A DESIGN METHOD FOR PRODUCT FAMILY CONFIGURATION WITH SPATIAL LAYOUT REQUIREMENTS

A Thesis
Presented to
The Academic Faculty

By

John-Travis Smith Hansen

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Mechanical Engineering

Georgia Institute of Technology

December 2017

A DESIGN METHOD FOR PRODUCT FAMILY CONFIGURATION WITH SPATIAL LAYOUT REQUIREMENTS

Approved by:

Dr. David Rosen, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Roger Jiao
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Katherine Fu
School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: December 4, 2017

ACKNOWLEDGEMENTS

I would like to acknowledge my advisor, Dr. David Rosen, for his patience, guidance , and understanding throughout this endeavor. He guided me with insightful comments and advice, throughout my research. I would also like to thank my committee members, Dr. Roger Jiao and Dr. Katherine Fu for their time, expertise, and feedback in the completion of my Masters thesis.

I would would also like to thank the other members of my lab for making the office a great learning and research environment. Specifically, I would like to thank Chad Hume, Narumi Watanabe, Sang-in Park, Paula Xian, Peter Zhao, and Mahmoud Dinar. Without their support and insight it would have been very difficult to finish my work.

I would finally like to thank my friends and family for all of the support they have given me. I am extremely grateful to both my mother and father for the love, support, and encouragement they have given me throughout my work. I thank my brother for his support and encouragement. I would like to thank my roommates Kyle and Clayton, who helped me navigate grad school. Finally I would like to thank my friends who have supported me, including Troy, Kelsey, Cecily, and many more.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

NOMENCLATURE

$\alpha_{j,k}$	Elements of Cf_j, a subset of $Confs$
δ^+	Positive deviation variable for modified simplex method
δ^-	Negative deviation variable for modified simplex method
A	Coefficient of variables for the simplex method
b	Constants for the simplex method
C^+	Set of Required Pair Constraints
C^-	Set of Dissallowed Pair Constraints
$C^=$	Set of Required Components Constraint
c_i^{in}	Element of $Comps^{in}$
c_n	The number of defined layout constraints
$C_{i,j}$	Matrix index for the presents of absence of a component in a product
Cf_j	Element of $Confs$ containing a subset of configuration options, $\alpha_{j,k}$
$Combinations^{all}$	All Combinations of components and configuration options still containing $\{\oslash\}$ elements
$Comps^{in}$	A set of uniquely named components for a product family
$Confs$	Set containing configuration option subsets as elements
$ConfSpc$	The Configuration space containing all feasible products from the combinatorial generation process
$ConstrainedSets$	Subset elements of $SetCombinations$ after constraints have been applied
$E_{Products}$	Full enumeration of available products after constraints are applied
FH_k	Y dimension of fascia k
$FilteredConstraints$	Sets of components and/or configuration options connected via constraints
FW_k	X dimension of fascia k

FX_k	X location of fascia k
FY_k	Y location of fascia k
$GeneratedSet$	Set of subsets containing combinations generated from $ConstrainedSets$ and the unconstrained components remaining in $Parts^{all}$
H_i	Y dimension of component i
Mc	MultiConstraint List for use in the Algorithms presented in Chapter 4
Mct	Set of constrained combinations of the components from Mc
Mx	MixedConstraint list for use in the Algorithms presented in Chapter 4
Mxt	Set of constrained combinations of the components and configuration options from Mc
N_u	Number of Unconstrained Components
N_{cn}	Number of constrained combinations of components and configuration options
N_{Comps}	Total number of combinations of only components
$N_{ConfigOpts}$	Total number of combinations of only configuration options
N_{mc}	Number of combinations from the contents of Mct
N_{Total}	Total number of combinations of components and configuraiton options
$ObjectGroups$	Set containing subsets of Components (with $\{\oslash\}$) and/or full sets of configuration options used for the combintorial generation process.
$Parts^{all}$	Set created by combining $Comps^{in}$ and $Confs$
$SetCombinations$	Sets containing all combinations of the elements of the subsets of $ObjectGroups$
$TempList$	Set Containing Required Pair and Disallowed Pair Constraints. Only used as an intermediate variable
Uc	List of unconstrained components for use in the algorithms in Chapter 4
Uct	Set of combinations of unconstrained components from Uc

v The number of component variables for the simplex layout procedure

W_i X dimension of component i

X_i X location of component i

X_{vars} Matrix of X location variables for components in the simplex layout procedure

Y_i Y location of component i

Y_{vars} Matrix of Y location variables for components in the simplex layout procedure

SUMMARY

The use of product platforms facilitates product variety and reductions in cost and lead times, enabling companies to compete in the global marketplace. A common platform allows companies to generate variety by adding, removing, or substituting modules within products to target a specific market. The product platform acts as the foundation for the technology, function, and physical arrangement, common to the entire product family. Configuration design methods are used to identify product platforms based on a set of components and their relationships in a product. From a configuration perspective, the physical locations and spatial requirements of components within a product can influence the physical layout of other members of the product family. This can lead to difficulty in obtaining the desired variety and spatial layout of components.

Methods for solving configuration design problems utilizing discrete mathematics and graph theory to model configuration problems from multiple viewpoints have been developed. The current viewpoints consider neither desires of the designer in regards to variety, nor the physical requirements of the product family in later design stages. The generation of product configurations is a combinatorial and discrete problem, while physical layout is combinatorial and algebraic. Resulting in a complex problem utilizing multiple forms of mathematics. The work of this thesis seeks to address this problem through the consolidation of configuration design and object layout methods to determine the effects of configuration design decisions on component layout.

The development of a design method to generate a product family with configuration constraints and the generation of common component layouts based on spatial constraints is presented in this thesis. This method facilitates product platform selection by determining if the designers' desired configurations and layouts are feasible. To demonstrate the use of the method presented in this thesis, a GUI-based software application was developed. This software implements the work of this thesis into a user-friendly program. The proposed

method and the software are demonstrated through a series of examples.

CHAPTER 1

INTRODUCTION

The overall objective of this thesis is to present a configuration design method for generating product configurations of a modular product family with designer defined, configuration and spatial constraints. Within the context of this thesis, configuration design refers to the issues of 1) which components are or are not in a Product and 2) their arrangement and relationships. The focus of this work is to identify relationships between configuration constraints and spatial layout as it pertains to the development of a product platform and increasing commonality throughout the product family. The main research contributions of this work are 1) the development of a configuration design space for the modeling of designer preferences within a feature-based product family, 2) a constraint grammar and optimization method for the modeling of spatial constraints in product families and 3) a method for the configuration and spatial-layout of feature-based product families. A code implementation and a GUI - based software application have been created to demonstrate the methods presented in this thesis. Example problems are used to illustrate the application of this research to the development of a product family. The example problems will develop product configurations and layouts for coffeemaker, car center console, and car instrument panel product families.

This chapter introduces the problem being addressed in this thesis and the research objectives. Section 1.1 discusses configuration design/ component layout problem addressed in this thesis. Section 1.2 presents the research questions addressed in this thesis. Finally Section 1.3 presents the organization of this thesis.

1.1 Configuration Design Problems

Product family design is a process of developing a set of products based around a common product platform. The product platform acts as the foundation for the technology, function, and physical arrangement common to the entire product family. Unique products are created by adding, removing, or substituting components or features within the common product platform. Product development is aided by systematic design methods during conceptual and embodiment design [1]. In conceptual design, function structures are used to connect physical components to the functions they perform. Embodiment design encourages the use of ergonomics and layout as design factors to meet function and usability requirements. This thesis explores the relationship between function structures and the physical arrangement of components to help develop product platforms.

The development of product families is a large problem; this thesis focuses on the development of configuration design methods that address the issues of which components are in a product and/or product family, and the physical arrangement and location of componets within products. Scaling issues in product family design are not addressed, although scaling is often used. Our work is focused on investigating problems related to commonality, modularity, and standardization. The questions addressed in this thesis are concerned with the Platform Commonization (PC) problem (the development of a reasonable common platform for the product family) and the Platform Supported Product Variety (PSPV) problem, (the creation of a product family from a common platform) proposed by Siddique [2]. From a configuration perspective, the physical locations and spatial requirements of components within a product can influence the physical layout of other members of the product family.

Configuration design is combinatorial in nature and therefore can be evaluated using discrete mathematics. Using the platform as a base, product family members are generated by selecting different combinations of optional components or modules to achieve the desired product variety. The development of the platform itself requires the identification and evaluation of component combinations and relationships. Representations of products and platforms in configuration design are usually modeled using graphs, sets, and matroids. Previous research addressing combinatorial problems in configuration design focused on the objectives of: (1) representation and reasoning of combinatorial design spaces, (2) modeling constraints and their effects on these spaces,(3) determination of feasible regions within the design space and enumeration of designs. Previous work presented discrete design spaces for modeling, functional intent, physical connections, assembly, and flow of energy and materials viewpoints [2–4]. Due to the combinatorial nature of configuration design, many of the configurations generated may be infeasible. To prevent the generation of infeasible configurations, constraints are applied to products, platforms, and product families such that only feasible configurations are generated.

When configuring product families, the physical arrangement of components can result in a multitude of alternatives due to the complex nature of 2D and 3D design. When product family members are arranged individually, the spatial requirements for optional components may not be taken into account and require unique components during manufacturing.

The purpose of this thesis is to present a method for configuring product families with constraints on combinations of configurations and the generation of common component layouts to aid in platform commonization. The development of this method is beneficial because it enables the evaluation of the entire product family and changes made to the product family early on in the design process. Allowing configurations deemed infeasible by the designer to be discarded. The current viewpoints do not consider product usability and the desired variety of the designer, nor do they consider the design of the product family during embodiment design. This results in numerous configurations, deemed feasible by other viewpoints, that may not meet the physical design requirements decided by the designer. The process presented in this thesis allows designers to develop entire product families and determine whether feasible layouts can be achieved based upon the functional requirements of components along with the aesthetic and ergonomic desires of the designer. The work in this thesis is focused on the 2D layout of components within the user interface of products, but 3D layout of entire products is addressed in future work.

1.2 Research Questions and Hypothesis

The primary goal of this research is the synthesis of a method for the configuration design of product families with combinatorial and spatial layout constraints. To achieve this overall goal two research questions are addressed. The first addresses the desire to express spatial relationships for components prior to embodiment design

Research Question 1

How can spatial relationships within products be simply captured and represented prior to embodiment design?

Hypothesis 1

A spatial grammar can be used to represent spatial relationships with a mathematical representation

The second question addresses the effects of configuration design decisions on physical design and the ability to determine the effects of configuration design decisions.

Research Question 2

How can configuration design methods be augmented to determine the effects of design decisions during conceptual design on embodiment design?

Hypothesis 2

Constraints and optimization methods using both discrete and continuous mathematics *can be used to determine the effects of configuration design decisions on physical design spaces.*

The result of this work is a configuration design method that will allow designers to generate initial component layouts prior to embodiment design and determine configuration feasibility within the conceptual and embodiment design stages.

1.3 Organization of This Thesis

In the next chapter related research in product family design, configuration design, and spatial layout is presented. The Configuration-Layout Design problem is formulated and the mathematical modeling for the constrained generation of configurations and spatial layout of components is presented in Chapter 3. The presentation of Chapter 3 is aided by a running example for development of a coffeemaker product family. Chapter 4 presents the data structures, algorithms, and application built to implement the methods presented in this thesis. Chapter 4 will also include a detailed demonstration of the application for the development of the coffeemaker product family present Chapter 3. To demonstrate the work presented in this thesis, case studies for the development of product families for Center Console and Instrument Panel car sub-systems are presented in Chapter 5. Concluding remarks, answers to research questions, and future work are presented in Chapter 6.

CHAPTER 2
LITERATURE REVIEW

In this chapter a survey of relevant work in the areas of product family design and object layout will be presented. This will serve as the groundwork for the research presented in this thesis. This chapter provides an overview of product family and configuration design concepts and methods. An overview of spatial layout and packing research is also given. This will include forms of representation and existing solution methods.

2.1 Product Family Matters

The implementation of a common product platform has been embraced by many companies to provide product variety while keeping manufacturing costs low. In the global marketplace companies need to produce a variety of products to reach diverse customer needs across different market segments. To reach these market segments, companies could design each product individually to the required specification. However multiple variations of a product are often designed at the same time based upon a set of products or a common structure.

A set of products based on a common core structure is known as a *product family* . The structure shared by members of the product family is the *product platform*. These basic product family terms are common throughout all product family design. In this section current research in the areas of product family design and configuration design will be presented. Topics covered include product variety, commonality, product architectures, and modeling. This will provide the base for the product family research presented in this thesis.

2.1.1 Product Variety: Building A Full House

Product variety is developed by the generation of unique product variants based on the product platform. Ulrich defines product variety as the diversity of products that a production system provides to the marketplace[5]. The range of variety produced is dependent on the number of market segments and customer bases being targeted.

Research performed in the areas of Mass Customization, Design for Product Variety, Designing a Family of Products, etc. has emphasized the development of a common product platform. A common platform allows for the desired variety to be achieved within the entire product family. Martin and Ishii [6] recognized commonality, modularity, and standardization as primary characteristics of product families. When designing a product family, the identification of modules is required to reach commonality and standardization goals. Different research groups have introduced methods for the identification of modules based on the functional descriptions of products [7]. Research has included emphases on, functions and customer demand [8], brands [9], and life-cycle viewpoints [10]. Measurements of commonality and standardization from these viewpoints have been attempted [10–12].

The research presented in product family design and modular architectures does not systematically search for all feasible product platforms and product family architectures. To reach this target, the generation of all feasible platforms and product family configurations based on given a set of constraints is required.

2.1.2 Research in Product Family Reasoning

The platform commonization problem is concerned with determining which components form the core product and which form the product family core and their relationship to the platform. Configuration design concerns itself with what components make up a design and their spatial and logical arrangements. Normally, product architecture design occurs during configuration design [1]. Product architectures can be divided into modules based on component function similarity, organization structure, manufacturing considerations, or the need for upgrade or add-on modules [13].

Research in general configuration design has focused on the development and application of heuristic methods for searching combinatorial design spaces [14]. Work has been performed on developing mathematical operations including language-based approaches using shapes and graphs [15–17]. This research produced mathematically elegant algorithms, but resulted in a computational complexity that makes implementation infeasible. Rosen [13] presents a formal foundation for, configuration design for product life-cycle that emphasizes product modularity and its influence on life-cycle issues. The foundation is derived from discrete mathematics (set theory and combinatorics) to represent and reason with combinatorial design spaces that encompass all combinations of components, modules and their relationships. Research in combinatorial design spaces has extended this work and developed mathematical modeling methods for discrete design spaces to configure product families for component intent, connections, assembly considerations, and flow of information and material viewpoints [2, 4, 18]. The combinatorial configuration work in this thesis takes considerations from this work as a mathematical basis for the methods presented.

2.2 Research in Layout and Packing

In this section object layout and packing research will be presented. Research in packing problems is immense and because of this, a variety of different optimization methods have been used to solve various packing problems. Different methods, from heuristic to stochastic will be presented along with drawbacks to the solution method. Research in object layout covering various fields will also be presented because the spatial layout problem being presented in this thesis does not tend to fall within the confines of regular packing problems. Finally forms of geometric representation will be presented, since objects and shapes can be mathematically represented in different ways.

2.2.1 Overview

The component layout or packaging problem involves the placement of components in a set space such that a set of objectives can be optimized while satisfying spatial or functional constraints. Cagan et al. [19] discusses the basic parts of a layout problem, which include: topological connections, objectives, components, and constraints. All of which are fed into an optimization search algorithm. A number of research groups have proposed heuristic layout methods to evaluate packing problems [20, 21], while others have proposed methods using traditional optimization methods such as, branch & bound [22], tree search [23], and lagrangian relaxation [24] to search the design space for optimal solutions. Aladahalli et al. [25] proposed the use a sensitivity metric to aide in pattern for 3D component layout. Szykman et al. [26], proposed a method for component placement by only using objectives to define the goals of the optimization process and using simulated annealing to solve. Hanna Landry and Cagan [27] proposed the use of Evolutionary Multi-Agent Systems in 3D object packing problems. While these methods often reach optimal or near optimal solutions, they are limited by constraint types, object shape, accuracy, and/or computation time.

Research in more general spatial layout outside of the standard packaging problem has focused on different objectives. Research has proposed the use of grammar-based constraints as a means of aiding object layout [28]. Borning et al. [29], suggests that objects within a computer user interface can be constrained linearly and given lexicographic solving preference. This allows soft constraints to have a satisfaction priority and a valid solution to be found without full constraint satisfaction while deciding which constraint is able to be violated. From this concept Badros et al. [30], proposed a constraint solving algorithm based on the simplex optimization method making use of constraint preference. This research was limited to digital user interfaces and objects as single points rather than multi-dimensional objects. The work presented in this thesis builds upon and adapts this research for use in the configuration of product families.

2.2.2 Existing Methods

A review of a variety of object layout approaches is presented in this section, along with a review of different geometric representations. Cagan discusses the basic parts of a layout problem, which include: topological connections, objectives, components, and constraints, all of which are fed into an optimization search algorithm [19]. These may all be defined differently depending on the method being used.

2.2.2.1 Heuristic Packing Methods

Heuristic methods are rule-based algorithms which use rules to pack objects within containers. Heuristic methods allow for solutions to be quickly found, since these algorithms follow a set path in the solution space. Due to their simplicity, many heuristics have a narrow scope but are easy to follow and will often lead to a good quality solution for problems within their scope [19]. The largest benefit of heuristic methods is due to their deterministic nature which will always yield the same final result for the same set of initial conditions. This consistency is one of the largest advantages of heuristic methods. At the same time this deterministic nature may not lead to an optimal solution, because the accuracy of these methods are often impossible to determine.

Many of the heuristic methods start by placing the object with the largest volume, or height. Such as the "first-fit decreasing height" or "best-fit decreasing height" mentioned by Ortman et al. [21]. Similarly Grbz [20] presented the "Largest Area First-Fit" (LAFF) heuristic which falls into the same category of heuristic methods. Most heuristic methods are limited to the packing of 2D or 3D cuboid objects. Since more variation in geometry adds more complexity to the problem, a more complicated set of steps would be required to able to produce a result and this still may not yield an accurate result.

2.2.2.2 Traditional Optimization Methods

Most traditional optimization methods use a mathematical representation of the objectives of the optimization process, called an “objective function”, and variables to represent the objectives. Variables will often appear in equality or inequality constraints which are required to be satisfied for a solution to be considered feasible. Along with the constraints, the objectives often are given weighting values which determine the priority of the objectives. The optimization processes typically search the solution space for the variable values to minimize the objective function. The methods in this section are different processes for searching the solution space for the optimal solution.

The bound and branch (B & B) method searches the solution space of the problem for the optimal solution. This sort of solution method is normally frowned upon due to the exponential number of possible solutions, but B & B adds bounds to the solution function. This is done by continuously removing branches from the subspace by determining whether or not the space can contain the optimal solution by comparing it to the current best solution [31]. The solution subsets that cannot possibly contain an optimal solution are then removed, therefore removing a portion of the solution space that has to be explicitly explored. In [22] the B & B algorithm is combined with heuristics to determine an optimal solution for the cutting of stock in 1 dimension. The heuristic is a greedy algorithm which picks the best local optima at a stage of the search with the theory that this will lead to the global optimal.

The linear optimization model for non-guillotine cutting of square and rectangular objects in a 2D plane created by Beasley [32] uses a combination of Lagrangian relaxation, subgradient optimization, and a tree search algorithm to determine the optimal solution. Lagrangian relaxation simplifies the calculation of an optimization problem by moving constraints on variables to the objective function, which removes hard constraints on variables and instead penalizes variable selections that violate the constraints [8]. This can produce a near optimal solution and sometimes the optimal solution. Subgradient optimization is an iterative process that uses the gradient of the previous solution to slowly converge on an optimal solution. The use of these allows this method to continually work towards an optimal solution and the tree search algorithm allows for backtracking if an optimal solution is not being found upon the current path. The use of penalty functions for constraint violations rather than hard constraints allows for easier movement in the solutions space. Kim and Gossard [24], proposed a method with an objective function which is a combination of objective goals to satisfy inequality constraints and penalty functions which penalize object overlap and the use of a gradient method to search the solution space.

While these methods are often able to achieve optimal or near optimal solutions, there are disadvantages to be considered. Since these methods will often search the solution space in a linear fashion, an excessive amount of time could be required to find the optimal solution, especially if the search algorithm has a poor initial direction. This is a large disadvantage of the tree search algorithm if a large amount of back tracking is required. While B & B is able to remove portions of the solutions space, it is possible that a large and poorly defined solution space could take more time [19]. Since a Lagrangian relaxation penalizes the objective function rather than keeping hard constraints it is possible that the optimal solution may not be found if the penalty functions are not weighted correctly. The effects of these disadvantages can be minimized with well-defined objective functions, constraints, and good initial search directions.

2.2.2.3 Stochastic Methods

Stochastic methods use a similar objective function to represent the packing objectives being optimized, but rather than searching the solution space in an order or pattern, the variable values are randomly generated to find possible solutions. Simulated Annealing (SA), is an optimization method which relies on random moves for the generation of solutions. SA relies on the continuous evaluation of random changes made to the system to reach an optimal result. SA methods use a weighted objective function to determine the quality of a change made to the system. SA methods make random changes to the solution and evaluate them against the previous best solution. If the new solution is better it is always accepted, if it is worse there is a chance that it can still be accepted which is dependent on the temperature of the system which decreases over time. SA escapes local optima due the random chance of accepting a worse solution. The temperature (taken from the physical annealing process), represents the randomness in the system. Higher temperatures have a higher chance of accepting an inferior solution.

Simulated Annealing has been explored and expanded upon over the years by different authors. Szykman and Cagan [26, 33] presented an SA method for packing objects and were able to add constraints to the system. They presented a basic objective function which focused on packing density, component overlap and object containment with in the container. Their method used a series of moves to change the layout of objects. The used three different movement types; translate, rotate, and swap, with 15 moves per temperature. The objective function was checked after every move. The details of the moves were determined randomly. This method was successful and validated by packing objects with known solutions. However the scope of their method was limited to cylindrical and cuboid geometries, and object rotation was limited to 90 degree increments. Xu developed a SA method that used different objectives for packing polygons without these constraints but found difficulty reaching known solutions [34].

Genetic algorithms (GA) is another stochastic method and relies on the continuous evaluation of a weighted fitness function to determine the quality of new solutions. GAs make changes to a family of possible solutions via mutations which change 1 bit of the child solution. Which set of solutions is kept and which are discarded is determined by the evaluation of the fitness function. Large changes are made via crossover by randomly switching parts of two parent strings into the next generation. The expectation is that after enough generations the optimal solution will be found.

The random nature of stochastic methods allows for a wider search of the solution space, but can possibly miss optimal or feasible solutions due to this. Since these methods do not search in an ordered fashion the optimal solution may be in an area of the solution space that was not search by these methods.

2.2.3 Overview Geometric Representation

Geometry was represented in various ways throughout the methods explored. Some chose to represent objects as solid bodies, boundary representation, Octree, or even by bounding volume. There are advantages and disadvantages to any representation. Starting with solid models which will give the greatest amount of detail but are also the most computational expensive due to the amount of information in a solid model. Boundary representations are usually an ordered list of vertices connected by straight lines to represent object geometry. This is a computationally light representation as only position information of the vertices is required but loses detail due the assumed straight line connections between vertices.

Octree represents objects by subdividing a cubic volume into eight fully occupied, partially occupied, or unoccupied octants, where the fully and partially occupied octants are represented as solid and unoccupied octants are represented as empty space. Partially occupied octants can re-subdivided to for more detailed representation of the object. This is a simple way of representing occupied space but the accuracy and computation expense of this method is dependent on the resolution of the edges. where a higher resolution has higher accuracy but is more computationally expensive. The use of bounding volumes is the simplest and least computationally expensive representation but at the expense of geometric accuracy. While this method works for simple geometries, packing efficiency decreases as geometric complexity increases.

To represent objects with more complex geometries a partitioning method breaks down complex geometries and represents them as a series of connected cubes and cylinders. Allowing for complex shapes to be represented to a greater accuracy without the computational expense of complicated curve calculations. Agarwal created product specific shape grammars to replicate existing products as well as create new products [35].

2.3 Summary

In this Chapter an overview of product family configuration and object packing methods and concepts were presented. Product family and configuration design research has introduced a variety of methods for the development of product families and evaluation of product platforms. A multitude of optimization methods have been applied to object packing problems. While these methods are able to produce optimal solutions, most of them have limits in scope or object placement. Object geometry was the most common limitation discovered in the review with the use simple rectangular or circular representations of objects. In the next chapter the problem specific to this thesis is discussed and the problem formulation is modeled.

CHAPTER 3

DESIGNER PREFERENCE CONFIGURATION-LAYOUT METHOD

The problem being addressed by this thesis is presented in Section 1.1. This chapter will present the formal problem formulation and the methods developed do address this problem. The problem being addressed in this thesis is the Configuration-Layout Design problem. This problem contains the 1) product family configuration and 2) component layout sub-problems. The sub-problems will be presented and then combined into the Configuration-Layout Design problem. The Configuration-Layout method, developed to address the Configuration-Layout Design problem, will be presented and modeled. The component layout portion of this method currently only refers to the placement of user interface components in a or fascia. An example for the development of a coffeemaker product family will be presented throughout the chapter. This example will be used to illustrate each part of the problem and methods as they are presented in the chapter. The next section will present the information that is given at the start of the Configuration-Layout problem.

3.1 Beginnings of a Product Family

This section will introduce terminologies and concepts needed to start presenting the problem formulation and methods presented in this chapter. This will provide background for the various parts of the Configuration-Layout problem and the methods presented to address it. Each concept will be introduced as part of an example for the development of a coffeemaker product family which will continue throughout the chapter. This will start with the introduction of ”components”. For this thesis, components are considered objects that perform a specific function and also have a physical presence in the fascia, or region in which components can be placed. This means each component must have both physical dimensions and a function description. A component can be a sub-assembly of smaller components as long as it is able to be given a fully defined set of dimensions. Table 3.1 presents the set of components for the coffeemaker product family. While many of the components likely will not be square or rectangular, the objects are represented by their bounding boxes.

Table 3.1: Dimensions of Coffeemaker Component

	Components	Dimensions (mm)	Fascia Choice
1	LCD Clock	(60,20)	Top
2	Power Button	(20,20)	Bottom
3	Set Hour	(10,10)	Top
4	Set Minute	(10,10)	Top
5	AM/PM Set	(5,5)	Top
6	Brew-Delay Set	(10,10)	Top
7	Temp Control	(10,10)	Top
8	Brew Strength Btn	(7,7)	Top
9	1 - 4 Cup Btn	(7,7)	Top

Each component is also given a number which will be used for indexing at a later time. Notice that each component is also given a set of 2D dimensions and a Fascia Choice. The dimensions are formatted as (Width, Height). Notice that every component required to make a coffeemaker is not listed in Table 3.1 such as a Burner, Carafe, Water Tank, etc. This is because many of the components will be common to the product family or will not have an effect on the component layout process. These were omitted to simplify the example.

Fascias in the product family are defined as the specific regions of a product in which the user interface components can be placed. When a fascia is defined it is given a name, dimensions, and location in the product. The coffeemaker product family has 2 fascias to choose from. The ”Top” fascia which is right in-front of the filter tray and a ”Bottom” fascia which is right below the carafe warmer. Table 3.2 presents these fascia, their dimensions, and global coordinates.

Table 3.2: Dimensions and Location of Coffeemaker Fascia

	Fascia	Dimensions (mm)	Location (X, Y)
1	Top	(150,100)	(0,130)
2	Bottom	(150,30)	(0,0)

The dimensions are formatted as (Width,Height) and the locations are global coordinates for the lower left corner of the fascia. The last concept to be introduced in this section are Configuration Categories and Options. Configuration Options are modules, components, or design features used to create product variety but may not have any distinct dimensions or play a roll in the component layout process. Configuration options are part of distinct Configuration Categories which contain all similar Configuration Options. The categories are required because each product product must select one and only one Configuration Option from each Configuration Category. Product variety can be created by selecting different Configuration Options while keeping the same component composition. Table 3.3 presents the Configuration Options and Categories for the coffeemaker product family example.

Table 3.3: Configuration Categories and Options for Coffeemaker Product Family

Configuration Category	Configurations Options	
Bean Grinder	10	w/ Grinder
	11	w/o Grinder
Single Cup Module	12	w/ Single Cup
	13	w/o Single Cup

There are 2 configuration categories in this example, Bean Grinder and Single Cup. In this case the options in these categories determine the presence of a feature but in others there could be multiple feature options. Notice that each configuration option is also assigned a number that as continued from the component numbering in Table 3.1. This is because the configuration options can also be used in configuration constraints during the configuration stage. The information presented in this section provides the background required to present the problem formulation for the Configuration-Layout problem. The next section will present the problem formulation for the Configuration-Layout problem.

3.2 Formulating the Configuration-Layout Problem

The problem addressed in this thesis is complex in nature and can be divided into two sub-problems: (1) combinatorial configuration of product families, (2) spatial layout of components. These sub-problems are represented differently mathematically and will be presented separately. A final combined problem formulation will then be presented. The overall goal of this process is to determine feasible configurations that satisfy both, configuration and layout constraints, and evaluate feasible configurations to minimize unique components.

3.2.1 Combinatorial Configuration

Components and configuration options used to create variety result in a combinatorial configuration problem. To generate product configurations, one needs to systematically enumerate elements from the power set of the Components then generate products from combinations of components and configuration options. This results in a rapidly increasing number of possible combinations. To resolve this, constraints can be applied to combinations of components and configuration options to reduce the number generated. This produces a set of feasible products. From this set, designers can select products to develop the product family. For this problem, a set of components and configuration options are given. Constraints are specified for combinations of components and configuration options. The combinatorial design space is calculated to determine all feasible combinations. A common platform can then be identified to minimize the number of unique components.

3.2.2 Spatial Layout

The spatial layout of user interface components is a vital task, as designers must focus on both ergonomics and aesthetics on top of other configuration viewpoints. Additionally, the layout of user interfaces is highly dependent on the product dimensions and target markets the designers are attempting to reach. This can limit fascias and designers must ensure that components do not overlap while reaching these goals. To reach these goals, designers can utilize spatial layout constraints to satisfy the requirements of these viewpoints and evaluate the layouts for component overlap.

3.2.3 Combined Problem Formulation

The previous sections described the sub-problems this thesis is addressing; the combined problem utilizes the parts from the individual problems to formalize the entire Configuration-Layout Design problem. The configuration layout problem takes the information given from the sub-problems as the starting point. An overview of the steps of the Configuration Layout Problem is provided here:

A. Identify functions, components and configuration options across the product family. This set of configuration information will create the base combinatorial space. Identify target product dimensions to determine the size of the fascia for which components are being arranged.

B. Specify constraints on combinations of configuration elements. This will aid in the reduction of the configuration design space. Specify spatial layout constraints between components being arranged. These constraints maybe aesthetic or functional constraints required for proper product function. The size of the fascias should also be specified to provide the bounds for the components.

C. Compute feasible products and configurations for the product family. This step generates all possible configurations that satisfy the previously defined configuration constraints.

