1-2014

Assembly Time Estimation: Assembly Mate Based Structural Complexity Metric Predictive Modeling

Joseph E. Owensby
Clemson University

Joshua D. Summers
Clemson University, jsummer@clemson.edu

Assembly Time Estimation: Assembly Mate Based Structural Complexity Metric Predictive Modelling

Joseph E. Owensby, Joshua D. Summers

Mechanical Engineering, Clemson University, Clemson, SC, USA

Contact Author: Joshua D. Summers, Professor
250 Fluor Daniel Building
Mechanical Engineering
Clemson University
Clemson, SC 29634-0921
jsummer@clemson.edu

This paper presents an automated tool for estimating assembly times of products based on a three step process: connectivity graph generation from assembly mate information, structural complexity metric analysis of the graph, and application of the complexity metric vector to predictive artificial neural network models. The tool has been evaluated against different training set cases, suggesting that partially defined assembly models and training product variety are critical characteristics. Moreover, the tool is shown to be robust and insensitive to different modelling engineers. The tool has been implemented in a commercial CAD system and shown to yield results of within +/- 25% of predicted values. Additional extensions and experiments are recommended to improve the tool.

Keywords: Design for Assembly, DFA, Assembly Time, Complexity, Artificial Neural Networks

1 Motivation: An Automated Tool for Assembly Time Estimation

The authors present a new computational design tool for estimating assembly times. This tool consists of three major components: a graph generator from computer aided design (CAD) assembly models, a structural complexity metrics generator, and an artificial neural network (ANN) modeller to predict assembly times. The tool uses the assembly mates defined within a CAD model, as defined by the designer, to create a connectivity graph. This graph is then evaluated against a suite of structural complexity metrics that are fed into an ANN based predictive model. This tool has been integrated into a commercial CAD software package and evaluated with respect to training size, assembly model authorship, and level-of-mate definition.

This paper presents the motivation for developing an assembly time estimation tool based on design for assembly methods and a review of previous efforts. This is followed by a discussion on the algorithm for automated assembly time estimation based on graphs resulting from assembly mate models. The tool is validated through external testing and a sensitivity analysis on the impact that different approaches to creating the mating models has on the estimation effectiveness. Finally, the limitations of this approach is discussed and future extensions identified.

1.1 Design for Assembly (DFA)

Design for Assembly (DFA) methods have been evolving since the 1960's, progressing from basic rules and guidelines to the creation of automated analysis tools, as detailed in Table 1 [1–4]. DFA works by estimating time for the assembly and providing recommendations for changing the components to improve this time. The first function (estimating time) is of interest here.

Table 1: Existing DFA Methods

DFA Method	Description	Developer	Date	Ref.
Methods-Time Measurement (MTM)	Assign operations with pre-defined assembly times to parts	Academic (Maynard)	1948	[5,6]
Manufacturing Producibility Handbook	Reference manual of manufacturing and assembly guidelines	Corporation (GE)	1960	[2]
Boothroyd and Dewhurst DFA	DFA based on minimum part criteria and handling and insertion difficulties	Academic & Consulting (Boothroyd and Dewhurst)	1977	[2,7]
Assembly Evaluation Method (AEM)	DFA based on one motion for one part	Corporation (Hitachi)	1980	[2,8–10]
Design for Assembly and Cost Effectiveness (DAC)	Uses 30 key words to evaluate design	Corporation (Sony)	1988	[2,11]
Assembly Oriented Product Design	Accesses a parts functional value	Academic (Warnecke and Bassler)	1988	[2]

Lucas DFA Method	Set of questions to determine assembly time	Academic & Consulting (Miles and Swift)	~1986	[2,12]
MOSIM	Focus of implementing DFA through CAD software	Corporation (Angermuller & Moritzen of Siemens)	1990	[2]
DFA Sandpit	Proactive DFA software based on original Lucas method	Academic (Swift and Jared)	2000	[13,14]

In the 1980's, the original guidelines published in the manuals of the 1960's were integrated into systematic qualitative/quantitative DFA analysis tools to help designers predict the product assembly times based on extensive time studies. Upon creation of these table based methods, researchers began to implement DFA using computer software to improve speed and ease of the analysis. These industrial tested DFA methods have proven advantageous in reducing a product's total part count, manufacturing cost, production lead time, inventory, assembly time, and assembly cost [15,16]. There are recognized limitations to these methods, however, namely the subjectivity of inputs [13,17], significant user inputs [18], and the reactive nature of the tool [19,20]. It is these limitations the authors address through the assembly mate based time estimation system. Specifically, i) system inputs are entirely objective as the assembly mates defined by the designers; ii) additional user inputs are not needed, and iii) the tool can be used in real time once assembly models are available in the CAD system.

1.2 Previous Efforts in Automated Time Estimation

The Connectivity Complexity DFA is one method used to solve the subjective issues of existing DFA methods preventing automation [21]. Developed using linear regression to identify a relationship between a product's assembly time and the complexity of the inter part connections; this method predicts assembly times from products inter connectedness complexity. The advantage of over existing methods is

that the physical connections between parts in an assembly can be identified objectively. The initial results predicted assembly times within +/- 15% of the training times used, proving that a product's connection complexity can be used to determine product assembly times [21].

To assess the potential utility, the Connectivity Complexity DFA method was compared to the Boothroyd Dewhurst DFMA software based on i) approximate time for analysis, ii) predicted assembly time, iii) amount of required input and subjective information, and iv) the number of redesign features [18]. It was determined that the Boothroyd Dewhurst DFMA software required users to answer forty nine questions per part, sixteen of which were subjective. The Connectivity Complexity method, however, only required that users answer five questions per part, none of which are subjective. The predicted assembly times of the Connectivity Complexity method ranged from 13.11% to 49.71%, lower than the predicted times of the DFMA software considered as the baseline. Both methods required a similar implementation time. Though the evaluation suggests that the Boothroyd DFMA software is effective, extensive subjective user inputs which are difficult to program are required. Based on this evaluation, though the Connectivity Complexity method can be automated as it only requires objective information, its accuracy can be improved [18]. This estimation method using manual graph generation and regression fit is V1 in the evolution of using structural complexity metrics to predict assembly times of Figure 1.

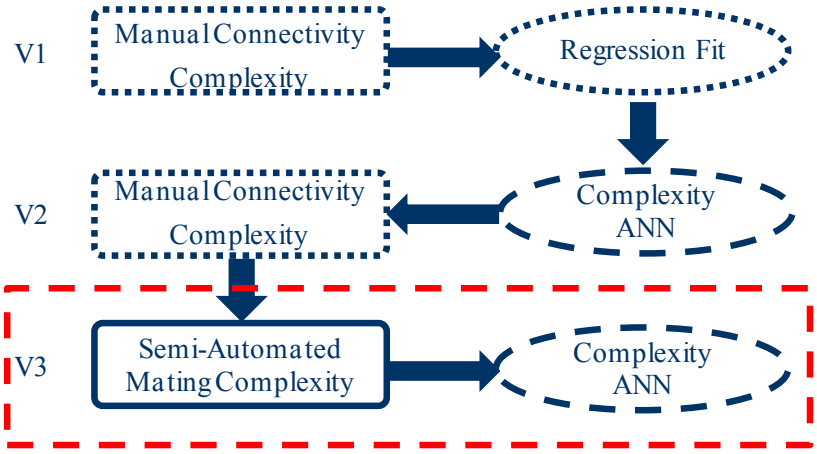

Figure 1: Connectivity Complexity DFA development flow chart

The original work (V1) used linear regression training and acted as a proof-of-concept to show the use of physical connections between parts to determine product assembly times [21]. The continuation of the work (V2) implemented the ANN training to improve the accuracy of the predicted assembly times [22]. The work presented here relates to the third attempt to develop an objective and automated assembly time estimation tool. During the early development of the structural complexity method, part connections within a product were identified early in the design process [18]. The inter-part connections required here can be extracted from sketches and 3D CAD models which are generated as early as the conceptual design phase, making it applicable throughout the design process [23,24]. Extracting the connections from assembly models also enables creation of a program to automate this method. The rest of this paper presents the development of an automated structural complexity metric based assembly time prediction method.

2 Automation of Structural Complexity Assembly Time Prediction Tool

This automated time estimation tool has three basic steps: graph generation, complexity analysis of graph, and application of ANN predictive model. Figure 2 shows a flow diagram of the SolidWorks (SW) mate extraction add-in, its required inputs, the information processing steps, and the assembly time output.

