nine F1 scores.

One of the benefits of traditional cross-validation is that the testing set is withheld from the training set to reduce bias in the results. With this method of validation, when the testing set and training set come from different datasets, the folds contain some overlapping data. To combat bias, we made sure to remove all products in the fold for the testing set that were also in the folds for the training set.

Step 4. Apply Grammar Rules to Determine Linear Functional Chain. After analyzing and organizing the results of the top 70% of the functions and flows, we apply the grammar rules described in section 2.1 to the results to determine the linear order of the functions in the functional representation. For the electric cord example in Table 2, the three functions are import, transfer, and export. They all have the same flow of electrical energy between them. The grammar rules developed

Table 3: Validation Cases

Validation Number	Testing Set Source Dataset	Training Set Source Dataset
1	Component-specific	Component-specific
2	Component-specific	Consumer products
3	Component-specific	Black and Decker products

by both Nagel et al. [7] and Bohm et al. [16] state that the import function occurs first and only once per flow in a chain of components, so it is placed first. The grammar rules also state that export is the last function in a chain of components, which leaves transfer as the middle function in this chain. A visualization of this process can be seen in Figure 3.
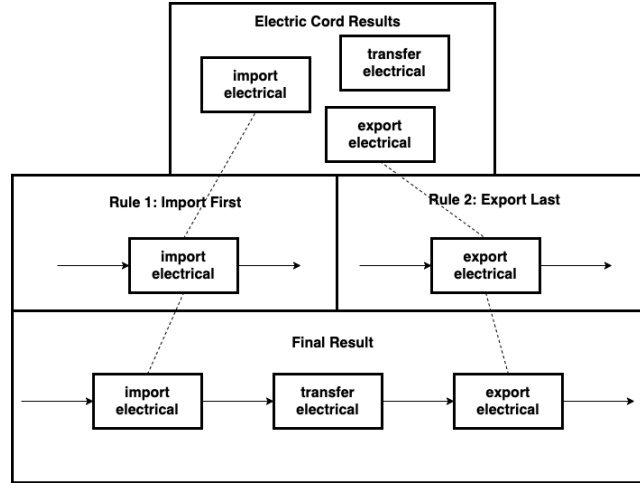

Figure 3: Example of Grammar Rule application

4   RESULTS

4.1   SQL Query

The results of our SQL query can be seen in Figure 4. The total number of CFF combinations is the number of times a component has a particular function and flow regardless of the number of times they repeat in the dataset. The number of unique combinations is the number of times a component has a particular function and flow at least once, and additional instances of that combination are no longer unique. The number of products in each dataset can be seen in Table 4.
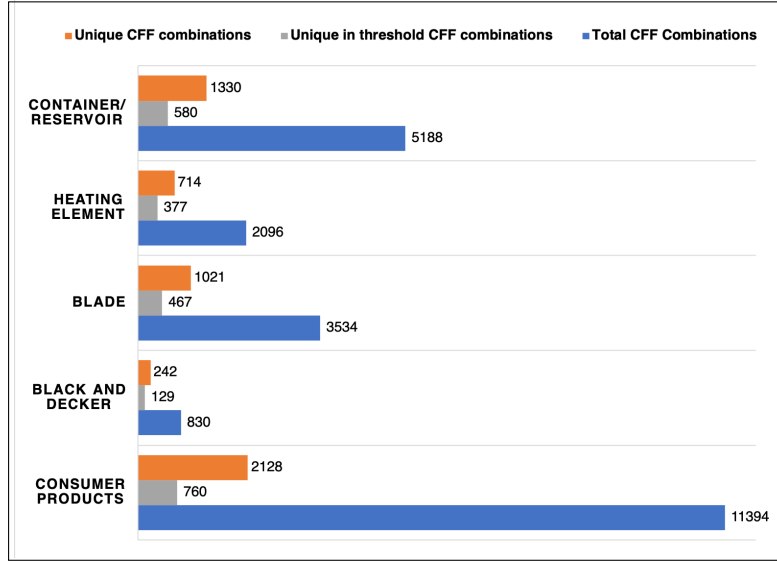
Figure 4: Query Results