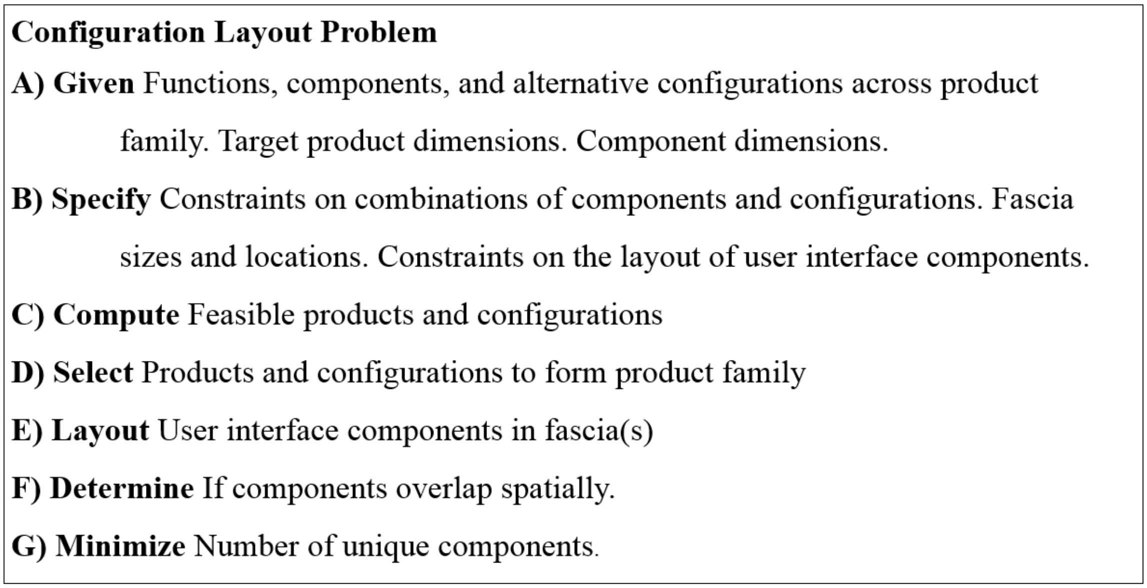
D. Select products to form the product family. These products are comprised of configurations generated in the previous step and should encompass the desired variety of the product family.

E. Compute the locations of all the components within the selected product family. The component locations should be calculated to maximize common component locations across the entire product family.

F. Determine if any of the components being arranged overlap spatially with one another and ensure that they are fully contained within the fascia.

G. Determine the common product platform by ensuring common component locations across the entire product family. Utilizing common locations to minimize product specific components and increase product family commonality.

The steps of the Configuration Layout Problem allow designers to configure product families and determine feasibility of the configurations produced and of dimensional targets prior to large-scale design. The word formulation of the Configuration-Layout problem, given in Figure 3.1 summarizes these steps and organizes them using the same labels.



Configuration Layout Problem

A) Given Functions, components, and alternative configurations across product family. Target product dimensions. Component dimensions.

B) Specify Constraints on combinations of components and configurations. Fascia sizes and locations. Constraints on the layout of user interface components.

C) Compute Feasible products and configurations

D) Select Products and configurations to form product family

E) Layout User interface components in fascia(s)

F) Determine If components overlap spatially.

G) Minimize Number of unique components.

Figure 3.1: Constraint Pair Connections.

3.3 The Designer Preference Configuration Design Layout Method

In this section the method presented in this thesis is modeled and related to the steps of the problem formulation. The configuration design portion of the problem utilizes set-theory and constraints on combinations. This will address the first research question and present models for the designer preference viewpoint. In the layout portion of the method linear programming and a spatial grammar will be presented to represent spatial constraints and perform constraint satisfaction. This will address the second research question and allow for spatial representation of products and components during conceptual design. As the method and its concepts are presented the example of a coffeemaker product family will be presented as well as the part of the problem formulation being addressed. Step ”A” of the Configuration-Layout problem was presented previously in Section 3.1.

3.3.1 Designer Preference Configuration-Design Space

During configuration design the use of constraints on combinations of components is recommended to reduce the number of possible combinations produced. Previous research has developed constraints to reduce combinations by component intent, connection, assembly and flow of information or materials viewpoints [4, 18]. These viewpoints are able to create large reductions in the number of combinations generated, but are limited in scope. Often designers may want to constrain a combination of components or features to create the desired variety, usability, or other functional reasons. The work presented in this thesis proposes configuration constraints that enable the designer to apply constraints that do not fall within the viewpoint specific constraints proposed in previous research. This is defined as the configuration design space, $ConfSpc$.

3.3.1.1 The Unconstrained Design Space

This section presents the representation of the Unconstrained Configuration design space. The representation of the unconstrained process has a great impact on how and when constraints can be applied. This will provide valuable insight to prior to the application of configuration constraints (steps B and C in the Configuration-Layout Problem). The set of components is denoted as $Comps^{in} = \{c_1^{in},\ c_2^{in},\ \ldots, c_m^{in}\}$ and configuration options are denoted as $Confs = \{Cf_1,\ Cf_2,\ \ldots Cf_n\}$. Where $Confs$ is a set of sets where Cf_j represents the configuration categories, j, of $Confs$ and $\alpha_{j,k}$ are the configuration options of Cf_j. Where j is the Configuration category and k is are the configuration options. $|Confs|$ is the number of configuration categories. Each element across both $Comps^{in}$ and $Confs$ their subsequent subsets is required to be distinct and well defined. This is because later processes will disrupt the ordering and require each element to be located by its identifier. The identifier can consist of a distinct name, number, or symbol. An example for the form $Comps^{in}$ for the coffeemaker product family is presented in Figure 3.2.

$$\begin{aligned} Comps^{in} &= \{\text{LCD Clock, Power Button, Set Hour, Set Minute, AM/PM Set,} \\ &\quad \text{Brew-Delay Set, Temp Control, Brew Strength Btn,} \\ &\quad \text{1 - 4 Cup Btn}\} \\ &= \{1,2,3,4,5,6,7,8,9\} \\ &= \{c_1^{in}, c_2^{in}, c_3^{in}, c_4^{in}, c_5^{in}, c_6^{in}, c_7^{in}, c_8^{in}, c_9^{in}\} \end{aligned}$$

Figure 3.2: $Comps^{in}$ example for the coffeemaker product family

In this figure the number and name forms of identifying the components are presented. Below them is the set, c_i^{in}, representation is presented. The $Confs$ set for the coffeemaker product family is presented in Figure 3.3.

$$\begin{aligned} Confs &= \{\{\text{ w/ Grinder, w/o Grinder }\}, \{\text{w/ Single Cup, w/o Single Cup}\}\} \\ &= \{\{10,11\},\{12,13\}\} \\ &= \{Cf_1, Cf_2\} \\ &= \{\{\alpha_{1,1}, \alpha_{1,2}\}, \{\alpha_{2,1}, \alpha_{2,2}\}\} \end{aligned}$$

To Clarify:

$$\begin{aligned} Cf_1 &= \{\alpha_{1,1}, \alpha_{1,2}\} = \{10,11\} \\ Cf_2 &= \{\alpha_{2,1}, \alpha_{2,2}\} = \{12,13\} \end{aligned}$$

Figure 3.3: $Confs$ example for the coffeemaker product family

Notice that the name and number identifying forms are given as possible set contents. This is because they are organized to be easily referenced such as from Tables 3.1 and 3.3. Figure 3.3 also presents how Cf_j and $\alpha_{j,k}$ components of the $Confs$ set fit.

All combinations of components can be generated by applying a power set relationship to $Comps^{in}$. Using the Cartesian product operation allows for configuration options to be properly selected and only one option from each category be selected.

$$ConfSpc = \wp(Comps^{in}) \times \prod_{j=1}^{|Confs|} Confs_j \tag{3.1}$$

This definition of the generation process allows for the simple generation of all combinations of components and configuration options in a single step. While this definition is technically correct, due to the presence of the power set, constraints can only applied after all combinations are generated. Resulting in a large number of infeasible combinations being generated. The power set prevents constraints being applied prior to generation. To reduce the number of infeasible combinations being generated from the process, sets of components or configuration options must be able to be manipulated by the constraints prior to enumeration. This means each component and its absence (null set) must be present in the formulation prior to the enumaration process. Changes were made to Equation 3.1 to account for these issues. Equations 3.2, 3.3, and 3.4 present the modified unconstrained generation process.

$$\begin{aligned} Comps^{bin} &= Comps^{in} \times \{\{\oslash\}\} \\ &= \{(c_1^{in}, \{\oslash\}), (c_2^{in}, \{\oslash\}), \ldots, (c_m^{in}, \{\oslash\})\} \\ &= \{c_1^{bin}, c_2^{bin}, \ldots, c_m^{bin}\} \end{aligned} \tag{3.2}$$

Equation 3.2 creates $Comps^{bin}$ which consists of subsets containing the components and a non-empty set containing an empty set. The use of the non-empty set of an empty set,$\{\oslash\}$, establishes the empty set as an element (rather than a subset) and allows for the absence of a component to be represented and manipulated by set theory operations. The ability to represent and manipulate the general absence of an object is an important concept as it provides the ability to show that an object is missing from a set. In this case, $\{\oslash\}$ does not represent the absence of a specific component but a general absence of an object. The set produced by the Cartesian product operation in 3.2, creates subsets containing a component and its absence. The power set in Equation 3.1 does not have to represent object absence because it automatically enumerates all component combinations, but to interact and apply possible constraints prior to enumeration the general absence of an object needs to be able to be represented.

$$\begin{aligned} Parts^{all} &= Comps^{bin} \cup Confs \\ &= \{(c_1^{in}, \{\oslash\}), (c_2^{in}, \{\oslash\}), \ldots, (c_m^{in}, \{\oslash\}), Cf_1,\ Cf_2,\ \ldots Cf_n\ \} \\ &= \{p_1, p_2, \ldots, p_m, p_{m+1}, p_{m+2}, \ldots, p_{m+n}\} \end{aligned} \tag{3.3}$$

Equation 3.3, combines the sets containing the sets of components and the set containing the sets of configuration options to create $Parts^{all}$. This will allow for constraints to manipulate components and configuration options. This also simplifies the unconstrained generation process. While the Cartesian Product operation results in an ordered set, since all objects of concern in this process are distinct, the ordered set can be changed into a normal (un-ordered) set of objects by removing any duplicate items. Equation 3.4 then applies a power set operation to all the elements of $Parts^{all}$ to create, $Combinations^{all}$ which contains all unconstrained combinations of Components and configuration options. However, $Combinations^{all}$ still contains $\{\oslash\}$ elements.

$$\begin{aligned} Combinations^{all} &= \prod_{i=1}^{|Parts^{all}|} Parts_i^{all} \\ &= \{(c_1^{in}, \{\oslash\}) \times (c_2^{in}, \{\oslash\}) \times (c_3^{in}, \{\oslash\}), \cdots \times (c_m^{in}, \{\oslash\}) \\ &\qquad \times Cf_1 \times Cf_2 \times Cf_3 \cdots \times Cf_n\} \\ &= \{p_1 \times p_2, \cdots \times p_m \times p_{m+1} \times p_{m+2} \cdots \times p_{m+n}\} \end{aligned} \tag{3.4}$$

$$ConfSpc = \{\beta \setminus \{\oslash\} | \beta \in Combinations^{all}\} \tag{3.5}$$

Equation 3.5 removes the empty set, $\{\oslash\}$, elements from all of the generated products, leaving only the components and configuration options in each set with in $ConfSpc$. This representation of the unconstrained generation of products allows for constraints to be easily applied.

The total number of unconstrained combinations of components and configuration options can be calculated prior to enumeration. The total number of component combinations for N_{Comps} components across the entire product family is given by the size of the power set of the $|Comps^{in}|$ components. The process of calculating the maximum number of possible combinations of components and configuration options, N_{Total}, is given below:

$$N_{Comps} = \sum_{k=0}^{|Comps^{in}|} \frac{|Comps^{in}|!}{k! * (|Comps^{in}| - k)!} \tag{3.6}$$

$$N_{ConfigOpt} = \prod_{i=1}^{|Confs|} |Cf_i| \tag{3.7}$$

$$N_{Total} = N_{Comps} * N_{ConfigOpt} \tag{3.8}$$

Where k is the number of components in a product. N_{Comps} yields the number of combinations of components that could be generated from the $|Comps^{in}|$ components. $N_{ConfigOpt}$ is the number of combinations of configuration options produced with no empty sets, meaning all possible unconstrained combinations. This is calculated by multiplying the number of configuration options, $Confs$, available in each of configuration category from 1 to $|Confs|$. The calculation is a modification of the $nchoosek$ calculation that determines the total number of combinations for each set size and also accounts for configuration options since one option from each configuration category must be selected for a product to be considered valid. The unconstrained total for the components and configuration options of the coffeemaker product family is 2,048. The calculation for this is presented in Figure 3.4.

$$
\begin{aligned}
N_{Comps} &= \sum_{k=0}^{|Comps^{in}|} \frac{|Comps^{in}|!}{k! * (|Comps^{in}| - k)!} \\
&= \sum_{k=0}^{|9|} \frac{|9|!}{k! * (|Comps^{in}| - k)!} \\
&= 1 + 9 + 36 + 84 + 126 + 126 + 84 + 36 + 9 + 1 \\
&= 512
\end{aligned}
$$

$$
\begin{aligned}
N_{ConfigOpt} &= \prod_{i=1}^{|Confs|} |Cf_i| \\
&= 2 \ \times \ 2 \\
&= 4
\end{aligned}
$$

$$
\begin{aligned}
N_{Total} &= N_{Comps} * N_{ConfigOpt} \\
&= 512 \ \times \ 4 \\
&= 2,048
\end{aligned}
$$

Figure 3.4: Unconstrained combination calculation for coffeemaker product family example

This section presented the unconstrained combinatorial generation process. The purpose of this was to present how the components and configuration options are represented and discuss how the unconstrained generation process is represented. The way the unconstrained generation process is formed can greatly effect when and how constraints can be applied. For the purpose of this thesis it was important to be able to manipulate components and configuration options prior to enumeration. The next section will take the unconstrained generation process and modify it so configuration constraints can be applied.

3.3.1.2 The Constrained Design Space

This section will present the constrained process for the generation of component and configuration option combinations. As the constrained process is presented the coffeemaker product family will also be developed. To start the constrained generation process, constraints must first be defined. Three constraints are presented in this thesis. The first constraint, Required Pairs (Eqn 3.9), requires a pair of components/ configurations options to both be present in any combination either are present and is represented by C^{c+}. The second constraint, Disallowed Pairs (Eqn 3.10), disallows a pair of components/ configurations options from being present in any feasible combination, and represented by C^{c-}. The third constraint, Required Components (Eqn 3.11), specifies a component must be in any configuration for it to be feasible and is represented by $C^{c=}$. This allows for components that are required for the product to perform its core function to be included. These constraints are denoted by the superscript "c":

$$C^{c+} = \{Set\, of\, required\, configuration\, pairs\} \tag{3.9}$$

$$C^{c-} = \{Set\, of\, disallowed\, configuration\, pairs\} \tag{3.10}$$

$$C^{c=} = \{Set\, of\, required\, components\} \tag{3.11}$$

The Disallowed and Required Pair constraints are in pairs, rather than triplets or quadruplets because the paired constraints can easily be daisy chained to perform the same actions of different multiples. Figure 3.5 presents the configuration constraints for the coffeemaker product family. Notice that the constraints are presented as the names of the elements. This acceptable during the set theory processes since each object was given a distinct name. The number identifier given will be used during the layout process. Notice that Components and Configuration options are able to appear in these constraints.

$$
\begin{aligned}
C^{c+} &= \{\{\text{LCD Clock ,Set Hour}\}, \{\text{LCD Clock ,Set Minute}\},\\
&\qquad\qquad \{\text{LCD Clock ,AM/PM Set}\}, \{\text{LCD Clock ,Brew-Delay Set}\}\}\\
C^{c-} &= \{\{\text{w/ Single Cup, w/ Grinder}\}, \{\text{w/ Single Cup, 1 - 4 Cup Btn}\}\}\\
C^{c=} &= \{\text{Power Button}\}
\end{aligned}
$$

Figure 3.5: Configuration Constraints for the coffeemaker product family example

The step of defining configuration constraints refers to step B, in the problem formulation. These constraints will reduce the number of combinations generated. The individual effects of these constraints have been characterized individually, but their combined effects create a complex problem that has yet to be elegantly addressed. The C^{c+} and $C^{c=}$ constraints have the effect of reducing the number of combinations by 50% for each set of constraints and the C^{c-} causes a reduction of 25% for each pair. When combined, the effect on the number of feasible combinations is difficult to define but these basic characterizations can serve as a guide (upper bound) for predicting the number of combinations produced for the given constraints:

$$M_{C^+} = N_{Comps} * 0.5^{m_{c+}} \tag{3.12}$$

$$M_{C^=} = N_{Comps} * 0.5^{m_{c=}} \tag{3.13}$$

$$M_{C^-} = N_{Comps} * 0.75^{m_{c-}} \tag{3.14}$$

Equations 3.12, 3.13 and 3.14 characterize the behavior of the constraints. $M_{C^+}$is the number of combinations produced using the required pairs constraint. M_{C^-} is the number of combinations generated from the disallowed pair constraint. $M_{C^=}$ is the number of combinations generated from required components constraint. ${}_mC$ is the unconstrained number of combinations. Where m_{c+}, m_{c-} are the number of required and disallowed pairs in C^{c+} and C^{c-} respectively ($|C^{c+}|$ and $|C^{c-}|$). Finally $m_{c=}$ is the number of components in the required components constraint, $C^{c=}$.

The configuration constraints for the generation process were described above. The rest of this section will present the constrained combinatorial generation process, or step C from the problem formulation presented in Figure 3.1. The first step in the constrained generation process is to remove the Required Components from the main set of components. This is to ensure that they do not appear in the Cartesian product operation in Equation 3.2 since there is no reason to represent the absence of any required components. The required components will be added to all of the products generated after constraints have been applied. To remove the Required Components, a set difference operation is performed with the set of all components, $Comps^{in}$, and the set of Required Components, $C^{c=}$. Equation 3.15 presents this process.

$$Comps^{in} = Comps^{in} \setminus C^{c=} \tag{3.15}$$

$$\begin{aligned} Comps^{bin} &= Comps^{in} \times \{\{\oslash\}\} \\ &= \{(c_1^{in}, \{\oslash\}), (c_2^{in}, \{\oslash\}), \ldots, (c_{m-|C^{c=}|}^{in}, \{\oslash\})\} \end{aligned} \tag{3.16}$$

After this $C^{c=}$ is set to the side to be used during the final generation process. $Comps^{in}$ sent to the cartesian product operation for the remaining components to be placed in sets with $\{\oslash\}$ elements as seen in Equation 3.16. As stated previously , $\{\oslash\}$ does not represent the absence of a specific component but a general absence of an object. Figure 3.6 presents the processes from Equations 3.15 and 3.16 applied to the coffeemaker product family.

$$\begin{aligned} Comps^{in} &= Comps^{in} \setminus \text{Power Button} \\ &= \{\text{LCD Clock, Set Hour, Set Minute, AM/PM Set, Brew-Delay Set,} \\ &\qquad \text{Temp Control, Brew Strength Btn, 1 - 4 Cup Btn}\} \end{aligned}$$

$$\begin{aligned} Comps^{bin} &= Comps^{in} \times \{\{\oslash\}\} \\ &= \Big\{(\text{LCD Clock}, \{\oslash\}), (\text{Set Hour}, \{\oslash\}), (\text{Set Minute}, \{\oslash\}), (\text{AM/PM Set}, \{\oslash\}), \\ &\quad (\text{Brew-Delay Set}, \{\oslash\}), (\text{Temp Control}, \{\oslash\}), (\text{Brew Strength Btn}, \{\oslash\}), \\ &\quad (\text{1 - 4 Cup Btn}, \{\oslash\})\Big\} \end{aligned}$$

Figure 3.6: Removing Required Components and adding Component absence to $Comps^{in}$ for the coffeemaker product family example

Notice that $Comps^{bin}$ has all of the components except for the Power Button, which was removed from $Comps^{in}$, paired with an $\{\oslash\}$. The next steps are to apply the other constraints. The constraint application process starts with results of Equation 3.3, $Parts^{all}$. Which contained individual sets containing components and $\{\oslash\}$, and the sets of configuration options, $Confs$, where each set represents a different configuration category. Figure 3.7 presents Equation 3.3 used in the coffeemaker product family example.

$$\begin{aligned} Parts^{all} &= Comps^{bin} \cup Confs \\ &= \Big\{(\text{LCD Clock}, \{\oslash\}), (\text{Set Hour}, \{\oslash\}), (\text{Set Minute}, \{\oslash\}), (\text{AM/PM Set}, \{\oslash\}), \\ &\quad (\text{Brew-Delay Set}, \{\oslash\}), (\text{Temp Control}, \{\oslash\}), (\text{Brew Strength Btn}, \{\oslash\}), \\ &\quad (\text{1 - 4 Cup Btn}, \{\oslash\}), \{\text{ w/ Grinder, w/o Grinder }\}, \{\text{w/ Single Cup, w/o Single Cup}\}\Big\} \end{aligned}$$

Figure 3.7: Combines $Comps^{in}$ and $Confs$ for the coffeemaker product family example

$Parts^{all}$ now contains the components from $Comps^{bin}$ and the sets of configuration options from $Confs$. The remaining constraints need to be applied to $Parts^{all}$ as part of the generation process. However the objects (components and configuration options) must first be filtered to determine which objects are connected via constraints and if any objects appear in multiple constraints. Components in both constraint types must be checked. Equation 3.17 creates a temporary combined list of constraints in $TempList$. Equation 3.18 then creates sets of objects in $FilteredConstraints$ that are connected via constraints.

$$\begin{aligned} TempList &= C^{c+} \cup C^{c-} \\ &= \{t_1, t_2, t_3, \ldots, t_{|C^{c+}|+|C^{c-}|}\} \end{aligned} \tag{3.17}$$

$$\begin{gathered} FilteredConstraints = \{f_i | t_j \in f_i \wedge \{t_k \in f_i \Leftrightarrow t_j \cap t_k = \neg\emptyset\}\} \\ f \in FilteredConstraints \end{gathered} \tag{3.18}$$

Both $TempList$ and $FilteredConstraints$ are small intermediate variables that will be discarded. Figure 3.8 presents the use of Equations 3.17 and 3.18 to the coffeemaker product family example.

$$\begin{aligned} TempList &= C^{c+} \cup C^{c-} \\ &= \Big\{\{\text{LCD Clock, Set Hour}\}, \{\text{LCD Clock, Set Minute}\}, \\ &\quad \{\text{LCD Clock, AM/PM Set}\}, \{\text{LCD Clock, Brew-Delay Set}\}, \\ &\quad \{\text{w/ Single Cup, w/ Grinder}\}, \{\text{w/ Single Cup, 1 - 4 Cup Btn}\}\Big\} \end{aligned}$$

$$\begin{aligned} FilteredConstraints &= \{f_i | t_j \in f_i \wedge \{t_k \in f_i \Leftrightarrow t_j \cap t_k = \neg\emptyset\}\} \\ &= \Big\{\{\text{LCD Clock, Set Hour, Set Minute, AM/PM Set ,Brew-Delay Set}\}, \\ &\quad \{\text{Single Cup, w/ Grinder, 1 - 4 Cup Btn}\}\Big\} \end{aligned}$$

Figure 3.8: Filtered Objets for Application of Required Pair and Disallowed Pair constraints for coffeemaker product family example

Notice that $TempList$ contains the contents of both C^{c+} and C^{c-}. The $FilteredConstraints$ has 2 elements in the form of sets. Notice that the elements in each of the subsets appear in a constraint with atleast one other element in the subset. The constraints must be applied to the subsets as a whole. Now that all object connections have been found the sets from $Parts^{all}$ need to be separated into groups by whether or not they contain objects connected by constraints. Equations 3.19 and 3.20 separate the components connected via constraints into groups and remove them from $Parts^{all}$.

$$\begin{gathered} i : 1 \rightarrow |FilteredConstraints| \\ ObjectGroups = \{o_i^G | p_j \in o_i^G \Leftrightarrow \text{An element of } p_j \in f_i\} \end{gathered} \tag{3.19}$$

$$Parts^{all} = Parts^{all} \setminus ObjectGroups \tag{3.20}$$

Equation 3.19 is important because it allows for constraints to be applied to a smaller set of objects rather than the combinations of every object in the product family. To illustrate the results of these equations, Figure 3.9 presents the the coffeemaker product family result of these equations.

$$\begin{aligned} ObjectGroups &= \{o_1^G, o_2^G\} \\ &= \Big\{\Big\{(\text{LCD Clock}, \{\oslash\}), (\text{Set Hour}, \{\oslash\}), (\text{Set Minute}, \{\oslash\}), \\ &\quad (\text{AM/PM Set}, \{\oslash\}), (\text{Brew-Delay Set}\}, \{\oslash\})\Big\}, \Big\{(\text{1 - 4 Cup Btn}, \{\oslash\}), \\ &\quad \{ \text{ w/ Grinder, w/o Grinder } \}, \{\text{w/ Single Cup, w/o Single Cup}\}\Big\}\Big\} \end{aligned}$$

$$\begin{aligned} Parts^{all} &= Parts^{all} \setminus ObjectGroups \\ &= \Big\{(\text{Temp Control}, \{\oslash\}), (\text{Brew Strength Btn}, \{\oslash\})\Big\} \end{aligned}$$

Figure 3.9: Separation of objects connected via constraints into groups for coffeemaker product family

Notice that the objects in the sets in $ObjectGroups$ match the contents of $FilteredConstraints$. Once the objects from $ObjectGroups$ are removed, only the unconstrained components remain in $Parts^{all}$. The next step of this process is to apply a Cartesian Product operation to the elements of the subsets of $ObjectGroups$ and $Parts^{all}$. A Cartesian Product performed between them to generate all combinations of the elements of these sets, to allow invalid sets may be removed. While this does generate invalid sets, it is more efficient as it generates fewer invalid sets. This generation process is presented in Equation 3.21.

$$\begin{aligned} s_i &\in SetCombinations \\ i &: 1 \rightarrow |ObjectGroups| \\ s_i &= \prod o_i^G \end{aligned} \tag{3.21}$$

Figures 3.10 and 3.11 present the results of Equation 3.21 for the example for the coffeemaker product family. Notice that each subset of s_1 and s_2 is a combination of the produced from the Cartesian product of the elements of o_1^G and o_2^G. Also notice that $\{\oslash\}$ is only listed once in each subset even if more than one component is absent. This is because these are not ordered sets so each element can only appear once in a set. s_1 has 32 subsets and s_2 has 8, many of which will be deemed invalid when constraints are applied. This does result in a few invalid combinations being generated. However fewer invalid sets will be generated compared to generating the 2048 unconstrained combinations and removing invalid sets from there.

$SetCombinations = \{s_1, s_2\}$

$s_1 = \Big\{$ {LCD Clock,Set Hour,Set Minute,AM PM Set,Brew-Delay Set}

{LCD Clock,Set Hour,Set Minute,AM PM Set, {$\oslash$}}

{LCD Clock,Set Hour,Set Minute, {$\oslash$},Brew-Delay Set}

{LCD Clock,Set Hour,Set Minute, {$\oslash$}}

{LCD Clock,Set Hour, {$\oslash$},AM PM Set,Brew-Delay Set}

{LCD Clock,Set Hour, {$\oslash$},AM PM Set}

{LCD Clock,Set Hour, {$\oslash$},Brew-Delay Set}

{LCD Clock,Set Hour, {$\oslash$}}

{LCD Clock, {$\oslash$},Set Minute,AM PM Set,Brew-Delay Set}

{LCD Clock, {$\oslash$},Set Minute,AM PM Set}

{LCD Clock, {$\oslash$},Set Minute,Brew-Delay Set}

{LCD Clock, {$\oslash$},Set Minute}

{LCD Clock, {$\oslash$},AM PM Set,Brew-Delay Set}

{LCD Clock, {$\oslash$},AM PM Set}

{LCD Clock, {$\oslash$},Brew-Delay Set}

{LCD Clock}

{{$\oslash$},Set Hour,Set Minute,AM PM Set,Brew-Delay Set}

{{$\oslash$},Set Hour,Set Minute,AM PM Set}

{{$\oslash$},Set Hour,Set Minute,Brew-Delay Set}

{{$\oslash$},Set Hour,Set Minute}

{{$\oslash$},Set Hour,AM PM Set,Brew-Delay Set}

{{$\oslash$},Set Hour,AM PM Set}

{{$\oslash$},Set Hour,Brew-Delay Set}

{{$\oslash$},Set Hour}

{{$\oslash$},Set Minute,AM PM Set,Brew-Delay Set}

{{$\oslash$},Set Minute,AM PM Set}

{{$\oslash$},Set Minute,Brew-Delay Set}

{{$\oslash$},Set Minute}

{{$\oslash$},AM PM Set,Brew-Delay Set}

{{$\oslash$},AM PM Set}

{{$\oslash$},Brew-Delay Set}

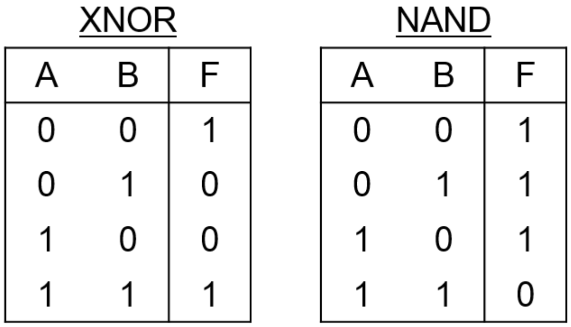
{{$\oslash$}} $\Big\}$

Figure 3.10: Results from the Cartesian Product of the Elements of o_1^G

$$\begin{aligned} s_2 = \Big\{ &\{\text{1-4 Cup Btn,Single Cup,w/ Grinder}\} \\ &\{\text{1-4 Cup Btn,Single Cup,w/o Grinder}\} \\ &\{\text{1-4 Cup Btn,No Single Cup,w/ Grinder}\} \\ &\{\text{1-4 Cup Btn,No Single Cup,w/o Grinder}\} \\ &\{\oslash\text{,Single Cup,w/ Grinder}\} \\ &\{\oslash\text{,Single Cup,w/o Grinder}\} \\ &\{\oslash\text{,No Single Cup,w/ Grinder}\} \\ &\{\oslash\text{,No Single Cup,w/o Grinder}\} \Big\} \end{aligned}$$

Figure 3.11: Results from the Cartesian Product of the Elements of o_2^G

Now that the components and configuration options are separated into sets of objects connected by constraints, the remaining constraints can be applied to the sets of $SetCombinations$. Both the Dissallowed Pairs and Required Pairs constraints can be broken down to represent basic logical operations. Where Required Pairs is a XNOR operation and Disallowed Pairs a NAND. The truth tables for these is presented in Figure 3.12.



