Proceedings of the ASME 2020 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC2020
August 16-19, 2020, St. Louis, Missouri, USA

IDETC2020-22498

USING DECISION TREES SUPPORTED BY DATA MINING TO IMPROVE FUNCTION-BASED DESIGN

Vincenzo J. Ferrero
School of Mechanical, Industrial and Manufacturing Engineering
Oregon State University
Corvallis, Oregon, USA
FerreroV@oregonstate.edu

Naser Alqseer
School of Mechanical, Industrial and Manufacturing Engineering
Oregon State University
Corvallis, Oregon, USA
AlqseerN@oregonstate.edu

Melissa Tensa
School of Mechanical, Industrial and Manufacturing Engineering
Oregon State University
Corvallis, Oregon, USA
TensaM@oregonstate.edu

Bryony DuPont*
School of Mechanical, Industrial and Manufacturing Engineering
Oregon State University
Corvallis, Oregon, USA
Bryony.DuPont@oregonstate.edu

ABSTRACT

Engineering designers currently use downstream information about product and component functions to facilitate ideation and concept generation of analogous products. These processes, often called *Function-Based Design*, can be reliant on designer definitions of product function, which are inconsistent from designer to designer. In this paper, we employ supervised learning algorithms to reduce the variety of component functions that are available to designers in a design repository, thus enabling designers to focus their function-based design efforts on more accurate, reduced sets of potential functions. To do this, we generate decisions trees and rules that define the functions of components based on the identity of neighboring components. The resultant decision trees and rulesets reduce the number of feasible functions for components within a product, which is of particular interest for use by novice designers, as reducing the feasible functional space can help focus the design activities of the designer. This reduction was evident in both case studies: one exploring a component that is known to the designer, and the other looking at defining function of an unrecognizable component. The work presented here contributes to the recent popularity of using product data in data-driven design methodologies, especially those focused on supplementing designer cognition. Importantly, we found that this methodology is reliant on repository data quality, and the results indicate a need to continue the development of design repository data schemas with improved data consistency and fidelity. This research is a necessary precursor for the development of function-based design tools, including automated functional modeling.

INTRODUCTION

Engineering designers are faced with solving multi-faceted global design problems and increasingly complex modern product design. In recent decades, design researchers are focusing on *DfX* areas such as designing for reduced environmental impact, full global market acceptance, optimized assembly, and end-of-life processing. Designers are challenged to meet design objectives through the use of memory, previous experience, and expertise, supplemented by external design tools. However, research finds that novice design engineers lack confidence in designing to meet complex *DfX* goals [1,2]. Today, there is a global push for creating multi-disciplinary design teams to supplement expertise and improve design innovation [3,4]. No longer is the designer title limited to classically trained engineers, but can now include medical, humanitarian, environmental, or trend-analysis professionals, to name a few. While being multi-disciplinary is important (as combining professional expertise results in improved designs), some foundational design principles, like function-based design, may

*Corresponding Author: bryony.dupont@oregonstate.edu

not be in the wheelhouse of these designers or novice design engineers, thus rendering these teams more reliant on other design tools.

Foundational design is rooted in designing tools, systems, and products that serve a purpose by completing functions [5]. In this regard, we define function as a critical basis of design. Function in design, specifically through the use of functional modeling, is a fundamental step in the development of new concepts and design ideas [6]. Functional modeling is a tool that is used to identify the functional interactions that occur within a concept. One potential benefit of functional modeling is that designers can use them to inspire creative ways to solve functional needs [7]. However, even in function-based engineering design, there is subjectivity in how to define a function or functional model. Functional models for the same design, but created by different designers, often are widely different [8].

Many previous works have looked into rectifying the subjectivity of function-based design. In 2000, Stone and Wood introduced standardized vocabulary to improve the readability, repeatability, and understanding of functional modeling [8]. In the following year, Hirtz et al. reconciled similar functional vocabulary methodologies into what is known as the *functional basis* [9]. The success of this research helped influence the creation of objective vocabularies to describe components, called the *component basis* terms [10]. Research since then has expanded the functional basis to include biological functions [11]. These works have undoubtedly reduced the subjectivity in function-based design by formalizing the vocabulary necessary for using function during the design process. However, when considering both the increasing complexity of design and the potential for less-experienced designers working on engineering design activities, we still observe the subjective assignment of functional basis terms to components.

Novice designers may be unaware of the nuanced differences between component functions; for example, the functions “couple” versus “secure.” Additionally, designers can be unfamiliar with components and the functions those components can perform. The assignment of functions to components is commonly thought to be an objective exercise, but can become subjective even when using the standardized functional basis terms. There is a need to explore how to limit the subjective assignment of component functions. In doing so, designers can make smarter decisions about designing for function.

In this paper, we use design repository data in conjunction with data mining techniques to limit the number of functions a component is likely to perform by considering both the components themselves and their neighboring components in the product assembly. We suggest that the methods presented here can limit a designer to only select from feasible, ‘correct’ functions while avoiding having to consider irrelevant ‘wrong’ functions. Providing a set of correct functions is relevant considering designers may not be familiar with the breadth of functions a component can perform or with a component itself, or the designer may lack significant experience in the function-based design domain.

PREVIOUS RESEARCH

Design repositories facilitate the use of design data in sophisticated methodologies and exist to enhance the availability of design knowledge [12,13]. Currently, design repositories house product information such as a bills of materials, cost data, CAD models, manufacturing processes, functional information, assembly data, and other *DfX*-relevant data. However, to encourage wider use of design repositories in both industry and research, efforts have been made to standardize product data schemas and the searchability of design repositories [14,15]. This research led to the creation of extensively peer-reviewed design repositories and the prevalence of repository data used in research [2,16,17].

Repository data has been used to create tools and methodologies to influence automated concept generation. Rajagopalan et al. created assembly models from repository data to support concept generation [18]. Bryant et al. explored computational techniques to automate concept generation using repository data [19]. The following year, Bryant et al. validated this methodology by testing the automated concept generation tool with undergraduate researchers [20].

Recent research in repository use has focused on solving modern design challenges related to the *DfX* areas of environmental and social sustainability. In 2012, Gilchrist et al. used repository data to understand if innovative products were less environmentally friendly [21]. This research influenced the exploration of eco-labeling practices of design repository products [22]. Soria et al. introduced a function-human design method for identifying human error during the functional decomposition of a product [23]. Later, the function–human error design method (FHEDM) was tested using data from 148 products sourced from a design repository [24]. Though this work is relevant, we suggest the future of repository data ‘learning’ is going to stem from data mining techniques rather than product comparison or research validation.