4.2   Automated Frequency Calculation and Thresholding Algorithm

The algorithm returned CFF combinations for all five datasets filtering out the combinations that were above the threshold. Figure 5 shows the CFF combinations for four components in the consumer products dataset, results above the black line are the CFF combinations found within the threshold. As seen in Figure 5, the 70% threshold is often the point where adding additional functions and flows for a component contributed a negligible delta to the sum of confidence. In order to remain consistent, all of the results are taken from the consumer database.

As seen in Figure 5 A, the component screw had one result above the threshold because the frequency result for *couple solid* is 91%, the remaining 17 functions and flows only contribute to 9% of the results . While screw only has one result, blade (Figure 5 D) is representative of a component with more function and flows returning 11 results above threshold, an additional 21 results below threshold were not shown, for clarity in the figure. Washer and heating element can also be seen in Figure 5. Additionally, components with the most results were reservoir, circuit board and wheel with 22, 20, and 16 function and flow combinations in threshold respectively. We found that 98% of the dataset has at least 2 or more CFF combinations per component.

Frequency is calculated as the ratio of the number of times the CFF combination occurs over the total number of CFF combinations for that component. Returning

to the screw example, the function and flow *screw couple solid* occurs the most at 589 times out of a total of 647. Conversely the consumer products dataset had 200 CFF combinations that only occurred once, which returns a ratio of 1/1 or 100% frequency. The other four datasets followed this same trend with a larger percentage of results occurring once or twice and a lower percentage occurring more than 20 times. As we expand the data in the repository we hope to decrease the number of times a CFF combination occurs only once.
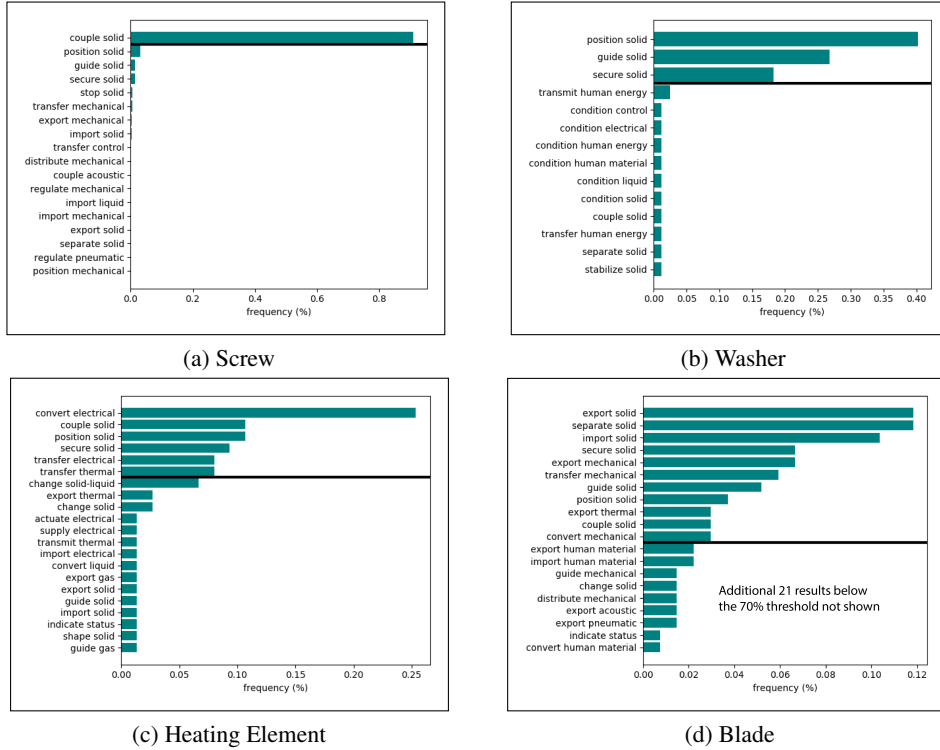