XNOR

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

NAND

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

Figure 3.12: XNOR and NAND truth tables

The set builder notation representation for the XNOR and NAND operations are the ”If and Only If”, $\Leftrightarrow$, and the ”NOT AND”, $\neg\wedge$, statements. To apply these constraints in the generation process, set builder notation must be used to specify valid sets of objects. This required that components and configuration options are be able to be manipulated prior to full generation. Equation 3.22 presents the constraint application of the Required Pairs constraint. This functions by checking the elements of each constraint against the elements of each set within s_i.

$$\begin{aligned}
&r^s \in ConstrainedSets\\
&t_j \in s_i\\
&\;i : 1 \rightarrow |SetCombinations|\\
&\;j : 1 \rightarrow |s_i|\\
&\;k : 1 \rightarrow |C^{c+}|\\
&\{t_j \in r_i^s | C_{k,1}^{c+} \in t_j \Leftrightarrow C_{k,2}^{c+} \in t_j\}
\end{aligned} \tag{3.22}$$

Figure 3.13 presents the results of applying the Required Pair constraints to the s_1 set of combinations for the coffeemaker product family. The constraints being applied were presented in Figure 3.5. This results in a reduction from 32 to 2 combination. This means 30 out of 32 original combinations were removed. This reduction is because the combination of Required Pair constraints on the {LCD Clock,Set Hour,Set Minute,AM PM Set,Brew-Delay Set} caused either all of the components to be present or none of them to be present for a product to be valid. This leaves the only options as the set of components and the null set.

$$\begin{aligned}
&i = 1\\
&j : 1 \rightarrow 32\\
&k : 1 \rightarrow 4\\
r_1^s = \Big\{&\{\text{LCD Clock,Set Hour,Set Minute,AM PM Set,Brew-Delay Set}\}\\
&\;\{\{\oslash\}\}\Big\}
\end{aligned}$$

Figure 3.13: Results from the Required Pair constraint the Elements of s_1

The last constraint to be applied is the Disallowed Pair constraint which disallows components from appearing in a combination together. Equation 3.23 presents process for applying the Disallowed Pairs constraint to a set of components and configuration options.

$$
\begin{aligned}
&r^s \in ConstrainedSets \\
&\quad i : 1 \rightarrow |SetCombinations| \\
&\quad j : 1 \rightarrow |s_i| \\
&\quad k : 1 \rightarrow |C^{c-}| \\
&\{t_j \in r^s_i | C^{c-}_{k,1} \in t_j \neg \wedge C^{c-}_{k,2} \in t_j\}
\end{aligned}
\tag{3.23}
$$

Figure 3.14 presents the results of applying the Disallowed Pair constraint to the sets of combinations of s_2. While each disallowed pair constraint does not have as great of an impact as the Required Pairs they allow for the designer to make smaller changes to the desired product family.

$$
\begin{aligned}
&i = 2 \\
&j : 1 \rightarrow 8 \\
&k : 1 \rightarrow 2 \\
r^s_2 = \Big\{&\{\text{1-4 Cup Btn,No Single Cup,w/ Grinder}\} \\
&\{\text{1-4 Cup Btn,No Single Cup,w/o Grinder}\} \\
&\{\{\oslash\}\text{,Single Cup,w/o Grinder}\} \\
&\{\{\oslash\}\text{,No Single Cup,w/ Grinder}\} \\
&\{\{\oslash\}\text{,No Single Cup,w/o Grinder}\}\Big\}
\end{aligned}
$$

Figure 3.14: Results from the Required Pair constraint the Elements of s_2

The effects of the constraints are clear in the equation and example applications presented. The final step of the constrained generation process is to generate the full set of products. To do this, all combinations of the unconstrained ($ConstrainedSets$), constrained ($Parts^{all}$), and required components ($C^{c=}$) must be generated. The set of

Equations 3.24 and 3.25 present this final combination process.

$$GeneratedSet = \prod_{i=1}^{|ConstrainedSets|} \{r_i^s\} \times \prod_{j=1}^{|Parts^{all}|} \{p_j\} \tag{3.24}$$

$$Combinations^{all} = \{g \cup C^{c=} | g \in GeneratedSet\} \tag{3.25}$$

Equation 3.24 generates the combinations of the combinations of the constrained objects in $ConstrainedSets$ and unconstrained objects in $Parts^{all}$. Equation 3.25 adds the required components to every set of $GeneratedSet$, where g is an element of $GeneratedSet$ and a set of components and configuration options. The demonstration of Equation 3.24 onto the coffeemaker product family is presented by Figure 3.15.

$$\boxed{GeneratedSet = \{r_1^s \times r_2^s\} \times \{p_1 \times p_2\}}$$

Figure 3.15: Equation 3.24 applied to the coffeemaker product family

Notice That all of the sets from $ConstrainedSets$ and $Parts^{all}$ are included in the Cartesian product operation. Figure 3.16 presents the application of Equation 3.25 to the coffeemaker product family example. Notice that after constraints have been applied there are 40 combinations of components and configuration options remaining. This is a a large reduction from the unconstrained total (2048) and a much more manageable number to sort through to build the product family. Notice that most of the sets have a $\{\oslash\}$ except for the sets that have all of the components possible in the list.

$Combinations^{all} = \Big\{\{\{\oslash\}$, Pwr Button,Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Pwr Button,Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, Temp Control Dial,Pwr Button,Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Temp Control Dial,Pwr Button,Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, Brew Strength Btn,Pwr Button,Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Brew Strength Btn,Pwr Button,Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, Temp Control Dial,Brew Strength Btn,Pwr Button,Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Temp Control Dial,Brew Strength Btn,Pwr Button,Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, Pwr Button,No Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Pwr Button,No Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, Temp Control Dial,Pwr Button,No Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Temp Control Dial,Pwr Button,No Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, Brew Strength Btn,Pwr Button,No Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Brew Strength Btn,Pwr Button,No Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, Temp Control Dial,Brew Strength Btn,Pwr Button,No Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Temp Control Dial,Brew Strength Btn,Pwr Button,No Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, Pwr Button,No Single Cup,w/ Grinder$\}$
$\{\{\oslash\}$, AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Pwr Button,No Single Cup,w/ Grinder$\}$
$\{\{\oslash\}$, Temp Control Dial,Pwr Button,No Single Cup,w/ Grinder$\}$
$\{\{\oslash\}$, AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Temp Control Dial,Pwr Button,No Single Cup,w/ Grinder$\}$
$\{\{\oslash\}$, Brew Strength Btn,Pwr Button,No Single Cup,w/ Grinder$\}$
$\{\{\oslash\}$, AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Brew Strength Btn,Pwr Button,No Single Cup,w/ Grinder$\}$
$\{\{\oslash\}$, Temp Control Dial,Brew Strength Btn,Pwr Button,No Single Cup,w/ Grinder$\}$
$\{\{\oslash\}$, AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Temp Control Dial,Brew Strength Btn,Pwr Button,No Single Cup,w/ Grinder$\}$
$\{\{\oslash\}$, Pwr Button,1-4 Cup Btn,No Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Pwr Button,1-4 Cup Btn,No Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, Temp Control Dial,Pwr Button,1-4 Cup Btn,No Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Temp Control Dial,Pwr Button,1-4 Cup Btn,No Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, Brew Strength Btn,Pwr Button,1-4 Cup Btn,No Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Brew Strength Btn,Pwr Button,1-4 Cup Btn,No Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, Temp Control Dial,Brew Strength Btn,Pwr Button,1-4 Cup Btn,No Single Cup,w/o Grinder$\}$
$\{$AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Temp Control Dial,Brew Strength Btn,Pwr Button,1-4 Cup Btn,No Single Cup,w/o Grinder$\}$
$\{\{\oslash\}$, Pwr Button,1-4 Cup Btn,No Single Cup,w/ Grinder$\}$
$\{\{\oslash\}$, AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Pwr Button,1-4 Cup Btn,No Single Cup,w/ Grinder$\}$
$\{\{\oslash\}$, Temp Control Dial,Pwr Button,1-4 Cup Btn,No Single Cup,w/ Grinder$\}$
$\{\{\oslash\}$, AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Temp Control Dial,Pwr Button,1-4 Cup Btn,No Single Cup,w/ Grinder$\}$
$\{\{\oslash\}$, Brew Strength Btn,Pwr Button,1-4 Cup Btn,No Single Cup,w/ Grinder$\}$
$\{\{\oslash\}$, AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Brew Strength Btn,Pwr Button,1-4 Cup Btn,No Single Cup,w/ Grinder$\}$
$\{\{\oslash\}$, Temp Control Dial,Brew Strength Btn,Pwr Button,1-4 Cup Btn,No Single Cup,w/ Grinder$\}$
$\{$AM PM Set,Brew-Delay Set,LCD Clock,Set Hour,Set Minute,Temp Control Dial,Brew Strength Btn,Pwr Button,1-4 Cup Btn,No Single Cup,w/ Grinder$\}\Big\}$

Figure 3.16: Results from Combinatorial Generation Process for the Coffeemaker Product Family

The final action in the generation of the product family is to remove the $\{\oslash\}$ elements from the sets of $Combinations^{all}$. The $\{\oslash\}$ elements can be removed because $\{\oslash\}$ as a whole is being treated as a unique object that can be manipulated by set theory operations. The process to remove these elements is presented in Equation 3.26.

$$ConfSpc = Combinations^{all} \setminus \{\oslash\} \tag{3.26}$$

For the coffeemaker product family the sets in Figure 3.16 would be the same except $\{\oslash\}$ would be removed. While the constrained generation process takes a few more steps, it is necessary to reduce the amount of invalid combinations generated by the process. This generation process accounts for step C in the problem formulation presented in Figure 3.1. This constrained generation process is expanded on in Chapter 4, when the software implementation of the generation process is discussed.

3.3.2 Transition From Configuration Design Space to Layout Space

The layout process is the next step. However prior to laying out components, step D from the problem formulation in Figure 3.1 must be performed. This step is to ”Select Products and Configurations to form a product family”. To do this products will be selected from the set of component and configuration option combinations that remain after constraints have been applied. This list of possible combinations appears in Figure 3.16. The configurations for the coffeemaker product family were selected for this example to simulate a product family being used to reach different market segments and price points. The configurations and products can be selected and defined at the designers discretion. The selected component combinations for the coffeemaker product family example are presented in Table 3.4 along with the names given to the products.

Table 3.4: Coffeemaker Product Definition

Component	Basic	Special	Ultimate
LCD Clock		X	X
Power Button	X	X	X
Set Hour		X	X
Set Minute		X	X
AM PM Set		X	X
Brew-Delay Set		X	X
Temp Control Dial		X	X
Brew Strength Btn			X
1-4 Cup Btn		X	

There were 3 sets of components decided upon to become products. The named products are the ”Basic”, ”Special”, and ”Ultimate” models. Each of these has a unique set of components selected. The configuration options for each product also need to be selected. Table 3.5 displays the selected configurations available for each product.

Table 3.5: Coffeemaker Product Configuration Selection

Configuration Options	Basic	Special	Ultimate
No Single Cup - w/ Grinder		X	
No Single Cup - w/o Grinder	X	X	
Single Cup - w/ Grinder			
Single Cup - w/o Grinder			X

While some of the configuration options are selected for multiple products since each product as a unique component combination, this makes each product completely unique. The selected configuration options show that 4 distinct products have been created. While there are only 3 models of product, since the ”Special” model is available in 2 configurations this means there are 2 unique products under the ”Special” model. Only 3 component layouts need to be produced because the configuration options do not effect the layout of the user interface components being placed.

To aid the transition from the discrete process to the semi-continuous layout process, a matrix of the selected combinations generated by the combinatorial process is formed:

$$\begin{aligned} C_{ij} &= Component\ i\ in\ product\ j \\ i &= 1\ \rightarrow m\ components \\ j &= 1\ \ to\ n\ products \end{aligned} \tag{3.27}$$

The constrained combinatorial process creates an $[n\ \times\ m]$ matrix where m is the number of components being laid out and n is the number of products produced. The i and j indices are permanent labels connected to specific components and products throughout the entire product family design process. C_{ij} is a binary variable to indicate the presence or absence of components within a product, where 1 indicates that a component is present in a product and 0 indicates it is absent. For the Coffeemaker product family, i is the set of identifying numbers given to each component when defined in Table 3.1. This matrix is built by creating a zeros matrix and changing the zeros to once when an object is present in a product.

$$C = \begin{array}{r|ccc|}
 & \text{Basic} & \text{Special} & \text{Ultimate} \\
\text{LCD Clock} & 0 & 1 & 1 \\
\text{Power Button} & 1 & 1 & 1 \\
\text{Set Hour} & 0 & 1 & 1 \\
\text{Set Minute} & 0 & 1 & 1 \\
\text{AM PM Set} & 0 & 1 & 1 \\
\text{Brew-Delay Set} & 0 & 1 & 1 \\
\text{Temp Control Dial} & 0 & 1 & 1 \\
\text{Brew Strength Btn} & 0 & 0 & 1 \\
\text{1-4 Cup Btn} & 0 & 1 & 0
\end{array},$$

Figure 3.17: C - Matrix for coffeemaker product family exmaple

Once all of the models in the product family are created the common architectures are identified along with the components common to the architectures. The available fascia for each product must also be defined. Fascias are areas within products in which components can be placed. This effectively adds bounds to component locations, simplifying the optimization process discussed later on. The fascias for the coffeemaker product family were defined in Table 3.2 and the fascia selection for each component defined in Table 3.1.

Some products can have a single fascia for all UI components (ie ATM, Movie Ticket Machines, Train Ticket Machines) and others have multiple fascia for component layout (ie. Coffeemakers, Cars, Arcade Machines). To have a well-defined, modular product family, common components should be placed in the same location in all products in which the components are present. For products with multiple user interface fascias, this limits common components to specific layout locations, with the advantage of increasing overall commonality across the product family. One of the motivations for commonizing component layouts is to reduce manufacturing costs, since each unique layout within a product family would required a different injection mold or additional unique mounting components. While not having common component placement does not create an infeasible product family, it greatly reduces the commonality and advantage of having a modular product family. For this process if no common fascia is available for a component, the designer should go back to redesign the fascias. In this process it is upto the designer to decide the fascia selection for each component. This will provide a designer greater control over the design process.

3.3.3 Designer Preference Component Layout

For the 2D layout process, a positive X-Y coordinate system is used, with the origin being the bottom left corner of the fascia. The coordinate origin location for the objects being placed is the bottom-left corner for square and rectangular objects or the closest equivalent to the bottom-left for non-rectangular objects. In Table 3.6 the spatial layout grammar is defined and the mathematical formulations are given. These definitions are for the 2D layout of objects. 3D layout would utilize a similar grammar and include mathematical definitions for the Z-axis. Objects “A” and “B” will be used in the table for the representation of the terms. The following variables will also be used as well:

- X_i : X coordinate of object i
- Y_i : Y coordinate of object i
- H_j : Height (Y axis length) of object j
- W_j : Width (X axis length) of object j

Table 3.6: Spatial Constraint Grammar

Grammar Term	Math Equivalence
A Left-of B	$X_A + W_A \leq X_B$
A Right-of B	$X_A \geq X_B + W_B$
A Below B	$Y_A + H_A \leq Y_B$
A Above B	$Y_A \geq Y_B + H_B$
A Top-Aligned B	$Y_A + H_A = Y_B + H_B$
A Bottom-Aligned B	$Y_A = Y_B$
A Left-Aligned B	$X_A = X_B$
A Right-Aligned B	$X_A + W_A = X_B + W_B$
A Horizontal Center-Aligned B	$X_A + \frac{1}{2} * W_A = X_B + \frac{1}{2} * W_B$
A Vertical Center-Aligned B	$Y_A + \frac{1}{2} * H_A = Y_B + \frac{1}{2} * H_B$
A at (X,Y)	$(X_A = X, Y_A = Y)$
A H. Center at X	$X_A = X - \frac{1}{2} * W_A$
A V. Center at Y	$Y_A = Y - \frac{1}{2} * H_A$
A H. Center in Fascia	$X_A + \frac{1}{2} * W_A = X_F + \frac{1}{2} * W_F$
A V. Center in Fascia	$Y_A + \frac{1}{2} * H_A = Y_B + \frac{1}{2} * H_B$

A set of constraints were created to demonstrate the application of the grammar presented in Table 3.6 for the coffeemaker product family. These constraints are presented in Table 3.7. For the 9 components defined for the coffeemaker product it took 26 constraints to reach the desired layout. When defining these types of constraints it is the responsibility of the designer to fully define the set of constraints.

Table 3.7: Cpffeemaker Layout Constraints

Object A	**Constraint**	**Object B/ Location**
LCD Clock	- X centered in fascia -	

Table Continued on Next Page

Continuation of Table 3.7

Object A	Constraint	Object B/ Location
LCD Clock	- Y centered in fascia -	
Set Hour	- below -	LCD Clock
Set Minute	- below -	LCD Clock
Set Minute	- right-of -	Set Hour
Set Minute	- bottom-aligned -	Set Hour
Set Hour	- Vertical Center at Y -	130 mm
AM PM Set	- Y-center aligned -	Set Hour
AM PM Set	- right-aligned -	LCD Clock
Set Minute	- left-of -	AM PM Set
Brew-Delay Set	- left-aligned -	LCD Clock
Brew-Delay Set	- left-of -	Set Hour
Brew-Delay Set	- Y-center aligned -	Set Hour
1-4 Cup Btn	- right-of -	LCD Clock
1-4 Cup Btn	- Y-center aligned -	LCD Clock
Pwr Button	- X centered in fascia -	
Pwr Button	- Y centered in fascia -	
1-4 Cup Btn	- Horizontal Center at X -	150 mm
Brew Strength Btn	- left-of -	LCD Clock
Brew Strength Btn	- Y-center aligned -	LCD Clock
Set Hour	- Horizontal Center at X -	95 mm
Set Minute	- Horizontal Center at X -	110 mm
Brew Strength Btn	- Horizontal Center at X -	65 mm
Temp Control Dial	- X-center aligned -	LCD Clock
Temp Control Dial	- above -	LCD Clock
Temp Control Dial	- Vertical Center at Y -	170 mm

End of Table

Linear programming is a method of optimization for a mathematical model that is represented by linear relationships. Dantzig's Simplex algorithm [36] is one of the most popular linear programming methods. The Simplex method is a linear optimization method used to minimize or maximize the value of an objective function based upon a series of linear equality and inequality constraints. Simplex uses a set of n real variables, constrained to be non-negative. For m constraints and n variables, Simplex follows the standard form of:

$$
\begin{aligned}
&Minimize\; f\left(x_1,\ x_2, \ldots,\ x_n\right) =\ c_1 x_1 + c_n x_2\ + \cdots + c_n x_n \\
&\quad Subject\; to: \\
&\qquad a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n = b_1 \\
&\qquad a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n = b_2 \\
&\qquad a_{3,1}x_1 + a_{3,2}x_2 + \cdots + a_{3,n}x_n = b_3 \\
&\qquad \vdots \\
&\qquad a_{m,1}x_1 + a_{m,2}x_2 + \cdots + a_{m,n}x_n = b_m \\
&\qquad\qquad x_1, x_2, \ldots, x_n \geq 0
\end{aligned}
\tag{3.28}
$$

Variables for the component locations and dimensions need to be defined. The location and dimension variables are defined using the same i indexes created in the combinatorial process. The location variables are defined with:

$$X_i = X \text{ location of component } i \tag{3.29}$$

$$Y_i = Y \text{ location of component } i \tag{3.30}$$

$$i = 1 \text{ to m components}$$

This creates two m length vectors that define the x and y positions of each component. This allows for easy indexing for final layout of components. For example the location of component 2 have the location vector of $(X_2,\ Y_2)$.

The dimensions of the components are defined in a similar fashion to the location variables. Component dimensions are defined by:

$$W_i = X \text{ dimension of component } i \tag{3.31}$$

$$H_i = Y \text{ dimension of component } i \tag{3.32}$$

$$i = 1 \text{ to m components}$$

This defines two m length vectors where W_i contains the component widths and H_i contain the component heights. These column vectors will be used in the component layout calculations.

Information about the fascias must also be defined. Similar notation is used to define the dimensions and locations of the fascias, except the fascia locations and dimensions will be fully defined prior to components being placed within them. Since there may be multiple fascias for components to be placed in, it is important to have a notation that clearly defines all of the separate fascias. The locations of the fascias are defined by:

$$FX_k = X \text{ location of fascia } k \tag{3.33}$$

$$FY_k = Y \text{ location of fascia } k \tag{3.34}$$

$$k = 1 \rightarrow p\ fascias$$

Similar to the component locations, the fascia locations are modeled using two $[n \times p]$ matrices containing the x and y locations of the fascias. The dimensions of the fascia have a very similar notation applied. The fascia dimensions are used in the definition of the bounds of the fascias to define bounds for the components being arranged. The fascia notation is

$$FW_k = X \text{ dimension of fascia } k \tag{3.35}$$

$$FH_k = Y \text{ dimension of fascia } k \tag{3.36}$$

$$k = 1 \ \rightarrow p \ fascias$$

Similar to the component dimensions, two $[p \times 1]$ column vectors represent the dimensions of the various fascias. All of these definitions are defined in such a way to aid in the calculation of the component locations.

The constraints for the entire product family are defined according to the constraint grammar and a series of equations is produced. For the purpose of solving the equations using the simplex method the constraints from Eqn. 3.28 must be converted to the matrix form of:

$$Ax = b \tag{3.37}$$

$$or$$

$$Ax \leq b \tag{3.38}$$

where A is the variable coefficient and b contains the constants for the constraints. Eqn. 3.37 can be expanded as:

$$\begin{bmatrix} A_{1,1} & \dots & A_{1,v} \\ \vdots & \ddots & \vdots \\ A_{c_n,1} & \cdots & A_{c_n,v} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_v \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_{c_n} \end{bmatrix} \tag{3.39}$$

$$v = (2 * m) \tag{3.40}$$

where v is the total number of variables in the product family and c_n is the number of constraints defined. If the product family contains variations of the same component (i.e.

credit card slots, receipt printers, keypads), each of these must also be fully defined within the product family. Since variations of the same component will never appear in the same model it is advised that the constraints for all of these components be the same, such that they can be nearly interchangeable depending on the model being produced and to minimize the impact of the layout throughout the product family.

Some of the variable matrices defined in previous steps will need to be concatenated to create the matrix to represent the x variable from Eqn. 3.37 and 3.38 and combine the constraints in the X and Y axis. This is because the matrices were created from the terms introduced in Eqns. 3.29, 3.30, 3.33, and 3.34. The matrix manipulations are shown below:

$$x = \begin{bmatrix} X_{vars} \\ Y_{vars} \end{bmatrix} \tag{3.41}$$

$$X_{vars} = \left(diag\left(C_{\forall i,j}\right) * X_{\forall i,\ j}\right)^T \tag{3.42}$$

$$Y_{vars} = \left(diag\left(C_{\forall i,j}\right) * Y_{\forall i,\ j}\right)^T \tag{3.43}$$

This produces a column vector that contains the variables for all of the components contained within the product being laid out. While the calculation for x will zero the variables for components not contained within the product, it will not remove the constraints. If the constraints are removed along with the variables the optimization process can result in no common layout being found.

The final part required to start the optimization process is an objective function. Since the goal of this process is feasibility, the objective of the optimization process is to minimize constraint error. Constraint error is the deviation between the constraint value and its target value. For an equality constraint this can be a positive or negative deviation, while for inequality the constraint error is either positive or negative. This allows for the measurement of error in the optimization process. The objective function for this is defined as:

$$min\, f = \sum_{i=1}^{c} |e_i| \tag{3.44}$$

where e_i is the error of constraint i. This sums the absolute value of all of the errors from the constraints for a general minimization problem with a known global minimum of zero (0). This means a completely feasible product family will have an objective function value of zero, and all of the constraints would be satisfied. Since simplex is a linear optimization process an absolute value cannot be used in the objective function. A quasi-linear adaptation based upon the Cassowary algorithm [30] is used. The error must be measured by a pair of deviation variables that are zero when the constraint is satisfied and a non-zero real number when a constraint is not satisfied. The deviation variables $\delta_i^+ \; and \, \delta_i^-$ are added to the end of each constraint to calculate deviation:

$$Ax_i + \; \delta_i^+ - \; \delta_i^- = b \tag{3.45}$$

Since each deviation variable only appears in a single constraint they will remain zero unless the constraint cannot be satisfied. Finally a constraint needs to be added that bounds components to only be placed within the fascia. The general form of this constraint is

For i = 1 → m where n = #of layout components in fascia k

$$\begin{aligned} X_i \; &\geq \; FX_k \\ X_i \; &\leq \; FX_k \; + \; FW_k \; - \; W_i \\ Y_i \; &\geq \; FY_k \\ Y_i &\leq FW_k + FH_k - H_i \end{aligned} \tag{3.46}$$

However even if all the constraints are satisfied a layout may not be feasible if the system is under constrained and there are components that overlap. Along those lines, a layout may not be completely infeasible if the components do not overlap and are within the specified fascia. Since the Simplex method is a linear optimization method, it is not possible to add a constraint to minimize component overlap since the overlap calculation involves non-linear relationships. Determination of overlap must occur after the use of the Simplex method. Overlap is check for each product in the product family. Overlap is only a problem if two components occur within the same product. Otherwise components may occupy the same space in the overall product family since overlap would not occur in a product. This ensures components are able to maintain the same location in every product they are present in, but not force blank spaces to be present in products that do not contain the component. To determine component overlap, the calculated component X and Y locations along with the component dimensions are utilized to create a polygon map and search for intersection of components. The polygon map places components in the fascia. The intersection is determined by checking the polygon map to check if any area is fully or partially occupied by the same components.

Algorithm 3.1 Overlap Calculation

```
n ← Number of Products Defined
for $i \leftarrow 1, n$ do
    m ← Number of Components in product i
    for $j \leftarrow 1, m$ do
        for $k \leftarrow i - 1$ do
            if $intersection\ (component_j, component_k) > 0$ then
                display $component\ overlap\ message\ for\ j\ and\ k\ inproducti$
            end if
        end for
    end for
end for
```

This process determines which components overlap and states which components will overlap. If none of the components overlap and all constraints are satisfied, the entire developed product family is valid. If component overlap is detected from the optimization process the current product is infeasible. Utilizing this information the designer can re-evaluate decisions made to either the configuration of the product family or the constraints in component layout and re-solve the problem to achieve feasibility. It will be up to the designer to decide on which changes should be made to reach feasibility, but the information provided by this method informs them of the problems they may encounter.

The full mathematical problem formulation for the component layout process creates a specialized simplex problem for the layout of components with the goal of feasibility. This formulation is:

$$\begin{aligned}
&Minimize\ f\left(x_1,\ x_2, \ldots,\ x_n\right) = \delta_1^+ - \delta_1^- + \delta_2^+ - \delta_2^- \ldots . +\ \delta_c^+ -\ \delta_c^- \\
&\qquad Subject\ to: \\
&\qquad a_{1,1}x_1 + a_{1,2}x_2\ + \cdots + a_{1,n}x_v + \delta_1^+ - \delta_1^- = b_1 \\
&\qquad a_{2,1}x_1 + a_{2,2}x_2\ + \cdots + a_{2,n}x_v + \delta_2^+ - \delta_2^- = b_2 \\
&\qquad a_{3,1}x_1 + a_{3,2}x_2\ + \cdots + a_{3,n}x_v + \delta_3^+ - \delta_3^- = b_3 \\
&\qquad \vdots \\
&\qquad a_{c_n,1}x_1 + a_{c_n,2}x_2\ + \cdots + a_{c_n,n}x_v + \delta_{c_n}^+ - \delta_{c_n}^- = b_{c_n} \\
&\qquad x_1, x_2, \ldots, x_n, \delta_1^+, \delta_1^-, \ldots, \delta_c^+, \delta_c^- \geq 0
\end{aligned} \tag{3.47}$$

where c_n is the number of defined constraints. v is the total number of location variables for the components as calculated in Eqn. 3.40, along with the bounding constraints defined in Eqn. 3.46. The constraint equations can be rewritten in matrix form for calculation.

$$A_{c,m}x = b_c \tag{3.48}$$

Where x is defined in Eqns. 3.41, 3.42, and 3.43. The $A_{c,m}$ and b_c coefficients are defined by the constraints for each axis but need to be concatenated for optimization:

$$A_{c,m} = \begin{bmatrix} A_x & zero\,(C_x,\ m) \\ zero\,(C_y,\ m) & A_y \end{bmatrix} \tag{3.49}$$

$$b_c = \begin{bmatrix} b_x \\ b_y \end{bmatrix} \tag{3.50}$$

Figure 3.19, at the end of the chapter presents the A matrix and the b vector with the constraints defined for the coffeemaker product family. Notice that the A does not have any of the δ_c^+ or δ_c^- listed. This is because there are would be 52 additional columns in the table and it is too large to fit on a single page. However the Objective function for the coffeemaker product family example is presented in Figure 3.18.

$$\begin{aligned} Minimize\ f\,(x_1,\ x_2,\ldots,\ x_m) =&\delta_1^+ + \delta_1^- + \delta_2^+ + \delta_2^- + \delta_3^+ + \delta_3^- + \delta_4^+ + \delta_4^- + \delta_5^+ + \delta_5^- + \delta_6^+ + \delta_6^- \\ &+\delta_7^+ + \delta_7^- + \delta_8^+ + \delta_8^- + \delta_9^+ + \delta_9^- + \delta_{10}^+ + \delta_{10}^- + \delta_{11}^+ + \delta_{11}^- + \delta_{12}^+ + \delta_{12}^- \\ &+\delta_{13}^+ + \delta_{13}^- + \delta_{14}^+ + \delta_{14}^- + \delta_{15}^+ + \delta_{15}^- + \delta_{16}^+ + \delta_{16}^- + \delta_{17}^+ + \delta_{17}^- + \delta_{18}^+ + \delta_{18}^- \\ &+\delta_{19}^+ + \delta_{19}^- + \delta_{20}^+ + \delta_{20}^- + \delta_{21}^+ + \delta_{21}^- + \delta_{22}^+ + \delta_{22}^- + \delta_{23}^+ + \delta_{23}^- + \delta_{24}^+ + \delta_{24}^- \\ &+\delta_{25}^+ + \delta_{25}^- + \delta_{26}^+ + \delta_{26}^- \end{aligned}$$

Figure 3.18: $Confs$ example for the coffeemaker product family

Figures 3.18 and 3.19 present the inputs into the simplex optimization process. The components for the coffeemaker product family can now be laid out. The constraints and objective function produce a set of coordinates for the components. These results are presented in Table 3.8. Note that the coordinates are from the lower left corner of the bounding box of the components.

Table 3.8: Coffeemaker Product Layout results

Components	Width	Height	X	Y
LCD Clock	60	20	70	140
Pwr Button	20	20	90	5
Set Hour	10	10	90	125
Set Minute	10	10	105	125
AM PM Set	5	5	125	127.5
Brew-Delay Set	10	10	70	125
Temp Control Dial	10	10	95	165
Brew Strength Btn	7	7	61.5	146.5
1-4 Cup Btn	7	7	146.5	146.5

Laying out the components completes step E of the problem formulation presented in Figure 3.1. The next step is to determine if any of the components overlap spatially, Step F from Figure 3.1. Following the procedure presented in Algorithm 3.1, it was determined that none of the components overlap. This was determined by taking the component compositions of each product and check each component against the others for overlap. Overlap can be checked in a variety of ways from sketching out the components to 'see' if there is overlap to performing a calculation to mathematically determine if there is overlap present. The last step of minimizing the number of unique components, Part G. This step can performed in a variety of ways. The simplest is eliminating unique fascias and mounting components for the components in the user interface. This is the final step and concludes the coffeemaker product family example. This aided in the explanation of the various parts of the Configuration-Layout Method presented in this chapter.