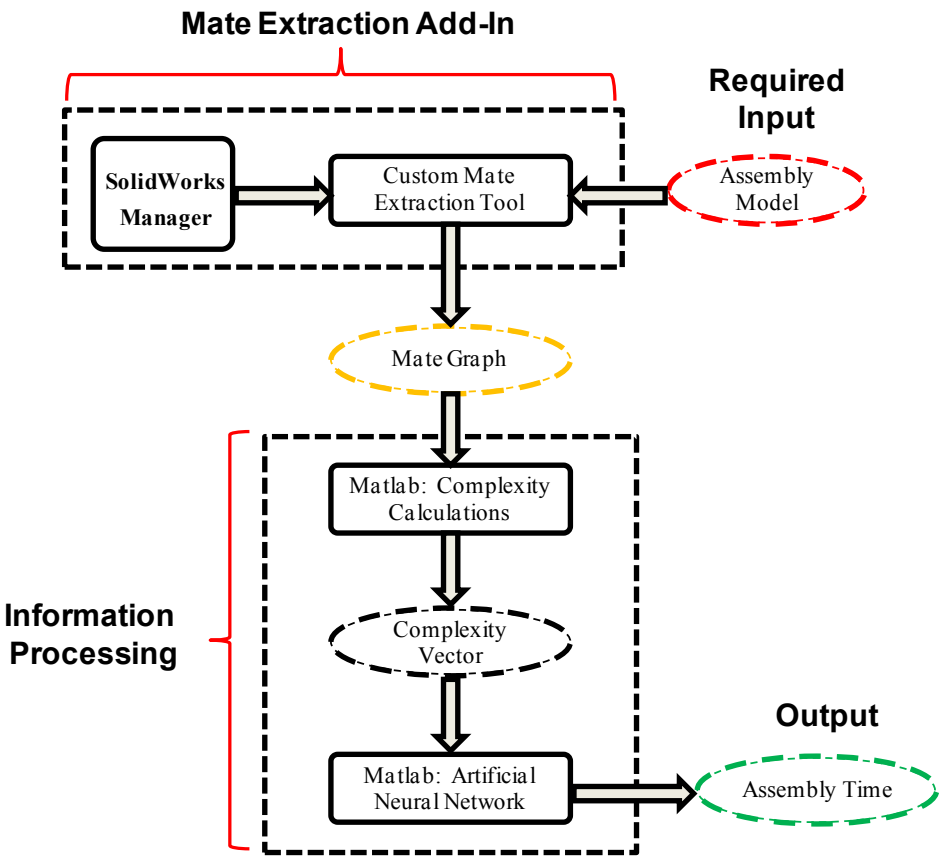

Figure 2: SW mate extraction add-in and information processing

The mate extraction add-in (top box of Figure 2) generates a connectivity graph that represents the product inter-part connections. This connectivity graph is processed external from the mate extraction add-in. The external processing is performed using MatLab where custom algorithms are used to generate a complexity vector of the mate graph; this vector along with previously trained ANNs is used to predict an assembly time. Before the information processing can be accomplished, the ANNs must be created and trained as explained below. Each of these steps is discussed in the following sections.

2.1 Step 1: Graph Generation

Two approaches for automated graph generation have been explored. The first, an implicit based approach [25] that extracts potential mating pairs of parts based on duplicate geometry [26,27], has limited efficiency and computational time. The

second approach, employing explicit information contained within CAD assembly models, is the focus of this paper and is fully reported elsewhere [28]. The explicit information chosen is the assembly mates defined within the models by the designer. In this manner, not only is an objective tool developed based on explicitly available information, this information is also closer to capturing the design rationale. Examples of mate relations within SolidWorks, the commercial package within which the tool is built, include concentric, coincident, angle, and locked mates. A challenge to this approach is that a single collection of parts can be mated in different ways, resulting in different connectivity graphs and the resulting structural complexity metric values. Consider the simple assembly model of Figure 3. These three parts (A, B, C) can be mated with different approaches to yield the same assembly (Table 2).

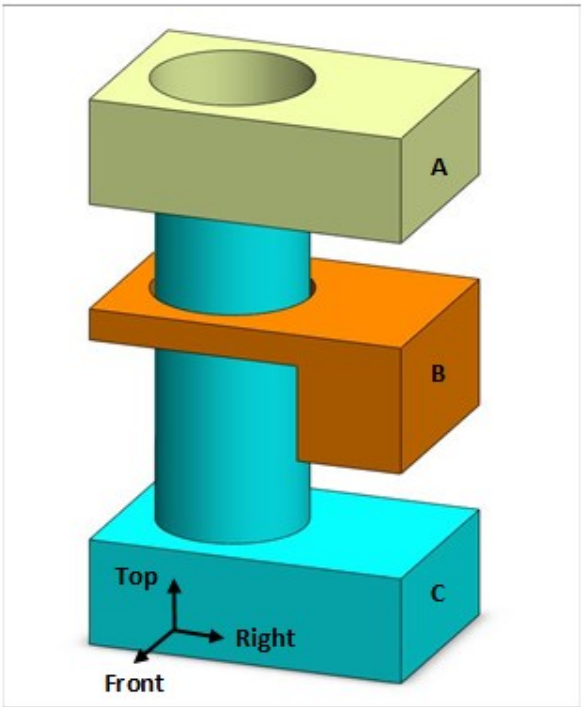

Figure 3: Part A, Part B, and Part C, mated or constrained in a variety of ways

Table 2: Mate configurations for Parts A, B, and C

Parts	Configuration 1	Configuration 2
C and B	C shaft concentric with B hole	C face right aligned with B face right
C and B	C face top coincident with B face bottom	C face top coincident with B face bottom
C and B	C face right parallel with B face right	C face front aligned with B face front
B and A	B hole concentric with A hole	B face right aligned with A face right
B and A	B face top coincident with A face bottom	B face top coincident with A face bottom
B and A	B face right parallel with A face right	B face front aligned with A face front

The tool used for extracting mate information from assembly models was developed using SolidWorks 2010 API Software Development Kit (SDK)[1]. SolidWorks (SW) is a commercial three dimensional modelling software package which provides an intuitive Graphical User Interface (GUI). The software offers two options to develop the SolidWorks API application, macros and add-in programming. Though macros tend to speed the development of automations, they are limited in scope as they replicate user actions within the GUI. If an automation component requires information that cannot be extracted from the GUI interface actions, then a separate add-in is required. This is the case for extracting mate information from SolidWorks assembly models. The algorithm implemented in the add-in programming environment, through C++ coding, is shown in Figure 4.

[1] http://www.solidworks.com/ (accessed September 17, 2012)

```
Get active assembly document
Get features list from feature manager tree
If feature = mate list
        Get Mate list from feature list
                For each mate in Mate list
                Get parts connected by mate
                Add parts to graph
        End
End if
```

Figure 4: Pseudo-code for Extracting Mate Information

To obtain the mate information from an assembly file, the program traverses through the feature types in the feature manager tree. A screen shot of the SW feature manager design tree for a Black & Decker Drill can be seen in Figure 5. This figure labels three main sections of the feature manager design tree: reference features, parts and sub-assemblies, and mates. Within the main assembly, everything in the feature manager design tree is recognized as an assembly feature. Information within the sections of the feature manager design tree may include annotations, co-ordinate planes, part names, part features, and part constraints.