In product design research (outside of the use of repository data), there have been extensive efforts to use data mining. Here we highlight the area of cellular phone design to showcase a concise representation for data mining in product design. Tucker and Kim used decision trees to create 46 feasible cellular phone design concepts from 40,000 simulated customer responses. In this work, feasibility is measured by a profitability approach [25]. Tucker and Kim also used text mining and regression to characterize trends in cellphone design space. In 2011, Bae and Kim used association rules and decision trees to define the feasible design space of future successful cellphone design based on customer data [26]. Recently, Zhan et al. used association rule mining to identify the correlation between customer preference and essential phone features, thus again highlighting feasible areas of cellphone design improvement [27]. Other product design domains follow a similar progression of data-driven design research. However, we observe that there is an overabundance of research using generated data and customer

data, and an under-representation of product data in data-driven product design research. Only recently has product data, sourced from a design repository, been used in data-mining-based product design research. Thus, we suggest there is a gap in implementing novel methodologies using repository data and data mining.

Wisthoff et al. leveraged the use of machine learning methods on repository data through quantifying the environmental impact of early design decisions using neural networks [28]. This work provides early evidence of success in using data mining to advance product design processes while using repository data versus consumer data. Soria et al. used association rule mining on repository data to provide an understanding of how humans interact with product systems and identifying human errors [29]. Tensa et al. used a similar methodology to advance techniques in automated functional modeling [30]. Based on the current state-of-the-art, our work continues the exploration of using data mining approaches on repository data to develop novel design methodologies that facilitate the design process.

RESEARCH MOTIVATION & SCOPE

Here we expand on previous research in data-driven design to improve the information quality provided to designers to supplement design cognition, particularly for novice designers or for designers working in novel *DfX* applications. Specifically, we aim to address empirical concerns in establishing repository-based data-driven design methods, First, there is a struggle to comprehend bulk data and distill that information into readily applicable design knowledge. As such, we observed much of repository-based research focuses on drawing high-level *DfX* conclusions from data rather than developing methodologies directly applicable by designers. Second, though data-driven design research offers increased utility, we observed that data-driven design employs the use of user data or generated test data. Our goal is to improve the utility of data-driven design by employing only high-fidelity function data from a product repository.

In this paper, we use *Classification based on Association (CBA)* and *CART (Classification and Regression Trees)* with repository-based data [31,32]. The results of our work can offer supplemental knowledge in the form of visual-based decision trees and hard-and-fast rules that a designer can use to aid in the identification of component function(s).

We hypothesize that this method can provide insight to designers and researchers alike by making function-based design approaches more truthful and useful. The goals of this research extend beyond just accurately classifying the function of known and unknown components. Additionally, we are looking to define and limit the feasible design space of function identification objectively. Reducing the subjectivity of function-based design will enable designers to make more informed design decisions, and to approach the more subjective and creative elements of the design process with a firmer grip on functional knowledge.

METHODOLOGY

In this research, we explore limiting the feasible space of component function identification through two case studies. Case Study 1 focuses on defining the feasible functional space of an *unknown* component, and Case Study 2 defines the feasible design space for a known component (in this case, a 'bracket'). The two case studies are selected to explore conditions in which a designer is familiar and unfamiliar with an observed component. Both case studies feature a data subset sourced from the Oregon State Design Repository [14,16]. Each datum represents a component, the component's function(s), and a list of neighboring components. Both of these case studies employ the use of *CART* decision trees and *CBA* analysis.

Data Selection and Processing

The Oregon State Design Repository (OSDR) [14,16]. features a peer-reviewed data schema that houses function-based product information, including components, functions, and assembly connections. In addition to these product data, the OSDR incorporates Design for Assembly (*DfA*), Design for Human Interaction (*DfHI*), and Design for the Environment (*DfE*) data for some products and systems.

The OSDR is selected as our data source due to the external validation of the functional analysis of products and components. *Function* in this context is the designed-for task that a component performs. The OSDR was created to help define objective aspects of functional design, which aligns closely with the goals of this research. One step toward this goal is the OSDR's inclusion of using functional and component basis terms.

From the OSDR, we sourced the common name and the component-basis name for components belonging to 234 unique products, including 7245 components. In the OSDR, components are described by a common name, assembly relationships (to other components), the name of the product in which they are found, material, function, and component basis. For the case studies, each datum represents a single component basis/function basis pair with three assembly-based data features: the parent, grandparent, and great-grandparent of the component.

Here we define the component *origin* as the primary component we are assessing. As with classic familial hierarchy, the origin component is connected to the parent component, the parent component is connected to a grandparent component, and a grandparent component to a great-grandparent component. It is important to note that data fidelity decreases as we explore further into the component connections. In this case, great-grandparents are often denoted by weakly descriptive basis names such as 'assembly' or 'system.' In light of this, we selected a depth of three familial components as further exploration is empirically deemed to be uninformative.

From the 7245 components and corresponding component relation information, we selected two data subsets (based on component basis terms) that represent a functionally diverse known component ('bracket') and set of unknown components that are functional defined but otherwise not classified by a component basis term. The component basis terms are chosen

instead of the common names as common names are more subjective. All components in each data set share the same defined component basis name, while their common names are widely unique. Table 1 demonstrates a sample data point from the known 'bracket' data set and unknown component data set.

TABLE 1. SAMPLE DATA POINT FROM THE KNOWN AND UNKNOWN DATASETS

	Known Component: 'Bracket'	Unknown Component
Common Name	Button Bracket	Controller Antenna
Component Basis Name	Bracket	Unclassified
Function	Secure	Export
Parent	Electric Switch	Circuit Board
Grand Parent	Housing	Assembly
Great Grand Parent	System	System

For the two data points shown in Table 1, the bracket can also be identified by its common name of 'Button Bracket' and the unknown component has a common name of 'Controller Antenna'. The common name is not used in this methodology but is retained for bookkeeping and can offer some insight into the components that designers are often misidentifying.

The data mining tools described in the next section learn by finding meaningful splits in the data between the classifier and correlating data features. In this case, function is the classifier we aim to define by the familial features of each component. From a data standpoint, we aim to limit the number of reasonably feasible functions (the feasible function space), given a set of designer-identifiable assembly features, that a component can perform.

