

The Part Count Tool (PaCT) for Design Concept Selection

Tarang Parashar¹, Kerry Poppa², Katie Grantham Lough³, Robert B. Stone²

¹*Department of Mechanical and Aerospace Engineering, Missouri University of Science and Technology, Rolla, USA*

²*School of Mechanical, Industrial and Manufacturing Engineering, Oregon State University, Corvallis, USA*

³*Department of Engineering Management and Systems Engineering,
Missouri University of Science and Technology, Rolla, USA*

E-mail: kag@mst.edu

Received July 28, 2011; revised November 1, 2011; accepted November 11, 2011

Abstract

This paper presents a part count tool that predicts the part count for a particular product concept during the conceptual design phase. The part count tool will also aid in ranking the design concepts by the criterion of number of components for a product. This tool utilizes existing automated concept generation algorithms to generate the design concepts. It extracts the available data from the Design Engineering Lab Design Repository to determine an average number of parts per component type in the repository and then calculates an average part count for new concepts. The part count tool also uses an algorithm to determine how to connect two non-compatible components through the addition of mutually compatible components. While emphasis is placed on the average parts per product in evaluating designs, the overall functional requirement of the product is also considered.

Keywords: Mechanical Design, Conceptual Design, Design for Manufacturing and Assembly

1. Introduction

Intense competition in the consumer market pushes designers to consider manufacturing costs more thoroughly and completely early in the design process. Typical cost reduction is achieved either by reducing the number of the components or by reducing the time required for manufacture or assembly or some combination of the two. In order to minimize the cost associated with production, time, and labor, Design for Manufacture (DFM) and Design for Assembly (DFA) techniques have been adopted. DFM techniques focus on the manufacturing aspects of a potential product during the design of the product whereas DFA focuses on the assembly operations during design. A major focus of both techniques is reducing the number of required parts which in turn reduces cost associated with inventory, material and overhead [1]. Approximately 70% of the final cost of a product is determined during the design process [1]. Design for Manufacture and Assembly (DFMA) combines DFM and DFA techniques to systematically reduce final product cost by guiding conceptual design decisions [1].

The Design Repository [2] is a heterogeneous collection of design knowledge about products, which supplies data for a variety of concept generation tools such as the matrix

based Morphological Evaluation Machine and Interactive Conceptualizer (MEMIC) [3], the grammar based Component Flow Graph (CFG) [4], and Morphological Chart Search [2]. Research is underway to develop and improve concept generation techniques to provide designers with a variety of tools for concept selection. The tool discussed in this paper, the Part Count Tool (PaCT), is designed to assist designers with concept selection based on the number of parts per product, usually referred to as part count in DFMA. The PaCT uses available Design Repository data to dynamically calculate the average number of parts for a given component type, predict the final part count and rank concepts accordingly. Given an estimate of part count, designers can further evaluate concepts based on predicted assembly and manufacturing cost of the final product.

2. Background

The PaCT algorithm builds upon several concept generation technologies that generally follow from the systematic approaches of Pahl and Beitz [5] and Otto and Wood [6]. The specific design approaches include functional modeling, design repositories and automated concept generation algorithms and are reviewed next.

2.1. Functional Modeling and Functional Basis

Functional modeling clarifies the overall function (*i.e.*, “what” the product must do to satisfy user requirements) of a particular product [5], and a functional model diagram is as a graphical representation of that model. Functional modeling is an important tool in the conceptual design phase since it places the emphasis on what the product must accomplish versus how (or the physical solutions) it is accomplished. Functional descriptions for a product describe the relationship between the input and the required output and take the form of a verb-object pair [6] where the function is the verb and the flow is the object. A black box model illustrates the overall function of a system based on the customer needs. More detailed chains of functions describe the changes undergone by different flows as they pass through a system [6]. These chains are then grouped together to form a functional model diagram for the whole system. An example functional model diagram is shown in **Figures 1 and 2**.

When first developed, functional modeling described only basic functions, and a common language did not exist to describe the flows [5,7-9]. Progress was made when the Functional Basis was introduced to provide a more systematic and useful design vocabulary for the designers and enable computing based on functionality [5,7-9]. The Functional Basis employs a set of definitions for the

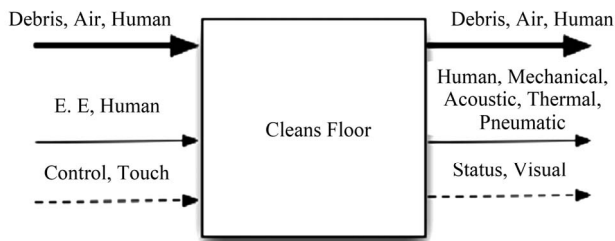


Figure 1. Black box for iRobot Roomba.

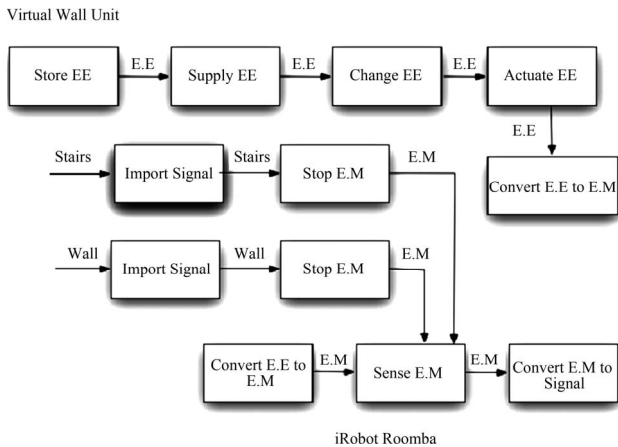


Figure 2. Snippet of a functional model for iRobot Roomba.

function and flow terms. Terms like transfer, actuate, import, etc. are examples of functions and terms like solid, liquid, status signal, etc. are examples of flows [7]. **Figure 1** is an example of a black box model diagram for the iRobot Roomba, a robotic floor vacuum. **Figure 2** illustrates a snippet of a functional model diagram of an iRobot Roomba representing one of its virtual walls, a device that creates a electromagnetic [EM] barrier to contain the robot in a desired area.