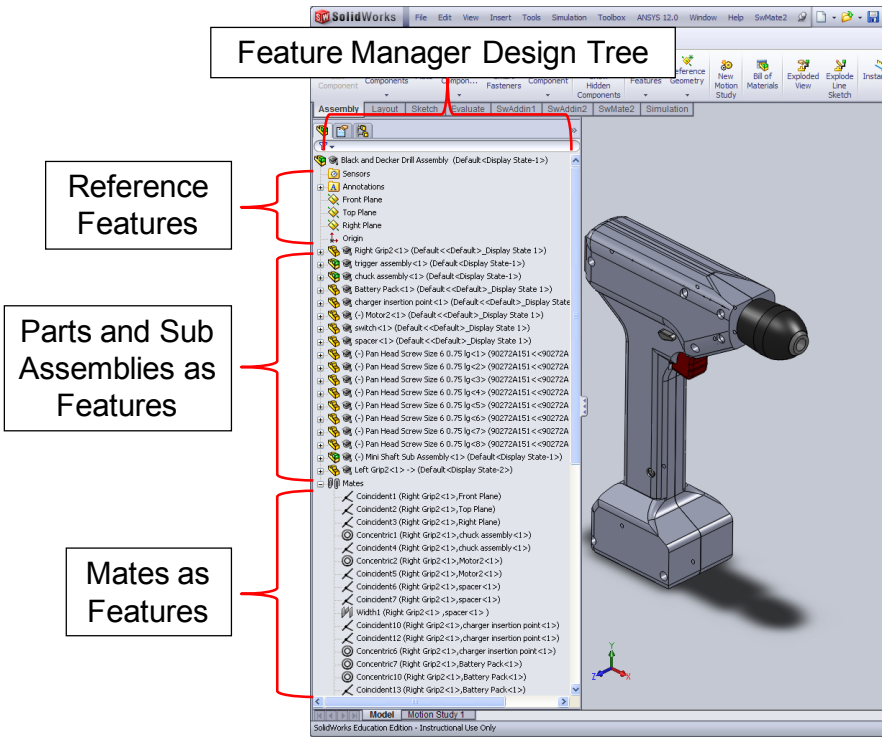

Figure 5: SolidWorks feature manager design tree

The program traverses through the feature manager tree until it reaches a container with mate information. Each mate consists of the name of the mate and the names of parts constrained by that mate. For each mate, the names of both parents (parts) are retrieved, indicating the connection between the parts. The names of the connected parts are then stored in a bi-partite table and saved as a *.csv file. This process is iterated until all connections between the parts are extracted from the feature manager tree.

2.2 Step 2: Complexity Metrics for Connectivity Graphs

Once the bi-partite table containing the mate connections found in the assembly file is generated, the complexity of the table based graph can be calculated using a custom MatLab program [29]. The program currently evaluates 29 distinct complexity metrics. Rather than evaluating a single complexity metric [30–32], the authors use a set of metrics to realize pattern discovery through the ANN models of the final step. The metrics evaluated are classified as size, interconnectivity, centrality, and decomposition [33].

Size is a common measurement used in complexity measurement. The size of an object is based on the count of some classification of the object within the system; as the value increases so too does the complexity [31]. While counts are the most intuitive form of complexity measurement, their contribution to complexity is non-linear [34]. When the count is low, the addition of one more is significant, while the opposite is true of high-count systems.

The interconnectedness of a graph can be evaluated through path length and flow capacities. Path length measurements are based on the number of relationships that must be passed through to travel from one element to another [35,36]. For example, a path length of two from node A to node C is necessary to travel through

the system A→B→C. Flow capacity measurements, in turn, are based on the number of unique paths between each pair of nodes. Here, the capacity is determined by the availability of edges, with each edge assumed to have a capacity of one and nodes assumed to have infinite capacity [37]. While shortest-path-length metrics address the existence of connection within the system, flow-capacity metrics elucidate the volume of information that is passed within the system.

Centrality, addressing relative importance of nodes within a system, assumes many forms in network analysis [38–41]. Two forms of centrality are employed here: betweenness centrality, a measurement on the number of shortest paths on which a node occurs [38]; and the clustering coefficient, a measure of the degree to which nodes are grouped within the system [42]. Regarding individual nodes, the clustering coefficient is a measure of the degree to which a given node and its neighbours will form a clique, or complete graph. This is defined as the percentage of nodes to which the given node is connected and which are connected to each other.

The final measurement is decomposability, used to inventory the requisite steps for structural disassembly of a system. As a measure of complexity, the decomposability score increases with ever larger and more complex systems; thus, what is measured is the difficulty of a disassembling a system set-by-set. The Ameri-Summers decomposability algorithm [43] is one measure of decomposability. Each step consists of removing those relationships that link to the elements with the fewest connections. Each additional step, relationship set, or relationships per separated element required to decompose the system is considered to increase the complexity. In an additional measure of decomposition, core numbers are the largest integer such that the given element exists in a graph where all degrees are at least that integer [44].

These degrees are subsequently separated into measurements relating to the in-degree and out-degree of each node in digraphs.

Table 3 classifies the metrics that are used in the graph analysis. This resulting complexity vector will be used along with Artificial Neural Networks (ANNs) to predict a products assembly time. For brevity, five of the metrics and their mathematical definitions are illustrated in Table 4. The comprehensive list and all associated algorithms are found in [29,33].

Table 3: Twenty-Nine Complexity Metrics Used in Graph Analysis

Class	Type	Metric
Size	Dimensional	Elements (DSE)
		Relationships (DSR)
	Connective	Connective Size (CS)
		Degree of Freedom (DOF)
Interconnection	Shortest Path Length	Total Shortest Path Length (TPL)
		Maximum Shortest Path Length (MPL)
		Average Shortest Path Length (APL)
		Shortest Path Length Density (PLD)
	Flow Capacity	Flow Capacity Sum ($\sum FC$)
		Maximum Flow Capacity (FC_{max})
		Mean Flow Capacity ($\overline{FC}$)
		Flow Capacity Density (FC_d)
Centrality	Betweenness	Betweenness Sum ($\sum C_B$)
		Maximum Betweenness ($C_{B_{max}}$)
		Mean Betweenness ($\overline{C_B}$)
		Betweenness Density (C_{B_d})
	Clustering Coefficient	Clustering Coefficient Sum ($\sum C_{CC}$)
		Maximum Clustering Coefficient ($C_{CC_{max}}$)
		Mean Clustering Coefficient ($\overline{C_{CC}}$)
		Clustering Coefficient Density (C_{CC_d})
Decomposition	General	Ameri-Summers (ASA)
	Core (In)	In Core Number Sum ($\sum CN_i$)
		Maximum In Core Number ($CN_{i_{max}}$)
		Mean In Core Number ($\overline{CN_i}$)
		In Core Number Density (CN_{i_d})
	Core (Out)	Out Core Number Sum ($\sum CN_o$)
		Maximum Out Core Number ($CN_{o_{max}}$)
		Mean Out Core Number ($\overline{CN_o}$)
		Out Core Number Density (CN_{o_d})

Table 4: Example Complexity Metrics Explored in the Interpretability Analysis Study [29,33]

Name	Description	Mathematical Definition
Connected Size (CS)	Number of arcs within the bipartite graph	$CS = \sum_{i=1}^{N_E} DEG\{N_i\}$
All-Pairs Shortest Path (TPL)	The sum of the lengths of the shortest path between each pair of entities. SP defines the shortest path between element pair	$TPL = \sum_{i=1}^{N_E}\sum_{j=1}^{N_E}(SP\{E_i, E_j\})$
Average Shortest Path Length (APL)	The average of all the shortest paths between each pair of entities.	$APL = \frac{TPL}{N_E^2 - N_E}$
Maximum Shortest Path Length (MPL)	The maximum path length from all shortest paths between each pair of entities.	$MPL = \max(SP\{E_i, E_j\})$
Path Length Density (PLD)	The Average Shortest Path Length divided by the number of relations	$PLD = \frac{APL}{N_R}$

2.3 Step 3: ANN Prediction Tool

The final step of the time estimation tool uses trained Artificial Neural Networks (ANN) based on the input pair of the complexity metric vector and the known assembly time. The trained ANN, currently implemented within the MatLab ANN toolbox, predicts new assembly times when given a complexity vector. Training an ANN requires a set of inputs and respective target values to effectively identify relationships between them. Once an effective set of inputs and targets has been compiled it can be reused in future implementations, thusly eliminating the training process from the final tool implementation. The next section describes the selection method for creating a database of assembly models and times that can be used for training.

2.3.1 Collecting Product 3D Assembly Models

To populate an effective ANN training set, a collection of 3D assembly models is required. For each model, an assembly time is needed and is generated based on the Boothroyd and Dewhurst (B&D) method [45], since the actual assembly times are not available. The models on which the method is applied are derived from direct reverse engineering of products, an on-line CAD repository[2], SolidWorks 3D Content, and from OEM assembly models available from past projects [46]. The example database of assemblies is found in Table 5. The reverse engineered models were created independently by different students within the CEDAR (Clemson Engineering Design Applications and Research) group as part of several other on-going projects separate from this effort.