3.4 Summary

This chapter presented the Configuration-Layout Design Problem and the method developed for this thesis to address it. To aid in the demonstration of the methods this chapter presents, an example for the development of a coffeemaker product family was presented. The methods presented provide adequate detail for use with other product families. This fully defines the optimization process for easy implementation into an

$$
A = \begin{array}{|cccccccccccccccccc|}
\text{LCD Clock-X} & \text{Power Button-X} & \text{Set Hour-X} & \text{Set Minute-X} & \text{AM PM Set-X} & \text{Brew-Delay Set-X} & \text{Temp Control Dial-X} & \text{Brew Strength Btn-X} & \text{1-4 Cup Btn-X} & \text{LCD Clock-Y} & \text{Power Button-Y} & \text{Set Hour-Y} & \text{Set Minute-Y} & \text{AM PM Set-Y} & \text{Brew-Delay Set-Y} & \text{Temp Control Dial-Y} & \text{Brew Strength Btn-Y} & \text{1-4 Cup Btn-Y} \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0
\end{array}, \qquad
b = \left|\begin{array}{c}
70 \\ 140 \\ 0 \\ 125 \\ 2.5 \\ 55 \\ 0 \\ 0 \\ 6.5 \\ 90 \\ 5 \\ 146.5 \\ 6.5 \\ 90 \\ 105 \\ 61.5 \\ 25 \\ 165 \\ -10 \\ -10 \\ -10 \\ -10 \\ -10 \\ -60 \\ -7 \\ -20
\end{array}\right|
$$

Figure 3.19: The A and b matrices without the deviation variables for the coffeemaker product family example.

automated process.

These concepts work to address the research questions presented in Chapter 1. The first research question on capturing spatial relationships and its hypothesis are addressed in this chapter. The set of layout constraints provided in Table 3.6 presents a layout grammar used to describe basic layout constraints and also provides their mathematical equivalence. This shows that a spatial grammar is feasible for determining component Layouts. The second research question of determining the effects of design decisions and its hypothesis addressed throughout the whole chapter. The hypothesis is a addressed in the development of the constrained combintorial generation process and the simplex layout method. The combined combinatoric and linear processes show how removing of infeasible products and defining a full product family at the same time can simplify the layout of components.

The next chapter will present the software developed to present the methods presented in this chapter. The same coffeemaker product family example is presented in the next chapter to demonstrate the software implementation.

CHAPTER 4

CONFIGURATION-LAYOUT METHODS IMPLEMENTATION

To properly present the advantage of the methods introduced in this thesis an example of its implementation into code and the product family design workflow is required along with case studies. The implementation produced for this thesis is a GUI based program built using Matlabs App Designer add-on. In this chapter an overview of the developed software, its data structures, capabilities, and workflow are discussed.

This software is capable of actively displaying the effects of the addition and removal of constraints on both combinations of components and desired layout of components within a product. This allows for designers to determine the effects of desired characteristics of the product family in development and go back and make changes on-the-fly to achieve the desired results. The configuration constraints used The code written for this software is capable of configuring product families with large numbers of components and constraints while maintaining efficient use of memory by not requiring full enumeration to determine the effects of constraints on the product family. The system also allows designers to determine locations for components common to multiple members of the product family and find locations for unique component while determining if the layout desires of the designer are feasible allowing for conceptual design to be completed and the early stages of embodiment design to be initiated.

This chapter presents the software developed for this thesis and provides background on its development. In section 4.1 the data structures and overall organization of information for the program is presented. Section 4.2 provides an overview of the functions written for the program and describes their behavior. The full GUI and operation of the program is presented in section 4.3 to display its capabilities. Section 4.4 presents a demonstration of the software for the example of the coffeemaker product family. Finally section 4.5 presents an overview of the chapter.

4.1 Data Structures

When implementing algorithms and methods into code it is important to develop logical and coherent data structures. This allows data to be easily indexed and segregates data as to remove complexity of indexing. The methods and algorithms presented in this thesis encompass a large number of data types, which are addressed at different times during the process. Since one of the advantages of a software implementation of these methods is to actively display the effects of changes made to the product family, all inputs and results must be stored. This means the data structures must be readily available for indexing and not highly impact memory using on the computer. To address these the data in this implementation is divided in to in to five main structures. These structures are sorted by processes they pertain to and the data type. The structures are listed below along with their dot notation reference in parenthesis:

- Components and Configurations (comp)
- Fascias (fascia)
- Constrained Configuration (confcons)
- Defined Products (confprod)
- Component Layout (lay)

The following subsections will detail the contents of the structures and reasons for their organization. This will provide an initial insight into the organization and function of the code.

4.1.1 Components and Configurations

This structure contains all the input and output information for all the components and configuration possibilities. In this case components are considered unique objects that have volume and are involved in both the constrained configuration and the product layout. Configurations mostly exist in concept, while they are able to impact the physical design of products they do not take a singular 2D/3D form. This definition is required because some of the fields of this structure refer to size and location. Both types of objects are included in this structure since they are both involved in the constrained combinatoric process are easier to call for calculation when from a single structure. To account for this, there is a field used to indicate whether an object is a component or configuration. Table 4.1 shows all of the available fields, their data type, and what they represent. The structures would be referenced by calling $comp.F$ in the code where F is the name of the field.

Table 4.1: Fields within Comp Structure

Field	Data Type	Description
$.name$	String	This field contains the name of the object being defined. No duplicate names are allowed because the name of the object is used during indexing to pull data about a particular object. Names cannot be changed after object creation.
$.type$	Integer	This defines the type of object being created. This is either a 1 or a 2. A 1 indicates a component and 2 a configuration. When indexing this field can be used to call only components or configurations. This is defined at object creation and cannot be changed.
$.category$	String	This field defines the category or type of configuration a configuration falls into. This is defined during object creation.
$.width$	Double	This field contains the width or x dimension of a component. This is initially zero but can be changed at anytime throughout the process.
$.height$	Double	This field contains the height or y dimension of a component. This is initially zero but can be changed at anytime throughout the process.
$.x$	Double	This is the x-location of an object from the origin to its lower left corner. This can be fully defined during the layout stage, but is set to zero until fully constrained or defined during the layout operation.
$.y$	Double	This is the y-location of an object from the origin to its lower left corner. This can be fully defined during the layout stage, but is set to zero until fully constrained or defined during the layout operation.
$.fasCh$	String	This field contains the designer chosen fascia for each component. This field must be defined for all components or the layout process cannot proceed. This can be changed at any time.

While all of the fields in the structure do not apply to either type of object enter-able into the structure, the ability to call all objects used in the combinatoric configuration portion of the application from a single structure provides the advantage of easier access. Since the unused fields of the structure remain empty when unused the impact on memory is low. Since this structure utilizes an element-by-element organization and all of the objects have a unique name, the name portion of the structure can be used to find the index location of a component to access the data connected to it. When the indexing location of a component is known, all the information connected to that component can be accessed with $comp(n)$ where n is the index location. The information entered into this structure is used through out the entire process. Figure4.1 is a visual representation of the structure used.

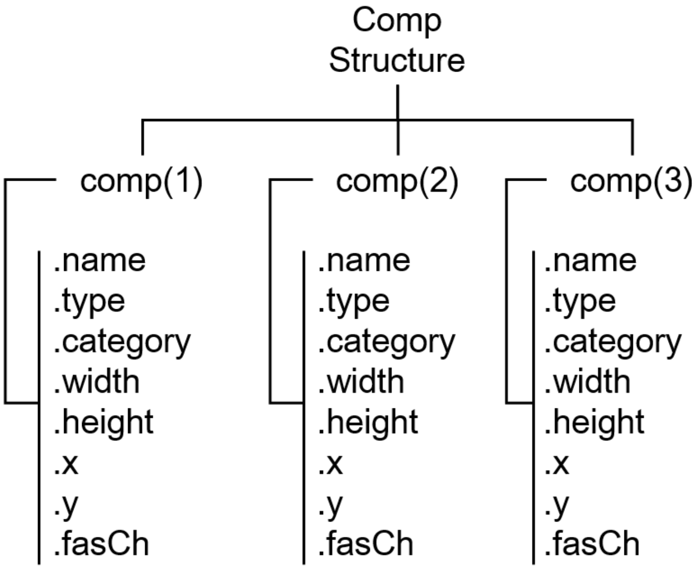


Figure 4.1: comp Structure Visualization

4.1.2 Fascias

The data in this structure is the information relating to the fascias of the product family being developed.Since the fascias are set prior to the layout process, only information relating to the layout process is required. This structure is similar to the $comp$ structure but all fields must be filled prior to the layout process as they effect the fascias are the bounds on the simplex operation. For the structure $fascia$ the existing fields appear in Table 4.2.

Table 4.2: Fields within Fascia Structure

Field	Data Type	Description
$.name$	String	This field contains the name of the object being defined. No duplicate names are allowed because the name of the object is used during indexing to pull data about a particular object. Names cannot be changed after object creation.
$.width$	Double	This field contains the width or x dimension of a component. This is initially zero and must be changed prior to the layout process.
$.height$	Double	This field contains the height or y dimension of a component. This is initially zero and must be changed prior to the layout process.
$.x$	Double	This is the x-location of an object from the origin to its lower left corner. This is initially zero and must be changed prior to the layout process.
$.y$	Double	This is the y-location of an object from the origin to its lower left corner. This is initially zero and must be changed prior to the layout process.

Similarly to the $comp$ structure the information relating to a particular fascia by determining the index based on the name of the fascia. The fascias must be defined prior to the layout process because the components require an initial spatial envelope for which the components can be placed. These bounds can later be changed but must be initially present for the components to properly be placed. However, due to the fluid nature of the software changes to the size and location of the fascia can be changed between layout operations as the designer determines the effects of design choices. There is no wasted memory since all fields must be used. Figure 4.2 displays the organization of this structure.

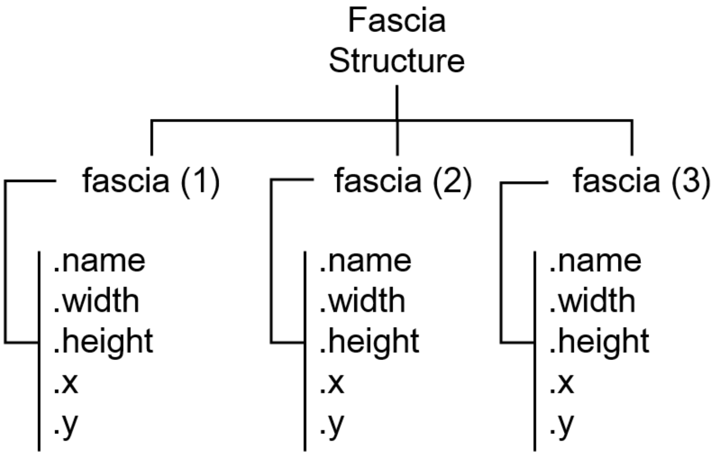


Figure 4.2: fascia Structure Visualization

4.1.3 Constrained Configuration

The constrained configuration (confcons) structure contains all of the information relating to the process of generating product configurations based on combinatoric constraints. This structure contains both user input and data generated by the code. The results of each step in the generation process is kept as the designer may not require the full enumeration of all possible products based on the constraints. The fields of his structure take advantage of the dot notation structures ability to handle multiple types of data containers. The fields in this structure consist of a mixture of matrices, cell arrays, vectors, and string arrays. This structure utilizes a mixture of plane and element-by-element organizations.

Table 4.3: Fields within Confcons Structure

Field	Data Type	Description
$.ReqComps$	Cell of Strings	This field contains the names of all components the designer decides are required for any product to be feasible. This is a single cell containing multiple entries.
$.ReqPairs$	Cell Array of Strings	This field contains contains all of the required pair constraints. Each cell contains the names of two components consisting of the required pair.
$.DisPairs$	Cell Array of Strings	This field contains contains all of the disallowed pair constraints. Each cell contains the names of two components consisting of the disallowed pair.
$.availconf$	Cell Array of Strings	This is is all possible combinations of configurations based upon the constraints on combinations, where each cell in the array represents a feasible configuration. Each cell contains one configuration option from each configuration category.
$.cnftabl$	Table of Mixed Data	This table contains the available configurations after constraints are applied and is organized for use in the GUI
$.NumProd$	Integer	This field contains the calculated number of feasible products after configuration constraints are applied.
$.objlbls$	Cell Array of Strings	Contains all component and configuration names
$.allttbls$	Binary Matrices	These are the binary matrices of the constrained products divided by sets of components connected via constraints.
$.seplbls$	Cell Arrays of Strings	These are the name of all components and configurations divided into cells of objects connected by constraints
$.septbls$	Cell Arrays of Strings	This contains the combined $allttbles$ and $seplbls$ fields. This is equivalent to the $ConstrainedSets$ variable presented in Eqation 3.22.
$.ProGen$	Cell Arrays of Strings	This contains the full enumeration of all possible products with constraints applied. The enumeration only exists if the designer chooses to generate it.

As different as this structure is from the previous ones it is because it contain different types of information including the inputs to the combinatoric and intermediate results. While the product enumeration will consume the largest amount of data in this structure, the enumeration field will remain empty unless the user explicitly decides to view the full enumeration. The details of this will be outlined in section 4.2 when the code is discussed. This structure attempts to conserve data by only fill fields when required. Figure 4.3

visualizes this structure. Notice that this structure contains a mixture of structure organization types. Where some parts of the structures contain singular elements and others are ordered.

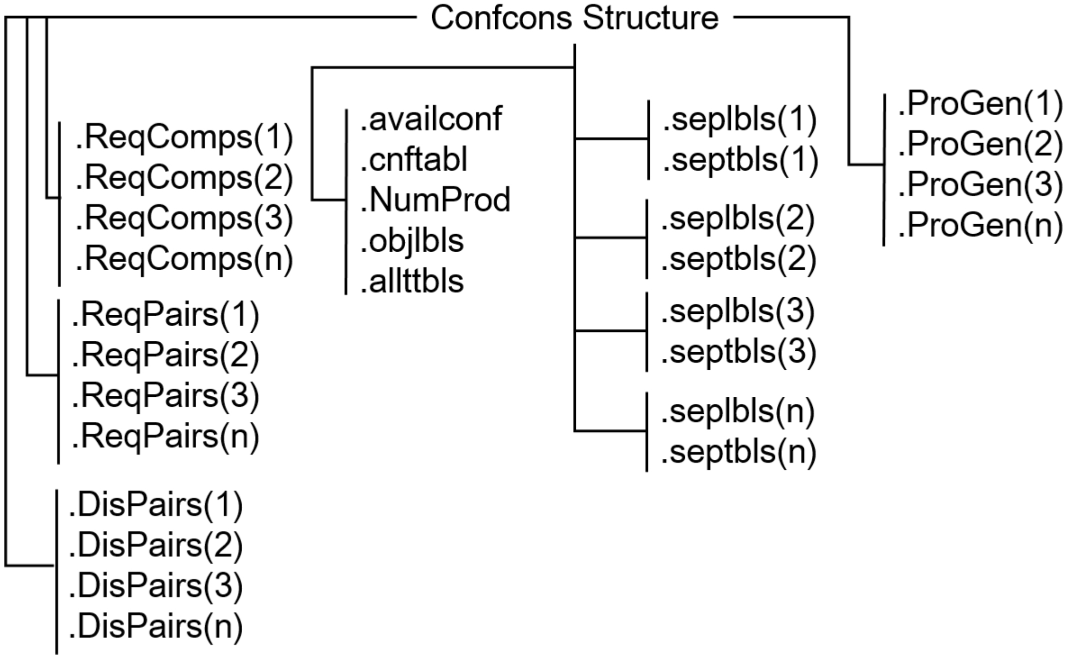


Figure 4.3: ConfCons Structure Visualization

4.1.4 Defined Products

This portion of the data structure contains the information connected to all products specified by the designer. Each product only requires a small amount of information to be defined since other structures contain the details of the individual components. This structure is organized element-by-element so that every index location ($confprod$(x).F), so the pertinent information for every product is easily accessible. Table 4.4 details the fields of this structure.

Table 4.4: Fields within Confprod Structure

Field	Data Type	Description
$.SpecName$	String	This field contains the name of the object being defined. No duplicate names are allowed because the name of the object is used during indexing to pull data about a particular object. Names cannot be changed after object creation.
$.ProdDef$	Cell Array of Strings	This contains all of the components the designer has decided belong in a specified product.
$.ConfgList$	Cell Array of Strings	This contains all of the possible configurations a product will be available in as defined by the designer.
$.Cnnctns$	Cell Array of Strings	This field contains any specified connections that occur between components.

As the table shows, this structure is quite small, but contains the products the designer decides will make up the product family. Figure 4.4 shows an example of how a product would be defined in the data structure. Notice how it follows a similar structure to the comp and fascia structure.

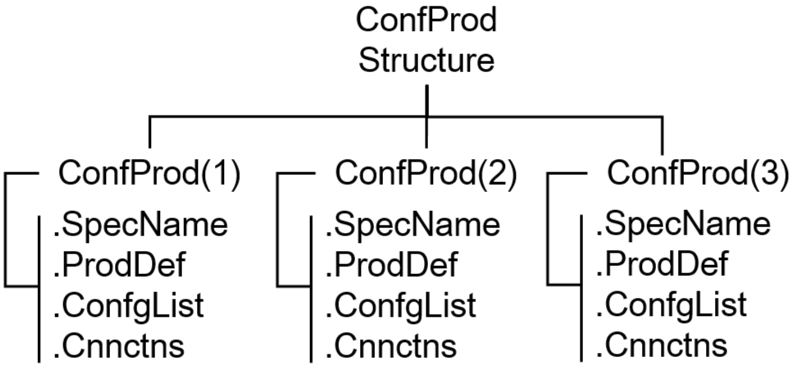


Figure 4.4: ConfProd Structure Visualization

4.1.5 Component Layout

The final data structure contains all the relevant information for the component layout process. Since the the layout process utilizes the simplex method, all of the components to set up the optimization process as well as components used in the GUI of the program. One difference between this structure and the rest is one of the fields contains a substructure. The constraints field contains a substructure because the constraint names need to be held in separate fields for when the parts of the simplex method are created so the individual dimensions and equations needed to create each constraint can be taken from the proper index. Table 4.5 shows the basic structure of the layout structure.

Table 4.5: Fields within Lay Structure

Field	Data Type	Description
$.Constraints$	Substructure	This is the substructure for the layout constraints.
$.Aeq$	Matrix of Numbers	This is the matrix containing the equality constrain coefficients.
$.Aneq$	Matrix of Numbers	This matrix contains the coefficients for the inequality constraints.
$.ObjectiveFcn$	Vector of Numbers	This vector represents the variable coefficients of the objective function. As described in Chapter 3 all coefficients except the deviation coefficients will be zero.
$.beq$	Vector of Numbers	This vector contains the numbers on the right side of the equations for the equality constraints
$.bneq$	Vector of Numbers	This vector contains the numbers on the right side of the equations for the inequality constraints.
$.lb$	Vector of Numbers	These are the lower bounds for all variables in the simplex operation. For components this is determined by the fascia chosen, for deviation it is zero.
$.ub$	Vector of Numbers	These are the upper bounds for all variables in the simplex operation. For components this is determined by the fascia chosen, for deviation it is infinity.
$.results$	Substructure	These are the results of the simplex optimization. Including the final objective function value and variable values.
$.overlap$	Cell Array	This contains information on whether any components overlap since the simplex method is incapable for determining it.

Almost no strings or cell arrays are required in this structure because the indexing within the matricies is set based on the index locations of the components in the GUI. Since this cannot be changed the order has already been set. Table4.6 provides information on the Constraints substructure.

Table 4.6: Fields within Constraints Sub-Structure

Field	Data Type	Description
$.FirstComp$	String	Name of the first component in the constraint.
$.SecondComp$	String or Double	Name of the second component in the constraint or a location.
$.Constraint$	Strings	This is the the name of the constraint being defined for these components. This is used along with a switch case to create the proper elements of the simplex operation.

Since most of the data in this structure is in the form of matrices and vectors of numbers there should be a low impact on memory. Especially considering that the sized of the matrices is directly tied to the number of components. Figure 4.5 presents the fields of the structure in a visualization of their organization. Notice how each of the .Constraint fields has a substructure containing three fields to represent the layout constraints.

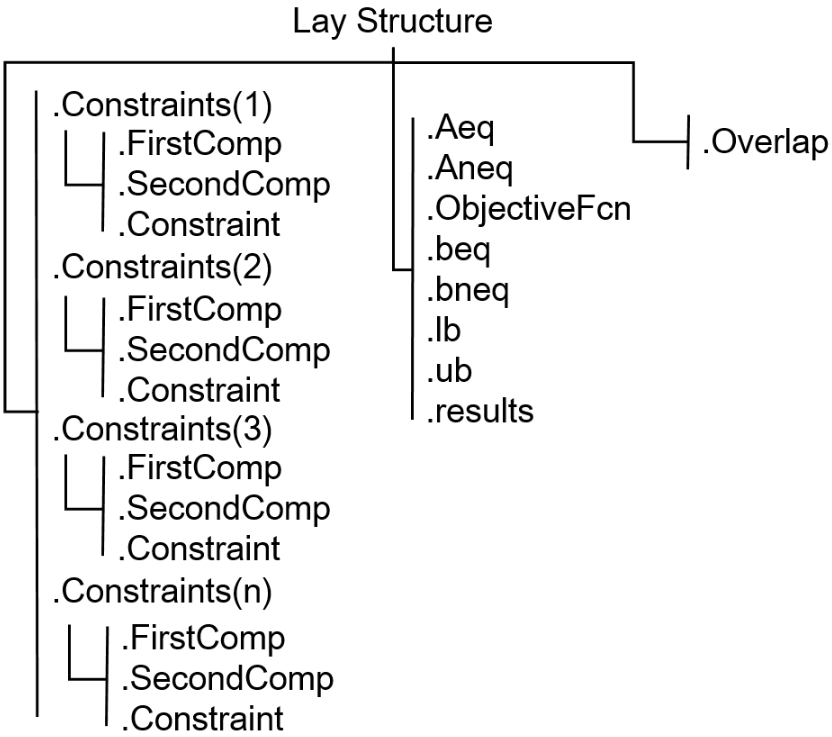


Figure 4.5: Lay Structure Visualization

4.1.6 Summary

Due to the number of data types this software handles through out its operation the data structures had to be well thought out and designed for ease of indexing. This lead to the structures presented in this section being developed; allowing for quick indexing and the ability for the user to go back and make changes at any step of the process. In the next section an overview of the code developed to accomplish the main goals (Constrained Configuration and Component Layout) of this software is given. This will show why the data structures for this program were developed in the fashion shown here.

4.2 Code Overview

The previous section described how the data was organized for this software, in this section will provide an overview of the main processes this code uses to both configure product families and layout components. Since this software was written in Matlab code, a more general description of the processes are provided for easier implementation into other languages. These will be described with a combination of written explanations, pseudo-code, and explain where the math methods presented in the previous are implemented. The written explanations will also include the reasoning behind some design decisions and the pseudo-code will provide a better insight into the implementation.

4.2.1 Combinatoric Generation Process

The code for the combinatoric process was developed to quickly and memory efficiently number and generate feasible products. This code was written based upon the Equations and methods presented in Section 3.3.1.2. To achieve this performance a set of objectives for the code were defined:

- Simplify generation process
- Not require full enumeration

- Minimize number of steps to determine feasibility of a product

The code was developed using these objectives as a guide. The initial step in the development of the algorithms for the generation of feasible products was to breakdown the main problem down into a series of steps to simplify the process. The purpose of these steps is to take the knowledge gained from the study of this problem and the effects of constraints to create a simple process. The generation process was divided into the following steps:

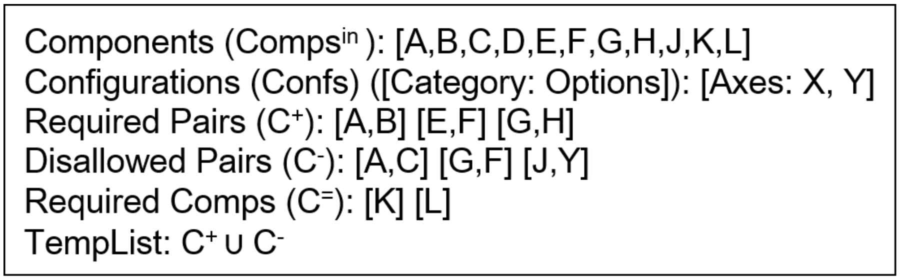
1. **Organize** constraints into groups based on the type of constraints, objects and possible interactions between constraints

2. **Evaluate** groups of constraints and generate groups of feasible components

3. **Calculate** the number of feasible products based on the smaller groups of components

4. **Generate** sets of feasible combinations of components to create products

These steps allowed for simplification of the larger task during algorithm development and code implementation. The following sections will present the the methods developed to complete each step. A small example is used to better demonstrate the actions of each step. The objects and constraints presented in Figure 4.6 are the inputs for this example.

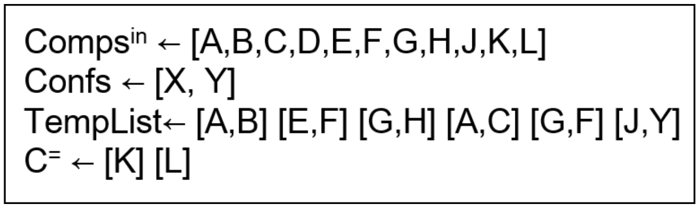


Components ($Comps^{in}$): [A,B,C,D,E,F,G,H,J,K,L]
Configurations (Confs) ([Category: Options]): [Axes: X, Y]
Required Pairs (C^{+}): [A,B] [E,F] [G,H]
Disallowed Pairs (C^{-}): [A,C] [G,F] [J,Y]
Required Comps ($C^{=}$): [K] [L]
TempList: $C^{+} \cup C^{-}$

Figure 4.6: Inputs for Combinatoric Example

4.2.1.1 Organize Constraints

In order to fully identify the the interaction between combinatoric constraints and their overall effect on the system, it is necessary to group together sets of constraints which will interact with each other. This process is code implementation of the processes that take place in Equations 3.15 to 3.20. Where Required Components and Components connected via constraints are removed from the main set of components and put into separate sets. It was observed during development that objects which appear in multiple constraints effect the behavior of the generation process. all objects directly or indirectly connected by constraints must be grouped together for evaluation. While this is a simple task it becomes more complicated when objects may not be directly connected by the same component. This leads to possible chains of constraints that need to be grouped together. The other grouping is any object or set of objects that are connected to a configuration by constraints must be grouped together with all configurations since configurations are internally constrained as well.



Comps^{in} ← [A,B,C,D,E,F,G,H,J,K,L]
Confs ← [X, Y]
TempList← [A,B] [E,F] [G,H] [A,C] [G,F] [J,Y]
C$^{=}$ ← [K] [L]

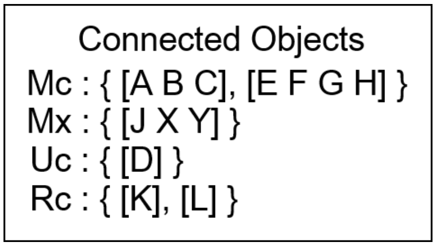
Figure 4.7: Input Variables Defined.

Algorithm 4.1 provides pseudocode to show how this task was completed via the code. The inputs of this algorithm are the list of Components ($Comp^{in}$), list of Configuration Options ($Confs$), list of Constraint Pairs ($TempList$), and list of Required Components ($C^{=}$). These input variables are the same as those presented in Chapter 3. At the end of this process constraint groups and singular constraints containing only components will be in the MultiConstraint List represented by Mc in the algorithm. Each element of Mc represents a single constraint or group of constraints that are connected. Groups of constraints that contain a mixture of both components and configuration options are placed into in the MixedConstraint,Mx in the algorithm. This is because configurations are already internally constrained to prevent two or more configurations of the same type from appearing in a product, thus any constraint containing a configuration effects all other configurations. Finally the list of components that don't appear in any constraint is put into the Unconstrained Component list Uc. Finally any constraint defined as required is put into the $C^{=}$ list. The data for Mc, Mx, Uc and $C^{=}$, is in the form of cell array containing component names so for Mc(x) contains the names of a set of components connected by constraints. The union of Mc and Mx ($Mc \cup Mx$) would be equivalent to $ObjectGroups$ from Chapter 3. Uc is equivalent to $Comps^{in}$ with the components that appear in constraints have been removed.The variables Mc, Mx, Uc and $C^{=}$ will be used in the other algorithms in this section.Figure 4.7 displays the contents of the inputs of Algorithm 4.1, $Comps^{in}$, $Confs$, $TempList$, and $C^{=}$ for the running example presented in Figure 4.6.

Algorithm 4.1 Constraint Organization Procedure

```
1: $TempList \leftarrow$ Set of all Constraint Pairs (Required Pairs and Disallowed Pairs)
2: $Comps^{in} \leftarrow$ Set of all Components
3: $Confs \leftarrow$ Set of all Configurations
4: $C^{=} \leftarrow$ Set of Required Components
5: Uc $\leftarrow \{\}$
6: Mc $\leftarrow \{\}$
7: Mx $\leftarrow \{\}$
8: procedure CONSTRAINT ORGANIZATION
9:     for i $\leftarrow 1$ to number of elements of $Comps^{in}$ do
10:        if $Comps^{in}(i) \notin TempList$ then
11:            Uc $\leftarrow Comps^{in}(i) \cup$ Uc
12:        end if
13:    end for
14:    p $\leftarrow 1$                              ▷ p is the current cell of MultiConstraint List $(mc)$
15:    for j $\leftarrow 1$ to number of elements of $TempList$ do
16:        Mc$(p) \leftarrow TempList(j)$
17:        repeat
18:            for all $TempList(k) \mid k \neq j \wedge TempList(k) \neq \oslash$ do
19:                if Mc$(p) \cap TempList(k)$ then ▷ search for any constraints that contain any component with in Mc$(p)$
20:                    Mc$(p) \leftarrow$ Mc$(p) \cup TempList(k)$
21:                    $TempList(k) \leftarrow \oslash$
22:                end if
23:            end for
24:        until Mc$(p)$ no longer changes
25:        $p \leftarrow p + 1$
26:    end for
27:    for m $\leftarrow 1$ to number of elements in Mc do
28:        if $\forall\, Confs \in$ Mc$(m)$ then
29:            Mx $\leftarrow$ Mx $\cup$ Mc$(m)$
30:            Mc$(m) \leftarrow \oslash$
31:        end if
32:    end for
33:    Remove Duplicate Components and Configurations from $Mc$ and $Mx$
34: end procedure
35: return $\{Uc, Mc, Mx, C^{=}\}$
```

The j for-loop on line 15 performs a large portion of the sorting action where j is 1 to the number of constraint pairs. On the previous line p is defined as 1 create the space for the first element of Mc. On line 15 Mc(p) is defined for the current j value and adding the first components to Mc(p). The following lines search all $TempList$ except for $TempList$(j) for components contained in Mc(p). This process is repeared for the current Mc(p) until no more elements are added. It is important to note on line 20 Constraints added to Mc(p) are removed from the main constraint list $TempList$ preventing infinite loops from occuring by finding repeat elements. After this process each element of Mc is search to determine if it contains a configuration option. If an element does contain a constraint option that group of components/configuration options is removed from Mc and move to Mx. The variables Uc, Mc, and Mx along with $C^{=}$ earlier were defined to represent the four different lists. These variables will be used in the other algorithms in this section.



Connected Objects
Mc : { [A B C], [E F G H] }
Mx : { [J X Y] }
Uc : { [D] }
Rc : { [K], [L] }

Figure 4.8: Mc, Mx, Uc, and Rc Defined.