Decision Tree and Data Mining Techniques

Here, we use two unique data mining approaches to give varying styles of rule interpretation that a designer that could use while exploring a product to define unfamiliar components or component functions. The algorithms used are *CART (Classification and Regression Trees)* and *CBA (Classification based on Association Rules)*. The programing language *R* is used to implement both algorithms [33].

CART (Classification and Regression Trees) will allow the designer to quickly visualize the assembly order about the component of interest (define the parent, grandparent, and great-grandparent components). The designer can follow the branches of the decision trees that best match their identification, finally reaching a node that can narrow down the available functions that component can perform and the likelihood of each function from the truncated set. As such, the proposed decision tree and ruleset has objectively reduced the feasible functional space. In theory, a designer can use their cognitive ability with other context clues to evaluate the subjective function of the unknown or known component.

CART, by default, creates an easy-to-understand diagram that a designer can follow to converge on a set of feasible functions for a component. Less visually, The *CBA* algorithm supplies direct text rules that can be used identify the feasible functions of a component. One purpose of using the *CBA* algorithm in this approach is to verify the design tree rules through an unrelated algorithm. In practice, if a user is still challenged by functional identification via decision trees, they can move to the hard-and-fast rules developed by the *CBA* algorithm. The approaches are chosen because they provide meaningful results that can directly be useful to the designer. This research marks our initial efforts to bridge the gap between information from data mining methodologies and usefulness to real-world designers.

CART (Classification and Regression Trees)

Classification and Regression Trees were introduced in 1984 by Breiman et al. [32]. The data mining classification method, *CART*, creates a decision tree that splits data based on *purity*. Purity is the percentage of data points that are classified toward the most popular label. If a given dataset has three possible classification labels, but 90% of the dataset is classified by one of the labels, then the dataset (or subset) is 90% pure. The algorithm examines each possible split—in our case parent, grandparent, and great grandparent—and quantifies the purity of the data if the algorithm were to split on a given feature. This split is representative of whether the feature is present or not. If the feature is split from the dataset and improves the purity of the most popular classification, then the split is accepted. The *CART* algorithm chooses to split on a feature that will most increase the purity. This splitting and purity search recursively continues until a purity criterion is met, where splitting the data no longer reasonably purifies the dataset.

The *CART* algorithm is implemented through the use of the *R* package, *RPart* [34]. This package creates a decision tree using the *Gini* splitting criteria (a type of purity measure). The program creates an exhaustive decision tree that is pruned according to a user-specified complexity parameter. This value is generally tuned to reduce cross-validation error, thus reducing overfitting the tree. Here we are looking for representative trees of our data rather than optimally fit trees. As such, we reduced the complexity parameter (CP) from 0.01 to 0.005. The complexity parameter is an advisory parameter that defines the threshold of purity improvement per split. The CP saves computing time by not fully exploring all possible splits.

RPart uses cross validation to help users define the 'best' CP for creating a pruned tree with the lowest classification error rate. Cross validation is the measure of 'risk' or classification error rate completed by leaving out a portion of the training set. The excluded portion is then used as a test set to quantify the error rate. However, the cross-validation error remained inconsequentially changed between a CP of 0.01 and 0.005. This finding indicates that the higher CP value over-prunes the

resulting decision trees. In this case, we are looking for exhaustive decision trees and set the CP at the lowest possible value without incurring a substantial change in classification error rate.

The trees developed in this research are not sufficiently accurate at classifying regardless of changes to the parameters of *CART*. This is due to the low purity of the terminal nodes, and the *CART* algorithm can no longer meaningfully split the data without overfitting. Fortunately, low classification error is not essential for the methodology presented. What is essential are the splits that do occur in the tree, as these splits improve the purity of the next set of nodes. In the context of this work, improving the purity of the data creates a feasible functional design space by reducing the number of infeasible functions presented to the designer.

Classification based on Association Rules (CBA)

Classification based on Association Rules was introduced by Lui et al. 1998 [31]. Association rules identify the frequency of data features and classifications occurring together as measured by support, confidence, and lift [35,36]. The *CBA* algorithm works by finding all the association rules in a given dataset and identifies the most reliable rules based on confidence. Then the algorithm tests the set of rules against the training dataset. If the classification is wrong, the algorithm will look at other, similar strong rules that will result in higher accuracy when classifying for that category. Once improvements in classification cannot be made by replacing rules, the algorithm reports the best set of rules.

The *CBA* algorithm is implemented with the *R* package *arulesCBA* [37]. We use this algorithm to identify association rules that are the most relevant to the reduction of the feasible function space. We measure importance by the lift parameter reported by the *CBA* algorithm. In both case studies, the confidence threshold is set to 30%, and the support threshold set to 0.05%. In the context of this work, confidence is the frequency that a function appears given a specific set of components. Support is the measure of the frequency of which the function/familial combination occurs in the entire data set. By setting the support low, the algorithm is encouraged to conclude from low-occurrence data-feature combinations. These parameters ensure that the resulting rules are representative of the most likely functions given a selection of parent, grandparent, and great grandparent components.

The *CBA* algorithm will not provide a designer with an exhaustive list of rules to follow. However, it provides additional information to a designer in addition to the information provided by a decision tree. Using the rules list, a designer can view what set(s) of components lead to the classification of a function. The rule list provides more information that the decisions tree as there is support, confidence, and lift reported for each rule. Using these metrics—especially lift—a designer can determine the trust to put in following a rule for identifying a function.

Case Studies

The two case studies we explored are representative of what a designer may face during the classification of a component function. Case Study 1 simulates a scenario in which a designer analyzing a product or product concept discovers an unfamiliar component and, as such, cannot define the purpose or function of the part. Case Study 2 demonstrates a scenario where a designer can identify the component but cannot distinguish function, likely due to the component's ability to perform multiple, unique functions. An example of such a component is a "bracket" that is capable of multiple functions, such as couple, position, secure, or guide. However, it is often challenging to define the specific function performed by the component.

Case Study 1 – Unknown Component

In Case Study 1, we examine the functional space of unknown components sourced from *unclassified* components in the design repository. We selected this data subset as a way to demonstrate the full breadth of the methodology as represented by a multitude of functions and components, rather than a single component with few unique functions. We theorize that the methods presented in this paper reach maximum utility by assuming the designer is entirely unaware of how to define a selected component and any possible functions the component can perform. The goal of this case study is to demonstrate the ability of these methods to reduce the number of tractable functions contributing to function-based design approaches, while still requiring the creative work of the designer to complete the functional analysis of the *unknown* component.