2.2. Design Repositories

Over time product design and manufacture activities have become more complex [7]. A wide variety of information pertaining to various aspects of design is required by designers to design or redesign a product so that it meets the needs of the customers [10]. Design repositories have been proposed as a hub for storing design information that can be searched and reused. The National Institute of Standards and Technology (NIST) Design Repository project [10] and the Design Engineering Lab Design Repository [11] represent two efforts to create a design database to aid designers in various aspects of design. The Design Engineering Lab Design Repository (hereafter referred to more succinctly as the Design Repository) hosts data for approximately 5630 artifacts extracted from 129 consumer products representing the electromechanical, hydromechanical and electronic domains. There are multiple ways of searching for artifacts in the database. For example, one may search for an artifact by its functionality or attributes. Textual description, physical parameters, manufacturing process, failure information, part number, and supporting functions are recorded as attributes. The product data in the Design Repository can be extracted in a variety of forms geared to support concept generation. Example formats include matrices such as the function component matrix (FCM) [2] and the design structure matrix (DSM) [2]. The FCM describes function-component interaction, and the DSM describes the component-component interaction.

The Design Repository schema is built on a PostgreSQL database [11]. The schema is divided into sections representing seven types of information: artifact-, function-, failure-, physical-, performance-, sensory- and media-related information. Artifact hierarchy is developed by using a parent-child relationship in the artifact table [11]. These tables provides information regarding the artifacts in terms of their names, type of assembly, quantity, corresponding component basis system name, input flow, and output flow.

2.3. Automated Concept Generation

In the past few decades, much research has been devoted

to automating the conceptual design process, reflecting a more systematic and refined approach to conceptual design. Mature design methods are often the target of research in the field as models to follow for automated concept generation algorithms [12]. Various methodologies have been developed to automate the design phase. These methodologies primarily focus on assigning appropriate solutions to the sub-functions of a functional model of a product and then assembling the individual solutions together to represent the final product [12].

Pahl and Beitz [5] and Hubka [13] have been pioneers in formulating systematic approaches for the conceptual design phase. However, few computational tools exist to assist the designers during the conceptual design phase. The tools in existence are typically limited to a particular application domain. For example, graph grammars [4,14] and catalog design [15,16] tackle only specific components on the basis of a required behavior and performance. This limited applicability led other researchers to expand the capabilities of concept generation algorithms to produce a broader area of possible solutions. Bryant *et al.* [3] developed a means of automating the conceptual design phase by introducing a concept generator for designers in the early stages of product design. The concept generator takes a functional description of the desired product as input and uses the Design Repository's database (in the form of the DSM and FCM) to generate concept variants. The concept generation process filters components that solve the input functionality based on component-component compatibility. The current concept generator approach can be summarized as a five-step process:

1) *Step 1. Creation of Functional Model*

On the basis of customer needs, a functional model for a product is created by identifying the flows and the functions acting on the flows while ensuring that all customer needs are met. The functional model is then converted to an adjacency matrix identifying the relationships among all of the sub-functions. **Figure 3** illustrates

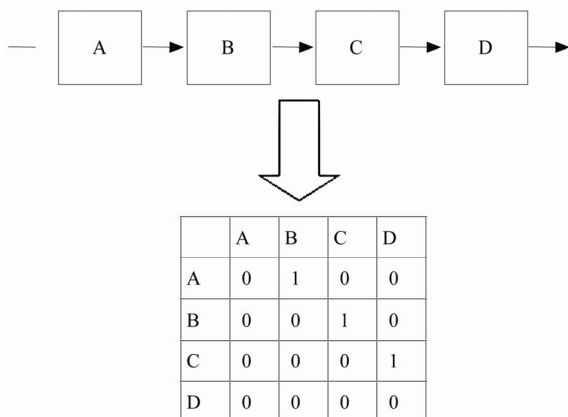


Figure 3. Example of functional model & its adjacency matrix.

a functional model and its associated adjacency matrix that captures the functional topology.

2) *Step 2. Generation of Concept Variants*

Next, concept variants are produced by supplying the functional model generated in Step 1 to an automated concept generator. These algorithms use matrix multiplication or graph grammar rules to find sets of components that form solutions for the specified functionality. The concept variant supplied to PaCT include the components need to produce the concept and their interconnections in a component adjacency matrix roughly analogous to a design structure matrix.

3) *Step 3. Determination of Component Compatibility*

At this point, component-component interaction data is used to determine compatibility among all the components in the chain. Two components are assumed compatible if the design repository has a record of them being connected in a previous product.

4) *Step 4. Identification of Feasible Concept Variants*

Finally, the process is concluded with the identification of a set of concept variants comprised of compatible components that solve the desired functionality. Each concept variant is computed by eliminating all the components that do not solve the function and are not compatible.

3. Research Approach: The Part Count Tool Algorithm

With the availability of automated concept generation tools, a large number of concept variants may be produced—typically too many for humans to process. Many methods have been proposed to evaluate and filter the concept variants such as customer satisfaction, risk [15] and performance, and research continues in this area. Thus far, researchers in the area have paid little attention to concept filtering based on Design for Manufacturing and Assembly (DFMA) concerns. Therefore, this work focuses on the part count for each concept variant as one criterion for filtering the concepts. A part count estimate not only helps to filter out the concept variants, but also provides needed information to evaluate the manufacturing and assembling time—factors that largely determine the final cost of the product. PaCT also addresses the case of incomplete concept variants, *i.e.*, those where a complete set of compatible solutions does not exist for an input functional model. Assumptions about function sharing and supporting components are made to determine the total number of parts.

In this section, PaCT is first placed in context of the typical workflow for a concept generation activity. Following that, the specific data required from a design knowledge base (such as the Design Repository in this work) is described along with a typical pseudo code

query to retrieve the data. Finally, the PaCT algorithm is presented along with an explanation of the assumptions embedded within it.