Figure 4.8 presents the results of the constraint organization process. This shows which components and configuration options are connected by constraints. The objects in brackets are connected via constraints and below them shows which list each one is contained in.

4.2.1.2 Evaluate Groups

The next step in the generation process is to evaluate the constraint groups created in the previous step. This step represents Equations 3.21 to 3.23 and the methods related to them. This is where each individual set of object types, whether it be unconstrained, constrained, or required are evaluated and any relevant constraints are applied. The constraints in each group need to be evaluated to determine feasible combinations of the components each group contains. This process involves applying the constraints to smaller groups of components and enumerating feasible combinations from the smaller groups. This reduces computational load by evaluating constraints only on the groups of components which they apply. This method generates all combinations of the smaller groups and then applies the constraints to the smaller groups. This avoids the combinatorial explosion that occurs with large set sizes. For this all combinations of the objects in each group are created and then reduced by applying the constraints within each group. This will result in a set of smaller truth tables that can be used to create the full enumeration of feasible products if required. Truth tables are used for these because the constraints can be defined in the form of basic boolean operations that can be applied to entire sets. The Required Pairs is a XNOR and the Dissallowed Pairs is a NAND.

Figure 3.12 from Chapter 3 displays the truth tables for these operations, where A and B members of the constraints and F is the feasibiliy based on the presence of and B. Pseudocode for this process is shown in Algorithm 4.2 to provide a simple overview of the evaluation process. Figure 4.8 presented the outputs of the previous algorithm for the on going example. Notice that each variable utilizes cell arrays to store the data. For Algorithm 4.2 the inputs are Mc, Mx, and Uc; the outputs are Mct, Mxt, and Uct. The outputs will contain the feasible combinations of each set of connected components. Figure 4.9 shows the output of the algorithm.

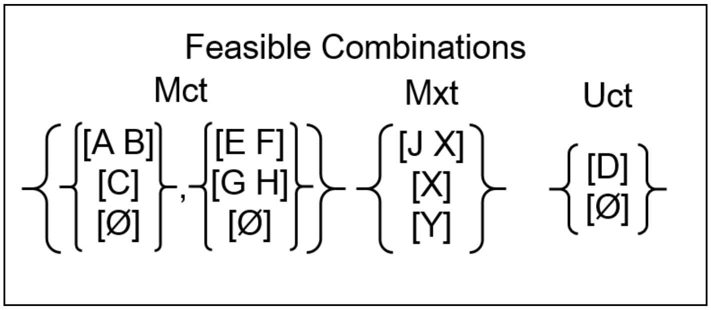


Figure 4.9: Individual feasible sets.

Notice that Mct, Mxt, and Uct are still in Cell array form but each cell has multiple sub cells containing the feasible combinations of the components in each cell of Mc, Mx, and Uc. The union of Mct and Mxt would be equivalent to $ConstrainedSets$. Uc would be equivalent to $Parts^{all}$ with constrained components removed. These feasible combinations will also be used in the next step to count the number of feasible solutions Algorithm 4.2 presents the process it takes to determine the outputs.

Algorithm 4.2 Constraint Evaluation Procedure

```
 1: $Uct \leftarrow \{\}$                                        ▷ unconstrained components in tables
 2: $Mct \leftarrow \{\}$                               ▷ Set of component only constraints tables
 3: $Mxt \leftarrow \{\}$                                         ▷ Set if Mixed constraint tables
 4: procedure EVALUATE CONSTRAINTS($Uc, Mx, Mc$)
 5:     for $i \leftarrow 1$ to length of Uc do
 6:         $Uct(i) \leftarrow \{\ Uc(i),\emptyset\}$
 7:     end for
 8:     for $j \leftarrow 1$ to number of cells in $Mc$ do
 9:         $Mct(j) \leftarrow \wp$ of the contents of $Mc(j)$
10:         for $k \leftarrow 1$ to number of constraints connected to the contents of $Mc(j)$ do
11:             Apply constraint $Mc(j,k)$ to $Mct(j)$
12:         end for
13:     end for
14:     Mxt $\leftarrow \wp$ of the contents of Mx
15:     for $m \leftarrow 1$ to number of constraints connected to the contents of Mx do
16:         Apply constraint $Mx(m)$ to Mxt
17:     end forreturn $\{Uct, Mct, Mxt\}$
18: end procedure
```

Algorithm 4.2 provides a brief overview of the process of constraint evaluation. To apply constraints first the combinations of the contents of $Mc(j)$ and Mx must be created. Lines 9 and 14 create the powerset of $Mc(j)$ and Mx respectively to create Mct and Mxt. Then lines 11 and 16 apply relevant constrains to Mct and Mxt. Both Mct and Mxt are changed and invalid combinations of components and/or configuration options are removed. This is a simplified version of the evaluation process compared to the version implemented in the software. While it may seem unnecessary to have objects with and without constraints containing configurations in separate variables, but constraints between only configurations must be applied first to prevent more than one configuration of the same type from being present in a product. This step completes the bulk of the processes required for the generation process. The next steps will take the results of the evaluation process to determine the effects of the combinatoric constraints and generate all feasible products if desired.

4.2.1.3 Calculate Products

The calculation of the total number of feasible products is a simple calculation as it just counts the number of rows left in each truth table and multiplies them all together. While a significant amount of work is required to determine the effects of the constraints, it is fairly computationally light because the constraint effects are compartmentalized and truth tables only need to be generated for the number of components in each group of components connected via constraints. This also provides the benefit of not having to generate all possible products to determine constraint effects, greatly reducing impact on computer memory. This calculation is similar to the calculation for the unconstrained number of products presented in Equation 3.8 but is modified to reflect the processes presented in the previous steps to apply constraints. The following Equations present the process of calculating the number of feasible products with combinatoric constraints applied.

$$N_{uc} = 2^{N_u} \tag{4.1}$$

$$N_{mc} = \prod_{i=1}^{|Mct|} |Mct_i| \tag{4.2}$$

$$N_{cn} = Ucn * N_{mc} * |Mxt| \tag{4.3}$$

Where N_u is the number of unconstrained components.$|Mct|$ is the number of elements in Mct, $|Mct_i|$ is the number of elements in Mct_i and $|Mxt|$ is the number elements in Mxt. Since Mct consists of subsets of multiple elements, an additional operation is required to determine the size of the subsets This calculation allows for the effects of constraints to quickly and easily be seen giving the user instant feedback on design decisions. The final step of the process is to generate the full set of all feasible products. This step is optional and in the software developed it is not used. Feasible product are developed by only allowing combinations of products in Uct, Mct and Mxt.

Figure 4.10 shows the process of calculating the number off possible products that are able to be produced from the running example. Notice that $C^{=}$ is not included because the required components are required for every product to be feasible and there are no other options for them. This results in 54 possible products available, which is considerably smaller than the unconstrained number of 4,096.

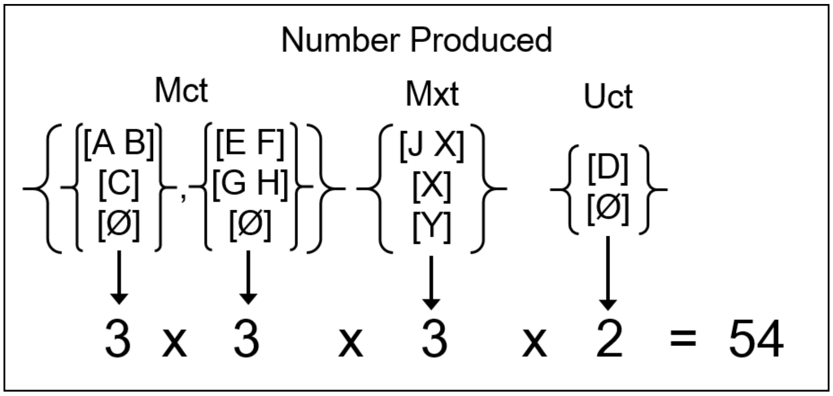


Figure 4.10: Number of possible combinations

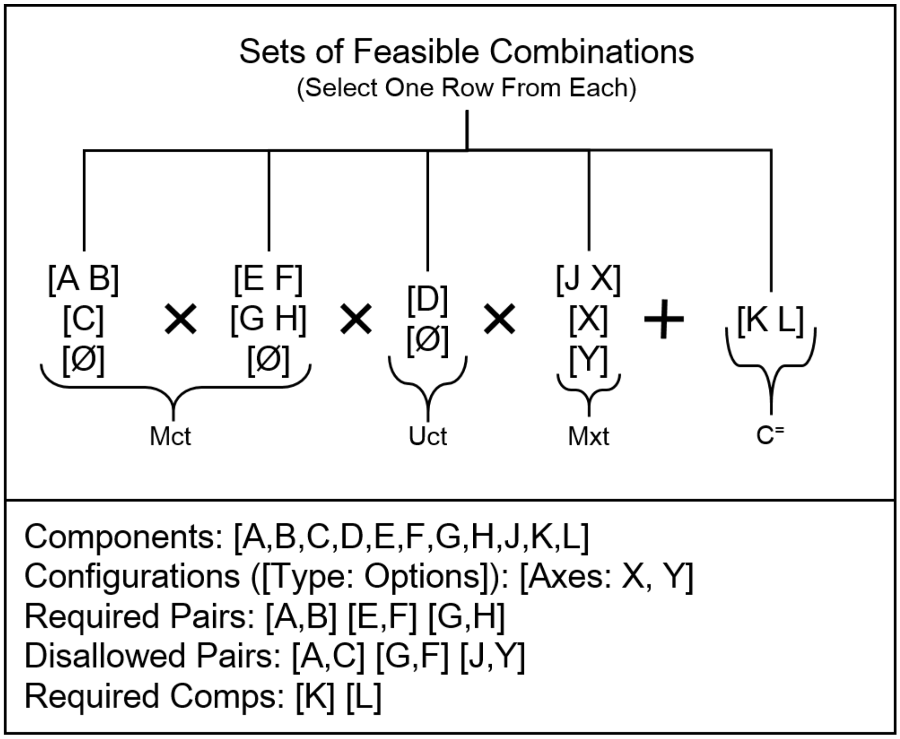
4.2.1.4 Generate

The process of generating all feasible products is also quite simple since much of the work was performed during the evaluation stage. This final step of the generation process implements Equations 3.24 and 3.26 and their related methods into code. Where the combinations are found for the elements of the different sets of objects are found and formatted for proper presentation. In this step products are created by taking one element from each sets (subset in the case of Mct), Uct, Mct and Mxt then combining with the required elements of $C^{=}$ to develop a product. All possible combinations of these generates the full set of feasible products. The math of this process relies of the set theory operation, Cartesian Products which creates a set containing subsets containing the combinations of the elements of 2 or more sets. Once the Cartesian product operation is complete a union operation is used to combine the generated products with the required components. Algorithm 4.3 presents the generation process:

Algorithm 4.3 Product Generation Procedure

```
1: $E_{Products} \leftarrow \{\}$                    ▷ Enumeration of all possible products
2: procedure PRODUCT GENERATION AND ENUMERATION
3:     $E_{Products} \leftarrow \left[\prod_{i=1}^{|Mct|} Mct_i \times Mxt \times Uct\right] \cup C^{=}$
4: end procedure
5: return $\{Uc, Mc, Mx, C^{=}\}$
```

Where $|Mct|$ is the number of subsets of Mct. This simple process utilizes set theory operations to generate all feasible products. This process is simple because the Evaluation step of the generation process eliminated any possible infeasible combinations of component or configuration options. Line 3 in the algorithm is the same computation that occurs in Equation 3.24. There is no need for the removal of the empty set elements that occurs in Equation 3.26 because the use of cell arrays in the implementation allows for empty sets to be represented and not effect the result.



Components: [A,B,C,D,E,F,G,H,J,K,L]
Configurations ([Type: Options]): [Axes: X, Y]
Required Pairs: [A,B] [E,F] [G,H]
Disallowed Pairs: [A,C] [G,F] [J,Y]
Required Comps: [K] [L]

Figure 4.11: Full Generation Visual

Figure 4.11 provides a simple illustration of the full enumeration process for the running example presented in Figure 4.6. Each column in the figure represents a set of components connected via constraints. A feasible product is generated by selecting one set from each column. All of these, except for the last column, are combined using the Cartesian Product operation. The last column represents components with the ”Required” constraint. These are added to the set since all elements of $C^{=}$ are required. The constraints and components from Figure 4.11 could generate 54 products. While this seems like a large number of products it is considerably smaller than the 4,096 unconstrained combinations. Figure 4.12 presents a Tree Diagram of the example product family showing how quickly it expands. Notice how $Mct(1)$ and $Mct(2)$ are specified since both subsets need to be addressed.

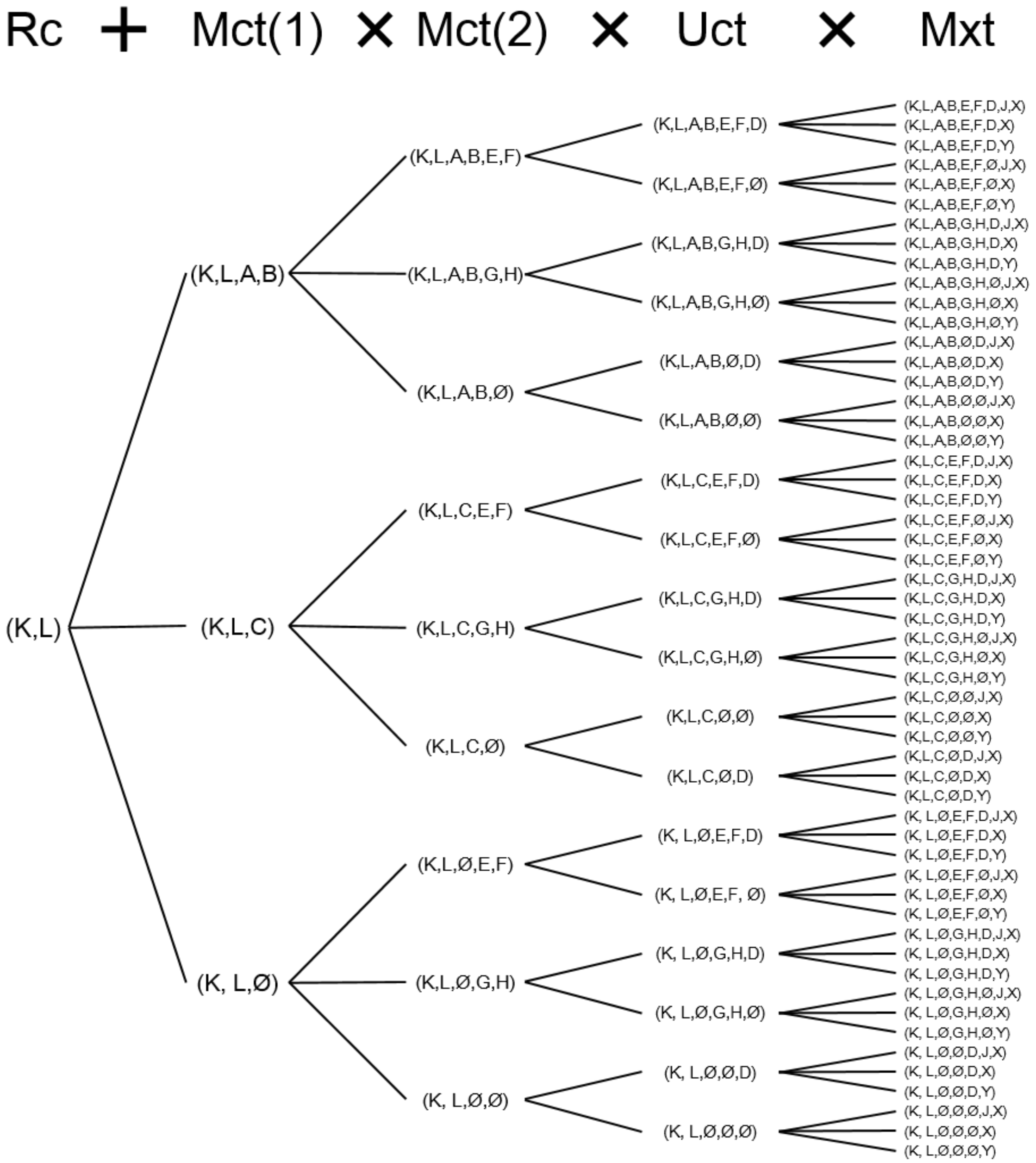


Figure 4.12: Generation Tree Diagram

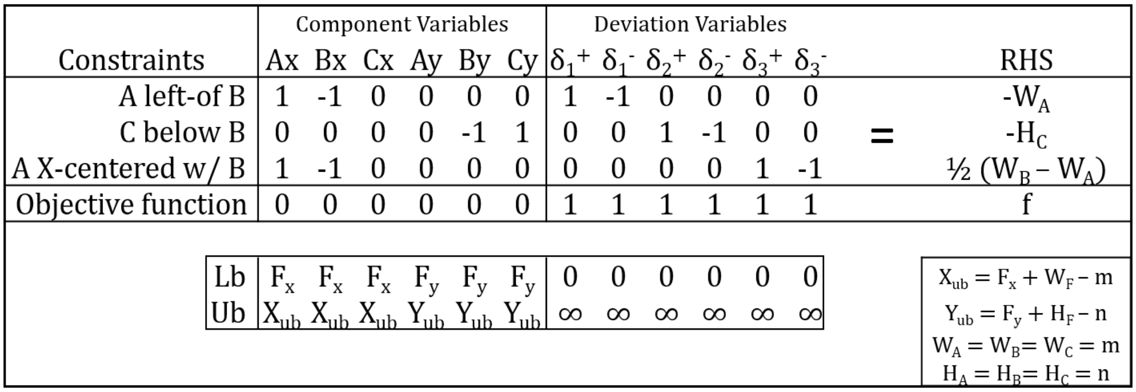
These steps provide an overview of the approach used to implement the method for constrained combinatoric generation of products presented in this thesis. This quick and efficient implementation allows for fast results and the user to easily flow through the software. In the next section an overview of the implementation of the component layout portion of this thesis is presented.

4.2.2 Component Layout Process

The layout process is the final step in the product family generation method presented in this thesis. The implementation of this method is simple since it is based on linear programming and the simplex method. This portion of the software very much follows the steps presented in Section 3.3.3. This section will provide a brief summary of how the different parts of the simplex algorithm were setup and any required post processing.

4.2.2.1 Simplex Setup

Matrices and vectors are used for the inputs of the process. For the constraint matrix the locations of the variables are based on the order of the components stored in the $comp$ structure, allowing for ease of indexing and creating constraints in the proper locations. The layout constraints in this software are the same ones presented in Table 3.6. The elements of the Simplex operation at this state are in the form that appears in Equation **??**. Once all the constraints are assembled the deviation variables δ are appended to the end of each constraint since the total number of δ variables is twice the total number of constraints. The deviation variables are defined in Equation 3.45. Since the objective function for this method contains only the deviation variables from the constraints, the objective function can not be defined in the code until all the constraints defined. The objective function and constraints defined with the deviation variables is the implementation of Equation 3.47. For the variables that represent the coordinates of the various components the upper and lower bounds are applied based on the chosen fascia.



	Component Variables						Deviation Variables							
Constraints	Ax	Bx	Cx	Ay	By	Cy	δ_1^+	δ_1^-	δ_2^+	δ_2^-	δ_3^+	δ_3^-		RHS
A left-of B	1	-1	0	0	0	0	1	-1	0	0	0	0		$-W_A$
C below B	0	0	0	0	-1	1	0	0	1	-1	0	0	=	$-H_C$
A X-centered w/ B	1	-1	0	0	0	0	0	0	0	0	1	-1		$\frac{1}{2}(W_B - W_A)$
Objective function	0	0	0	0	0	0	1	1	1	1	1	1		f
Lb	F_x	F_x	F_x	F_y	F_y	F_y	0	0	0	0	0	0		
Ub	X_{ub}	X_{ub}	X_{ub}	Y_{ub}	Y_{ub}	Y_{ub}	∞	∞	∞	∞	∞	∞		

$X_{ub} = F_x + W_F - m$
$Y_{ub} = F_y + H_F - n$
$W_A = W_B = W_C = m$
$H_A = H_B = H_C = n$

Figure 4.13: Example of Simplex Algorithm Setup

Equation 3.46 shows the process of defining the bounds on component locations, where the fascia for each component is chosen by the designer. The bounds for the deviation variables must also be defined, since the values of the deviations can only be zero or a positive number, the bounds for all deviation variables is defined as all number from zero to infinity, $[0, +\infty]$. Figure 4.13 presents an example of the how each aspect of the simplex method is implemented for a set of general constraints. Notice how the coefficient of the location variables are zero in the objective function since only the deviation from the desired is meant to effect the optimization process. This allows for layout constraints to be violated and a feasible solution found even if the designer the desired layout is infeasible.

4.2.2.2 Overlap Check

However even if no layout constraints are violated a layout may be infeasible if components overlap. Since the simple algorithm is a linear method it is not possible to apply a constraint to eliminate overlap. The components must be properly constrained. The final step of the layout process is to determine if there is any over lap between components in products. It is very possible that when every component is place components may appear to overlap, but the layout is only infeasible of the components that overlap appear in the same product. To ensure there are no overlaps the list of components from every defined product is checked for overlap based on the procedure in Algorithm 3.1. If there is no overlap in any product the product family is feasible, but anytime any change is made to that may effect component location overlap must be rechecked.

4.2.3 Comments

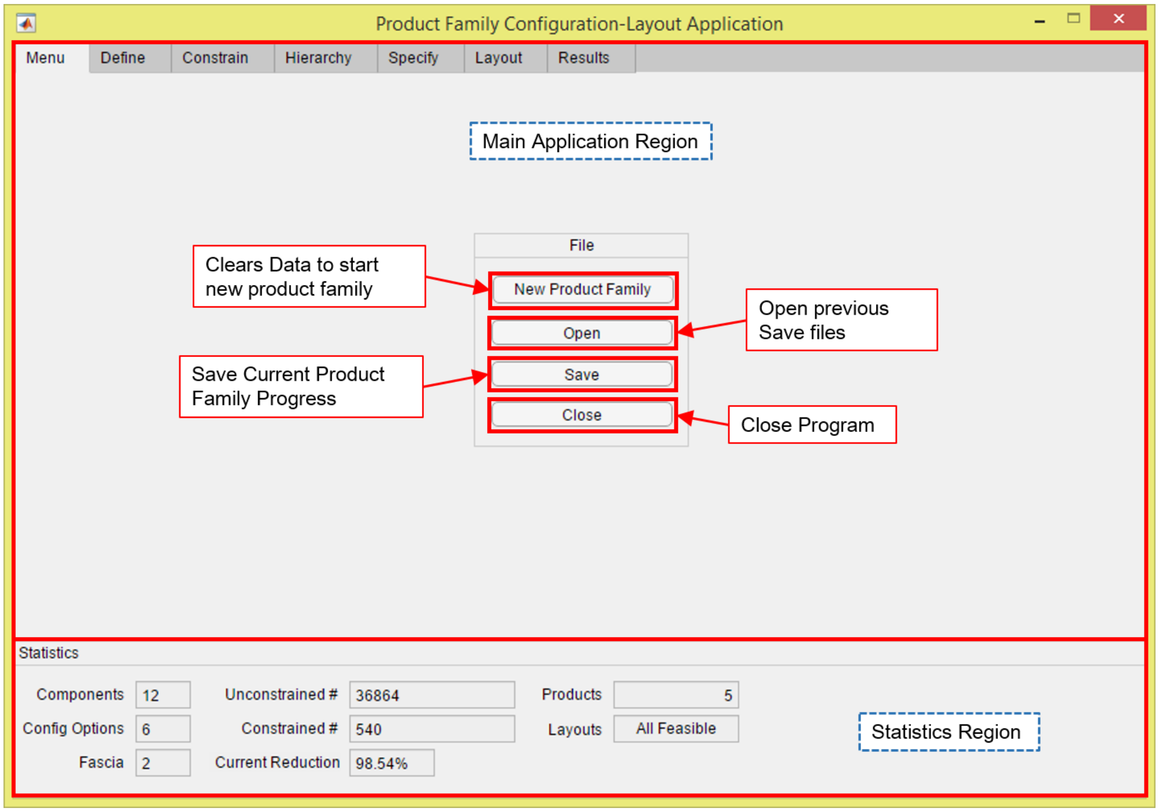
This section presented an overview of the algorithms and processes used to implement the work of this thesis into code. The code for combinatoric generation implementation demonstrates the ability to determine the consequences of design decisions on product families without generating all valid products. The layout implementation provides the ability to complete initial design based on a basic constraint grammar and determine whether the desired layout is feasible. Both of these represent the backbone of the GUI-based software presented in the next section.

4.3 Software Workflow

In this section the software developed for this thesis is presented. This program implements the methods presented in Chapter 3 into an fast GUI-based program that allows designers to both generate a product family based on combinatoric constraints and determine common component layouts across the entire product family. This step allows the designer to determine the effects of design choices and make changes at any point in the process to achieve the desired result. This software is divided into seven tabs (Menu, Define, Constrain, Hierarchy, Specify, Layout, Results) and a constantly updating statistics region. The goals of these regions was to create a simple and intuitive GUI to allow users to quickly learn and include in their design process. Each of these will be presented and information about their function explained.

4.3.1 Menu

The ”Menu” tab is the screen any user will see when starting up the program. This will allow uses to open any previous save files or close the program if it was accidentally opened. Figure 4.14 presents the application where the upper portion of the application set on the ”Menu” tab is the Main region and the strip at the bottom is the Statistics region. The contents of the Main region changes as different tabs are selected while the Statistics always contains the same items.



Components 12
Config Options 6
Fascia 2
Unconstrained # 36864
Constrained # 540
Current Reduction 98.54%
Products 5
Layouts All Feasible

Figure 4.14: Menu Tab of Product Family Configuration Layout Application

Notice in the Menu portion there are four labeled buttons (”New Product Family”, ”Open”, ”Save”, ”Close”) used to perform all of the primary functions of this tab. The basic functions of these buttons are outlined below.

”New Product Family” - This allows the user to clear then entire workspace currently in use by the programand resets all user interface components. Allowing the user to either scrap an unsaved project or start a new project after saving another.

”Open” - Allows the user to open previously product family projects created by this program. Allowing the user to make changes or continue developing an incomplete product family.

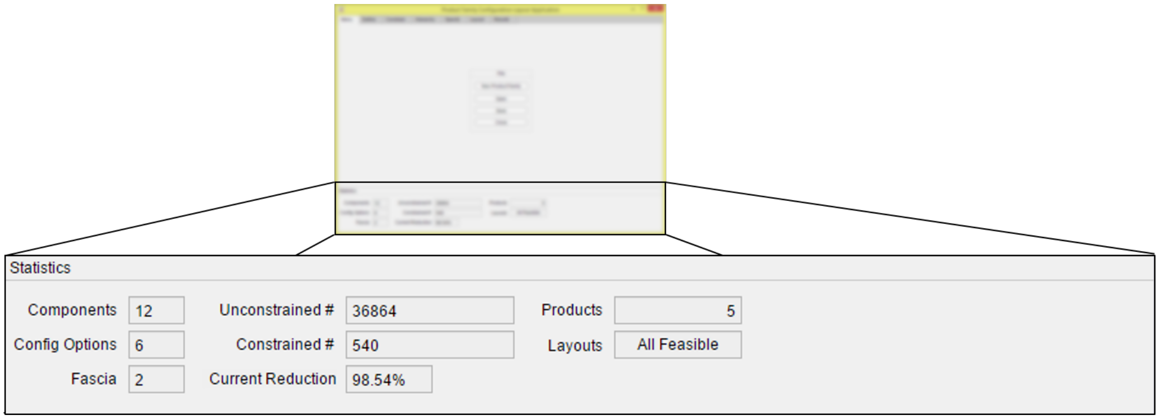
”Save” - Allows the user to save the data currently input and output from the program into a save file that the user is able to name. The specific data being saved is all of the data within the structures presented in Section 4.1. This also allows Users to develop multiple versions of the same product family and save them for comparison.

”Close” - Allows the user to close the program and performs the same function as clicking the ”X” button on the top-right of the window. When clicked, a dialog box pops up to confirm that the user would like to exit the program.

While minor in the overall program, these functions provide essential functions to the user with the ability, to save, open, and create new projects in a quick and effienient manner as well as allow the user to only purposely close the program to prevent lost of data and changes made.

4.3.2 Statistics Panel

Mentioned in the Menu section, the statistics region is panel of fields containing current statistics, calculations and results. These fields are actively updated as changes are made throughout the product family development process. Figure 4.15 presents the stats panel.



Statistics

Components 12 | Unconstrained # 36864 | Products 5

Config Options 6 | Constrained # 540 | Layouts All Feasible

Fascia 2 | Current Reduction 98.54%

Figure 4.15: Statistics Panel of Product Family Configuration Layout Application

The panel has 8 fields, each representing a different statistic, calculation, or other indicator. These provide the user with information about the product family at all times. Below is a brief description each of the fields present in the panel.

Components - The number of components defined in this product family.

Config Options - The number of configuration options defined.

Fascia - Number of Defined Fascia.

Unconstrained # - Unconstrained number of possible products based on the number of defined components and configuration options.

Constrained # - The reduced number of possible feasible products based upon configuration constraints.

Current Reduction - Calculation of the reduction from Unconstrained to Constrained.

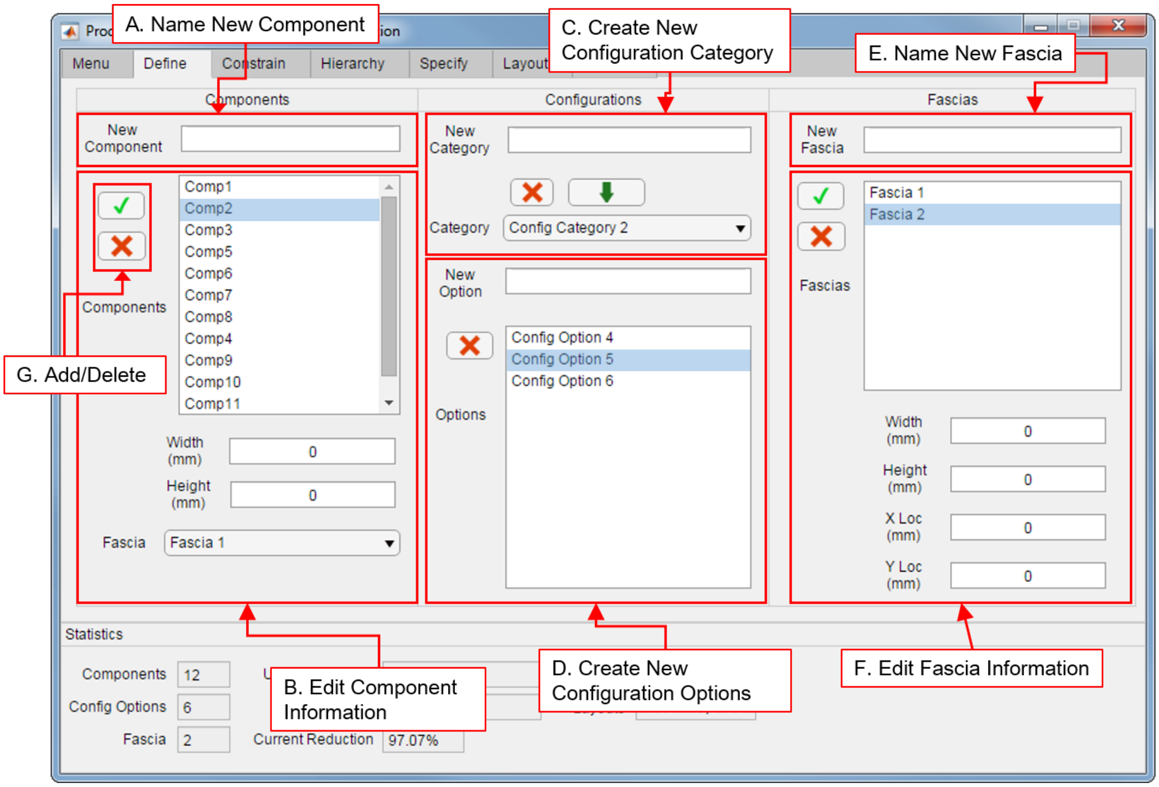
Products - The number of defined products.