[2] http://gicl.cs.drexel.edu/wiki/Main_Page (accessed September 17, 2012)

Table 5: Collection of product assembly models

#	Product	Assembly Model Generation
1	G2 Pen	Reverse Engineered
2	Pencil Compass	Reverse Engineered
3	Solar Yard Light	Reverse Engineered
4	Pony Vise	Reverse Engineered
5	Black and Decker Drill	Reverse Engineered
6	Paper Pro Stapler	GICL Website[2]
7	6" MagLight	SW 3D Content[1]
8	Indoor Electric Grill	SW 3D Content[1]
9	Shift Frame LH	OEM
10	Wide Flag	OEM

An example of an exploded view for one of the OEM components is found in Figure 6.

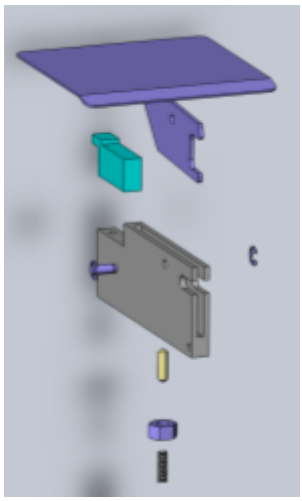

Figure 6: Exploded view of OEM Wide Flag Assembly

Each of these assembly models are defined within SolidWorks with the mates that are available within the CAD system. Complexity vectors are generated automatically for each of these products, and assembly times are developed for each product. Should a company wish to deploy this system in their design group, company specific assembly models can be collected and used for training purposes

with known product assembly times. These historical models, ideally collected from different projects, have been authored by different designers with different levels of component count and mating resolution. Specific strategies for selecting and developing ANN training models are reserved for future work.

Though the physical products for items 1-6 in Table 5 were obtained, items 7-10 could not be located or lacked a specific consumer product to match the SolidWorks model including product generational changes that did not match exactly. Without the physical product, applying the Boothroyd DFA method is difficult since the objective and subjective analysis questions typically require a true understanding of how the product is assembled. To solve this problem a combination of DFA analyses were conducted, evaluated, and used. First a "virtual" Boothroyd DFA analysis was conducted on the SolidWorks Assembly model. The challenge with this "virtual" method is that without disassembling and holding the actual parts, an understanding of the product structure, function, assembly sequence, handling difficulties, and insertion difficulties cannot be obtained which is essential when applying the Boothroyd DFA. The challenges of determining the handling and insertion difficulties come because such information requires the designer to answer subjective questions about the product [17]. For example, if a part is either difficult to grasp or has resistance to insertion, it is challenging to assess this difficulty without physically picking up the part and inserting it.

Once the "virtual" Boothroyd DFA was completed, if a physical product was present that matched the SolidWorks model, it was disassembled and the DFA analysis was conducted as well. The "virtual" Boothroyd DFA method was always conducted first to reduce the chance that a handling or insertion difficulty experienced during the physical analysis would influence the designer during the "virtual"

analysis. Between the Boothroyd DFA analyses on the physical products and the virtual products a total of sixteen assembly times to match the respective CAD assembly models were determined.

2.3.2 Training of Mate Complexity DFA Method

The research on the connectivity complexity method previously conducted used ANNs to increase the accuracy of the original connectivity complexity DFA method [21] Artificial neural networks were selected to identify the relationship between the products connectivity complexity vector and respective assembly times because they are often used to complete nonlinear statistical analyses [47]. The complexity vectors and assembly times of the Pencil Compass, the 6 Inch MagLight, and the Black and Decker Drill from Table 5 were held back for use as test inputs once the ANN training was completed. These three products were chosen for testing because their part counts and assembly times form a good representation of the training set.

To train the ANNs for this research, 189 architectures were generated, consisting of one to three layers with up to fifteen neurons per layer depending on the configuration. Each architecture was given the training set 100 times so that probability densities could be used to better approximate the relationship. The probability density plots can be generated for each product based one ANN structure replicated 100 times (Figure 7). In Figure 7, the function is shown with the target time illustrated as the vertical line near the function peak. The ANN training inputs consisted of eleven complexity vectors for eleven of the sixteen assembly times. If a product had both a virtual and physical Boothroyd DFA predicted assembly time then the same complexity vector for that product would be trained towards the two different assembly times. Once the training inputs and targets were compiled, the

different ANN architectures were trained with the best selected and evaluated for later use as described above.

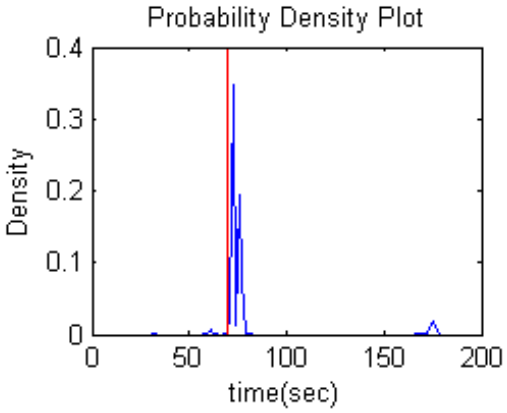

Figure 7: Example Probability Density Plot

Three separate Artificial Neural Networks training sets using different inputs and targets were evaluated to determine if the number of mates affected the predicted results. The first training set (Case 1) was generated using complexity vectors based on all of the SW models being fully defined, indicating that assembly parts are fully constrained by mates and cannot move. The second training set (Case 2) was generated using complexity vectors based on the partially defined SW models, achieved by having the designer mate the assembly model to the point where parts are constrained due to design intentions. The third training set (Case 3) was generated using both the complexity vectors generated for the fully defined and partially defined SW assembly models, indicating that Case 3 had twice as many training inputs and targets than either Case 1 or Case 2.

The average probability for all 189 architectures for predicting the assembly time was then found and compared to determine that which would be most effective at predicting an assembly time within the specified target range. The five architectures with the highest average probabilities were selected for evaluation. Table 6 shows these architectures selected for the three training schemes.

Table 6: Selection of top 5 ANN architectures for each testing case

Case 1 (F. Def.)		Case 2 (P. Def.)		Case 3 (F&P Def.)	
Arch.	Avg. Prob.	Arch.	Avg. Prob.	Arch.	Avg. Prob.
95	0.601	56	0.999	109	0.992
173	0.541	64	0.963	45	0.736
79	0.537	174	0.789	154	0.699
90	0.500	147	0.753	30	0.639
99	0.500	52	0.737	133	0.625

Case 2, trained with the partially defined products, yielded the overall best top five architectures based on the probability density curves. ANN training Case 3 which used fully and partially defined products was next, while training Case 1 which used only fully defined products was least effective. The mates added to parts in an assembly define the constraints of that part within that assembly. If a designer must add more mates than required, the original constraint definition may either be lost or negatively affected. As this may reduce the predictive capacity of fully defined assembly models, a detailed investigation into this issue is reserved for future work. For comparison, the times for each of the top five architectures for each training case, were compared across the three test products.

To determine the effectiveness of each ANN training scheme, their predicted assembly times are compared using the top five architectures for each ANN training scheme (Table 7). Shaded cells illustrate the level of accuracy for various tests (green - returned values are within +/- 25% tolerance; yellow - values are within +/- 50% tolerance). Again, these tolerance ranges are sought as they are comparable to the +/- 50% that is recognized as a limitation of the benchmark B&D method [48].

Table 7: Comparison of predicted assembly times for each training case

Product Test Case	**Level of Definition (Test)**	**Target Time (s)**	**Case 1 (Fully Defined Training) (s) (+/- % Error)**	**Case 2 (Partially Defined Training) (s) (+/- % Error)**	**Case 3 (Fully and Partially Defined Training) (s) (+/- % Error)**

Pencil Compass	Fully	68.3	121.4 (+77.5)	NA	94.5 (+38.2)
	Partially		NA	96.6 (+41.2)	82.5 (+20.6)
MagLight	Fully	75.4	118.3 (+56.9)	NA	70.2 (-6.9)
	Partially		NA	65.1 (-13.7)	75.7 (+0.5)
Black & Decker Drill	Fully	189.6	226.3 (+19.3)	NA	319.3 (+68.4)
	Partially		NA	186.1 (-1.9)	202.3 (+6.7)

For training Case 1, both test cases and the training set were fully defined models. For training Case 2, again, both test cases and training set were all partially defined models. As Training Case 3 used a combination of fully defined and partially defined models for training, both fully defined and partially defined models were used for testing.