The *unknown* component data subset features 279 unclassified components that perform one of 25 unique functions. In this dataset, there are 12 possible parents, five grandparents, and four great-grandparents. Table 3 shows the 25 unique functions performed by the *unknown* components. Table 4 lists the component basis terms and familial locations of the components that are attached to an *unknown* component in at least one occurrence.

TABLE 3. CASE STUDY 1 – UNKNOWN COMPONENT OBSERVED FUNCTIONS

#	Observed Functions	#	Observed Functions	#	Observed Functions
1	Actuate	10	Import	18	Sense
2	Change	11	Inhibit	19	Separate
3	Condition	12	Position	20	Stabilize
4	Convert	13	Process	21	Stop
5	Couple	14	Regulate	22	Store
6	Detect	15	Remove	23	Support
7	Distribute	16	Rotate	24	Transfer
8	Export	17	Secure	25	Transmit
9	Guide	-	-	-	-

TABLE 4. CASE STUDY 1 – UNKNOWN COMPONENT ASSEMBLY DATA

#	Component	Parent	G_Parent	GG_Parent
1	Actuator	✓	×	×
2	Assembly	✓	✓	✓
3	Biological	×	×	✓
4	Blade	✓	×	×
5	Circuit Board	✓	×	×
6	Converter	✓	×	×
7	Electric Motor	✓	×	×
8	Fan	✓	×	×
9	Housing	✓	✓	×
10	Insert	✓	×	×
11	Sensor		✓	×
12	Support	✓	×	×
13	System	✓	✓	✓
14	Unclassified	✓	✓	✓

× signifies a component is not present in that position
✓ signifies a component is present in that position

Case Study 2 – Known Component

In Case Study Two, we explore how the methodology could be applied to accurately identify the function of a component that a designer can only define by name. This study looks into applying the methodology on a narrower, better-defined set of potential functions. It is unclear whether the current functional basis definitions are detailed enough to provide meaningful results when considering a known component. In this case study, we can start to conclude how we may further define/classify function, thus increasing the robustness of this methodology.

In design practice, we theorize that by providing defined feasible functions for a standard part that are limited by the familial relationship to other components, designers can use the provided decision tree and ruleset to more accurately define component functions. The results of Case Study Two can provide meaningful insight into inexperienced designers who may be unable to define the most likely function of a component accurately.

The known component data subset features 108 *bracket* components that perform one of 10 unique functions (Table 5), with 12 possible parents, four grandparents, and two great-grandparents. Table 6 lists the component basis terms and familial locations of the parental components that are attached to a *bracket* component on at least one occurrence.

TABLE 5. CASE STUDY 2 – KNOWN COMPONENT – BRACKET- OBSERVED FUNCTIONS

#	Observed Functions	#	Observed Functions
1	Couple	6	Position
2	Distribute	7	Regulate
3	Export	8	Secure
4	Guide	9	Support
5	Import	10	Transfer

TABLE 6. CASE STUDY 2 – KNOWN COMPONENT (BRACKET) ASSEMBLY DATA

#	Component	Parent	G_Parent	GG_Parent
1	Assembly	✓	✓	✓
2	Bearing	✓	×	×
3	Belt	✓	×	×
4	Circuit Board	✓	×	×
5	Cover	✓	×	×
6	Electric Switch	✓	×	×
	Handle		✓	×
7	Housing	✓	✓	×
8	Lever	✓	×	×
9	Reservoir	✓	×	×
10	System	✓	✓	✓
11	Unclassified	✓	×	×
12	Wheel	✓	×	×

× signifies a component is not present in that position
✓ signifies a component is present in that position

Assumptions and Limitations

The data selected from the OSDR are subject to the assumptions used during the creation of the repository. The case study data are defined by the component basis name rather than the common component name. Using component basis names limits the uniqueness and subjectivity of empirically named components. Furthermore, we found that common-named components that were named similarly to a basis term often are represented by a differing basis term (e.g., a brake bracket represented as the basis term *support*).

The familial (assembly) component data in the ODSR is limited to a depth of three. Many of the identified component assembly chains terminated on the *system* basis term by the great-grandparent component; *system* is synonymous with product. The fidelity of the component basis name also reduces further in the chain. It is observed that the grandparent and great-grandparent components are often denoted as *assembly* rather than a more descriptive basis term.

A low number of data features and low feature fidelity of the data limits the predictive capability of the algorithms used in this study. In the context of this research, our data has three features (parent, grandparent, great grandparent). Though we are not particularly looking to classify new data accurately, the effects of low data fidelity can limit the amount the presented methodology can reduce the feasible design space. In future implementations, higher data fidelity (such as expanding our data features to include children and great-great-grandparents) will lead to further defined solution space and reduce the number of presented feasible functions.

RESULTS AND DISCUSSION

In this section, we show the *CART* diagrams and *CBA* rules for each case study. The first figure shown in each section denotes the spread and frequency of functions observed in each case study. In addition, the order of the functions observed in Figures 1 and 3 are representative of the order of the functions in bar graphs located at the bottom of the *CART* decision trees shown (Figures 2, 4, and 5).

Case Study 1 – Unknown Component Results

Figure 1 shows that the unknown components found in the OSDR are completing a wide array of functions. The most popular functions—secure, transfer, position, guide, and couple—appear to be generally well-known functions. In addition, the less common functions - import, export, transfer, transmit, and convert – are also subject to being assigned to unknown components. We have observed that the trained design engineers responsible for the construction of the OSDR seemed to have challenges naming components during the reverse-engineering process, even if the components were completing well-known functions.