Layouts - Displays whether the layouts for the defined products are feasible based on whether there is component overlap.

The fields in this panel each update as changes are made to the product family, allowing designers to instantly see the effects of decisions. The Constrained # statistic is calculated from the Equations presented in Section 4.2.1.3.

4.3.3 Define

In the ”Define” tab the user defines the components, configurations, and fascias that will appear in the product family. Along with naming the components and fascia, physical information use in the layout process is also defined here. The ”Define” tab is presented in Figure 4.16. The boxes and arrows name and simply explain the various parts of the tab.


A. Name New Component

B. Edit Component Information

C. Create New Configuration Category

D. Create New Configuration Options

E. Name New Fascia

F. Edit Fascia Information

G. Add/Delete

Category: Config Category 2

Config Option 4
Config Option 5
Config Option 6

Fascia 1
Fascia 2

Width (mm): 0
Height (mm): 0
X Loc (mm): 0
Y Loc (mm): 0

Comp1
Comp2
Comp3
Comp5
Comp6
Comp7
Comp8
Comp4
Comp9
Comp10
Comp11

Width (mm): 0
Height (mm): 0
Fascia: Fascia 1

Statistics

Components: 12
Config Options: 6
Fascia: 2
Current Reduction: 97.07%

Figure 4.16: Define Tab of Product Family Configuration Layout Application

Each of the parts of this tab are used to define different parts of the product family. Below, the various aspects of the tab are explained, to provide a more detailed description.

An important note that applies to this area is that the program will not allow the user to assign 2 components/configurations/fascias the same name. This is because this program indexes data using the names of objects, requiring each object to have a unique name.

A. This edit field is used to name new components to be added to the product family. Once a name is entered the user can either press “Enter” on the keyboard, or Press the Green Checkmark below. This name is then added to the appropriate data structure and will appear at the end of the listbox in B.

B. This are is used to enter and edit information about each individual component. When a component is initially created its Height and Width will be set to zero and there will be no chosen fascia. When an item in the listbox is highlighted the Width, Height, and Fascia fields will display the information relating to the highlighted component.

To change the component dimensions: (1) Highlight the desired component. (2) change the dimension information in each field, pressing “Enter” on the keyboard after each change. To Change the fascia choice, click on the drop down menu and click on the desired fascia. It should be noted that while the layout information for each component is not required for the configuration process. The layout process cannot proceed without it.

To delete a component, highlight the component desired for deletion and click on the Red ”X” button next to the list box. This will remove the component from the list box and delete it from the appropriate data structure.

C. This area is used to define new categories of configuration options. The Categories are different types of configuration possibilities such loading type for a coffee maker and the options are the different possible configurations for the particular category. For the coffeemaker example loading types could be Front, Top, or Side. While these configuration options affect available product variety and may effect the physical design of a product, configuration options do not always exist in the form of individual components or necessarily represent particular physical objects. There is also the requirement that each product must have one and only one option selected from each category to be valid. Due to these differences, they are defined separately from components and are not given physical shape information.

How ever the use of this area in the program is similar to the Component definition area, where once the name is entered the user can either press ”Enter” or (in this case) click on the Green ”Down Arrow”. This will add the new configuration Category to the drop-down list. The options for a particular configuration Category can be viewed by changing the selection of the drop-down list.

To delete a configuration Category, select the chosen Category from the drop-down and click the Red ”X”. An important note is that when configuration Category is deleted, all configuration Options associated with that Category are also deleted.

D. This area allows for the addition or removal of configuration Options from the Category selected in the Category drop-down. A configuration option is added by typing the name into the New Option field and pressing ”Enter” on the keyboard.

An option can be deleted by highlighting, the option and clicking on the Red ”X”. Since the configuration options for the product family do not necessarily represent a particular physical object no other information is required.

E. This edit field is used to add new fascias for the product family. This process is exactly the same as defining components as shown in A.

F. This area allows the user to view and edit fascia data. Similar to components the dimensions of each fascia must be entered. In addition to dimensions, the location of the fascias must also be defined. The location of fascias are defined from the bottom-left corner of the object to the global zero of the 2D design area.

Fascia data can be edited the same way as with components by highlighting the fascia requiring changes and edit the numbers in the fields, pressing ”Enter” on the keyboard after every change. The procedure for fascia deletion is also the same as components.

G. The add (checkmark) and delete (x) buttons will be seen any other tabs across this program and perform the same action.

While its required that objects be created at the beginning of the process, the flexibilty of this program allows users to go back, make changes, or create new objects at any time during the process. Allowing for the effects of new components or features to be seen on the product family and layout feasibility. The information about Fascia size and location will be used to determine the bounds of components for the layout processes.

4.3.4 Constrain

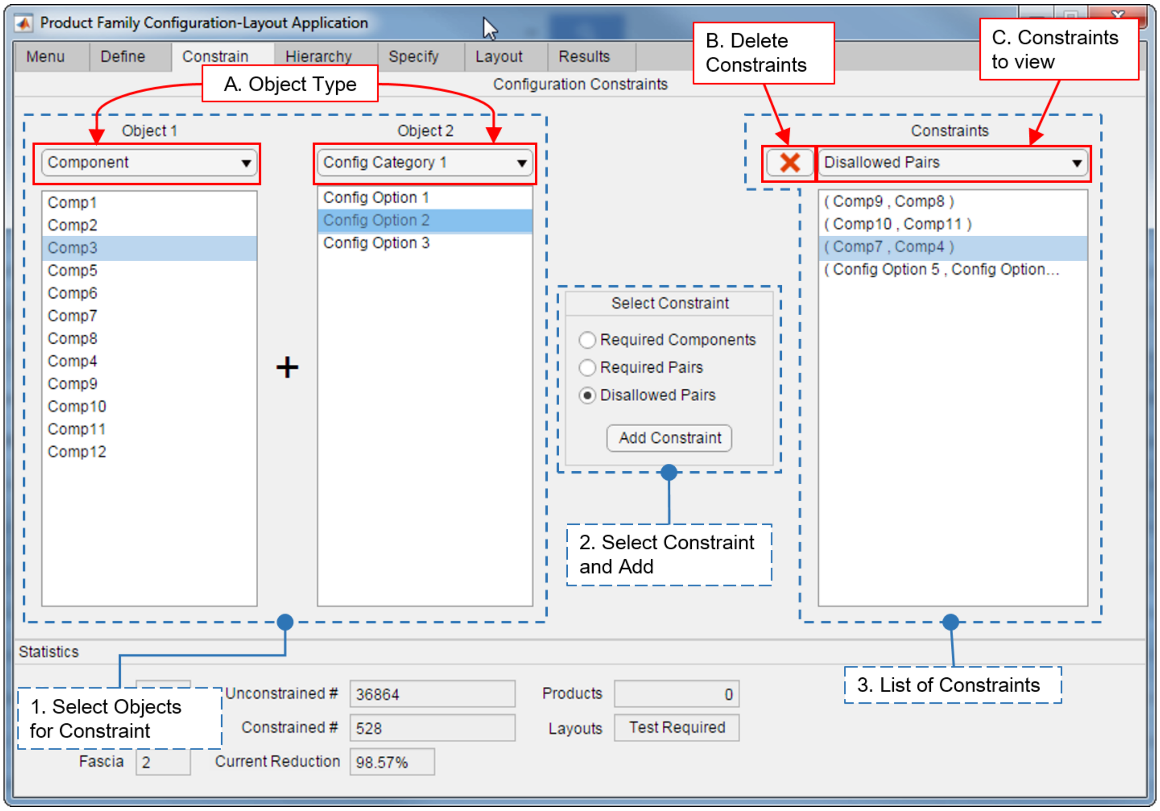
This tab is used to define the configuration constraints of the product family. These constraints enable the designer to dramatically reduce the number of possible product from a set of components and decide which combination of features or components are disallowed or required. This tab allows user to select component(s) and configuration option(s) and define configuration constraints. The configuration constraints, defined in section 3.3, are Required Components, Required Pairs, and Disallowed Pairs. Required Components are listed individually, and the Required and Disallowed Pairs are listed in object pairs. The ”Constrain” tab presented in Figure 4.17 has the following features.

A. The drop-down menus allow the user to select objects by category, where the categories are “Components” and the names of the various configuration categories. Allowing users to easily find specific objects.

B. A delete button used to delete the constraint highlighted in the constraint list.

C. A drop-down menu used to view constraints by type. When a different constraint is selected in the menu, the list box below will display all constraints of that type.

Figure 4.17 shows the simple interface used to define configuration constraints. Where objects for constraint are listed in the boxes to the left. The constraint being defined can be selected and then viewed in a list box on the right. Every time a constraint is added or removed the processes in Algorithms 4.1 and 4.2 are performed to create the groups of feasible component combinations. These are used to determine the effects of the current constraints by calculating the constrained number as presented in Section 4.2.1.3. The dashed boxes in the figure provide a basic instructions for constraint creation.



A. Object Type

B. Delete Constraints

C. Constraints to view

Object 1: Component

Comp1
Comp2
Comp3
Comp5
Comp6
Comp7
Comp8
Comp4
Comp9
Comp10
Comp11
Comp12

Object 2: Config Category 1

Config Option 1
Config Option 2
Config Option 3

Select Constraint:
○ Required Components
○ Required Pairs
● Disallowed Pairs

Constraints: Disallowed Pairs

(Comp9 , Comp8)
(Comp10 , Comp11)
(Comp7 , Comp4)
(Config Option 5 , Config Option...

1. Select Objects for Constraint

2. Select Constraint and Add

3. List of Constraints

Unconstrained #	36864	Products	0
Constrained #	528	Layouts	Test Required
Fascia	2	Current Reduction	98.57%

Figure 4.17: Constrain Tab of Product Family Configuration Layout Application

Below the instructions from the Figure are expanded upon in more detail.

1. Select objects to be constrained by highlighting the desired items in the list box(es). The user can change the category of the list to find items from different categories. The Required Pairs and Disallowed pairs require the user to highlight an object from

both Object 1 and 2 list boxes. The “Required Components” constraint requires only 1 object per constraint and when selected the Object 2 list box is disabled establishing to the user that the object highlighted in the Object 1 list box is the object to be constrained.

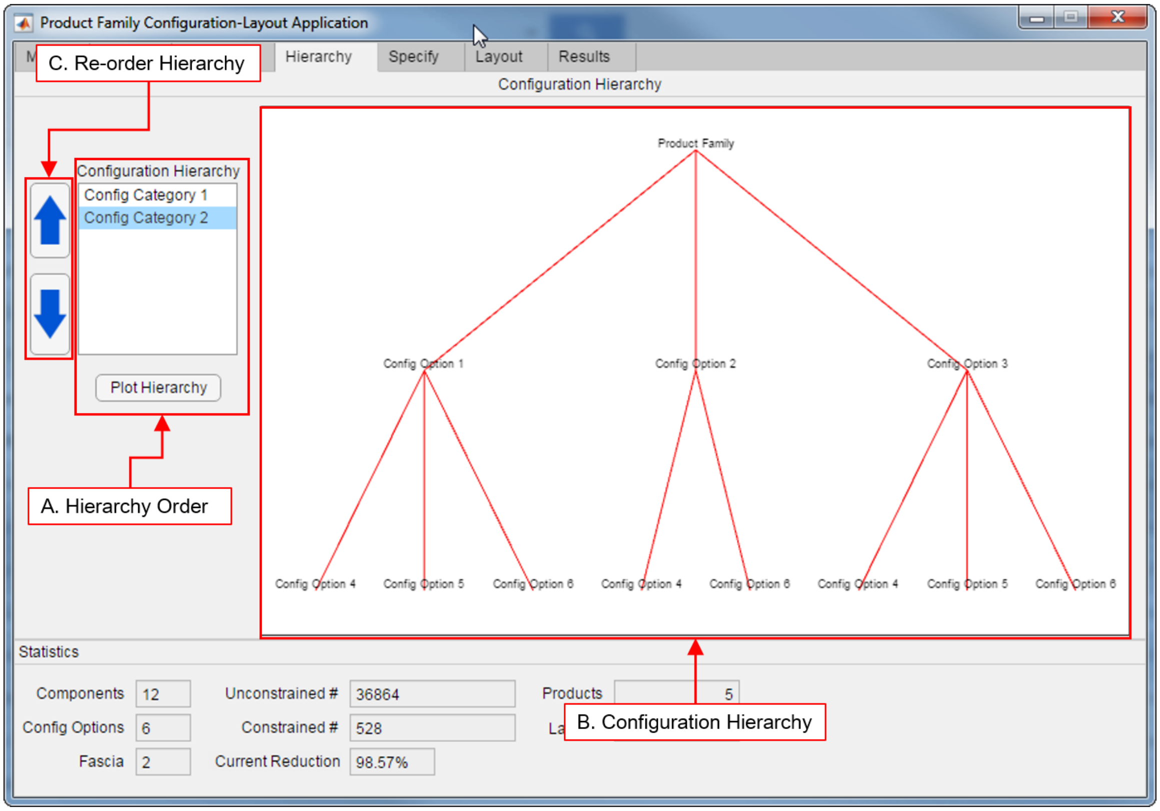
2. Select the constraint from the list and click “Add Constraint”. When the constraint is changed in the “Select Constraint” box, the list box to the right will also change to display the constraints of that type that already exist. When ”Add Constraint” is clicked, the corresponding constraint is added to the list.

3. This list box display already defined constraints. Changing the item in the drop-down menu allows the user to choose the type of constraints previously defined. Constraints can be deleted by highlighting a constraint and clicking the delete button (B).

The goal of this tab was to provide users with as much control as possible in defining configuration constraints but disallowing invalid combinations of constraints. An object that appears in a Required Component constraint is not allowed to appear in any other constraint because it may disallow a component from appearing in any product. An additional feature of this tab is that as constraints are add/removed, the “Constrained #” in the Statistics panel is constantly updated to reflect the effects of the current set of constraints. Testing this feature with a product family containing 22 components and 8 configuration options resulted in 0.75 second computation time to determine the effects of the constraints. This time may vary depending on the number of components, configuration options and constraints defined. This provides the user with instant feedback on design decisions.

4.3.5 Hierarchy

The ”Hierarchy” tab performs the simple task of allowing the user to choose a hierarchical order for the configuration categories and plot the hierarchy after constraints on combinations have been applied. Figure 4.18 shows the Hierarchy tab with a hierarchy plotted. Whenever configuration constraints are changed that involve configuration options, the Hierarchy must be replotted to display the correct hierarchy as well as any time the hierarchy is changed. Algorithms 4.1 and 4.2 are also performed to reflect the new ordering of the configuration categories.



Components 12

Config Options 6

Fascia 2

Unconstrained # 36864

Constrained # 528

Current Reduction 98.57%

Products 5

Figure 4.18: Hierarchy Tab of Product Family Configuration Layout Application

This tab only has a few components since a large portion of it is used to display the plot. It is important to note that any time the hierarchy is regorganized, product configurations will have to be re-selected. This will be addressed further in section 4.3.6. It’s basic features are:

A. The list of configuration categories in their hierarchy order, where the hierarchy is in descending order. The hierarchy can be plotted by clicking the ”Plot Hierarchy” button.

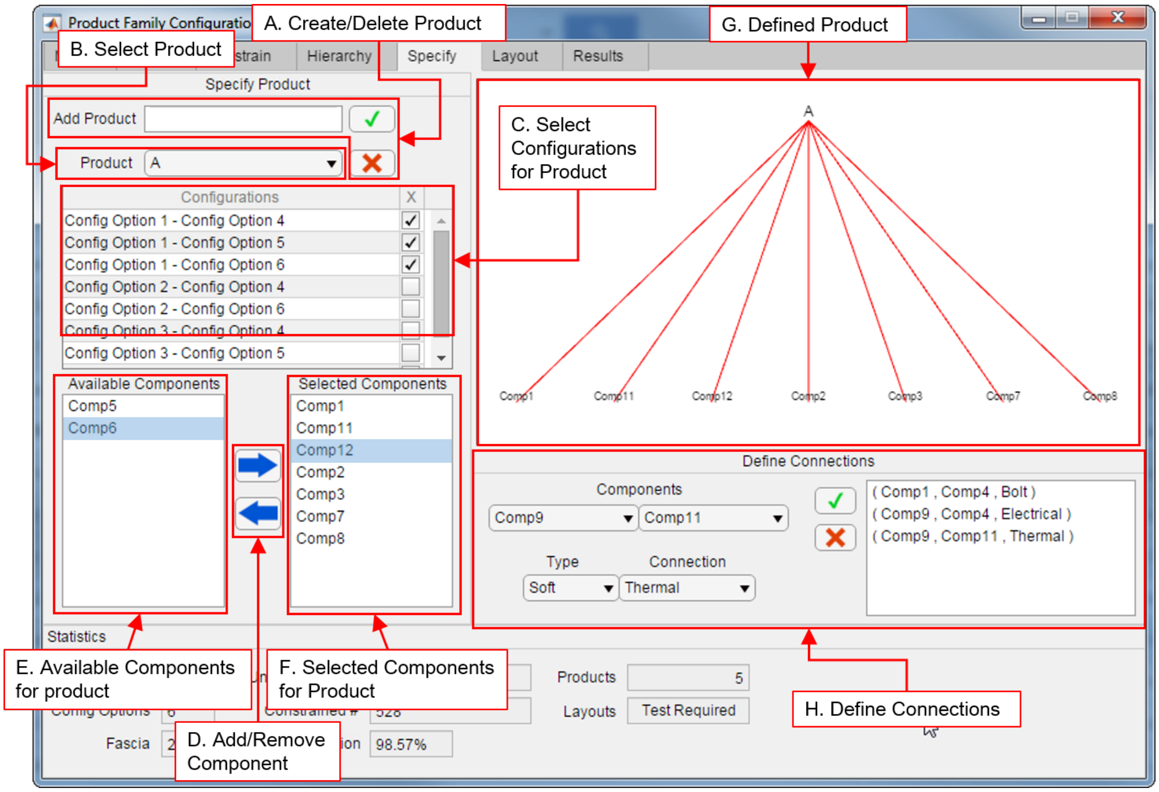
B. This is the plot of the configuration hierarchy. The plot in this figure is a hierarchy that has had constraints applied to the configuration options.

C. These up and down arrows are used to re-order the hierarchy if the designer desires. To re-order, highlight a Category and click the up or down arrow until the desired position is reached.

While this tab is simple it provides with the ability to assign levels of importance to configuration categories and the plot provides a good visual aide to see the affects of constraints.

4.3.6 Specify

The ”Specify” tab is the user is able to start defining the products that will make up the product family and full effects of the configuration constraints. The user is able to name a product and decide what configurations the product is produced in and which components and features the product contains. This tab also allows users to define connections between components which can be noted by the designer during the physical design process. Figure 4.19 presents the interface for this part of the program.



Add Product ✓

Product A ✗

Configurations	X
Config Option 1 - Config Option 4	☑
Config Option 1 - Config Option 5	☑
Config Option 1 - Config Option 6	☑
Config Option 2 - Config Option 4	☐
Config Option 2 - Config Option 6	☐
Config Option 3 - Config Option 4	☐
Config Option 3 - Config Option 5	☐

Available Components: Comp5, Comp6

Selected Components: Comp1, Comp11, Comp12, Comp2, Comp3, Comp7, Comp8

Components: Comp9, Comp11

Type: Soft

Connection: Thermal

(Comp1 , Comp4 , Bolt)
(Comp9 , Comp4 , Electrical)
(Comp9 , Comp11 , Thermal)

Config Options 6

Constrained # 528

Fascia 2

ion 98.57%

Products 5

Layouts Test Required

Figure 4.19: Specify Tab of Product Family Configuration Layout Application

There is a number of parts to this and for the most part there is no proper order as products can be in multiple steps of development. The features will be presented in the basic order of defining a product and component connections.

A. This Field allows users to create a product and define is name. Once the name is entered the user may press ”Enter” on the keyboard or click the Green ”Check Mark”. A product can be deleted by selecting the product from the Drop down menu in (B) and clicking the Red ”X”.

B. This drop-down menu allows the user to select the product being configured and view its current configuration. When a new product is created in (A) the new product is automatically added to the list.

C. This table contains the available configurations to the product and allows the user to select which configurations each individual product will be available in. If a configuration is not allowed to be selected based on the products selected components the program will disallow selection. It should also be noted that the number of products in the Statistics panel is the sum of the number of configurations each product is available in.

D. These arrow buttons are used to add and remove components from a product definition. The right arrow takes the highlighted component from the Available Components list and Adds it to the Selected Component list. The Left Arrow removes the highlighted component from the Selected components list and adds it to the Available components list.

E. The Available Components list shows the components available to add to a product after constraints have been applied. Any time a configuration selected or a component is added or removed from a product the configuration of the product is checked versus the constraints and the Available component list is adjusted so that only components that will make a feasible product can be added. Additionally components connected via Required Pairs are automatically added to the selected components. This ensures the user is only able to create feasible products from the configuration constraints.

F. The Selected Components list is the list of components the product currently contains. Components can be added or removed using the Add/Remove component buttons from D. Every a new product is created the selected components for the product will automatically contain any and all components from the ”Required Components” constraint

G. This plot displays the components the product being defined contains. This plot adjusts as changes are made to the product.

H. This area allows the user to define constraints between components. While these connections apply to the entire product family, they only exist in a product the two components being connected appear in the same product. These will allow the user to note connections for detail design. Table 4.7 presents the available Types and connections available in the program.

The Available Components list and Configurations list are adjust to remove components and configurations that would create infeasible products based on the Algorithm 4.3 and the methods presented in Section 4.2.1.4. Where everytime a component is added, the relevant group of objects is checked to determine whether components are required to be added with it and objects that may not appear with that object in a product.

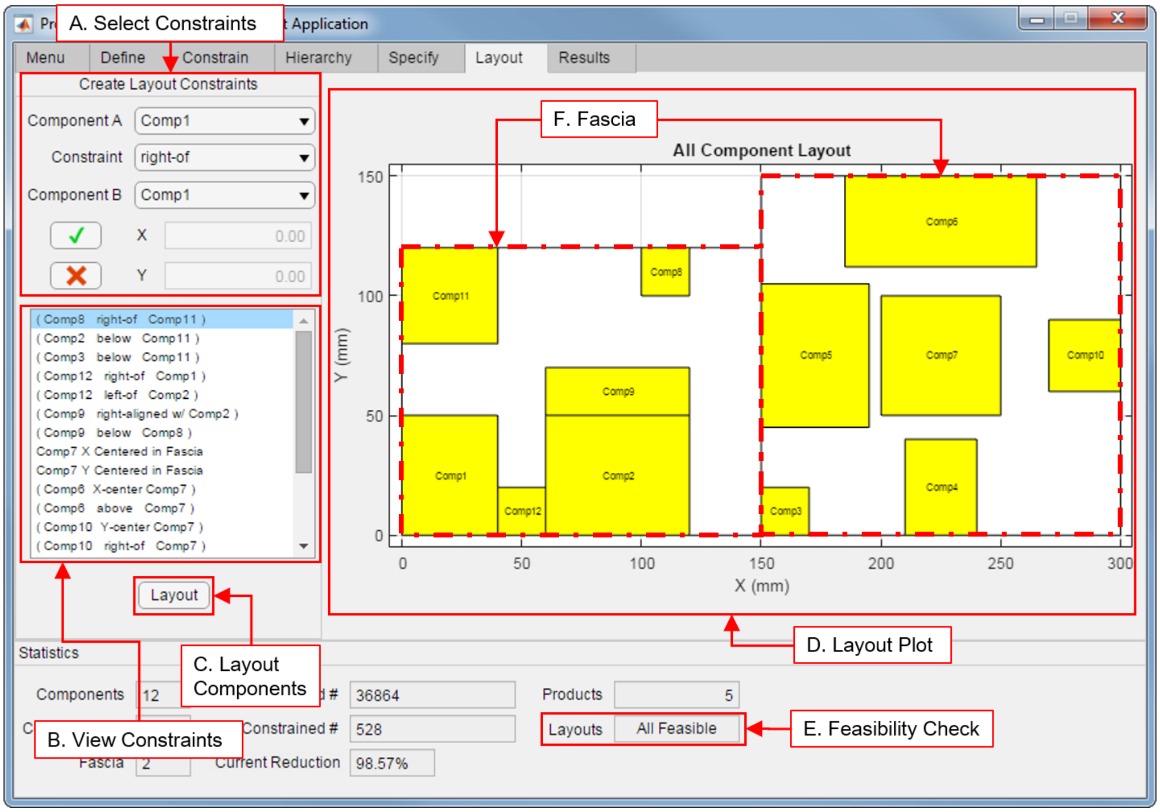
Table 4.7: Connection Type in Product Family Configuration Layout Application

Type	Connection
Rigid	Bolt
	Weld
	Adhesive
	Rivet
	Strap
Kinematic	Pin
	Hinge
	Slider
	Cylindrical
Soft	Electrical
	Fluid
	Thermal
	Information

Since the program constantly tracks the number of products made the user is able to create the number of products desired. The ability to change product definitions and add more products at any time will greatly aide designers when attempting to develop multiple products initially and to later go back and expand the product family by adding more products. While the user is able to go back and make changes at any time to the configuration constraints, any time a constraint is added/deleted or a new configuration category is defined the product configurations and selected products must be redone. This is because changes in constraints can radically change product feasibility and make a feasible product, infeasible. When a new configuration option or category is added more options are available, but the previous options no longer exist since one option from each configuration category must be chosen. This ensures there is no possibility for an infeasible product to be defined.

4.3.7 Layout

The Layout tab is the final step of the product family generation process. In this tab the user is able to define layout constraints from Table 3.6 to the components of a product family and determine whether the layout desired by the user is feasible. Figure 4.20 presents the Layout tab. When the desired constraints are added the will click the ”Layout” button and the operation for setting up the Simplex calculation from Section 4.2.2.1 is performed.



Component A: Comp1
Constraint: right-of
Component B: Comp1
X: 0.00
Y: 0.00

(Comp8 right-of Comp11)
(Comp2 below Comp11)
(Comp3 below Comp11)
(Comp12 right-of Comp1)
(Comp12 left-of Comp2)
(Comp9 right-aligned w/ Comp2)
(Comp9 below Comp8)
Comp7 X Centered in Fascia
Comp7 Y Centered in Fascia
(Comp6 X-center Comp7)
(Comp6 above Comp7)
(Comp10 Y-center Comp7)
(Comp10 right-of Comp7)

Components: 12
C
Fascia: 2
d #: 36864
Constrained #: 528
Current Reduction: 98.57%
Products: 5
Layouts: All Feasible

Figure 4.20: Layout Tab of Product Family Configuration Layout Application

The interface of this tab allows the user to easily define layout constraints on components, both relative and global constraints. Users can view the defined constraints and delete constraints that create an undesirable layout. A basic overview of the parts of the tab is provided below.

A. This portion of the tab allows the user to define layout constraints for components within the product family. The constraints available are defined in Table 3.6. Some constraints require one component for the input and other require two. To guide the user the ”Component B” , ”X”, and ”Y” fields are disabled or enabled based on the requirements of the selected constraints to ensure easy constraint definition.

Constraints can then be added with the Green ”Checkmark” button.

B. This list box contains the all of the currently defined constraints. Constraints can deleted by highlighting the chosen constraint in the Constraint list in B and clicking the Red ”X” from A.

C. This button is used to compile the constraints, perform the simplex optimization process, and plot the component layout. Component overlap is also checked at this stage and is indicated in the Statistics panel (E). However the program will show an error and not proceed to the optimization process if a fascia has not been chosen for each component. Computation times on the layout process are often less than a second. Testing on layouts of 22 components has had computation times in the 0.15 to 0.25 second range.

D. This plot contains the layout of the components in the product family after constraints have been applied. In Figure 4.20 the layout is just an example for the layout for a series of components between 2 fascias.

E. This field displays the results of the overlap check between the components in the product family. It is important to note that overlap is checked for each individual product rather than all components in the product family which is shown in the plot (D). This is because components can occupy the same location in the product family as long as they do not appear in the same product. This ensures that components are able to have the same location in every product they are present in but not require empty spaces to be present in products they do not.

F. The white region in which the yellow components are laid out is the fascia of the product.

The overlap process in the program follows the process presented in Algorithm 3.1. The ”Layout” field in the Statistics panel has 3 states (Test Required, All Feasible, Overlaps Found). The ”Test Required” state is the default state and only changes when the ”Layout” button is clicked in the Layout tab and the layouts are generated and overlap checked.

When checked if overlap is detected the field will display ”Overlaps Found” if not, it will display ”All Feasible”. From these states if a change is made to the components, fascias, or constraints the field will revert back to ”Test Required”. Figure 4.21 displays a flow chart of the basic overlap check process.

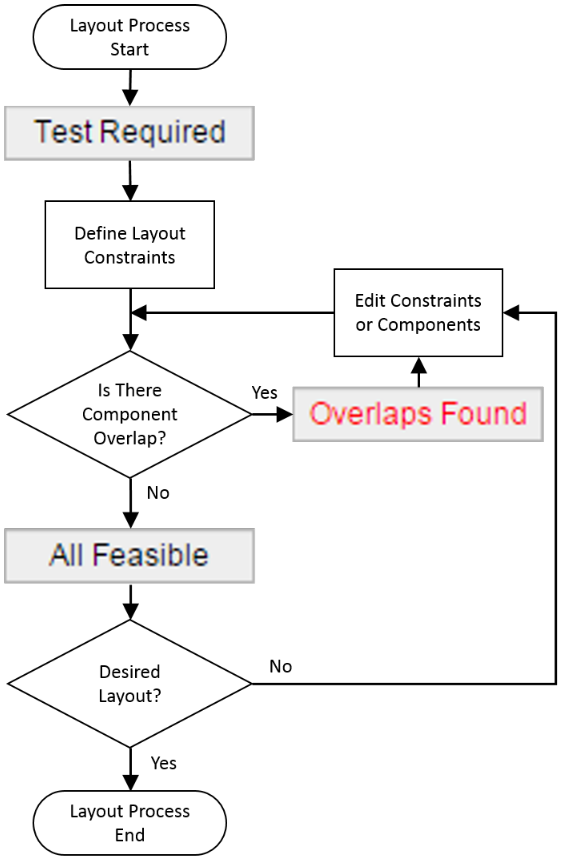
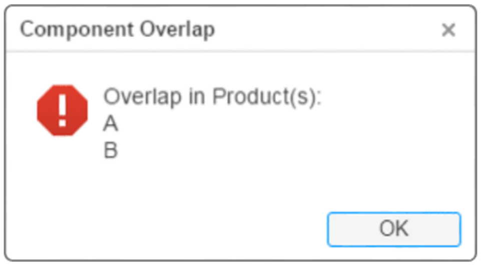


Figure 4.21: Flow Chart of Overlap Check

Additionally when overlaps are detected a dialog box pops up and displays the products in which the overlaps occur. An example of this window is presented in Figure 4.22. Knowing which products the overlap occurs allows the user to look at the products in the next tab, Results, and determine which components are overlapping.