Test results indicate that using training Case 3 which had fully and partially defined models resulted in predicted assembly times closest to the target times. The percent error of the predicted assembly times for four of the six inputs decreased by using the training Case 3 as opposed to the first two cases. However, the size of the training set was doubled with Case 3. Therefore, it is not clear whether a combined training set or simply a larger training set is preferred. Training cases using partially defined models are more effective than those using fully defined models. Based on these results, *future training cases could use only partially defined models*.

To investigate the effect of training input variability, three different training cases were assembled (Case 4, 5, 6) by increasing the number of analysed products. Based on the limited success of downloading product assembly models from online databases, the number of models was increased by reverse-engineering five additional consumer products, the list of which is in Table 8. Only certain combinations of the first ten assembly models shown were used to train Case 1, 2, and 3. The last five

products were added to the training set to replace the repeated training inputs (physical and virtual times) used in the first three test cases. The last three columns of Table 8 show Case 4, 5, and 6 where the products used to train each case are labelled "Training" and the products used as test inputs are labelled "Test". All of these are for partially defined modelled, similar to what would be expected to be modelled by an engineer.

Table 8: Increased product collection and training case products for training/testing

#	Product	Assembly Model Generation	Case 4	Case 5	Case 6
1	G2 Pen	Reverse Engineered	Training	Training	Training
2	Pencil Compass	Reverse Engineered	Training	Training	**Test**
3	Solar Yard Light	Reverse Engineered	Training	**Test**	Training
4	Pony Vise	Reverse Engineered	Training	Training	Training
5	Black and Decker Drill	Reverse Engineered	Training	**Test**	**Test**
6	Paper Pro Stapler	GICL	**Test**	Training	Training
7	6" MagLight	SW 3D	**Test**	Training	**Test**
8	Indoor Electric Grill	SW 3D	Training	Training	Training
9	Shift Frame LH	OEM	Training	Training	Training
10	Wide Flag	OEM	Training	Training	Training
11	One Touch Chopper	Reverse Engineered	Training	**Test**	Training
12	Computer Mouse	Reverse Engineered	Training	Training	Training
13	Boothroyd Piston Assembly	Reverse Engineered	Training	Training	Training
14	3 Hole Punch	Reverse Engineered	Training	Training	Training
15	Durabrand Hand Mixer	Reverse Engineered	**Test**	Training	Training

Since all of previous products were the subject of virtual Boothroyd Dewhurst DFA analyses, the new ANN trainings, Case 4, 5, and 6, only use virtual Boothroyd predicted assembly times as their targets which are trained with unique complexity vector inputs for each product. The results of these ANN training cases are in Table 9. Each test yielding estimations within the +/- 25% tolerance range are shaded.

Table 9: Comparison of predicted assembly times for the last three ANN training sets

Product Test Case	Level of Definition (Test)	Target Time (s)	Case 4 (s) (+/-% Error)	Case 5 (s) (+/-% Error)	Case 6 (s) (+/-% Error)
Pencil Compass	Partially	68.3	NA	NA	60.2 (-12.0)
MagLight	Partially	75.4	69.8 (-7.5)	NA	65.4 (-13.3)
Black & Decker Drill	Partially	189.6	NA	199.4 (+5.1)	233.8 (+23.3)
Paper Pro Stapler	Partially	123.5	118.3 (-4.2)	NA	NA
Durabrand Blender	Partially	263.2	271.8 (+3.3)	NA	NA
Solar Yard Light	Partially	128.8	NA	113.1 (-12.2)	NA
One Touch Chopper	Partially	316.6	NA	318.7 (+0.7)	NA

As shown in Table 9 the results for training Case 4, Case 5, and Case 6 have less than 14% error of the target time except in one time generated by Case 6, which exhibited an error of 24%. In that none of the first three training Cases investigated has percent errors this low for all test products, providing a more diverse training set that does not reuse test inputs will increase the overall accuracy of the set. Case 4 generally has the lowest overall percent error out of all training cases. The percent errors for Case 4 range from -7.5% to +3.3% and is closely followed by Case 5 with has percent errors ranging from -12.2% to +5.1%. This additional testing suggests that variety of training has a positive impact on accuracy. Additional training experiments can be found in [28].

2.3.3 Using the ANN Models

Once the ANN models are trained, new assemblies can be analysed and their respective times estimated. This analysis/estimation is done by supplying to the ANN program within MatLab the complexity vectors calculated for the assembly models in

a “use” mode rather than “training” mode. The MatLab interface provides an assembly time display.

To predict an assembly time using the developed assembly time prediction tool, nine steps must be completed (user actions-green and program executions-red):

- **User: Opens SolidWorks assembly model**
- **User: Click on SWMate2 Add-in**
- *Program: Extracts mates and builds the bi-partite table*
- *Program: Opens Matlab and calls custom complexity algorithm passing the generated file name as the input*
- *Program: Complexity algorithm reads mates from the bi-partite table and calculates a respective complexity vector*
- *Program: Calls custom Matlab ANN function (accepts generated complexity vector as input)*
- *Program: Loads previously determined ANN training case that uses top five selected architectures*
- *Program: Mate connection complexity vector is given to custom ANN assembly time prediction function as test input and the function outputs replicated results*
- *Program: Results are interpreted and a predicted assembly time is displayed*

3 Validating the Tool

Two different validations are used to test the tool. First, an external assembly model never before used in any previous training or testing is used to ensure the objectivity of the test. The second validation test entails exploring how assembly models of different users influence predicted times.

3.1 External Testing

To test the developed assembly time prediction tool, a product not previously used for training or the interpretation of results is identified and used for testing. A Durabrand Electric Knife was selected because of similarity in size, part count, and product family to the products and assembly models used for training. Though the SolidWorks assembly model generated for the Electric Knife forms a rough representation of the actual product, it is not exact. Moreover, the assembly model was constrained by a practicing engineer partially, in a manner consistent to typical industry practice. Once the Electric Knife assembly model was generated, a virtual B&D analysis was conducted (taking approximately 2,000 seconds to complete compared with roughly 60 seconds for the automated tool analysis time) and which predicted an assembly time of 212.34 seconds. The new assembly time prediction tool is evaluated by opening the assembly model for the Electric Knife and clicking on the assembly time prediction SolidWorks Add-in.

The Electric Knife assembly model was tested using the top five selected architectures for each case. This testing was repeated for all six training cases, the predicted assembly times of which are tabulated in Table 10. The cells in the table are shaded to illustrate the level of accuracy for the different tests; green shading indicates that the values returned are within the +/- 25% tolerance range and the yellow shading indicates that the values are within the +/- 50% tolerance range.

Table 10: Predicted assembly times for an electric knife using a fully automated assembly time predication tool

Training Set Name	Electric Knife Target Time (s)	Predicted Time from Loaded Training Set (s)	% Error (+/-)	Analysis Time (s)
Case 1	212.34	457.83	+54	68
Case 2		665.87	+68	67
Case 3		315.23	+33	67
Case 4		251.7	+16	67
Case 5		204.59	-4	68
Case 6		225.34	+6	68

The percent error in the predicted time for the training sets ranges from -4% to +68% errors (**Error! Reference source not found.**). If the cases are discretized into general categories, the same conclusions inferred in the previous training case investigation are again made. Though Training Case 1 and Case 2 had a training size of eleven inputs and targets, training inputs were reused, resulting with the highest percent errors ranging from 47% to 68% error. Training Case 3 had twice the training size, twenty-two, but reused training inputs, in turn resulting in a percent error of 33%. Training Case 4, Case 5, and Case 6 had training sizes of twelve inputs and targets, all of which are unique. This resulted in the lowest percent error ranging from -4% to +16% errors, well within the +/- 50% errors that are possible with the B&D method [45].

Running the analysis on this test product while loading trained neural networks took less than 111 seconds once MatLab was opened. The total time to run the analysis, including opening and initializing MatLab which takes approximately another 120 seconds, yielded a total approximate analysis time of 330 seconds. This is a significant improvement when compared to the nearly 2,000 seconds for analysis time for the B&D tool. Fully integrating a trained ANN in C++ within the add-in, therefore, can improve the execution time.