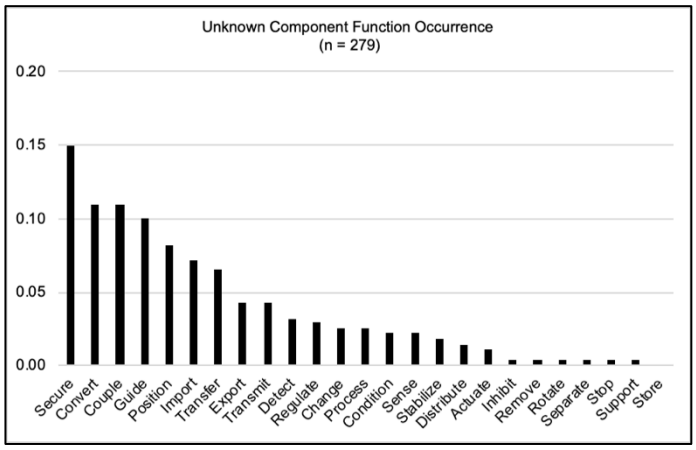

FIGURE 1: GRAPHICAL REPRESENTATION OF THE FUNCTIONAL SPREAD OF THE UNKNOWN COMPONENT CASE STUDY

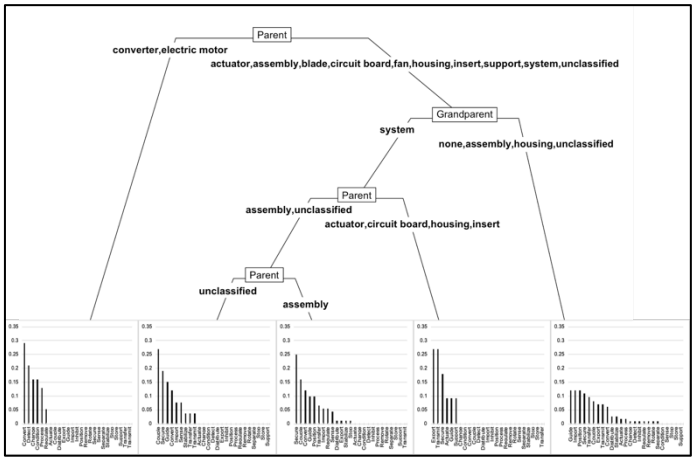

FIGURE 2: ABRIDGED CART DECISION TREE REPRESENTING THE DECISION PATHS TAKEN WHEN DECIDING THE FUNCTION OF AN UNKNOWN COMPONENT

The resulting decision tree from the *CART* algorithm for the *unknown* component case study is shown in Figure 2 (this diagram is an abridged version to improve clarity; the full diagram is shown in Figure 5 in the appendix) The full tree has several branches; a designer can reach a terminal node by identifying the neighboring components around the component whose function they wish to identify. The resultant bar plots represent the probability that an unknown component, with a given set of neighbors, will have a particular function.

As shown in Figure 2, if the parent of the unknown component is a converter or electric motor (branch 1), the most probable set of functions are *change*, *condition*, *covert*, *detect*, *process*, and *regulate*; with *convert* being the most likely function. Thus, it is concluded that the component connected to an electric motor or converter is probably performing a *performance*-based function commonly associated with electrical motors. This branch shows how the proposed methodology produces a set of probable function suggestions that limit the feasible design area of function definition. We suggest that the designer can more easily use design intuition to explore these six functions versus the original 25.

The left branch shown in Figures 2 and 5 denotes the first meaningful split where the parent component is a converter or electric motor. The *CART* algorithm will choose to split the data on the most purifying split. For the unknown component *CART* decision tree, we observe that splitting the 'parent - converter and electric motor' off creates a pure subset (left branch) that does not meet the splitting criteria to continue splitting. Concisely, the grandparent and great-grandparent features data in the left branch are not relevant enough to pose another split. Thus, the left branch is the purest split as only one feature is needed to define the functional space of the component that has a parent of converter or electric motor.

The right branch in Figures 2 and 5 includes all observed components other than a converter or electric motor. This data benefits from further splitting based on data features. As such, we observe multiple splits based on parent, grandparent, and great-grandparent components. When observing a well-defined path, such as the branch ending in 'parent – actuator, circuit board, housing, insert,' we see that the feasible function space is further reduced when compared to a path heavy in the components *assembly*, *system*, and *unclassified*. However, the results of these low-fidelity branches still indicate that the functional space can be reduced even when operating under higher uncertainty. These branches can also stimulate a reduction in the functional space when a designer using these trees can only poorly define the surrounding components, either due to a lack of knowledge or obscurity in design or component locations.

We have observed that well-defined paths result in a better reduction of the feasible functional space. However, when observing the 'parent – actuator, circuit board, housing, insert' terminal node, it can be seen that the feasible functions lack the cohesion found in the 'parent – converter, electric motor' node. The former node graph includes *actuate*, *export*, *guide*, *secure*, *support*, and *transmit*. A novice engineer may use this tree and assume a 'parent – actuator' would likely have a child

component function of *actuate*. However, it is unlikely that a child component is actuating onto an actuator. Thus, we include the *CBA* rule set to improve confidence in identifying function from a low-cohesion terminal node, as shown in Table 7.

TABLE 7. CASE STUDY 1 – UNKNOWN COMPONENT FUNCTION DEFINING ASSOCIATION RULES

#	Observed Component	Function	Lift
1	Grandparent = Assembly G_Grandparent = Assembly	Couple	9.00
2	Parent = Actuator	Export	23.50
3	Parent = Insert	Export	23.35
4	Parent = Electric Motor	Convert	8.71
5	Parent = Circuit Board Grandparent = System	Transmit	23.25
6	Parent = Circuit Board Grandparent = Assembly	Import	13.95
7	Parent = Unclassified Grandparent = Unclassified G_Grandparent = System	Stabilize	27.90
8	Parent = Support	Remove	139.50
9	Parent = Blade	Position	6.07
10	G_Grandparent = Assembly	Couple	3.60
11	Grandparent = Assembly G_Grandparent = Biological	Import	5.58
12	G_Grandparent = Unclassified	Position	4.04
13	Parent = Assembly G_Grandparent = Biological	Transmit	7.75
14	Parent = Fan	Guide	3.32
15	Parent = Unclassified G_Grandparent = System	Stabilize	16.74
16	Parent = Assembly Grandparent = Unclassified G_Grandparent = System	Transfer	4.65

The *CBA* ruleset shows promise in further defining the functional space of a terminal node. In the actuator example, we observe that the *CBA* ruleset is suggesting that the designer consider the function *export* for an actuator parent. The lift value denotes how reliable the rule is. Higher lift values should promote a designer to strongly consider subsequent rules, and lower lifts should at least be considered. However, we notice that there is sometimes a high lift value assigned to illogical combinations such as 'support – remove'. Figure 1 shows that *remove* makes up a small portion of the functions in the *unknown* dataset. The *CBA* results suggesting 'parent – support' is related to *removing* are likely due to a few data points classified as '*remove*,' but all share the same parent. This result suggests a fringe case that should not be discounted but solidifies the point that the *CBA* rules should be explored if the decision tree is unable to reduce the functional feasibility enough to inspire confidence by the designer.