3.1. General Approach

The general approach for the PaCT is to apply the algorithm to a set of candidate solutions as an aid to concept selection. As implemented in this work, the PaCT workflow consists of inputting a functional model of the desired product to an automated concept generator to create a set of concept variants, and importing the generated concept variants into the PaCT application to compute the part count estimate for each concept variant and to display the results. All concept variants can be compared in PaCT's Bill of Materials (BOM) view. The user of PaCT can select different options to account for needed fasteners and to refine the part count estimates. The part count estimate can then be used when developing concept variant rankings to guide the selection of a concept for further development. The general steps involved in generating the concepts and executing the PaCT algorithm to compute the total number of parts are outlined in **Figure 4**.

3.2. Data Collection

In order to compute the part count for a given concept, two intermediate calculations are needed for each component that composes a concept variant. The first is the average number of parts in the general component. The second is a table of possible connections between the general component and all other potential components, whether they are included in the current concept variant or not. Both of these calculations rely on observations from previous products and utilize data in the Design Repository database. For each calculation, the logic used to gather the data is presented as well as an example of the collected data.

3.2.1. Average Number of Parts Per Component

This value represents the average number of parts that make up a component's occurrence in a product. For example, the component term "housing" typically occurs as two or more parts in a product, e.g., left and right housing. This value is needed in cases where the returned concept variant may suggest the same component to solve two or more successive functions. In order to accommodate this opportunity for function sharing, the string of connected components can be replaced with the average value. The following pseudo code queries were used to extract the data from the database:

1) Logic:

(a) COUNT all Artifacts of the Component Type

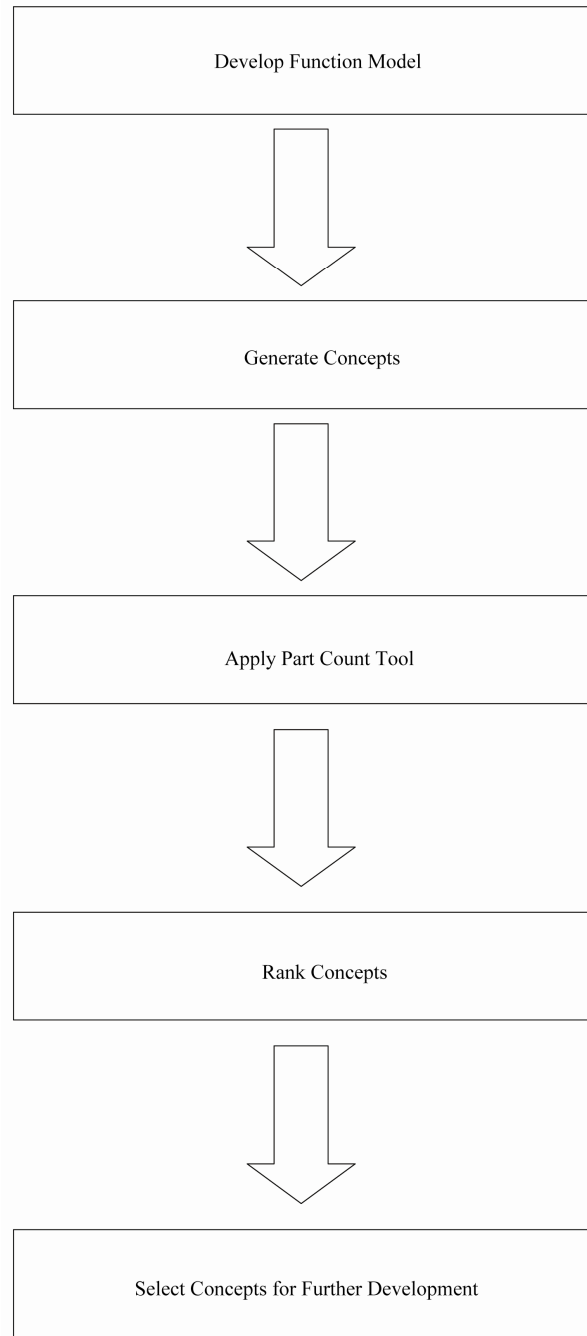


Figure 4. Steps involved in generating concepts for the part count tool.

(b) COUNT the number of Systems with Artifacts of the Component Type

2) Example: Tube

Average values for artifacts classified as a "tube" are calculated by dividing the value obtained from Step (a) by the value obtained from Step (b). A value of 2.7931 was assigned to the tube component, meaning on average when a tube appears in a product, it is composed of more

than two parts.

3.2.2. Component Interaction Table

This table defines how components interact with other components in the Design Repository. The following pseudo code queries are used to determine component interaction:

1) Logic:

a) SELECT the Input Artifact of every Artifact of the specified Component Type

b) SELECT the DISTINCT Component Types of the Input Artifacts

c) SELECT the Output Artifact of every Artifact of the specified Component Type

d) SELECT the DISTINCT Component Types of the Output Artifacts

2) Example:

Following results were generated for tube component.

Table 1 exhibits the result of the component interaction generated from the data base.

Table 1. Result of Component-Component Interaction for Tube

Figure 6 is used for predicting the number of parts for a product.

The concept variants are first checked for the component compatibility. If there is no interaction between the first two components the algorithm identifies the next component in the chain and checks for compatibility with

Table 1. Result of component-component interaction for tube.