3.2 Mate Sensitivity Testing

If this tool is to be effective, it should be generally insensitive to modelling preferences of different designers. To test such preferences, a set of products are provided to different designers to create assembly models. The assembly models and their associated connectivity graphs and complexity vectors are used to estimate the assembly times for comparison against B&D predicted assembly times. Three separate products were chosen for this study: the Solar Yard Light, the Black & Decker Drill, and the One Touch Chopper. These three products and their respective part count, B&D predicted assembly times, and their product structures are listed in Table 11.

Table 11: Selected products for mate sensitivity study

Product	Part Count	B&D Predicted Assembly Time (s)	Product Structure
Solar Yard Light	15	128.79	Linear
Black & Decker Drill	26	186.65	Clam Shell
One Touch Chopper	43	316.67	Combo: Clam Shell & Stackable

Table 11 represent the totality of products (i.e. assembly time, part count, and general product structure) used in the different training sets. All products differ for all three products listed. Linear product structures are composed of products where the majority of components are inserted along the same axis. Clam shell product structures sandwich the majority of parts between two halves. Stackable product structures have some type of base or foundation where other parts are stacked atop one another to create the assembly. Products also have structures that are based on any combination of these.

The assembly models for each product were prepared by creating an assembly file with all individual components for that product without any mates and by creating a separate reference assembly file that illustrates how the product is assembled,

through which students view the assembly process. To prevent the designers from being influenced by the reference assembly, parts were fixed and all mates were deleted. An exploded view of the reference assembly model, the Black & Decker drill in Figure 8 was created to help determine the assembly sequence.

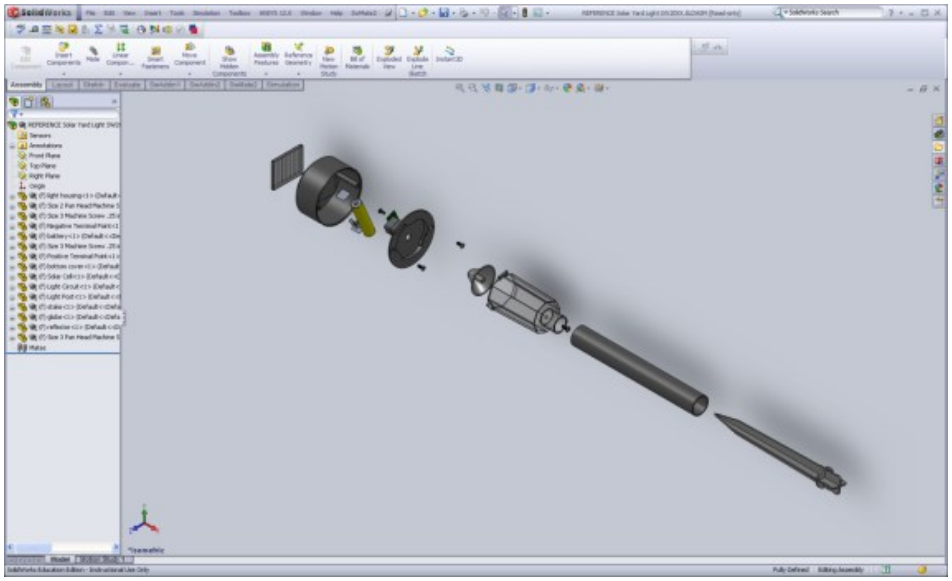

Figure 8: Exploded view of Solar Yard Light Reference Assembly

The exploded view of the reference assemblies is collapsible so that the exact location of parts within the assembly is visible. The product assembly file provided to the students included all of the product parts in the general location with respect to the parts to which they will be mated. The students must position the parts in the correct location and then add mates to the assembly as they see fit. Figure 9 shows the Solar Yard Light assembly model provided so students may add mates as needed. Note that the parts are out of position, requiring including mating constraints to create the proper model.

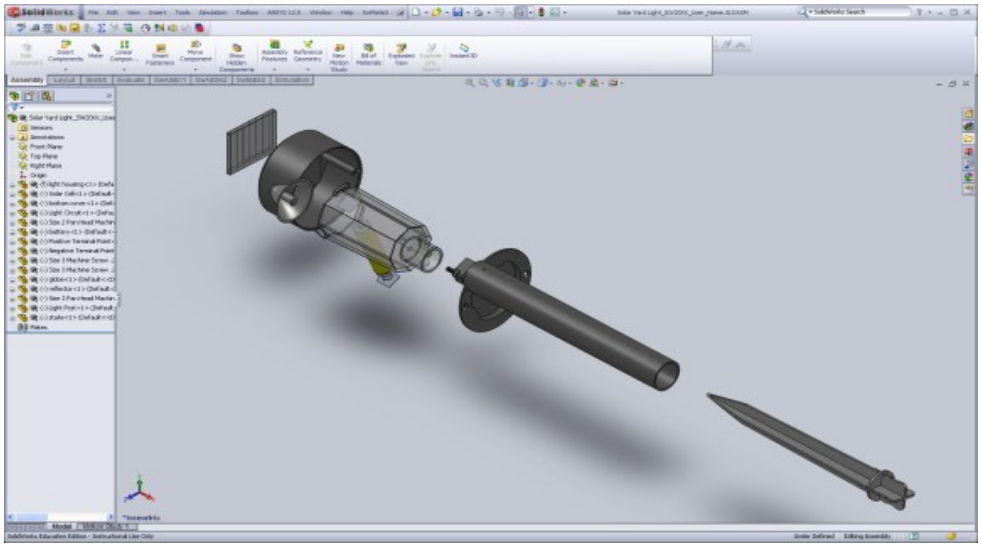

Figure 9: Solar Yard Light assembly model provided to students with no mates

The assembly models and reference assembly models for all three products were distributed to mechanical engineering seniors and graduates enrolled in a Design for Manufacturing course. The students added mates to the unmated collection of parts as appropriate, and the final mated assemblies were used to analyse assembly estimation time with the developed tool.

Demographic information (level, experience with SW, frequency of use of SW) is collected from each student (Table 12), and they were asked to self-report on the time necessary for generating the assembly models from the part collections. The demographics suggest that the students are drawn from a generally novice population and that the students did put forth some effort in creating the assemblies. If an either an expert modeller was found or a student spent less than 15 minutes on one of the activities, then that sample would have been withdrawn.

Table 12: Form results from mate sensitivity study of the assembly time prediction tool

Student	Under Grad. / Graduate	SW Assembly Experience	SW Assembly Usage Frequency	Mate Time Light (min)	Mate Time Drill (min)	Mate Time Chopper (min)
S1	UG	Low	Low	30 < t < 45	45 < t < 60	NA
S2	UG	Low	Low	60 < t < 90	NA	NA
S3	UG	Low	Med.	15 < t < 30	NA	NA
S4	Grad	Low	Med.	15 < t < 30	45 < t < 60	NA
S5	Grad	Med.	Med.	30 < t < 45	t < 15	60 < t < 90
S6	Grad	Med.	High	NA	30 < t < 45	30 < t < 45
S7	UG	Med.	Med.	15 < t < 30	45 < t < 60	30 < t < 45
S8	Grad	Low	Med.	45 < t < 60	t > 90	45 < t < 60
S9	Grad	Med.	Med.	30 < t < 45	45 < t < 60	45 < t < 60
S10	Grad	Low	High	45 < t < 60	t < 15	NA
S11	UG	Med.	Low	15 < t < 30	NA	NA

Once all of the mated assemblies were compiled, the automated assembly time prediction tool was used to predict a respective assembly time using the average of the top five architectures for the best performing training set (Case 4). The number of mates the students added, the target time, the predicted assembly times for each student's assembly, the percentage error in the predicted time, and the MatLab analysis times for the Solar Yard Light are shown in Table 13. Table cells are shaded to illustrate the level of accuracy for various tests (green - returned values are within the +/- 25% tolerance range, yellow– returned values are within the +/- 50% tolerance range).