Case Study 2 – Known Component Results

In Figure 3, we observe that the bracket component's most popular functions (*secure*, *guide*, *position*, and *couple*) follow a logical functional assignment. In contrast, the functions *transfer*, *export*, *import*, *regulate*, and *distribute* are unlikely functions for a bracket. The unlikely functions may be hard for a novice designer to consider given they may be unaware that a bracket can perform such functions. The function frequency results suggest that the drive for this research is meaningful in that there are is a significant frequency of non-classical functions assigned

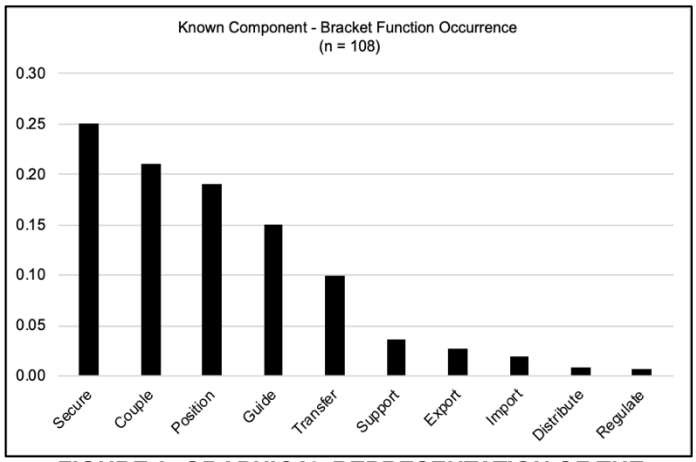

FIGURE 3: GRAPHICAL REPRESENTATION OF THE FUNCTIONAL SPREAD OF THE KNOWN (BRACKET) COMPONENT CASE STUDY

to the known component that likely would remain unexplored by novice engineers.

In Figure 4, we observe a reduction of the feasible design space in each terminal. In each terminal node bar graph, at least two functions are removed from consideration. Accordingly, the untraditional functions are overshadowed by the popular functions found for brackets. The overshadowing observed could lead to limited consideration of less common functions by designers using this decision tree. The issues raised from this result are only reinforced by the results shown in Table 8—the

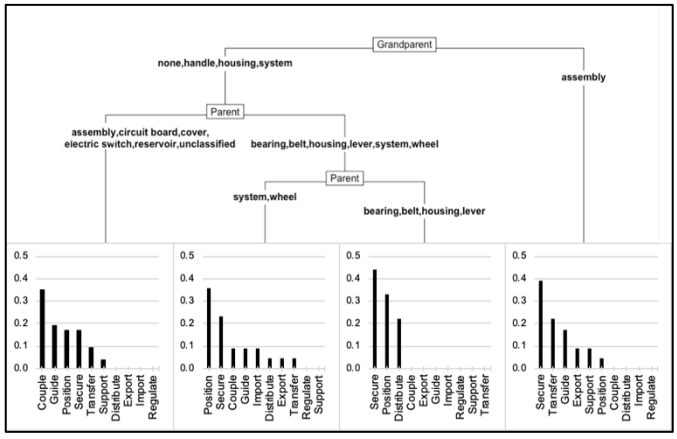

FIGURE 4: CART DECISION TREE REPRESENTING THE DECISION PATHS TAKEN WHEN DECIDING THE FUNCTION OF AN KNOWN COMPONENT

CBA rules developed for identifying the functions of a known (bracket) component. The rules do not suggest that a designer considers any of the untraditional functions. These results identify areas of improvement, such as defining a function-type schema for design repositories.

TABLE 8. CASE STUDY 2 – KNOWN- BRACKET - COMPONENT FUNCTION DEFINING ASSOCIATION RULES

#	Observed Component	Function	Lift
1	G_Grandparent = Assembly	Secure	4.00
2	Parent = Belt	Secure	4.00
3	Parent = Cover	Couple	4.69
4	Parent = Unclassified	Couple	2.82
5	Parent = Lever	Position	2.57
6	Parent = Reservoir	Guide	3.37
7	Grandparent = Assembly	Secure	1.57
8	Grandparent = None	Position	1.89
9	G_Grandparent = System	Secure	1.43
10	Parent = Circuit Board	Guide	2.25
11	Parent = Housing	Position	1.71
12	Parent = Wheel	Guide	2.25

CONCLUSION

In this paper, we introduce a novel approach aimed at increasing function-based design knowledge for novice designers by using data mining techniques. The methodology presented employs *Classification and Regression Trees (CART)* and *Classification based on Association Rules (CBA)* algorithms to create rules and visual guides that help novice designers identify the function of components. The decision trees and rules generated are based on the identification of neighboring components about the component whose function needs to be identified. The goal of this work is to provide an objectively defined functional design space. The results of our work help reduce the feasible function space for function-based design, which will enable novice designers to more accurately undertake functional assignment. To do this, we created visual decision trees and provided hard-and-fast function defining rules. The decision trees and rules aid untrained designers to determine component function by identifying the components around the defined component. The methodology is tested in two case studies.

In case study one, we explored defining the function of a component that is unfamiliar to the designer. In case study two, we tested the methodology with a component a designer can likely identify—in this case, a bracket—but whose functions may be nebulous. The results suggest that our method is viable in limiting the number of feasible functions given the identification of components. However, we discovered that the rules provided by the *CBA* algorithm could be misleading if used alone to identify the component function. In this regard, we maintain that a designer should use the decision tree first, then confer with the set of *CBA* rules if necessary. We also found that an overabundance of popular functions can obscure the ability to consider uncommon-but-important functions. This sentiment is apparent in case study two, where the generated *CBA* rules only implicate popular functions. It does not indicate a scenario where a designer may need to consider a specialized or less traditional function.

In terms of designer utility, our proposed methodology is still reliant on (and thus limited by) designer cognition, specifically in subsequent product design and functional analysis processes. However, we assert that efforts explored here allow for designers with incomplete domain knowledge to define function more successfully. The limited design space increases focus on a likely set of functions for a component. We suggest that even with weak domain knowledge, this methodology can provide a less daunting feasible function space that the designer can succinctly explore.

We suggest future work explore increasing the utility of the presented methodology and repository data by creating a data schema that categorizes functions on a high level. This new schema can promote designers to consider weakly represented but essential functions. Using high-level function classification can be used as a new data feature, or simply as a grouping method to signal a difference of function-type in the resulting decisions trees and rules.