Compatible Input Components	Compatible Output Components
guiders	screw
nozzle	distributor
housing	mixer
tube	material filter
cover	housing
friction enhancer	tube
ic engine	clamp
electric motor	cap
heating element	reservoir
container	container
bracket	
nut-bolt	

3.3. Part count Tool Algorithm

After generating the concept variants (a typical view is shown in **Figure 5**) the PaCT algorithm shown in the

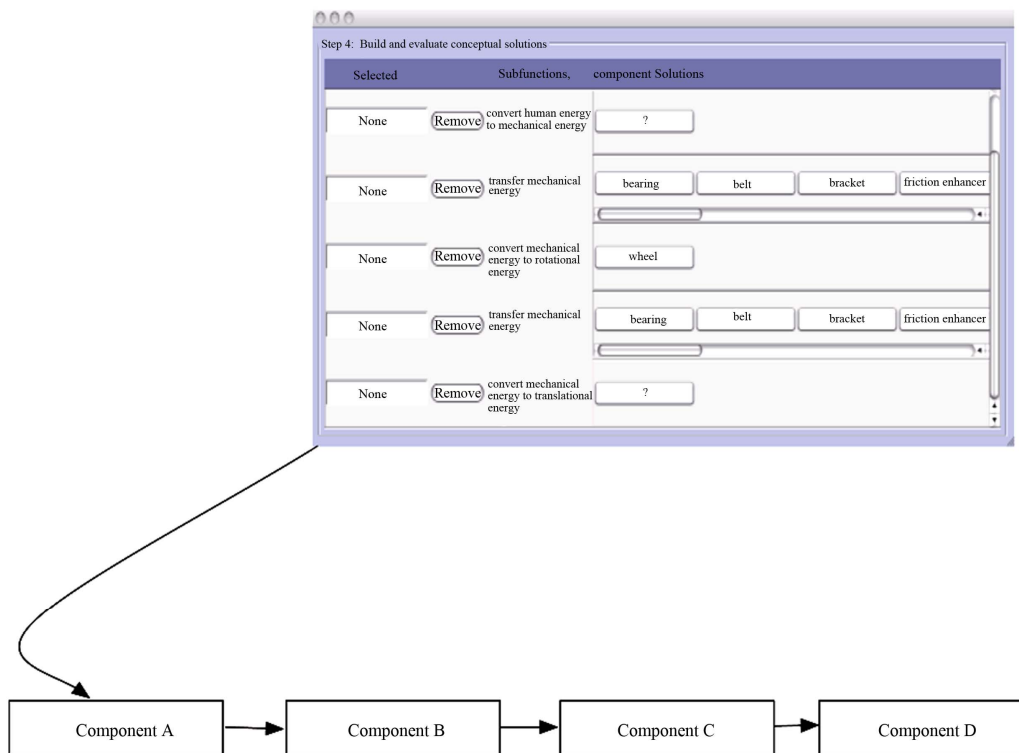


Figure 5. Concept variant shown in the concept generator interface.

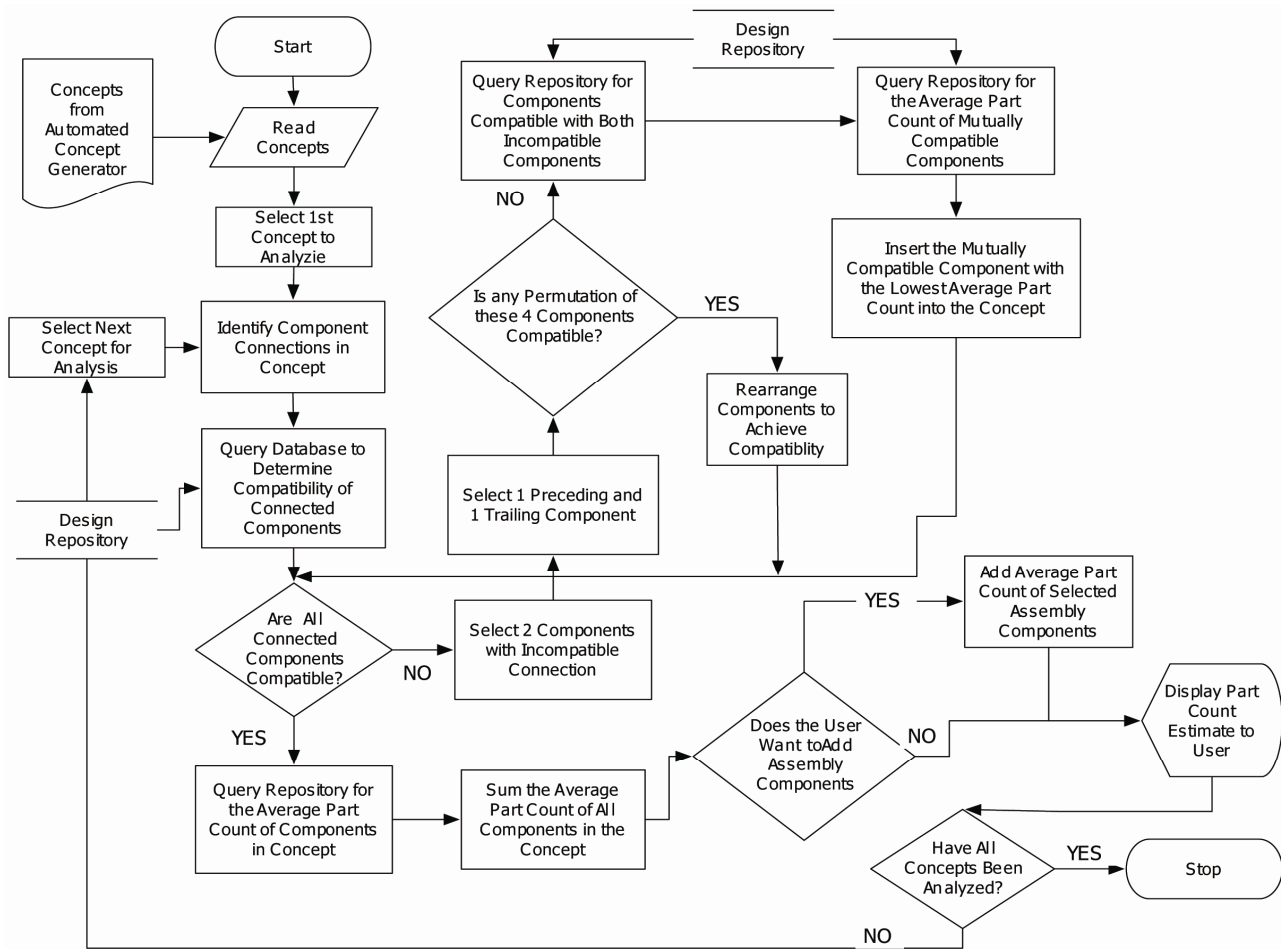


Figure 6. Algorithm for part count tool.

the initial component. If an interaction is discovered, the positions of the components are switched and the compatibility continues with the next component in the chain (Figure 7). If no interaction exists with any components, a linking component is selected that interacts with each of two components (Figure 8). Once all components are checked for interaction, the average value is assigned to each component and the sum of the values are calculated to determine the average number of parts required for the product.