Table 13: Mate sensitivity analysis for Solar Yard Light

Student	Solar Yard Light Target Time	# of Mates	Predicted Time from Loaded Training Set	% Error (+/-)	Analysis Time (s)
Student 1	128.79	33	129.56	+1	67
Student 2		32	110.99	-16	71
Student 3		25	88.71	-45	68
Student 4		36	121.08	-6	69
Student 5		38	115.95	-11	70
Student 7		36	145.95	+12	64
Student 8		35	131.32	+2	65
Student 9		41	107.08	-20	63
Student 10		36	125.39	-3	64
Student 11		36	111.3	-16	64

Of the ten assembly configurations analysed (one student did not complete the analysis), the percentage error in the predicted assembly time ranged from -45% to +12% error with the average of the absolute values being 13% error. The number of mates each student added does not appear to directly relate to the predicted assembly time and the percentage error. Though student one used thirty three mates and student two used thirty two mates, the predicted assembly times had +1% and -16% errors respectively. Likewise, though students four, seven, ten, and eleven all used thirty six mates, the percentage errors were -6%, +12%, -3%, and -16% respectively. Student three used the least number of mates, twenty five, and had the largest percentage error, -45%. Since the number of mates does not appear to directly relate to the predicted assembly time, the significantly higher percentage error for Student 3 could possibly be caused by different assembly definition, emphasis on one type of mate usage, or usage of reference geometry to mate parts. To fully understand the cause of this localized increase these errors error, a detailed study investigating the types of mates used and the respective complexity vectors created must conducted, and which will be pursued in future research.

All student mated assemblies were within +/- 50% of the target time and nine of the ten were within +/- 25% of the target. Excluding the predicted time from the model from Student 3's, the percentage error range changes from -20% to a +12% error. The analysis time to predict these assembly times was less than seventy-two seconds for each model per model, which does not include the time for MatLab to open and initialize (approximately 120 seconds). The original target assembly time for the Solar Yard Light was predicted using a Virtual B&D analysis, taking 3,300 seconds (55 minutes) to complete the analysis manually.

Table 14 shows the results for the Black & Decker drill assembly and Table 15 the results for the One Touch Chopper. In both, the error is less than 25%, well within the +/- 50% variance estimated with B&D [45].

Table 14: Mate sensitivity analysis for Black & Decker Drill

Student	Black & Decker Drill Target Time (s)	# of Mates	Predicted Time from Loaded Training Set	% Error (+/-)	Analysis Time (s)
Student 1	189.65	52	205.73	+8	68
Student 4		46	188.4	-1	67
Student 5		59	220.69	+14	68
Student 6		53	240.25	+21	64
Student 7		59	232.04	+18	65
Student 8		62	190.21	+0.3	64
Student 9		50	224.9	+16	63
Student 10		48	213.6	+11	65

Table 15: Mate sensitivity analysis for One Touch Chopper

Student	One Touch Chopper Target Time (s)	# of Mates	Predicted Time from Loaded Training Set	% Error (+/-)	Analysis Time (s)
Student 2	316.62	89	336.91	+6	65
Student 6		90	357.1	+11	67
Student 7		91	322.17	+2	68
Student 8		104	325.07	+3	65
Student 9		86	352.57	+10	64

Table 16 lists a summary of the products each student mated and the errors of predicted assembly times.

Table 16: Summary of % errors for each student for each product

Student	Solar Yard Light % Error (+/-)	Black & Decker Drill % Error (+/-)	One Touch Chopper % Error (+/-)
Student 1	+1	+8	NA
Student 2	-16	NA	+6
Student 3	-45	NA	NA
Student 4	-6	-1	NA
Student 5	-11	+14	NA
Student 6	NA	+21	+11
Student 7	+12	+18	+2
Student 8	+2	+0.3	+3
Student 9	-20	+16	+10
Student 10	-3	+11	NA
Student 11	-16	NA	NA

All of the percentage errors shown in **Error! Reference source not found.** are within +/- 45% error of the target assembly times for the given product, placing them within the +/-50% tolerance range. If the predicted assembly time is removed for Student 3's Solar Yard Light, the range of errors drops to +/- 21%. It should also be noted that the highest errors for the Black & Decker Drill and the One Touch Chopper were from both from Student 6 who had a medium level of SW assembly experience and a high SW assembly usage frequency. No significant variance of percentage errors of across the three products **Error! Reference source not found.** suggests that the automated tool performs well for the variety of test products used in this study (**Error! Reference source not found.**). Though admittedly not statistically significant, this preliminary study does illustrate the potential insensitivity of the tool to the designer-choice-for-mating-approaches.

4 Concluding Remarks and Recommended Future Studies

A method and implemented tool, demonstrably effective for estimating assembly times, is based entirely on objective information explicitly found within the assembly models of a commercial CAD system. Experimentation was used to develop recommendations for developing the training sets. Moreover, the tool is validated against a withheld training case of an electric knife. Finally, the tool is demonstrated

to be robust against user variability through a study with models generated by several student engineers.

Even though the automated assembly time prediction tool addresses the goals of eliminating subjective information dependency, reducing user input requirements, and allowing earlier use of the tool in the design process prior to physical reverse engineering, it still has limitations that must be addressed in future research. The limitations here encompass three discrete categories related to the ANN training cases used, the mating scheme sensitivity, and the robustness of the mate extraction add-in. Each of these limitations is addressed in the following sub sections.

4.1 Limitation with Regards to ANN Training Cases

The case used to train the ANNs affects the results of the predicted assembly times. For example, the predicted times for the Electric Knife test case ranged from -4% to +68% depending on the training case used (**Error! Reference source not found.**). It was recommended that *future training cases should use a set of at least eleven unique training inputs and targets composed of partially defined assembly models* to improve the accuracy of the predicted assembly times. These investigations into ANN training case types used for such recommendations are only preliminary, however. For more effective or specific recommendations, larger sample sizes must be used, which is the subject of future research. Such studies should also investigate if the test inputs are either internal or external to the training sets used. Internal test inputs would be products that have part counts, component counts, and complexities within the range of the training case and external inputs would have values outside of the range of the training case.

During tool development, several different training cases were evaluated to determine their effect on the predicted assembly times and to select five ANN

architectures to use with the automated tool. Though the selection process for choosing the five ANN architectures is repeatable, it may not select the overall best architectures. A formalized architecture selection process that chooses the five most effective architecture structures should be the subject of future research.

4.2 Limitation with Regards to Mating Sensitivity

The results of the designer modelling preference study showed that for a given product the % errors are within +/- 25% error for all cases except for one outlier with a -45% error. The mate sensitivity study only evaluated the variability between different test subjects' assembly times, and the specific effect of the different mating styles on the predicted assembly times was not explored. Further investigation into this mating variability and its effect on the predicted assembly time using this tool will be undertaken in future research.

4.3 Limitation with Regards to Program Robustness

The automated assembly time prediction tool is a SolidWorks custom add-in that extracts the defined mates from an assembly model and uses the complexity of the mate connection graphs to predict an assembly time based using trained ANNs. The automated tool has successfully predicted assembly times in less than five minutes. Though effective, the limitations of this tool must be resolved in future research, as summarized thusly:

(1) Does not extract mates from subassemblies;

(2) Does not handle part patterns within assemblies;

(3) Extracts suppressed mates;

(4) Requires MatLab to perform computations.

The first three limitations can be addressed in future versions of this tool with a more robust development of the SW API program. The fourth limitation can also be

addressed through the development of a standalone complexity analysis module and ANN module for integration into the tool. This would more seamlessly integrate the program, requiring fewer external calls and allowing for easier portability of the code. Moreover, it should improve the time spend in running the program as a significant portion is dedicated to opening the MatLab program to access the various toolboxes.

4.4 Extendibility of Current Tool

The current method employs an exclusive use of complexity metrics on connectivity graphs to create the trained ANNs, initially undertaken to reduce the amount of subjectivity and designer interaction required. As shown [17], however, much of the subjectivity of the B&D method is related to the *insertion* activity. The *handling* activity is more objective. Therefore, in the next version of the tool, this additional information about the parts might be integrated into the predictive models.

5 References

[1] O'Grady P., 1991, "A Review of Approaches to Design for Assembly," Concurrent Engineering, Research, and Applications, **1**(3), p. 5.

[2] Boothroyd G., and Alting L., 1992, "Design for Assembly and Disassembly," CIRP Annals-Manufacturing Technology, **41**(August), pp. 625–636.

[3] Wang L., Keshavarzmanesh S., Feng H.-Y., and Buchal R. O., 2008, "Assembly process planning and its future in collaborative manufacturing: a review," The International Journal of Advanced Manufacturing Technology, **41**(1-2), pp. 132–144.

[4] Chiu M.-C., and Kremer G. E. O., 2011, "Investigation of the applicability of Design for X tools during design concept evolution: a literature review," Internatinoal Journal of Product Development, **13**(2), pp. 132–167.