We find that over-simplified repository data limits the ability to converge on a reasonably reduced feasible design space. In this regard, the repository data used in this research can benefit from increasing the fidelity of naming neighboring components. Finally, in the future, we aim to test tools developed from this methodology by observing novice designers using the decision trees and rule sets in human studies research.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. CMMI-1826469. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Ford, P., Meadwell, J., and Terris, D., 2016, "The Need for a Holistic Approach to Sustainability in New Product Development from the Designers Perspective," *Proceedings of the 18th International Conference on Engineering and Product Design Education: Design Education: Collaboration and Cross-Disciplinarity, E and PDE 2016*, pp. 71–76.

[2] Szykman, S., Sriram, R. D., Bochenek, C., and Racz, J. W., 1999, "The NIST Design Repository Project," Adv. Soft Comput. - Eng. Des. Manuf., pp. 5–19.

[3] Johnson, A., and Gibson, A., 2014, "Chapter 1 - Engineering Design," *Sustainability in Engineering Design*, Elsevier, pp. 1–19.

[4] Alves, J., Marques, M. J., Saur, I., and Marques, P., 2007, "Creativity and Innovation through Multi-disciplinary and Multisectoral Cooperation," Creat. Innov. Manag., **16**(1), pp. 27–34.

[5] Rosenman, M. A., and Gero, J. S., 1998, "Purpose and Function in Design: From the Socio-Cultural to the Technophysical," Des. Stud., **19**(2), pp. 161–186.

[6] Pahl, G., Beitz, W., Feldhusen, J., and Grote, K.-H., 2007, *Engineering Design*, Springer London, London.
[7] Ullman, D., 2009, *The Mechanical Design Process*, McGraw-Hill Education.
[8] Stone, R. B., and Wood, K. L., 2000, "Development of a Functional Basis for Design," Asme, **122**(December 2000).
[9] Hirtz, J., Stone, R. B., McAdams, D. A., Szykman, S., and Wood, K. L., 2002, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," Res. Eng. Des. - Theory, Appl. Concurr. Eng., **13**(2), pp. 65–82.
[10] 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*.
[11] Cheong, H., Chiu, I., Shu, L. H., Stone, R. B., and McAdams, D. A., 2011, "Biologically Meaningful Keywords for Functional Terms of the Functional Basis," J. Mech. Des. Trans. ASME, **133**(2), pp. 1–11.
[12] Bohm, M. R., and Stone, R. B., 2004, "Product Design Support: Exploring a Design Repository System," Am. Soc. Mech. Eng. Comput. Inf. Eng. Div. CED, pp. 55–65.
[13] Szykman, S., Sriram, R. D., Bochenek, C., Racz, J. W., and Senfaute, J., 2000, "Design Repositories: Engineering Design's New Knowledge Base," IEEE Intell. Syst. Their Appl., **15**(3), pp. 48–55.
[14] Bohm, M. R., Stone, R. B., Simpson, T. W., and Steva, E. D., 2008, "Introduction of a Data Schema to Support a Design Repository," CAD Comput. Aided Des., **40**(7), pp. 801–811.
[15] Phelan, K., Wilson, C., and Summers, J. D., 2014, "DEVELOPMENT OF A DESIGN FOR MANUFACTURING RULES DATABASE FOR USE IN INSTRUCTION OF DFM PRACTICES," *ASME 2014 International Design Engineering Technical Conferences*, pp. 1–7.
[16] Ferrero, V., Wisthoff, A., Huynh, T., Ross, D., and DuPont, B., 2018, "A Sustainable Design Repository for Influencing the Eco-Design of New Consumer Products," EngrXIV, p. Under Review.
[17] Oman, S., Gilchrist, B., Tumer, I. Y., and Stone, R., 2014, "The Development of a Repository of Innovative Products (RIP) for Inspiration in Engineering Design," Int. J. Des. Creat. Innov., **2**(4), pp. 186–202.
[18] Rajagopalan, V., Bryant, C. R., Johnson, J., McAdams, D. A., Stone, R. B., Kurtoglu, T., and Campbell, M. I., 2005, "Creation of Assembly Models to Support Automated Concept Generation," *Volume 5a: 17th International Conference on Design Theory and Methodology*, ASME, pp. 259–266.
[19] Bryant, C. R., McAdams, D. A., Stone, R. B., Kurtoglu, T., and Campbell, M. I., 2005, "A Computational Technique for Concept Generation," Proc. ASME Int. Des. Eng. Tech. Conf. Comput. Inf. Eng. Conf. - DETC2005, **5**, pp. 267–276.
[20] Bryant, C. R., McAdams, D. A., Stone, R. B., Kurtoglu, T., and Campbell, M. I., 2006, "A Validation Study of an Automated Concept Generator Design Tool," Proc. ASME Des. Eng. Tech. Conf., **2006**, pp. 1–12.
[21] Gilchrist, B. P., Tumer, I. Y., Stone, R. B., Gao, Q., and Haapala, K. R., 2012, "COMPARISON OF ENVIRONMENTAL IMPACTS OF INNOVATIVE AND COMMON PRODUCTS," *2012 International Design Engineering Technical Conferences &Computers and Information in Engineering Conference*, ASME, Chicago, pp. 1–10.
[22] Ferrero, V., Raman, A. S., Haapala, K. R., and DuPont, B., 2019, "Validating the Sustainability of Eco-Labeled Products Using a Triple-Bottom-Line Analysis," Smart Sustain. Manuf. Syst., **3**(1), p. 20190022.
[23] Soria Zurita, N. F., Stone, R. B., Demirel, O., and Tumer, I. Y., 2018, "The Function-Human Error Design Method (FHEDM)," *Volume 7: 30th International Conference on Design Theory and Methodology*, American Society of Mechanical Engineers.
[24] Soria Zurita, N. F., Stone, R. B., Onan Demirel, H., and Tumer, I. Y., 2020, "Identification of Human–System Interaction Errors During Early Design Stages Using a Functional Basis Framework," ASCE-ASME J Risk Uncert Engrg Sys Part B Mech Engrg, **6**(1).
[25] Tucker, C. S., and Kim, H. M., 2009, "Data-Driven Decision Tree Classification for Product Portfolio Design Optimization," J. Comput. Inf. Sci. Eng., **9**(4), pp. 1–14.
[26] Bae, J. K., and Kim, J., 2011, "Product Development with Data Mining Techniques: A Case on Design of Digital Camera," Expert Syst. Appl., **38**(8), pp. 9274–9280.
[27] Zhan, Y., Tan, K. H., and Huo, B., 2019, "Bridging Customer Knowledge to Innovative Product Development: A Data Mining Approach," Int. J. Prod. Res., **0**(0), pp. 1–16.
[28] Wisthoff, A., Ferrero, V., Huynh, T., and DuPont, B. L., 2016, "Quantifying the Impact of Sustainable Product Design Decisions in the Early Design Phase through Machine Learning," *ASME 2016 International Design Engineering Technical Conference & Computers and Information in Engineering Conference*, pp. 1–10.
[29] Zurita, N. F. S., Tensa, M. A., Ferrero, V., Stone, R. B., DuPont, B., Demirel, O. H., and Tumer, I. Y., 2019, "An Association Rule Approach for Identifying Physical System-User Interactions and Potential Human Error Using a Design Repository," *ASME 2019 International Design Engineering Technical Conferences &Computers and Information in Engineering Conference*, pp. 1–10.
[30] 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," Proc. Des. Soc. Int. Conf. Eng. Des., **1**(1), pp. 1713–1722.
[31] Liu, B., Hsu, W., and Ma, Y., 1998, "Integrating Classification and Association Rule Mining," Kdd, **98**, pp. 80–86.
[32] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. ., 1984, *Classification And Regression Trees*, Chapman and Hall/CRC, Belmont, CA.
[33] The R Foundation, "R Programming Language."
[34] Therneau, T., Atkinson, B., and Ripley, B., 2019, "Rpart: Recursive Partitioning for Classification, Regression and Survival Trees. R Package Version 4.1-15."
[35] Agrawal, R., Imielinski, T., and Swami, A., 1993, "Mining Association in Large Databases," Proc. 1993 ACM SIGMOD Int. Conf. Manag. data - SIGMOD '93, pp. 207–216.
[36] Agrawal, R., 1994, "Fast Algorithms For Mining Association Rules In Datamining," Int. J. Sci. Technol. Res., **2**(12), pp. 13–24.
[37] Johnson, I., Hahsler, M., and Giallanza, T., 2020, "Package 'ArulesCBA .'"