3.3.1. Part Count Tool Assumptions:

Several assumptions are required to arrive at the total part count for a given concept variant. They are described next.

1) If one component immediately follows an identical component in sequence, only one component is selected. However, if an identical component is found at a different place, it is considered twice. See Figure 9 for illustration.

2) When checking interactions, only the next component complexity at this stage in design tool development. See

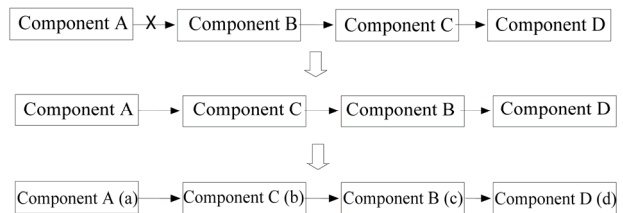


Figure 7. Example of a component switching.

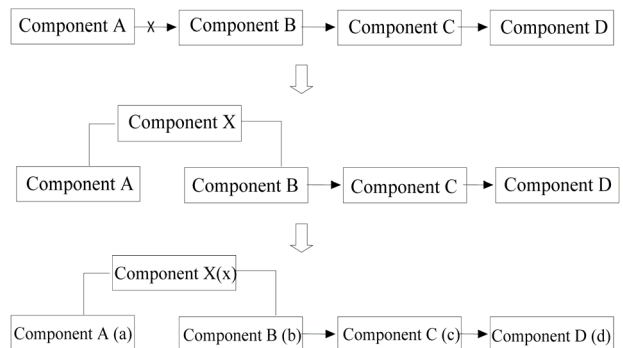


Figure 8. Example of addition of linking component.

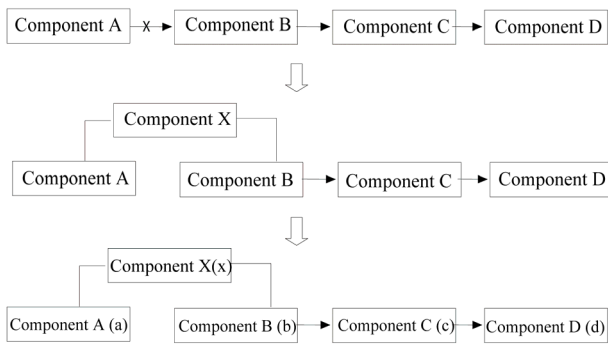


Figure 9. Illustration of assumption 1.

in a series is checked for compatibility in order to avoid **Figure 10** for illustration.

3) If there are more than two components capable of providing a link between two incompatible components, then preference is given to the component with the lowest average value in order to consider the fewest components for a particular product. See **Figure 11** for illustration.

4) The final concepts are ranked according to number of parts, and not functional accuracy. The ranking assumes that the concepts with the least number of parts is most desirable.

3.3.2. Assembly Functions

Components that perform assembly functions (*i.e.*, parts needed to secure, guide, position or couple other components together) are added to the part count tool based on user input in order to generate a more accurate picture of the total number of parts. These components include the screw, nut-bolt, solder, fastener, and rivet. To validate the choices of these components, a check was made to ensure that all assembly functions were covered including guide, position, secure, and couple. For example, the screw, nut-bolt and fastener all solve a coupling connection, based on the definition of coupling [17] which states that two components exhibit a coupling connection if an intermediate artifact aids in the assembly of those components. No external components are required to perform the secure function [17]. The guiding connection requires no intermediate component because by definition mating surfaces must have a moving interface nor does the last assembly function [17]. Position does not require any extra components since this function requires a connection that restricts the movement of the two components in multiple directions and allows the artifact to come loose with the application of additional extra force [17]. Concepts are then ranked based on the total number of parts where the fewest parts implies the highest ranking.

4. Case Studies and Discussion

To illustrate the initial validation of the tool, two cases are

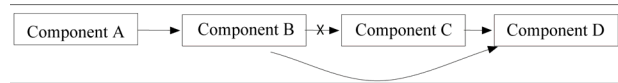


Figure 10. Illustration of assumption 2.

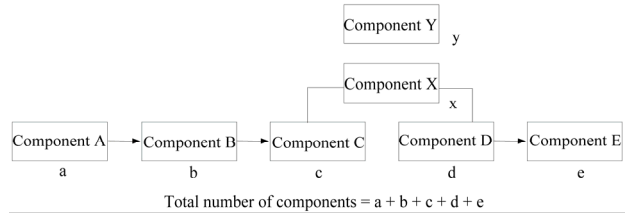


Figure 11. Illustration of assumption 3, where $x < y$.

presented. First, a product that already exists in the Design Repository is checked to make sure the algorithm can accurately recall part count. Second, a functional model for an existing product not currently included in the Design Repository is predicted. Both products' functional models are processed through an automated concept generator algorithm and the output that matches the actual product configuration is submitted to PaCT and the estimated part count is compared to the actual part count of the product.

4.1. Case Study I

The first product to be tested was a Fisher Price racing dog toy for children. Initially, a functional model (**Figure 12**) was created and the interrelationships among the sub functions were expressed in an adjacency matrix. An application to draw functional models, known as FunctionCAD and available at designengineeringlab.org/functioncad/, was used to draw the model and automatically export an adjacency matrix of the functions to the concept generator application. The concept generator provided many design concepts that solved the required functionality. The generated concept that most closely matched the racing dog toy was saved in a component adjacency matrix format that was directly input to the part count tool. Then the part count tool estimated the number of parts required to build the concept.