[5] Maynard H., Stegemerten G. J., and Schwab J., 1948, Methods-Time Measurement, McGraw-Hill, New York, NY.

[6] Laring J., 2002, "MTM-based ergonomic workload analysis," International Journal of Industrial Ergonomics, **30**(3), pp. 135–148.

[7] Boothroyd G., and Walker J., 1996, “Design for Assembly,” Handbook of Manufacturing Engineering, J. Walker, ed., Marcel Dekker, New York, NY, pp. 1–50.

[8] Ohashi T., 2002, “Extended Assemblability Evaluation Method (AEM),” JSME International Journal: Series C - Mechanical Systems, Machine Elements, and Manufacturing, **45**(2), p. 567.

[9] Mohd Naim Z., 2009, “Design for assembly and application using Hitachi assemblability evaluation method,” Universiti Malaysia Pahang.

[10] Miyakawa S., and Ohashi T., 1986, “The Hitachi Assemblability Evaluation Method (AEM),” Conference on Product Design for Assembly, New Port, RI, pp. 15–17.

[11] Kroll E., Lenz E., and Wolberg J. R., 1988, “A Knowledge-Based Solution to the Design for Assembly Problem,” Manufacturing Review, **1**(2), pp. 104–108.

[12] Swift K. G., 1989, “Expert system aids design for assembly,” Assembly Automation, **9**(3), pp. 132–136.

[13] Tate S., Jared G., Brown N., and Swift K. G., 2000, “An Introduction to the Designers’ Sandpit,” International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, ASME, Baltimore, MD, pp. 84–87.

[14] Dalgleish G. F., Jared G. E., and Swift K. G., 2000, “Design for Assembly Influencing the Design Process,” Journal of Engineering Design, **11**, pp. 17–29.

[15] Boothroyd G., 1994, “Product Design for Manufacture and Assembly,” Computer-Aided Design, **26**(7), pp. 505–520.

[16] Esterman M., and Kamath K., 2010, “Design for Assembly Line Performance: The Link Between DFA Metrics and Assembly Line Performance Metrics,” Volume 6: 15th Design for Manufacturing and the Lifecycle Conference; 7th Symposium on International Design and Design Education, ASME, pp. 73–84.

[17] Namouz E., Summers J. D., and Mocko G. M., 2012, “Reasoning: Source of Variability in the Boothroyd and Dewhurst Assembly Time Estimation Method,” International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Chicago, IL, pp. DETC2012–71075.

[18] Owensby J. E., Shanthakumar A., Namouz E., Rayate V., and Summers J. D., 2011, “Evaluation and Comparison of Two Design for Assembly Methods: Subjectivity of Information,” ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pp. DETC2011–47530.

[19] Barnes C. J., Dalgleish G. F., Jared G. E. M., Mei H., and Swift K. G., "Assembly oriented design," Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning (ISATP'99) (Cat. No.99TH8470), IEEE, pp. 45–50.

[20] Kim G. J., Bekey G. A., and Goldberg K. Y., "A shape metric for design-for-assembly," Proceedings 1992 IEEE International Conference on Robotics and Automation, IEEE Comput. Soc. Press, pp. 968–973.

[21] Mathieson J., Wallace B., and Summers J. D., "Estimating Assembly Time with Connective Complexity Metric Based Surrogate Models," International Journal of Computer Integrated Manufacturing, **on-line**(in press).

[22] Miller M., Mathieson J., Summers J. D., and Mocko G. M., 2012, "Representation: Structural Complexity of Assemblies to Create Neural Network Based Assembly Time Estimation Models," International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Chicago, IL, pp. DETC2012–71337.

[23] Ault H. K., "3-D Geometric Modeling for the 21st Century," Engineering Design Graphics Journal, **63**(2).

[24] Veisz D., Namouz E., Joshi S., and Summers J. D., "The Impact of the Disappearance of Sketching: A Case Study," Artificial Intelligence in Engineering Design Analysis and Manufacturing.

[25] Shanthakumar A., 2012, "Development of a Feature Recognition Algorithm for Automated Identification of Duplicate Geometries in CAD Models," Clemson University.

[26] Namouz E. Z., Mears L., and Summers J., 2011, "Lazy Parts Indication Method: Application to Automotive Components," SAE 2011 World Congress & Exhibition, pp. 2001–2011.

[27] Griese D., Namouz E., Shankar P., Summers J. D., and Mocko G. M., 2011, "Application of a Lightweight Engineering Tool: Lazy Parts Analysis and Redesign of a Remote Controlled Car," ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pp. DETC2011–47544.

[28] Owensby J. E., 2012, "Automated Assembly Time Prediction Tool Using Predefined Mates from CAD Assemblies," Clemson University.

[29] Mathieson J. L., 2011, "Connective Complexity Methods for Analysis and Prediction in Engineering Design," Clemson University.

[30] Bashir H. A., and Thomson V., 2001, "An Analogy-Based Model for Estimating Design Effort," Design Studies, **22**, pp. 157–167.

[31] Shah J. J., and Runger G., 2011, "Misuse of Information-Theoretic Dispersion Measures as Design Complexity Metrics," ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, ASME, Washington, DC, p. DETC2011/DTM–48295.

[32] Singh G., Balaji S., Shah J. J., Corman D., Howard R., Mattikalli R., and Stuart D., 2012, "Evaluation of Network Measures as Complexity Metrics," ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, ASME, Chicago, IL, pp. DETC2012–70483.

[33] Mathieson J. L., and Summers J. D., 2010, "Complexity Metrics for Directional Node-Link System Representations: Theory and Applications," Proceedings of the ASME IDETC/CIE 2010.

[34] Barclay I., and Dann Z., 2000, "New-Product-Development Performance Evaluation: A Product-Complexity-Based Methodology," IEE Proceedings in Scientific Measurement Technology, **147**(3), pp. 41–55.

[35] Mathieson J., and Summers J., 2009, "Relational DSMs in Connectivity Complexity Measurement," Proceedings of 11th International DSM Conference, pp. 15–26.

[36] Pramanick I., and Ali H., 1994, "Analysis and Experiments for a Parallel Solution to the All Pairs Shortest Path Problem," IEEE International Symposium on Circuits and Systems, IEEE, New York, NY.

[37] Goldberg A. V., and Tarjan R. E., 1986, "A New Approach to the Maximum Flow Problem," Annual ACM Symposium on Theory of Computing, ACM, New York, NY, pp. 136–146.

[38] Freeman L., 1977, "A Set of Measures of Centrality Based on Betweenness," Social Networks, **40**, pp. 35–41.

[39] Freeman L., 1979, "Centrality in Social Networks: Conectual Clarification," Social Networks, **1**, pp. 215–239.

[40] Koschutzki D., Lehmann K. A., Peeters L., Richer S., Tenfelde-Podehl D., and Zlotowski, 2005, "Centrality Indices," Network Analysis: Methodological Foundations, U. Brandes, and T. Erlebach, eds., Springer Verlag, New York, NY.

[41] Sabidussi G., 1966, "The Centrality Index of a Graph," Psychometrika, **31**, pp. 581–603.

[42] Watts D. J., and Strogatz S., 1998, "Collective Dynamics of 'Small-World' Networks," Nature, **393**(6), pp. 440–442.

[43] Summers J. D., and Ameri F., 2008, "An algorithm for assessing design complexity through a connectivity view," Proceedings of the TMCE 2008.

[44] Bader G. D., and Hogue C. W. V., 2003, "An Automated Method for Finding Molecular Complexes in Large Interaction Network," BMC Bioinformatics, **4**.

[45] Boothroyd G., Dewhurst P., and Knight W., 2002, Product Design for Manufacture and Assembly, M. Dekker, New York, NY.

[46] Chavali S. R. K., Sen C., Mocko G. M., and Summers J. D., 2008, "Using rule-based design in engineer-to-order industry: An SME case study," Computer-Aided Design and Applications, **5**, pp. 178–193.

[47] Tu J. V., 1996, "Advantages and Disadvantages of Using Artificial Neural Networks versus Logistic Regression for Predicting Medical Outcomes," Journal of Clinical Epidemiol, **49**(11), pp. 1225–1231.

[48] Boothroyd G., Dewhurst P., and Knight W. A., 2011, Product Design for Manufacture and Assembly, CRC Press, Boca Raton.