(a) Screw

(b) Washer

(c) Heating Element

(d) Blade

Figure 5: Frequency Algorithm Results for Components in the Consumer Products Dataset

4.3   Using F1 Scores to Validate Accuracy

We used the 10-fold cross validation method to quantify the accuracy with precision, recall, and the F1 score when applying the top 70% of the most frequent functions and flows found for a component for each of the three testing datasets. The number of products in each dataset, the size of a single fold (which is also the size of a testing set), and the size of the remaining nine folds (the size of the training set) is

shown in Table 4.

Table 4: Sizes of Testing and Training Sets

Dataset	Number of Products	Size of Testing Set (Size of Single Fold)	Size of Training Set (Size of Nine Folds)
Blade	32	3	29
Heating Element	23	2	21
Reservoir	44	4	40
Consumer	142		
Black and Decker	12		

Each training dataset is tested against three testing datasets, itself, consumer products, and Black and Decker products. The results of the average F1 scores are shown in Table 5. For each of the three testing datasets, we performed a single factor ANOVA test to see if there is a significant difference in our F1 scores across the training datasets. We found that all three testing datasets (**blade, reservoir/container, and heating element**) were significantly different with $\alpha=0.05$. We then performed a two-sample t-Test assuming equal variances to determine within each testing dataset, which training sets were significantly different. Within each testing dataset, there was a significant difference found for all combinations, except the comparison of **<u>Blade and Black and Decker</u>** within the **Blade** testing set. The direction of the significant difference trends toward the consumer products dataset, which consistently had the highest F1 score.

Table 5: F1 Scores

Testing dataset: Blade		Testing dataset: Reservoir		Testing dataset: Heating Element	
Training dataset	**F1 scores**	**Training dataset**	**F1 scores**	**Training dataset**	**F1 scores**
Blade	0.4354	Reservoir	0.3967	Heating Element	0.4044
Consumer	0.4471	Consumer	0.4072	Consumer	0.4067
B&D	0.4419	B&D	0.2937	B&D	0.3215

4.4   Linear Functional Chains

In this section, we test the automation process described in the methods by building likely functional chains for four single components, which are the same four components featured in section 4.2, Figure 5. The results of the functional linear chains can be seen in Figure 6. Screw is a very simple example with only one function and flow. As demonstrated in the results, components vary in complexity and therefore vary in functional chains. This complexity is based both on the component itself, such as the difference between screw and blade, but is also based on the product in which

the component performs the function, for example blade within a knife versus blade within a more complex product like a jigsaw. We apply the following grammar rules adapted from Bohm and Stone, to the blade functional chain; a) import is automatically placed as the first function for a chain and b) export is automatically placed as the last function for a chain [16]. The grammar rules also dictate that the convert function has separate inflows and outflows; therefore the automation would branch off the function-flow export thermal from convert mechanical. This same rule is applied to the results of heating element, convert electrical is branched off into transfer thermal.
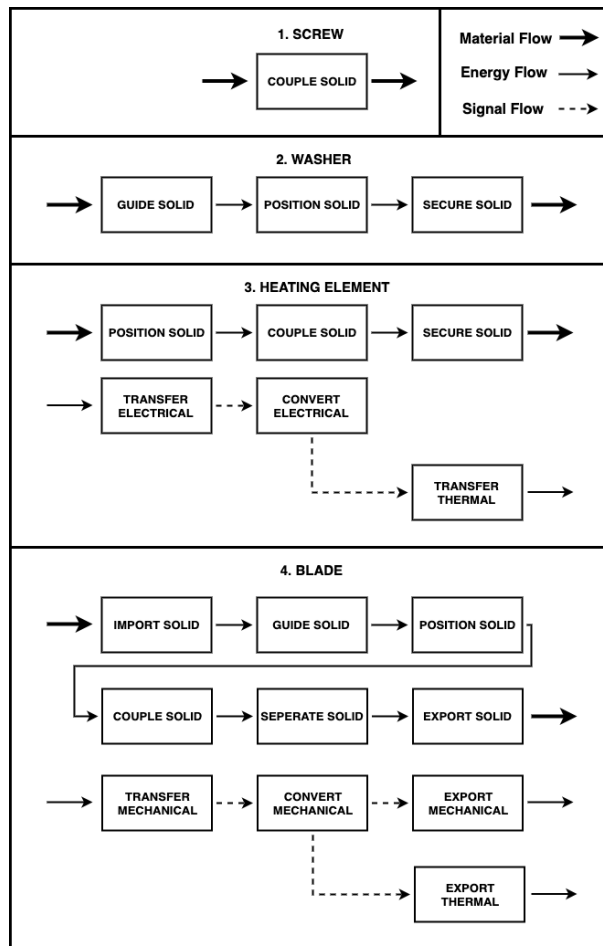