Table 2 shows that a total of 30 parts were used to manufacture the actual Fisher Price racing dog toy from Design Repository database. The PaCT (**Figure 13**) predicted 43.4 parts to manufacture the concept that most closely resembled the actual product. These 43.4 parts include parts that perform assembly functions. Comparing the parts listed in the part count tool with the actual data reveals that while basic component types are the same, the estimated of total number of parts is higher than the count observed in the real product. The PaCT algorithm appears to provide a conservative estimate of parts in this case.

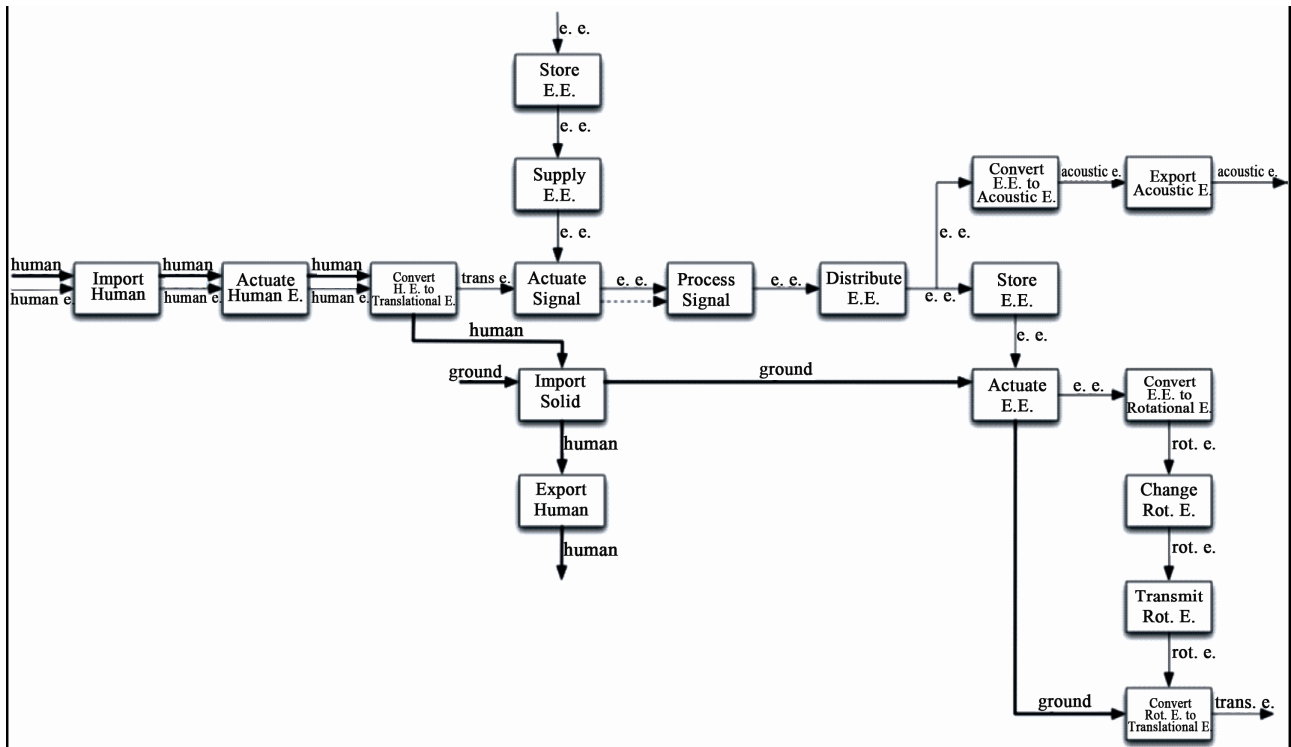


Figure 12. Functional model for the fisher price racing dog toy.

Table 2. Actual number of parts for the fisher price racing dog toy.

Component Name	Number of Distinct Parts
Switch	2
Wire	1
Back Wheel	2
Speaker	1
Motor	1
Screws	9
Shaft	1
Housing	3
Circuit Board	1
Gear	2
Battery Contacts	3
Spring	2
Wheel Assembly	2
Total Part Count	30

Component	Average	Supporting	Notes
1 handle	1.66667	None: +0	
2 gear	3.25926	None: +0	
3 electric swi...	1.54217	None: +0	
4 circuit board	1.52941	None: +0	
5 electric wire	4.67857	None: +0	
6 battery	1.13514	None: +0	
7 electric wire	4.67857	None: +0	
8 electric motor	1.14925	None: +0	
9 gear	3.25926	None: +0	
10 belt	1.18182	None: +0	
11 coupler	1.55556	None: +0	[added]
12 housing	3.59821	Screw: +5.24	
13 battery	1.13514	None: +0	
14 circuit board	1.52941	None: +0	[added]
15 speaker	1.14286	None: +0	
16 circuit board	1.52941	None: +0	[added]
17 housing	3.59821	None: +0	
18 Total	38.1689	43.4089	

Figure 13. Concept with supporting components.

4.2. Case Study II

The PaCT was next tested on a SKIL PowerTwist Screwdriver shown in Figure 14, a product that has not been



Figure 14. SKIL power screwdriver in assembled form.

incorporated in the Design Repository. The power screw driver was dissected (Figure 15) to determine that it was comprised of 24 distinct parts representing 9 general component types. Table 3 lists the components and their respective part counts. A functional model for the power screwdriver was constructed, shown in Figure 16, and submitted as input to the automated concept generator. The tool generated concepts based on the functional model diagram and the concept shown in Figure 17 was determined to be the closest match to the actual components in the screwdriver

The number of parts in a screwdriver predicted by the PaCT compares favorably to the number of parts found when the tool was dissected, as shown in Figure 18. In general, however, the part count tool added an extra housing for the power drill. The average part count for the component type housing is 3.61 based data in the repository. In this case the average is an overestimate because the real product is comprised of a two piece housing. The concept variant in Figure 18 uses only ten distinct component types to solve the functionality of the power screwdriver. The reduction in component types from the raw concept generator output is due to the assumptions of algorithm, the effects of which are visible in the use of a single housing for three consecutive functions rather than a separate housing for each (see Figure 17). Without this assumption, PaCT would significantly overestimate the total number of parts. In a similar case a single battery

permits the storage, supply, and transfer of electrical energy (see Figure 17) and makes the concept more modular than one that relied on an intermediate electric wire to transfer the electrical energy to the switch. By default, the part count tool does not consider the supporting components, but in this case screws were used in addition to snap fits to connect the two halves of the housing. The option of modifying the results with assembly components was chosen to mimic the actual SKIL power screwdriver. Data in the Design Repository indicates that a screw is a likely choice to join two housings. An average of 5.24 parts is found for products in the database that contain the general component type of screw. Adding these additional parts brings the total part count to 29.13. Figure 19 shows the concept with supporting components. Again, PaCT is found to give a conservative estimate for the products actual part count.