Component Overlap

Overlap in Product(s):
A
B

Figure 4.22: Overlap Notification Window

4.3.8 Results

The Results tab is the final tab in the program and allows the user to review the products that have been defined in the product family generation process. The user is shown the components each product contains, products dimensions and locations, the configurations selected for the product, and which, if any, components overlap. This final tab is presented in Figure 4.23. Since this tab is only for the user to view products and layouts, there is only a single object for the user to interact with. Below is a brief overview of the elements in this tab.

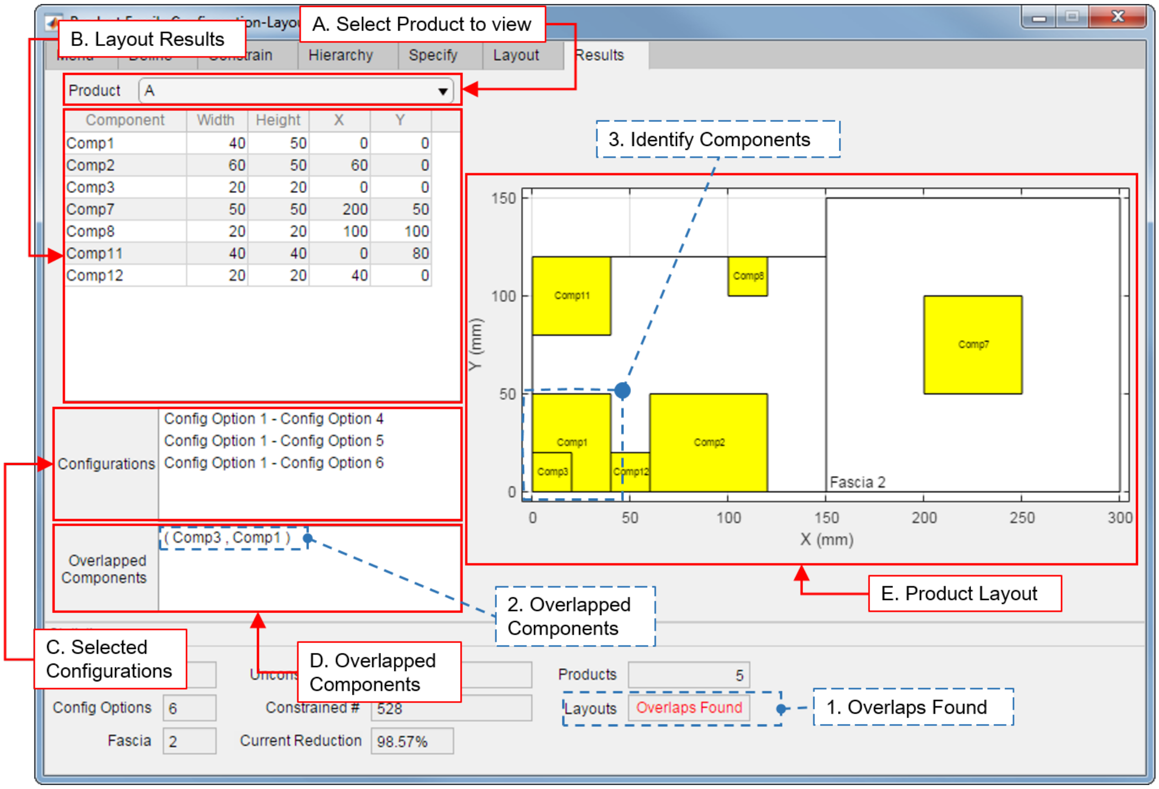
A. This drop down list is used to select between the products defined by the user. When a product is selected the rest of the objects in the tab are refreshed to reflect the definition of the product.

B. This area displays the components contained within the selected product, their dimensions and location in the product layout. This provides the details of the layout process.

C. This list box contains the configurations that the user decided the product will be available in. Since the statistics bar does not indicate the number of configurations for each product.

D. This lists the components that overlap within this particular product. The product in this figure contains an overlap between components ”Comp3” and ”Comp1”.

E. This is the plot of the layout for the product selected. Only components this product contains appear on this plot providing user with an understanding of the the layout of each particular product.



Product: A

Component	Width	Height	X	Y
Comp1	40	50	0	0
Comp2	60	50	60	0
Comp3	20	20	0	0
Comp7	50	50	200	50
Comp8	20	20	100	100
Comp11	40	40	0	80
Comp12	20	20	40	0

Configurations: Config Option 1 - Config Option 4
Config Option 1 - Config Option 5
Config Option 1 - Config Option 6

Overlapped Components: (Comp3 , Comp1)

Config Options: 6
Fascia: 2
Constrained #: 528
Current Reduction: 98.57%
Products: 5
Layouts: Overlaps Found

Figure 4.23: Results Tab of Product Family Configuration Layout Application

Since the plot in the Layout tab contains all components in the product family it can be difficult to understand the layouts of the individual products. Since this product contains a pair of overlapped components a short overview of the overlap checking process is provided in the dashed boxes. Once the overlapped components have been identified the user can follow the procedure outline in the flowchart from Figure 4.21.

4.3.9 Summary

In this section the software developed for this thesis was presented and and overview provided for each part of the program. This provided the general order and work flow of the program for ease of integration into the work flow of the user. The figures for this section only contained general inputs to demonstrate the different parts, in the next section the software is used to demonstrate the generation of a coffeemaker product family.

4.4 Coffeemaker Example

In this section an example of this programs application to the generation of a coffeemaker product family will be presented. This is the same coffeemaker product family example from Chapter 3. This will show how each part for the previous coffeemaker example carries over into this application. The example will present the step-by-step process for the generation of this product family. In this example each of the individual tabs of the program will be worked through and the final results shared. It should be noted that this Example does not define all the components that comprise a coffeemaker such as housings, water tank, heater, filter tray, etc. These components do not effect the product configuration process because they are common to every coffeemaker model. Including these components would clutter the tabs of the user interface and artificially inflate the unconstrained number of combinations.

4.4.1 Define Parts of Product Family

In the define tab all of the information about the different components, configurations and fascia is defined. The information for the components, configurations, and fascia is displayed in Tables 4.8, 4.9, and 4.10.

Table 4.8: Dimensions of Coffeemaker Component

Components	Dimensions (mm)	Fascia Choice
LCD Clock	(60,20)	Top
Power Button	(20,20)	Bottom
Set Hour	(10,10)	Top
Set Minute	(10,10)	Top
AM/PM Set	(5,5)	Top
Brew-Delay Set	(10,10)	Top
Temp Control	(10,10)	Top
Brew Strength Btn	(7,7)	Top
1 - 4 Cup Btn	(7,7)	Top

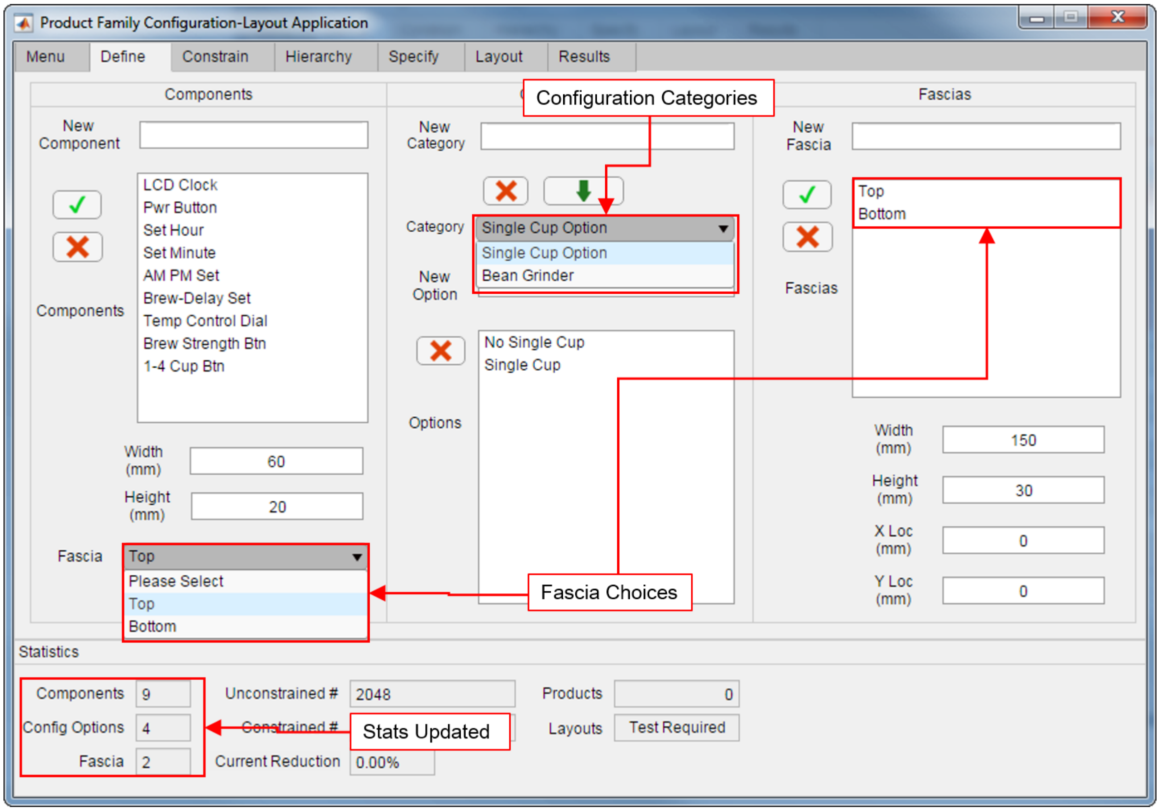
Table 4.9: Configuration Options for Coffeemaker Product Family

Configuration Category	Configurations Options
Bean Grinder	w/ Grinder
	w/o Grinder
Single Cup Module	w/ Single Cup
	w/o Single Cup

Table 4.10: Dimensions and Location of Coffeemaker Fascia

Fascias	Dimensions (mm)	Location (X, Y)
Top	(150,100)	(0,130)
Bottom	(150,30)	(0,0)

The data for these was entered into the program using the information for adding objects to the product family presented in Section 4.3. This resulted in a defined tab with the following fields and available options in the dropdown menus presented in Figure 4.24. This shows what all the fields will look like. Notice the Statistics fields have updated with the addition of these to the program.



Components: LCD Clock, Pwr Button, Set Hour, Set Minute, AM PM Set, Brew-Delay Set, Temp Control Dial, Brew Strength Btn, 1-4 Cup Btn

Width (mm): 60

Height (mm): 20

Fascia: Top (Please Select, Top, Bottom)

Category: Single Cup Option (Single Cup Option, Bean Grinder)

Options: No Single Cup, Single Cup

Fascias: Top, Bottom

Width (mm): 150

Height (mm): 30

X Loc (mm): 0

Y Loc (mm): 0

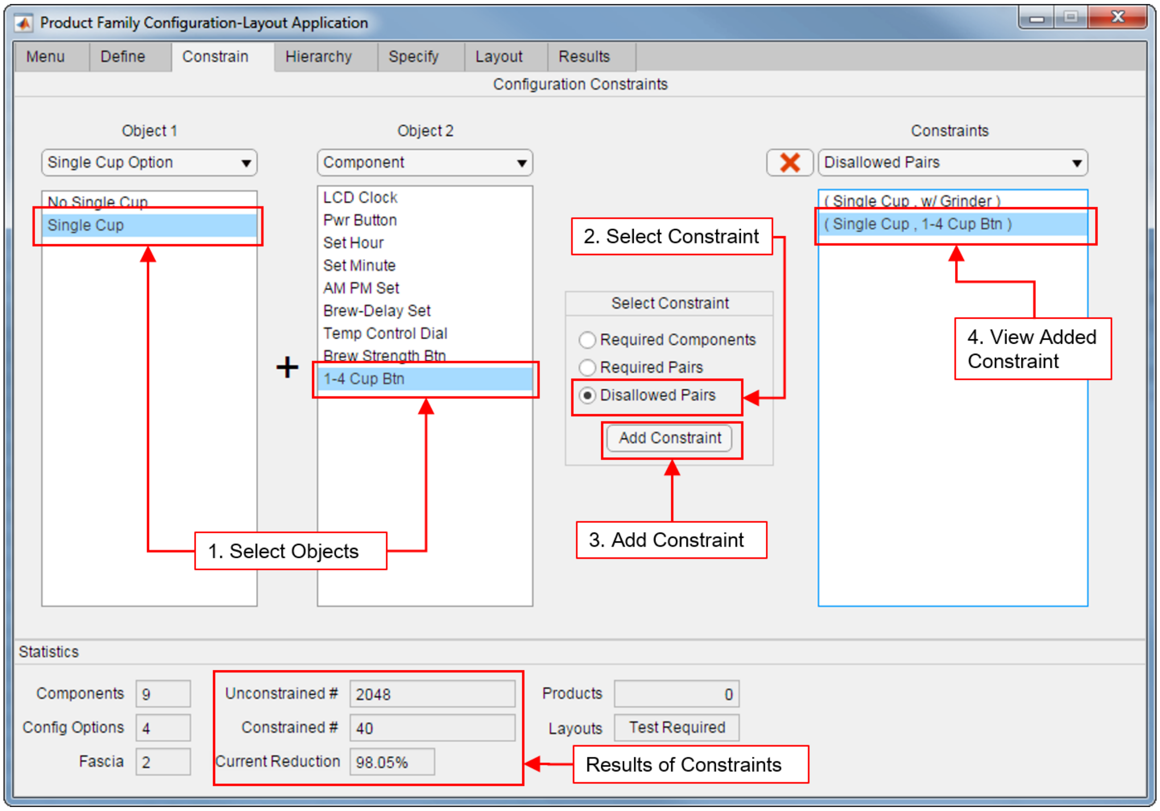
Components	9	Unconstrained #	2048	Products	0
Config Options	4	Constrained #		Layouts	Test Required
Fascia	2	Current Reduction	0.00%		

Figure 4.24: Define Tab for Example

The next step is to define constraints on component combinations.

4.4.2 Constrain Component and Configuration Combinations

Going on to the the Constrain tab to create the configuration constraints on the product family. The basic procedure for the creation of configuration constraints from Section 4.3 is followed. Figure 4.25 shows how the process for the creation of the disallowed pair constraint of $(Single\,Cup, 1-4\,CupBtn)$. Also notice how with the addition of constraints the ”Constrained $\#$ ” in the statistics panel has updated.



Object 1: Single Cup Option
- No Single Cup
- Single Cup

Object 2: Component
- LCD Clock
- Pwr Button
- Set Hour
- Set Minute
- AM PM Set
- Brew-Delay Set
- Temp Control Dial
- Brew Strength Btn
- 1-4 Cup Btn

Select Constraint:
○ Required Components
○ Required Pairs
● Disallowed Pairs

Constraints: Disallowed Pairs
- (Single Cup , w/ Grinder)
- (Single Cup , 1-4 Cup Btn)

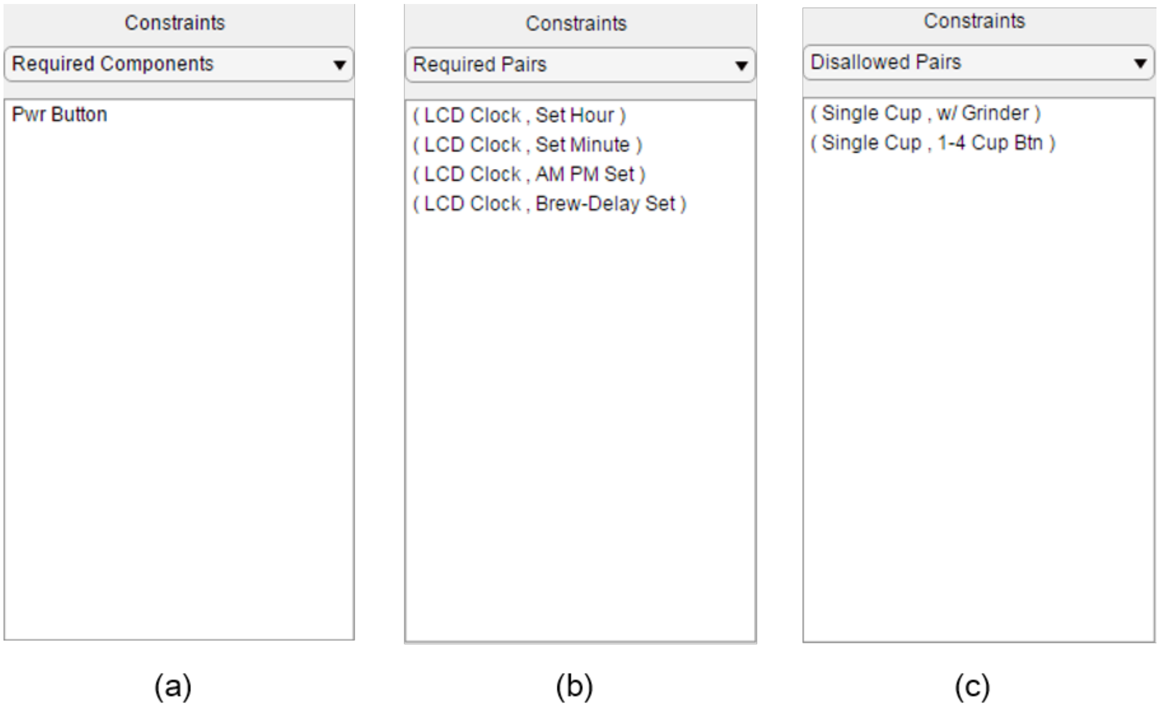
1. Select Objects
2. Select Constraint
3. Add Constraint
4. View Added Constraint

Components	9	Unconstrained #	2048	Products	0
Config Options	4	Constrained #	40	Layouts	Test Required
Fascia	2	Current Reduction	98.05%		

Results of Constraints

Figure 4.25: Constrain Tab for Example

This same procedure for constraint creation in the program was used for all constraints in the product family. Figure 4.26 displays all of the configurations created for the product family. The statistics panel in Figure 4.25 represents the results of all of these.



Pwr Button

(a)

(LCD Clock , Set Hour)
(LCD Clock , Set Minute)
(LCD Clock , AM PM Set)
(LCD Clock , Brew-Delay Set)

(b)

(Single Cup , w/ Grinder)
(Single Cup , 1-4 Cup Btn)

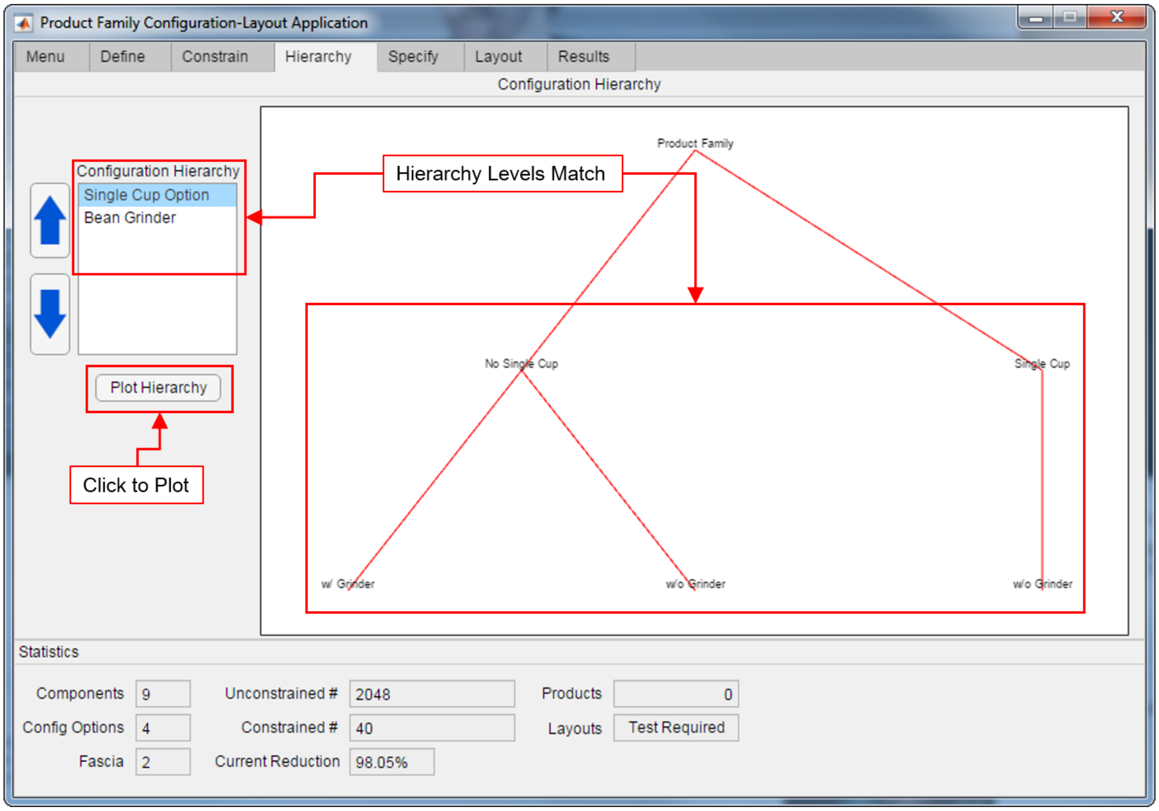
(c)

Figure 4.26: Configuration constraints added: (a) Required Components, (b) Required Pairs, (c) Disallowed Pairs

This tab showed the ease of adding configuration constraints through this program. The next step is to view the hierarchy of the configurations.

4.4.3 View Hierarchy

The Configuration hierarchy is defined in this tab. For this product family it was decided that the Single Cup option would be the higher Category in the hierarchy. Figure 4.27 presents the Hierarchy tab for the coffeemaker product family. Notice how the ordering in the list box matches the ordering of the categories in the plot. Since the order that the configuration categories were entered in the Define tab as the desired hierarchy order, no other changes need to be made before clicking plot.



Hierarchy Levels Match

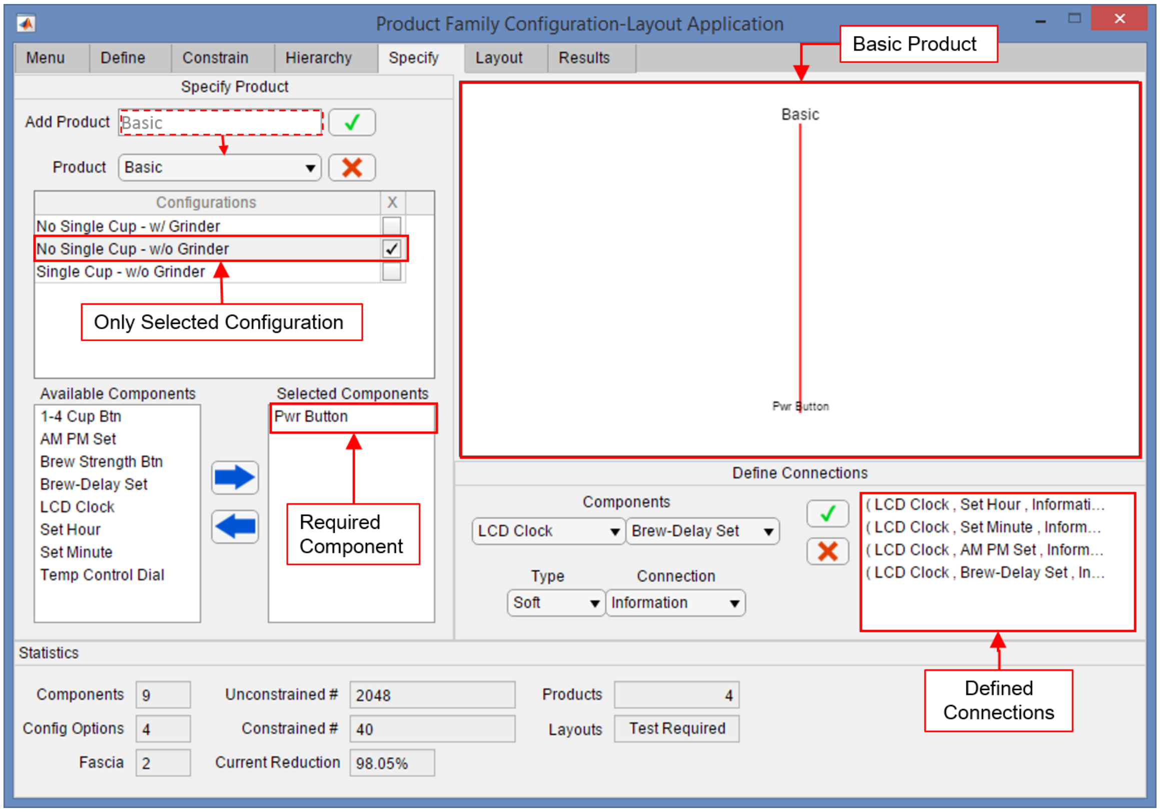
Click to Plot

Components	9	Unconstrained #	2048	Products	0
Config Options	4	Constrained #	40	Layouts	Test Required
Fascia	2	Current Reduction	98.05%		

Figure 4.27: Hierarchy Example

4.4.4 Specify Products

It is now time to star defining products for this product family. The designer in this case chose to have three levels of products. To start with the designer decided to have a bare minimum coffeemaker. This resulted in no components besides the required components being added, and the selection from the possible configurations being the minimum. Figure 4.28 presents the specification for the ”Basic” product developed.



Basic Product

Add Product: Basic

Product: Basic

Configurations	X
No Single Cup - w/ Grinder	☐
No Single Cup - w/o Grinder	☑
Single Cup - w/o Grinder	☐

Only Selected Configuration

Available Components:
1-4 Cup Btn
AM PM Set
Brew Strength Btn
Brew-Delay Set
LCD Clock
Set Hour
Set Minute
Temp Control Dial

Selected Components:
Pwr Button

Required Component

Components: LCD Clock, Brew-Delay Set

Type: Soft

Connection: Information

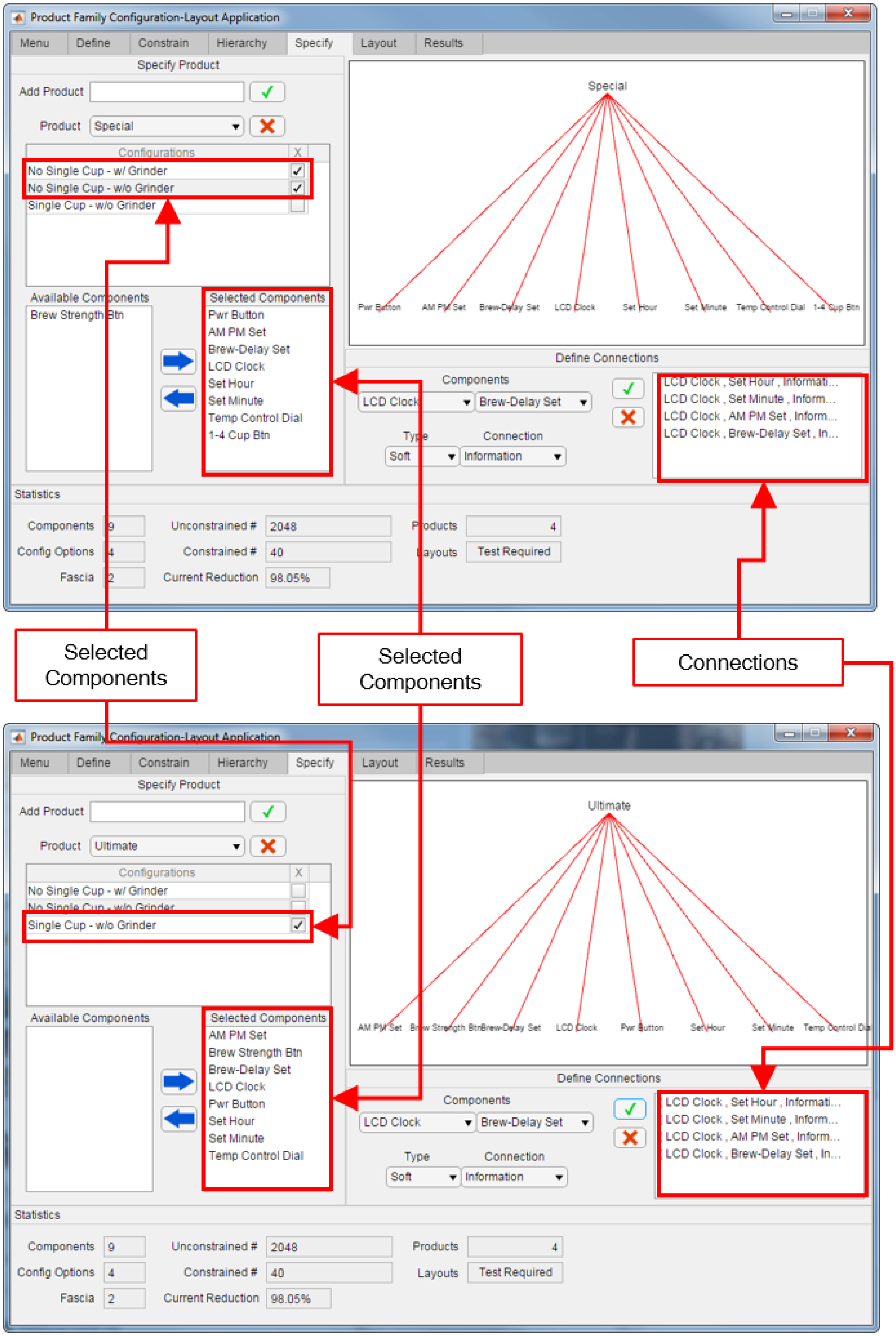
(LCD Clock , Set Hour , Informati…
(LCD Clock , Set Minute , Inform…
(LCD Clock , AM PM Set , Inform…
(LCD Clock , Brew-Delay Set , In…

Defined Connections

Components	9	Unconstrained #	2048	Products	4
Config Options	4	Constrained #	40	Layouts	Test Required
Fascia	2	Current Reduction	98.05%		

Figure 4.28: ”Basic” Coffeemaker Specification

Figure 4.28 points out the main parts of the product and also highlights connections that have been defined for the product family. Two more products have also been developed and are shown in Figure 4.29. These are the ”Special” (top half of figure) and ”Ultimate” (bottom half of figure) products. where the designer decided they wanted to attempt to create more variety in the ”Special” than the ”Ultimate” and has chosen 2 configurations for production. This could be a subjective decision or it could be a decision to attempt to reach a market section that had previously not been reached.



Product Family Configuration-Layout Application

Specify Product

Add Product:

Product: Special

Configurations	X
No Single Cup - w/ Grinder	☑
No Single Cup - w/o Grinder	☑
Single Cup - w/o Grinder	□

Available Components:
Brew Strength Btn

Selected Components:
Pwr Button
AM PM Set
Brew-Delay Set
LCD Clock
Set Hour
Set Minute
Temp Control Dial
1-4 Cup Btn

Define Connections

Components: LCD Clock, Brew-Delay Set

Type: Soft

Connection: Information

LCD Clock , Set Hour , Informati...
LCD Clock , Set Minute , Inform...
LCD Clock , AM PM Set , Inform...
LCD Clock , Brew-Delay Set , In...