Figure 6: Component Based Linear Functional Chains

5   DISCUSSION

In review, we mined the Design Repository for CFF combinations, we then applied a Pareto optimal threshold to find the most likely combinations, developed linear functional chains for individual components, and by validating the accuracy of the frequency calculation and thresholding algorithm, we were able to test our hypothesis. We hypothesized that restricting the training set to constitute products that all share a similar component would give more accurate results for automating the generation of linear functional chains. For example, products having the component blade would have more similar functionality with other products having the component blade as opposed to products outside that dataset. As stated before, we found support with this hypothesis in previous work, using one product, the Delta jigsaw, as a validation method [21].

In this research, we had four general findings: **Finding 1:** With more robust validation methods, the results in Table 5 show that *learning from the most possible products will return a higher accuracy than any restricted-size dataset.* The component-specific datasets had lower accuracy when cross-validated against component-specific data than when cross-validated against all consumer products. The Black and Decker dataset is the smallest, containing 12 products, and consistently had the lowest F1 score when used as the training set. The consumer products dataset is the largest, containing 142 products, and consistently had the highest F1 scores.

Finding 2: We suggest that because the F1 score is calculated for an entire testing set, which often contains rare components that might have only one function and flow in the testing set, this may decrease the overall accuracy of function and flow results per component. As is often the case in large datasets, the accuracy of the data input can be a concern. Over the 20 years of the development of the Design Repository, many different contributors have worked on this project. This turnover has led to some inconsistencies in the data; for example, container and reservoir are often used interchangeably or as seen in Figure 5, *screw* is 91% correlated with *couple solid* but there are 17 other results, which could be due to individual input variations. *This noise of the additional rare or mislabeled CFF combinations in the datasets can certainly reduce the accuracy of the results, especially for the larger consumer products dataset.*

Finding 3: While finding 1 suggests that learning from more data returns more accurate results, restricting the dataset based on the component may return more refined results for functionality. For example, the heating element, and reservoir/container component-specific datasets have six CFF combinations for the component *heating element*, the consumer products dataset has 10, and the blade dataset only returned one result. Heating element and reservoir/container

have a high overlap in products, such as coffee makers, but blade products are unlikely to contain heating element as a component. *There may be times when a designer desires more refined results and a smaller learning dataset can be used if the products have the component of interest in the learning set.*

Finding 4: In developing the linear functional chains, we demonstrated more simple examples, such as *screw* and *washer*. As complexity increases, grammar rules are necessary to order the function and flow results. Only two existing grammar rules applied to our findings in heating element and blade. *As we expand our work in developing linear functional chains, we will need to expand on the research around grammar rules to create additional rules required to connect flows at the interface of components.* Individual analysis allows for the development of new rules to handle each situation, but automation is possible based on investigating the interactions between component, function, and flow. While significant future work is required to fully automate the functional modeling of a product, these findings offer a starting point.