Figure 15. SKIL power screwdriver in disassembled form.

Table 3. Total number of parts of SKIL power screwdriver.

Component Name	Number of Distinct Parts
Housing	3
Screw	4
Planetary Gear Set	8
Spring	1
Switch	1
Battery	1
Motor	1
Chuck Assembly	4
Bit Holder	1
Total Part Count	24

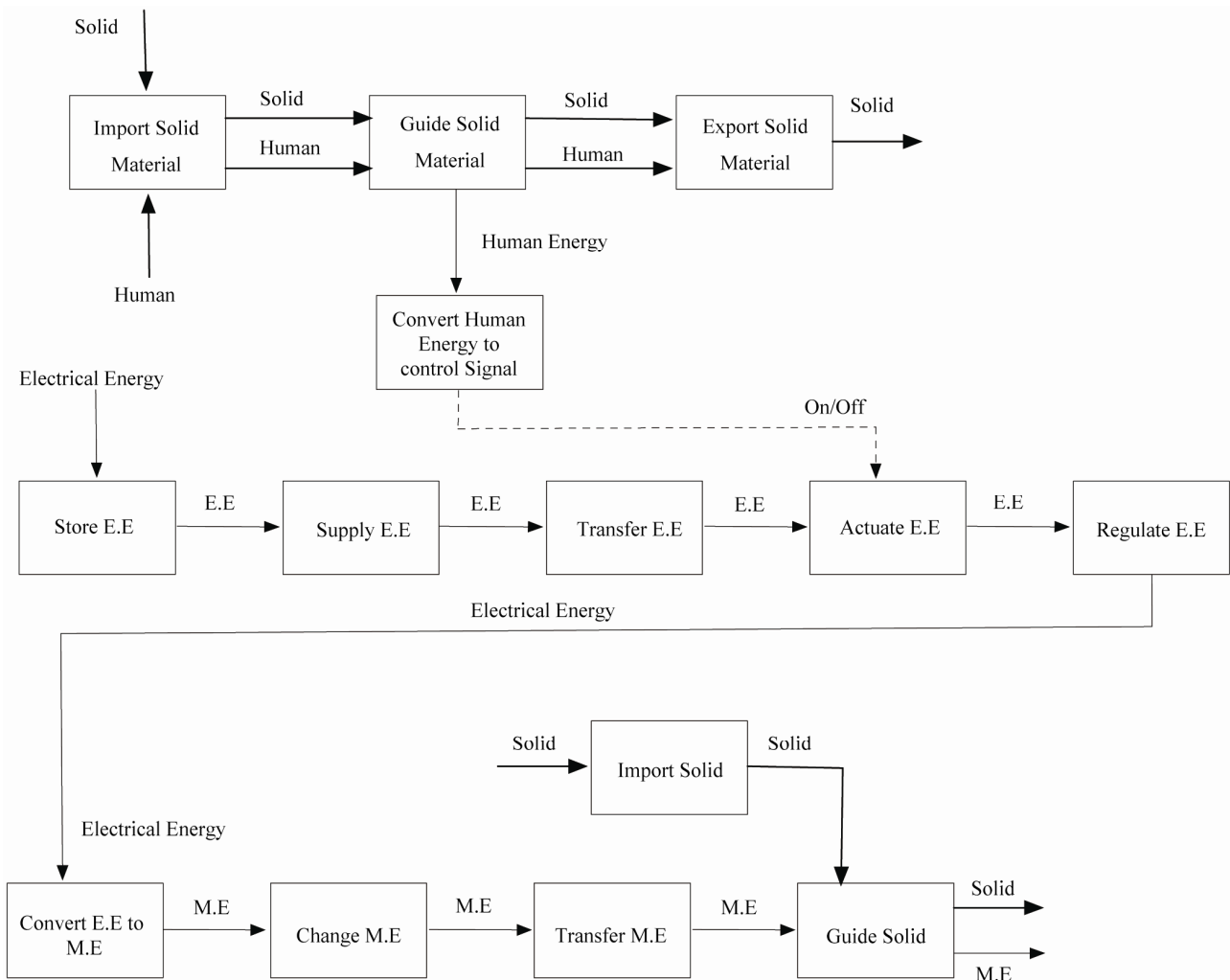


Figure 16. Functional model diagram for the SKIL power screwdriver.

Subfunction	Component Solutions
import solid material	housing (21%)
guide solid material	housing (29%)
export solid material	housing (11%)
convert human energy to control signal	electric switch (70%)
actuate electrical energy	electric switch (84%)
store electrical energy	battery (97%)
supply electrical energy	battery (80%)
transfer electrical energy	battery (3%)
regulate electrical energy	regulator (3%)
convert electrical energy to mechanical energy	electric motor (94%)
transfer mechanical energy	gear (26%)
import solid material	housing (21%)
change mechanical energy	gear (75%)
guide solid material	housing (29%)

Figure 17. Concept selected from the concept generator.