APPENDIX

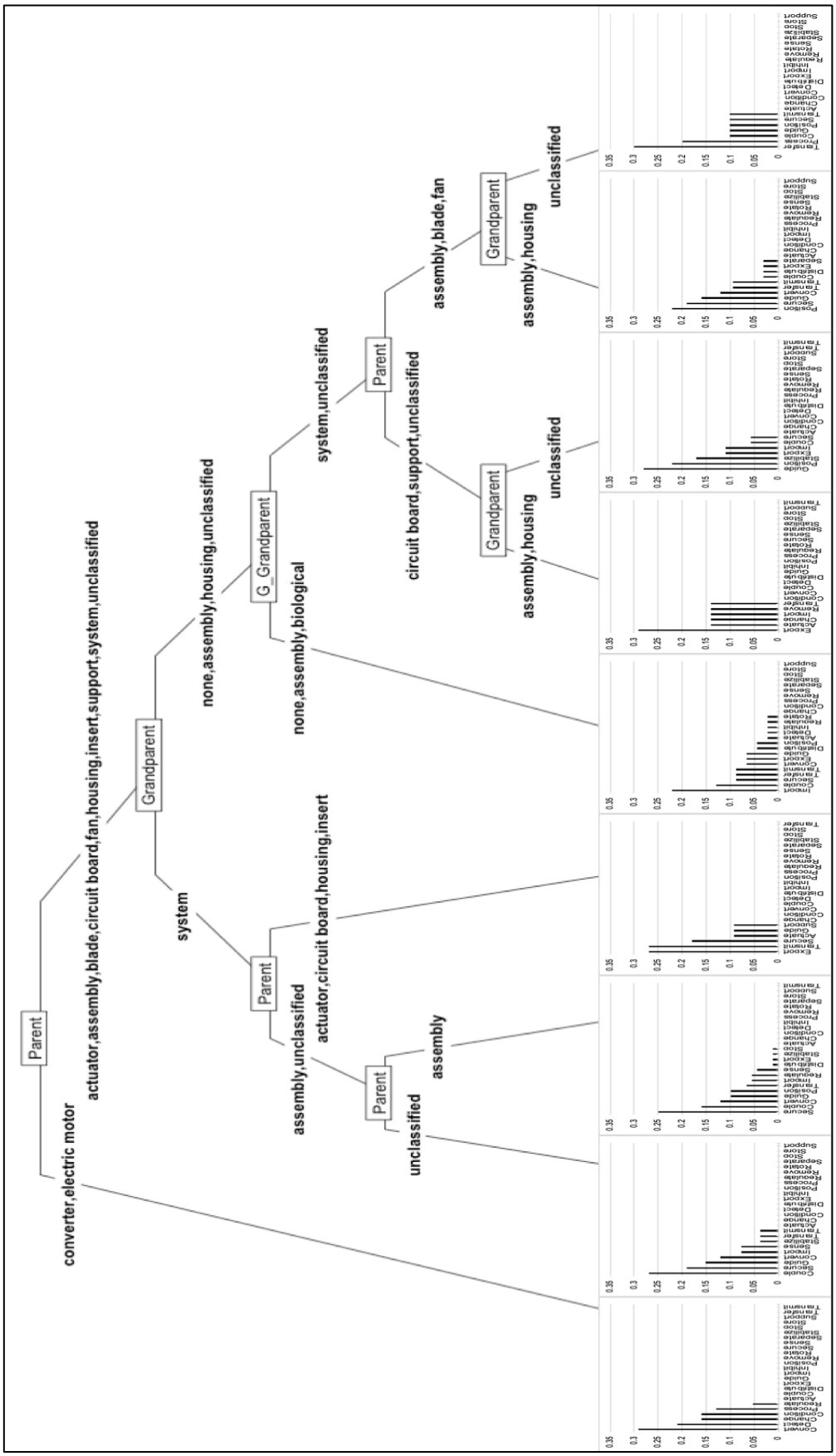

FIGURE 5: FULL *CART* DECISION TREE REPRESENTING THE DECISION PATHS TAKEN WHEN DETERMINING THE FUNCTION OF AN UNKNOWN COMPONENT