6 CONCLUSION

Functional modeling is a complicated and challenging process for both novice and expert designers. However, during concept generation in product design, it is imperative to derive the functionality of the intended product because the function of the product is critical in linking customer needs to a form solution. We know from research, designers often think and design in component-based solutions. Our research finds connections between component and function and flow, information that can provide the designer with the functional breakdown of components. A functional approach to design is specialized; functional design accounts for variance in design for different purposes. We used data from the Design Repository to find the CFF combinations for five datasets: all consumer products, a Black and Decker consumer product family, and consumer products with the component blade, heating element, and reservoir. Our automation algorithm orders the frequencies from largest to smallest, sums the frequencies of each CFF combination, and uses a threshold of 70% of the sum of frequencies of combinations for each component. This threshold is the point where most of the functionality is preserved with a minimal contribution of error.

We then applied existing grammar rules to create component-based linear functional chains, the first step in automating functional modeling. Our results confirm the notion that function and flow correlations can be used to build a linear functional chain of individual components within a product. We found that the accuracy of data mining depends on the size and quality of the learning set used, with larger

datasets providing more accurate results. However, using a broad or narrow dataset will depend on the goals of the designer.

Research has found inconsistency among designers when building functional models. We think that our future work towards an automated functional model generator will ultimately help standardize the language and syntax used in functional models, just as the work on Functional Basis and Component Basis terms have helped improve language and syntax consistency in the repository. As we have seen in the data in the design repository, designers are individuals dealing with human bias and perceptions; automation can help create more uniform functional models.

This uniformity will improve the process of designers contributing to the design repository, and enable more products to be added with higher consistency. The repository provides the user with a wealth of information; however, in its current form, the repository can be challenging to navigate for novice users. As stated above, streamlining the process of adding new products with automating functional modeling allows not only individual products to be added by users but also the addition of entire repositories. Enabling products to be entered by users will further increase the size and quality of the data in the Design Repository and increase the accuracy of our automation process. This automation will also allow engineers to design a new product based on components and receive the functionality of the components.

References

[1] Otto, K., and Wood, K., 2003. *Product design: techniques in reverse engineering and new product development*.

[2] Yang, M. C., 2009. "Observations on concept generation and sketching in engineering design". *Research in Engineering Design,* **20**(1), pp. 1–11.

[3] Ullman, D. G., 1992. *The mechanical design process*, Vol. 2. McGraw-Hill New York.

[4] Beitz, W., and Pahl, G., 1996. "Engineering design: a systematic approach". *MRS BULLETIN,* **71**.

[5] Stone, R. B., and Wood, K. L., 2000. "Development of a Functional Basis for Design". *Journal of Mechanical Design,* **122**(4), dec, p. 359.

[6] Hirtz, J., Stone, R., McAdams, D., Szykman, S., and Wood, K., 2002. "A functional basis for engineering design: Reconciling and evolving previous efforts". *Research in Engineering Design,* **13**, pp. 65–82.

[7] Nagel, R. L., Vucovich, J. P., Stone, R. B., and McAdams, D. A., 2008. "A Signal Grammar to Guide Functional Modeling of Electromechanical Products". *Journal of Mechanical Design,* **130**(5), may, p. 051101.

[8] Miller, S. R., and Bailey, B. P., 2014. "Searching for Inspiration: An In-Depth Look at Designers Example Finding Practices". In Volume 7: 2nd Biennial International Conference on Dynamics for Design; 26th International Conference on Design Theory and Methodology, ASME, p. V007T07A035.

[9] Eckert, C., and Stacey, M., 2000. "Sources of inspiration: a language of design". *Design studies,* **21**(5), pp. 523–538.