	Component	Average	Supporting	Notes
1	housing	3.59821	None: +0	
2	electric swi...	1.54217	None: +0	
3	battery	1.13514	None: +0	
4	electric motor	1.14925	None: +0	[swapped]
5	regulator	1.2	None: +0	[swapped]
6	coupler	1.55556	None: +0	[added]
7	gear	3.25926	None: +0	
8	housing	3.59821	None: +0	
9	gear	3.25926	None: +0	
10	housing	3.59821	None: +0	
11	Total	23.8953	23.8953	

Figure 18. Concept without assembly components.

	Component	Average	Supporting	Notes
1	housing	3.59821	Screw: +5	
2	electric swi...	1.54217	None: +0	
3	battery	1.13514	None: +0	
4	electric motor	1.14925	None: +0	[swapped]
5	regulator	1.2	None: +0	[swapped]
6	coupler	1.55556	None: +0	[added]
7	gear	3.25926	None: +0	
8	housing	3.59821	None: +0	
9	gear	3.25926	None: +0	
10	housing	3.59821	None: +0	
11	Total	23.8953	29.1353	

Figure 19. Concept with assembly components.

5. Conclusions and Future Work

The PaCT tool presented in this paper allows design for manufacture and assembly (DFMA) considerations to be evaluated at the concept selection stage by linking component part counts from existing products to product functionality. This finding provides designers with a more realistic prediction for the number of parts in a new design that lends itself to manufacture and assembly cost forecasting. This tool allows for the rapid comparison of the large set of results returned by automated concept generators.

Analysis of both the case studies reveals that the part count tool conservatively predicts a greater number of parts than exist in either product. For the Fisher Price racing dog toy and the Skil screwdriver the predicted number of parts was 20% - 30% higher than the actual number. The conservative estimates of the part count provided by PaCT are directly linked to the assumptions described in Section 3.2.1. While a conservative model is typical in engineering, increased accuracy can likely be found by improving the assumptions for handling repeated components within a concept. The addition of supporting components to the tool, however, does offer a more accurate picture of the components needed to assemble the main parts together and, thus, the total number of parts.

In addition to part count, PaCT also is able to find connections between two incompatible components by either rearranging the existing concept's components or adding a new component as an intermediary. This algorithm closes a gap in the current automated concept generation routines. However, the addition of a component that might introduce unintended functionality present issues that must be considered by the designer prior to acceptance or level one headings in your article.

6. References

- [1] G. Boothroyd, P. Dewhurst and W. Knight, "Product Design for Manufacture and Assembly," 2nd Edition, Marcel Dekker, New York, 2002.
- [2] M. Bohm and R. Stone, "Representing Product Functionality to Support Reuse: Conceptual and Supporting Functions," *Proceedings of Design Engineering Technical*, Salt Lake City, 28 September-2 October 2004.
- [3] C. Bryant, R. Stone, D. Mcadams, T. Kurtoglu and M. Campbell, "Concept Generation from the Functional Basis of Design," *Proceedings of International Conference on Engineering Design ICED 05*, Melbourne, 15-18 August 2005.
- [4] T. Kurtoglu and M. Campbell, "Automated Synthesis of Electromechanical Design Configurations from Empirical Analysis of Function to Form Mapping," *Journal of Engineering Design*, Vol. 20, No. 1, 2009, pp. 83-104. [doi:10.1080/09544820701546165](https://doi.org/10.1080/09544820701546165)
- [5] G. Pahl, W. Beitz, J. Feldhusen and K. H. Grote, "Engineering Design a Systematic Approach," Springer-Verlag, London, 2007.
- [6] K. Otto and K. Wood, "Product Design: Techniques in Reverse Engineering, Systematic Design, and New Product Development," Prentice-Hall, New York, 2001.
- [7] J. Hirtz, R. Stone, D. Mcadams, S. Szykman and K. Wood, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," *Research in Engineering Design*, Vol. 13, No. 2, 2002, pp. 65-82.
- [8] K. T. Ulrich and S. D. Eppinger, "Product Design and Development," McGraw-Hill/Irwin, Boston, 2004.
- [9] D. G. Ullman, "The Mechanical Design Process," 3rd Edition, McGraw-Hill, Inc., New York, 2002.
- [10] M. Bohm, R. Stone and S. Szykman, "Enhancing Virtual Product Representations for Advanced Design Repository Systems," *Journal of Computer and Information Science in Engineering*, Vol. 5, No. 4, 2005, pp. 360-372. [doi:10.1115/1.1884618](https://doi.org/10.1115/1.1884618)
- [11] M. R. Bohm, R. B. Stone, T. W. Simpson and E. D. Steva, "Introduction of a Data Schema: The Inner Workings of a Design Repository," *Proceedings of ASME International Design Engineering Technical Conferences*, Philadelphia, 10-13 September 2006.
- [12] C. Bryant, D. Mcadams, R. Stone, T. Kurtoglu and M. Campbell, "A Computational Technique for Concept Generation," *Proceedings of ASME Design Engineering Technical Conferences*, Long Beach, 24-28 September 2005.
- [13] V. Hubka and W. Ernst-Eder, "Theory of Technical Systems," Springer-Verlag, Berlin, 1984.
- [14] L. Schmidt and J. Cagan, "Ggreada: A Graph Grammar-Based Machine Design Algorithm," *Research in Engineering Design*, Vol. 9, No. 4, 1997, pp. 195-213. [doi:10.1007/BF01589682](https://doi.org/10.1007/BF01589682)
- [15] A. Ward and W. Seering, "Quantitative Inference in a Mechanical Design 'Compiler'," *Journal of Mechanical Design*, Vol. 115, No. 1, 1993, pp. 29-35. [doi:10.1115/1.2919320](https://doi.org/10.1115/1.2919320)

- [16] A. Ward, "A Theory of Quantitative Inference for Artifact Sets Applied to a Mechanical Design Compiler," PhD dissertation, Massachusetts Institute of Technology, Cambridge, 1989.
- [17] V. Rajagopalan, C. R. Bryant, J. Johnson, D. Mcadams, R. Stone, T. Kurtoglu and M. Campbell, "Creation of Assembly Models to Support Automated Concept Generation," *Proceedings of ASME Design Engineering Technical Conferences*, Long Beach, 24-28 September 2005.