[10] Szykman, S., Sriram, R., Bochenek, C., Racz, J., and Senfaute, J., 2000. "Design repositories: engineering design's new knowledge base". *IEEE Intelligent Systems,* **15**(3), may, pp. 48–55.

[11] Bohm, M. R., Stone, R. B., Simpson, T. W., and Steva, E. D., 2008. "Introduction of a data schema to support a design repository". *Computer-Aided Design,* **40**(7), jul, pp. 801–811.

[12] Bryant, C. R., McAdams, D. A., Stone, R. B., Johnson, J., Rajagopalan, V., Kurtoglu, T., and Campbell, M. I., 2008. "Creation of Assembly Models to Support Automated Concept Generation". In ASME 2005 International Design Engineering Technical Conferences, pp. 259–266.

[13] Kurtoglu, T., Campbell, M. I., Bryant, C. R., Stone, R. B., and McAdams, D. A., 2005. "Deriving a Component Basis for Computational Functional Synthesis". *ICED 05: 15th International Conference on Engineering Design: Engineering Design and the Global Economy*, p. 4061.

[14] Sridharan, P., and Campbell, M. I., 2005. "A study on the grammatical construction of function structures". *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM,* **19**(3), pp. 139–160.

[15] Sailer, K., Pomeroy, R., and Haslem, R. Data-driven design-Using data on human behaviour and spatial configuration to inform better workplace design. Tech. Rep. 3.

[16] Bohm, M. R., and Stone, R. B., 2010. "Form Follow Form - Fine Tuning Artificial Intelligence Methods". *Proceedings of the ASME 2010 International Design Engineering Technical Conferences and Computers and Information*, pp. 1–10.

[17] Sen, C., Summers, J. D., and Mao, X., 2019. "A Physics-Based Formal Vocabulary of Energy Verbs for Function Modeling Knowledge-based self-generating function modeling system". In Proceedings of the ASME 2019 International Design Engineering Technical Conferences IDETC/CIE2019.

[18] Kurfman, M. A., Stock, M. E., Stone, R. B., Rajan, J., and Wood, K. L., 2003. "Experimental Studies Assessing the Repeatability of a Functional Modeling Derivation Method". *Journal of Mechanical Design,* **125**, pp. 682–693.

[19] Conway, D., and White, J. M., 2011. *Machine Learning for Email: Spam Filtering and Priority Inbox.* O'Reilly Media, Inc.

[20] Lanzi, P. L., 2000. *Learning classifier systems: from foundations to applications*. No. 1813. Springer Science & Business Media.

[21] Tensa, M., Edmonds, K., Ferrero, V., Mikes, A., Soria Zurita, N., Stone, R., and DuPont, B., 2019. "Toward Automated Functional Modeling: An Association Rules Approach for Mining the Relationship between Product Components and Function". *Proceedings of the Design Society: International Conference on Engineering Design,* **1**(1), jul, pp. 1713–1722.

[22] Mikes, A., Edmonds, K., DuPont, B., and Stone, R. "Optimizing an algorithm for data mining a design repository to automate functional modeling". No. IDETC2020-22346, American Society of Mechanical Engineers.

[23] Geisser, S., 1975. "The predictive sample reuse method with applications". *Journal of the American statistical Association,* **70**(350), pp. 320–328.

[24] Stone, M., 1974. "Cross-validatory choice and assessment of statistical predictions". *Journal of the Royal Statistical Society: Series B (Methodological),* **36**(2), pp. 111–133.

[25] Kohavi, R., 2001. "A study of cross-validation and bootstrap for accuracy estimation and model selection".

[26] Duchesne, P., and Rémillard, B., 2005. *Statistical modeling and analysis for complex data problems*, Vol. 1. Springer Science & Business Media.

[27] Sokolova, M., Japkowicz, N., and Szpakowicz, S., 2006. "Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation". In Australasian joint conference on artificial intelligence, Springer, pp. 1015–1021.