Statistics

Components	9	Unconstrained #	2048	Products	4
Config Options	4	Constrained #	40	Layouts	Test Required
Fascia	2	Current Reduction	98.05%		

Product Family Configuration-Layout Application

Specify Product

Add Product:

Product: Ultimate

Configurations	X
No Single Cup - w/ Grinder	□
No Single Cup - w/o Grinder	□
Single Cup - w/o Grinder	☑

Available Components:

Selected Components:
AM PM Set
Brew Strength Btn
Brew-Delay Set
LCD Clock
Pwr Button
Set Hour
Set Minute
Temp Control Dial

Define Connections

Components: LCD Clock, Brew-Delay Set

Type: Soft

Connection: Information

LCD Clock , Set Hour , Informati...
LCD Clock , Set Minute , Inform...
LCD Clock , AM PM Set , Inform...
LCD Clock , Brew-Delay Set , In...

Statistics

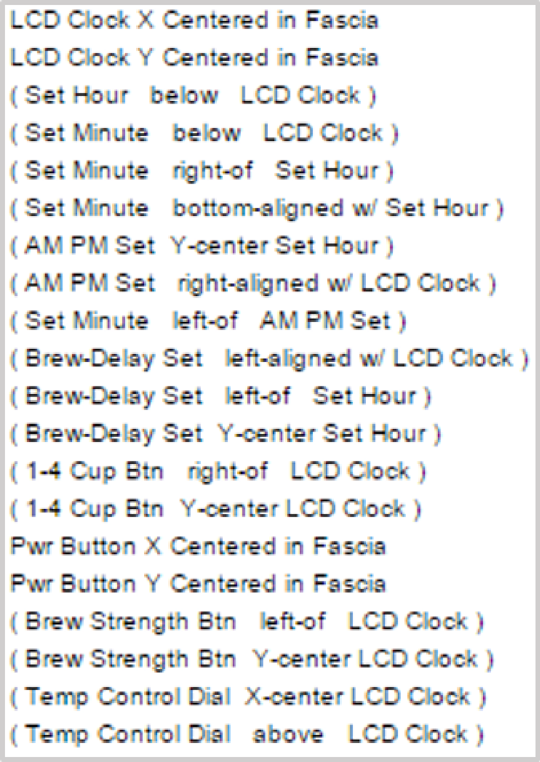
Components	9	Unconstrained #	2048	Products	4
Config Options	4	Constrained #	40	Layouts	Test Required
Fascia	2	Current Reduction	98.05%		

Figure 4.29: ”Special” and ”Ultimate” Coffeemaker Specification

This tab configured the products to be made in this product family. Out of the 40 possible products, 4 were chosen for production. Also notice that the connections box in each screen capture from Figures 4.28 and 4.29 contains the same contents regardless of the components each product contains. This is used to show all connections in the product family for the user to reference during later design stages. The next step is to determine the layouts of these products.

4.4.5 Layout Components

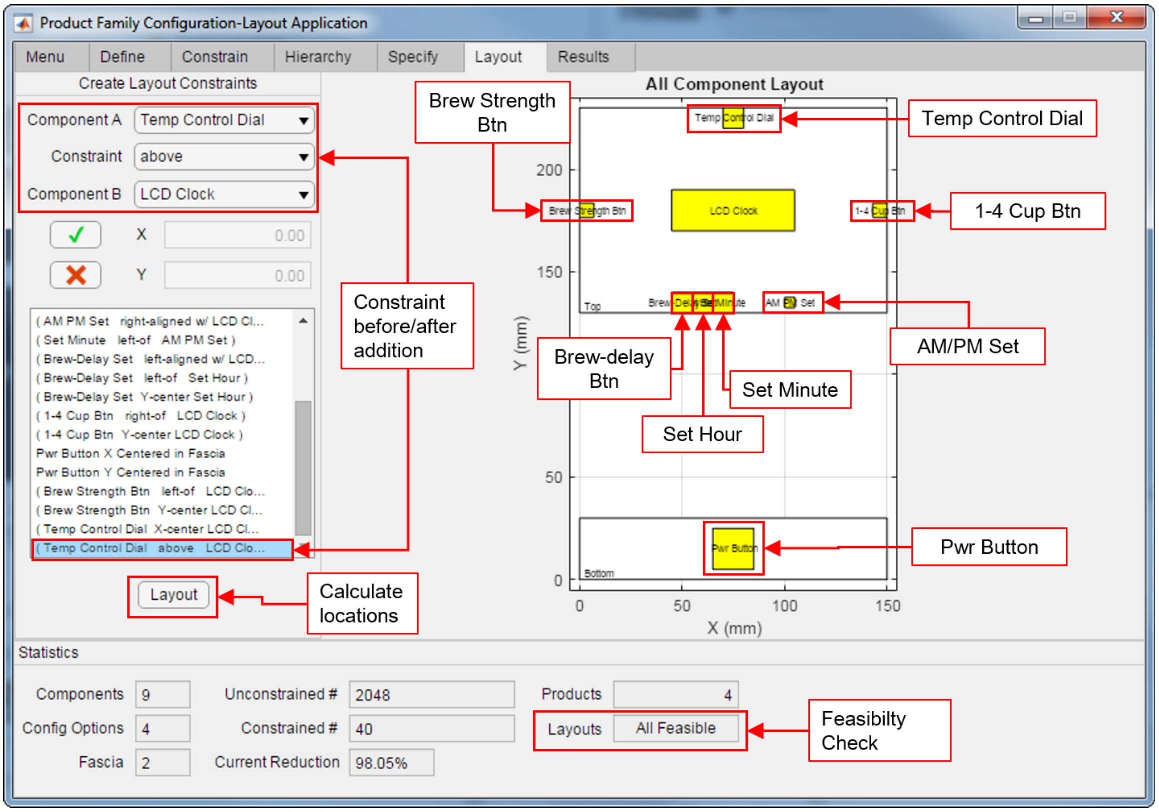
Now is time to layout the components within this product family. The layout decisions are a combination of the designers judgment on desired layout and usability of the product. While this won't result in a necessarily aesthetically pleasing layout it would allow the designer to see whether or not the the desired layout of components is feasible. Figure 4.30 lists the layout constraints defined for this product family.



LCD Clock X Centered in Fascia
LCD Clock Y Centered in Fascia
(Set Hour below LCD Clock)
(Set Minute below LCD Clock)
(Set Minute right-of Set Hour)
(Set Minute bottom-aligned w/ Set Hour)
(AM PM Set Y-center Set Hour)
(AM PM Set right-aligned w/ LCD Clock)
(Set Minute left-of AM PM Set)
(Brew-Delay Set left-aligned w/ LCD Clock)
(Brew-Delay Set left-of Set Hour)
(Brew-Delay Set Y-center Set Hour)
(1-4 Cup Btn right-of LCD Clock)
(1-4 Cup Btn Y-center LCD Clock)
Pwr Button X Centered in Fascia
Pwr Button Y Centered in Fascia
(Brew Strength Btn left-of LCD Clock)
(Brew Strength Btn Y-center LCD Clock)
(Temp Control Dial X-center LCD Clock)
(Temp Control Dial above LCD Clock)

Figure 4.30: Layout Constraints

These constraints result in a basic layout which appears in Figure 4.31. This shows that the designers basic layout is feasible. The figure also shows how a constraint appears prior to to be added and when a constraint is added to the list of constraints. Arrows are used to point out the labels on the plot as they may be difficult to read on the plot. Notice that both the Top and Bottom fascia are plotted in this figure. Showing the locations for every component in the product family. The ”Pwr Button” is the only component in the bottom fascia because it is the only component that as to be specified to be present in the bottom fascia. This means the ”Basic” model only makes use of the Bottom fascia and will not require components needed to configure a Top fascia.



Product Family Configuration-Layout Application

Create Layout Constraints

Component A: Temp Control Dial

Constraint: above

Component B: LCD Clock

X: 0.00

Y: 0.00

(AM PM Set   right-aligned w/ LCD Cl...
(Set Minute   left-of   AM PM Set)
(Brew-Delay Set   left-aligned w/ LCD...
(Brew-Delay Set   left-of   Set Hour)
(Brew-Delay Set   Y-center Set Hour)
(1-4 Cup Btn   right-of   LCD Clock)
(1-4 Cup Btn   Y-center LCD Clock)
Pwr Button X Centered in Fascia
Pwr Button Y Centered in Fascia
(Brew Strength Btn   left-of   LCD Clo...
(Brew Strength Btn   Y-center LCD Cl...
(Temp Control Dial   X-center LCD Cl...
(Temp Control Dial   above   LCD Clo...

Statistics

Components: 9

Config Options: 4

Fascia: 2

Unconstrained #: 2048

Constrained #: 40

Current Reduction: 98.05%

Products: 4

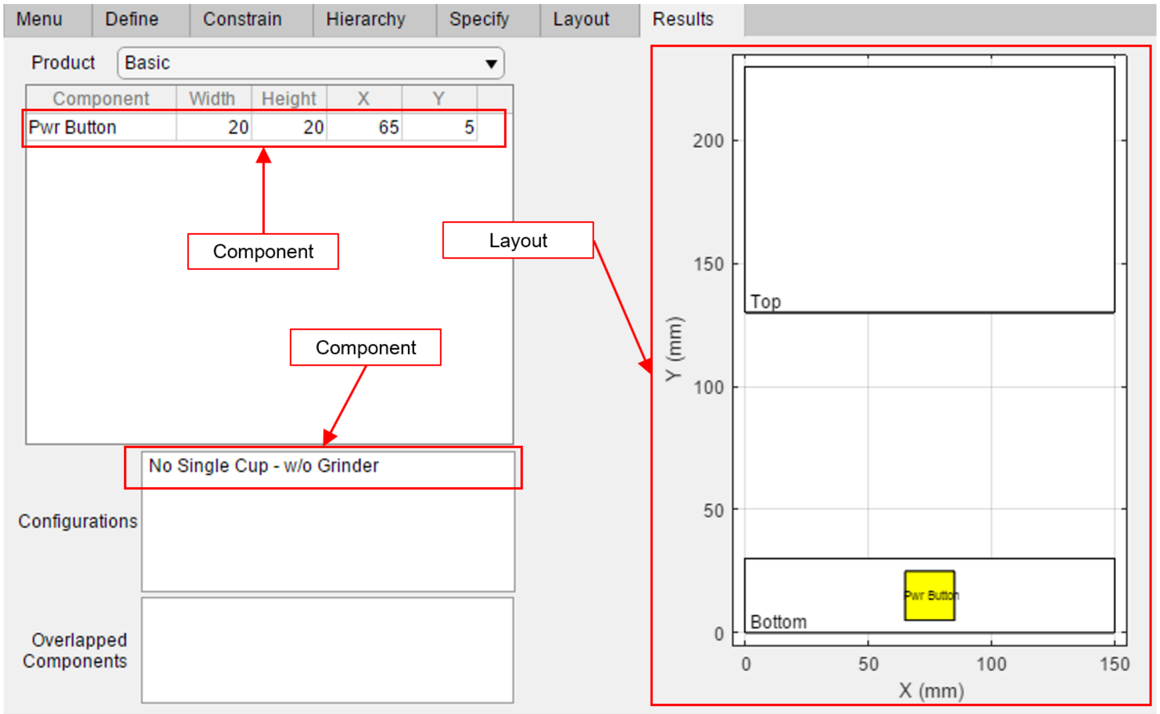
Layouts: All Feasible

Figure 4.31: Coffeemaker Product Family Layout

This layout represents the locations of components for any product which they appear in. Determining common locations of components throughout the product family. Since all products are defined and the layouts are feasible, the product family has been successfully developed the final step is to review the products and their layouts.

4.4.6 Review Results of Generation

The final step of the product family generation process is to review the products which have been defined and their layouts. Figure 4.32 presents the result for the generation of the ”Basic” coffeemaker. Note that the overlap box is empty since there are no overlaps in this product or the product family as a whole.



Product: Basic

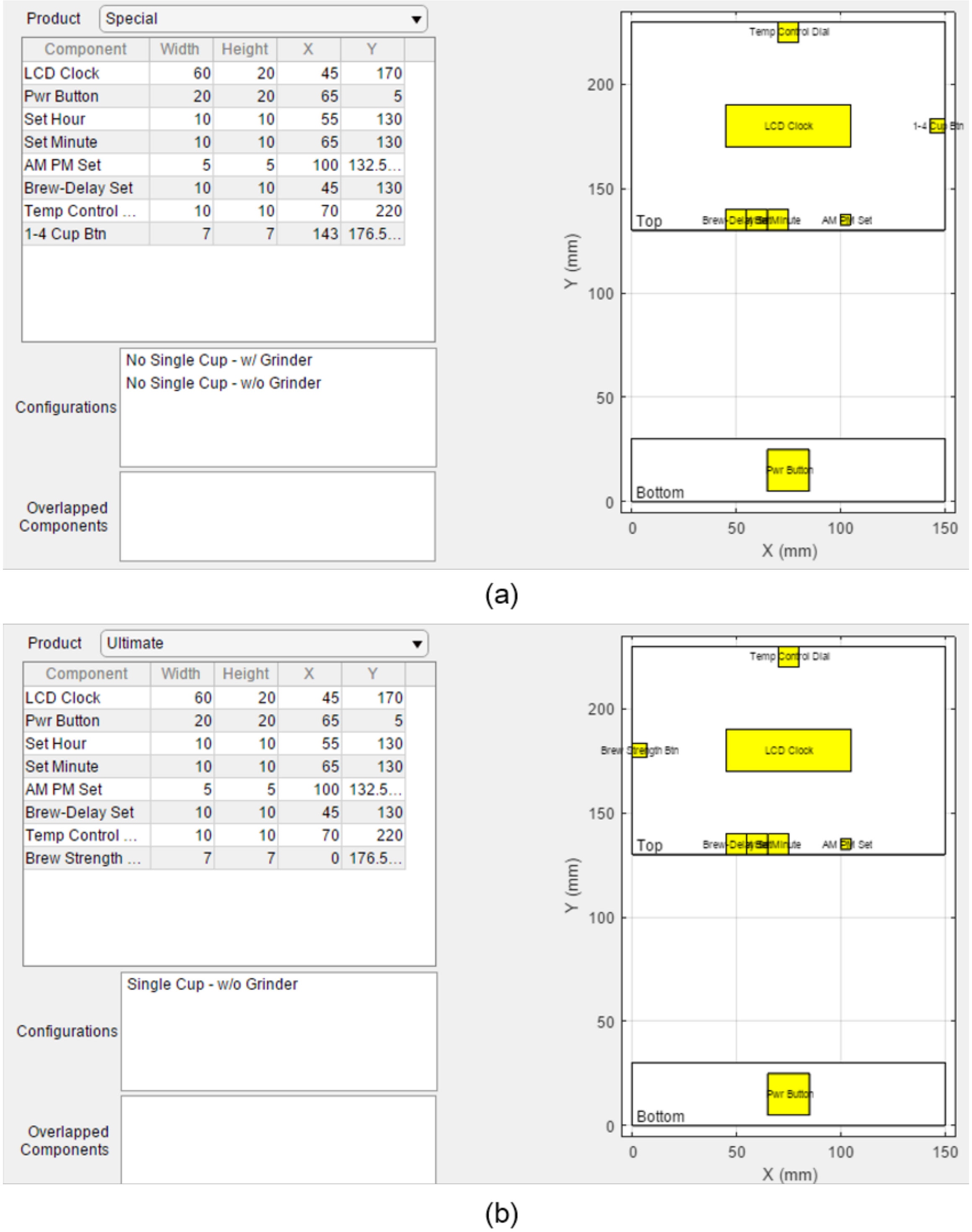
Component	Width	Height	X	Y
Pwr Button	20	20	65	5

Configurations: No Single Cup - w/o Grinder

Overlapped Components

Figure 4.32: ”Basic” Coffeemaker Results

Now that the results for a single product in the coffeemaker product family has been explored. an understanding of the results for the other two products can be seen. Figure 4.33 shows the results for the (a) ”Special” and (b) ”Ultimate” products. Notice the differences in the components each product contains, and their plots.



(a)

Product: Special

Component	Width	Height	X	Y
LCD Clock	60	20	45	170
Pwr Button	20	20	65	5
Set Hour	10	10	55	130
Set Minute	10	10	65	130
AM PM Set	5	5	100	132.5...
Brew-Delay Set	10	10	45	130
Temp Control ...	10	10	70	220
1-4 Cup Btn	7	7	143	176.5...

Configurations:
No Single Cup - w/ Grinder
No Single Cup - w/o Grinder

Overlapped Components:

(b)

Product: Ultimate

Component	Width	Height	X	Y
LCD Clock	60	20	45	170
Pwr Button	20	20	65	5
Set Hour	10	10	55	130
Set Minute	10	10	65	130
AM PM Set	5	5	100	132.5...
Brew-Delay Set	10	10	45	130
Temp Control ...	10	10	70	220
Brew Strength ...	7	7	0	176.5...

Configurations:
Single Cup - w/o Grinder

Overlapped Components:

Figure 4.33: (a) ”Special” and (b) ”Ultimate” Coffeemaker Results

This final step in the product family generation presents the results of the previous tabs and allows the designer to see the individual products that have been designed and decided if any changes should be made.

4.5 Summary

In this section the software developed for this thesis was presented along with its background, data structures, processes, work flow, and an example of its use. The data structures provide insight into how the data flows throughout the program and which data should organized for ease of access. The processes provide background as to how the set theory and linear programming portions of the methods presented in this thesis can be implemented into code. Pseudo code was provided to provide insight into the implementation. An overview of the GUI of the program was presented to show the capabilities and instruction to its use. Finally the example demonstrated the steps for the generation of a simple coffeemaker product family. This helped provide instructions for the use of the program on a simple product family.

The work of this chapter helps to back up the research questions and hypotheses presented in Chapter 1. This first research question on capturing spatial relationships prior to embodiment design is greatly aided in this chapter. The spatial grammar presented in Table 3.6 is used in the software created to present the work of this thesis. The use of these layout constraints in the software allow the user to automatically generate component layouts from a small number of simple constraints. This shows the feasibility of the desired layout prior to full on physical design. The users ability to create grammar based constraints and instantly see the results of the constraints allows for faster conceptual design.

The second research question of determining the effects of design decisions early on and its hypothesis is also addressed by the software as well. The software developed shows the effects of configuration constraints on the number of available products instantly. This feedback allows designers to better understand how each constraint effects the overall product family. The ability to create only feasible products is a great aid in allowing the designer to shape product families. In the area of component layout, the ability for the designer to create layout constraints and automatically see the results on both the overall product family but the individual products provides insight into the effects of desired layout and physical design during the conceptual design stage.

The work of this chapter presented how the methods presented in Chapter 3 could be implemented into a software application. This software helps to address the research questions proposed in 1. The next chapter presents case studies of the methods presented in this thesis using the software presented in this chapter. This chapter should provide insight into how the larger product families in the next chapter are set up and configured.

CHAPTER 5

CASE STUDIES: AUTOMOTIVE INTERFACES

The previous chapter presented the method and an example of the application. In this chapter a case study for the user interface of a car. A car provides sufficient configuration design space and tight spatial constraints to demonstrate the advantage of this method and address the second research question. This will show the effects of configuration design decisions on spatial layout as a matter of product family design. A car has multiple user interface areas, but for this case study the focus will be on the layout of indicators, dials, lights, buttons, and displays within the instrument panel and center console of the vehicle. Car manufacturers usually release multiple versions of the same vehicle with varying features and options to reach a different market sects. These case studies will provide demonstrate the generation process for a large number of possible combinations, which will be constrained to achieve the desired variety. This chapter will also demonstrate the methods presented in this thesis applied to different ends of the configuration design spectrum. Where the Center Console is often able to have larger amounts of variety since very few of its elements are crucial for vehicle operation and are more likely to vary to meet the desires of the user. While the Instrument panel is much more constrained in the elements that must be present in each form of the vehicle and their layout for the user to quickly and easily determine the information being displayed. To complement these case studies, reasons and explanations from the viewpoint of a theoretical designer are also included. This will proved examples of how the different constraints and user inputs can be used and possible reasons as to why particular decisions were made.

5.1 Center Console

The center console of a car usually houses elements for both the radio/stereo system and climate control of the vehicle. There can be a large amount of variation in the center console to allow the manufacturer to reach different priced points and create multiple variations of the same car model. There are often many components in each of center consoles but may also be few depending on the variety the designer is attempting to create within the product family. The stereo system and the climate controls are important aspects of the vehicle because they're only purpose is user comfort and enjoyment. This means the layout of these systems are important to allow the user to change settings with little effort and effect of the drivers ability to safely navigate the road. This case study presents the layout of the center console for multiple versions of the same automobile in the form factor of a 2-door sports car. In the later stages of the configuration process some of the decisions are made based on attempting to reach different types of consumer for this vehicle category. This case study will be presented in the flow of the software presented in the previous chapter. Starting with definition of the components and configuration options in the next section.

5.1.1 Define Components and Configuration options

In this section the components and configuration options for the center console product family is presented. For the layout portion of this product family there are two layout locations, the Entertainment Region and the Climate Control Region. In this product family there are 22 components between the entertainment and climate control regions of the center console. The components are defined along with their dimensions and fascia selection in Table 5.1.

Table 5.1: Dimensions of Center Console Components

Components	Dimensions (mm)	Fascia Choice
Volume Dial	(30,30)	Entertainment Region
Channel Dial	(30,30)	Entertainment Region
AM/FM Btn	(40,20)	Entertainment Region
Sound Btn	(40,20)	Entertainment Region
CD Slot	(125,7)	Entertainment Region
Insert/Eject Btn	(15,7)	Entertainment Region
Touch LCD w/ Integrated CD Slot	(150,85)	Entertainment Region
Favorite Stations Btns	(150,15)	Entertainment Region
CD Btn	(40,20)	Entertainment Region
Temperature Dial	(30,30)	Climate Control Region
A/C on/off	(40,20)	Climate Control Region
A/C Vent Selector	(40,20)	Climate Control Region
Rear Window Defroster Btn	(40,20)	Climate Control Region
Recirculate Air Btn	(40,20)	Climate Control Region
Max A/C Btn	(40,20)	Climate Control Region
Display Btn	(40,20)	Entertainment Region
Seek Btn	(20,20)	Entertainment Region
Track Btn	(20,20)	Entertainment Region
Basic Radio LCD	(150,40)	Entertainment Region
A/C Power Dial	(30,30)	Climate Control Region
Auxilary Audio Port	(5,5)	Entertainment Region
Menu Btn	(40,20)	Entertainment Region

Most of the Components are self explanatory but a couple require further explanation. The Sound Btn is used to access the menu to adjust audio settings such as bass, treble, fade, and balance. The Menu Btn allows the user to access other radio settings such as Bluetooth, Backup Camera or other Vehicle information. The Basic Radio LCD is the standard monochrome LCD that displays time, station and other basic information, while the LCD w/ Integrated slot is a full color touch screen LCD screen.

This product family also has 3 configuration categories which have atleast 2 configuration options. Table 5.2 defines the configuration category and options present in this product family. These represent different types of features that are desirable and maybe used to create variety among different versions of the same car.

Table 5.2: Configuration Options for Center Console Product Family

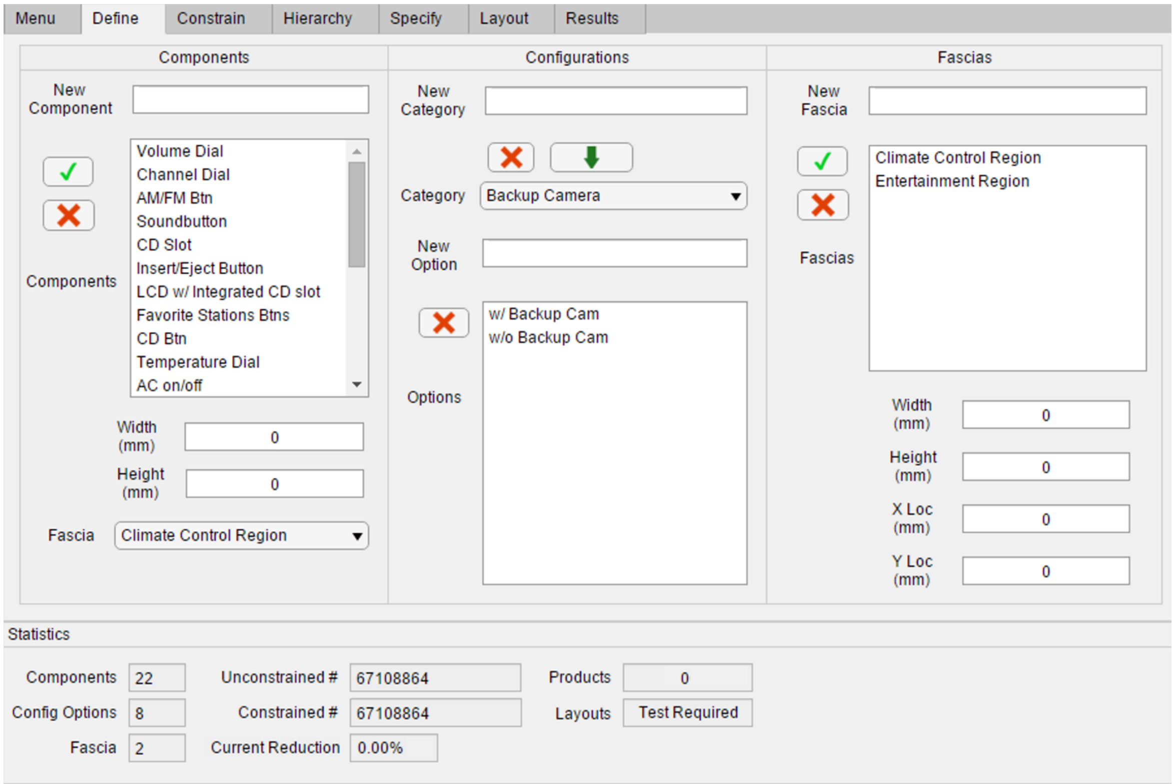
Configuration Category	Configurations Options
Backup Camera	w/ Backup Cam
	w/o Backup Cam
Auto Climate Control	w/ Thermostat
	w/o Thermostat
Auto Climate Control	w/ Sat Radio
	w/o Sat Radio
	w/ Sat Radio +Bluetooth
	w/o Sat Radio + Bluetooth

For clarification the inclusion of the ”Thermostat” provides the ability for the user to set a desired temperature rather than having to balance the A/C Power with the Temperature control dial and cannot be set to a specific temperature. Finally the the different regions of the fascia are defined. As mentioned previously this family has 2 different layout region, Entertainment and Climate Control. The Fascia regions, along with their Dimensions and Global coordinates from the lower left corner are presented in Table 5.3.

Table 5.3: Dimensions and Location of Center Console Fascia

Fascias	Dimensions (mm)	Location (X, Y)
Climate Control Region	(200,70)	(0,0)
Entertainment Region	(200,150)	(0,70)

The component, configuration, and fascia elements have been defined and are able to be entered into the Define tab of the software presented in the previous chapter. This is presented in Figure 5.1.



Components

New Component:

Components:
- Volume Dial
- Channel Dial
- AM/FM Btn
- Soundbutton
- CD Slot
- Insert/Eject Button
- LCD w/ Integrated CD slot
- Favorite Stations Btns
- CD Btn
- Temperature Dial
- AC on/off

Width (mm): 0

Height (mm): 0

Fascia: Climate Control Region

Configurations

New Category:

Category: Backup Camera

New Option:

Options:
- w/ Backup Cam
- w/o Backup Cam

Fascias

New Fascia:

Fascias:
- Climate Control Region
- Entertainment Region

Width (mm): 0

Height (mm): 0

X Loc (mm): 0

Y Loc (mm): 0

Statistics

Components	22	Unconstrained #	67108864	Products	0
Config Options	8	Constrained #	67108864	Layouts	Test Required
Fascia	2	Current Reduction	0.00%		

Figure 5.1: Define Tab for the Car Center Console Case Study

Notice that most of the elements defined in the previous tables are able to be seen the number of components, configuration options and fascia's listed in the Statistics panel reflect this. The number of Unconstrained combinations in the statistics panel, 67 million, should also be noted as this is a large number of possible combinations that will be reduced when configuration constraints are defined. This section defined the components and configurations that will make up the product family. In the next section constraints are defined to reduce the number of possible combinations of components and configurations to a more manageable number.

5.1.2 Constrain Center Console Combinations

In the previous section the different aspects of the center console were defined and the scale of the configuration problem was revealed in the number of unconstrained combinations of components and configuration options. In this section, configuration

]

constraints are defined to reduce the total number of possible combinations. These constraints are based on designer preference and Requirements for the product family to perform its basic functions. The constraints, Required Components, Required Pairs, and Disallowesd Pairs, were presented in previous chapters. To aid in understanding the use of these constraints, explanations and reasonings for some of the constraints will be provided. The first defined constraint is the Required Components. These components are defined as universal to all models and sometimes required to perform the base functions of the product. Table 5.4 presents the Required Components.

Table 5.4: Center Console Required Components

Required Components
A/C Vent selector
A/C on/off
Channel Dial
Rear Window Defroster Btn
Recirculate Air Btn
Volume Dial

For this product family, 6, components were designated as mandatory. The designer decided that the user should easily be able to change the station or radio volume without having to take their eyes of the road in all models. This requires a tactile input mechanism. The designer chose to make the Volume dial and Channel dial mandatory to accomplish this. These provide a tactile input that the user can find without looking. While these may not be necessary if the Automobile has the LCD Touch Screen, the designer believes having physical dials will improve the user experience. The designer required the A/C Vent Selector, on/off, and Recirculate Air Button to provide all models some form of A/C. The Rear Window Defroster Btn was included as a functional requirement as the designer would like all models to be able to defrost the rear window. The reasons for requiring these components varies from Safety to Comfort to Ease-of-use. These are all factors a designer may consider when configuring the product family.

The next constraint to be addressed is the Required Pairs constraint. These required components to always appear together in a product and the reasons behind them can vary. Table 5.5 presents the Required Pair constraints. Where each row contains a pair of components/configuration options.

Table 5.5: Center Console Required Pairs

Required Pairs		
AM/FM Btn	-	Basic Radio LCD
CD Slot	-	Basic Radio LCD
Sound Button	-	Basic Radio LCD
Seek Btn	-	TrackBtn
TrackBtn	-	Basic Radio LCD
w/ Backup Cam	-	Touch LCD w/ Integrated CD slot
Touch LCD w/ Integrated CD slot	-	w/ Thermostat
Basic Radio LCD	-	Menu Button
w/ Backup Cam	-	w/ Thermostat
Temperature Dial	-	A/C Power Dial
Insert/Eject Button	-	CD Slot

In this product family, 11 Required pairs were defined. Since the Basic Radio LCD can only display information, it requires other components to perform its basic functions as a radio. To ensure that Basic Radio LCD meets its minimal functional requirements the designer decided that the AM/FM Btn, CD Slot, Sound Button, Track, Seek and Menu Button need to be included with the Basic LCD. To create value in the models that may contain the Touch LCD the designer decided that Every model that contains the Touch LCD will have a Backup Camera and a Thermostat for temperature control. The constraints not discussed will have similar reasonings.

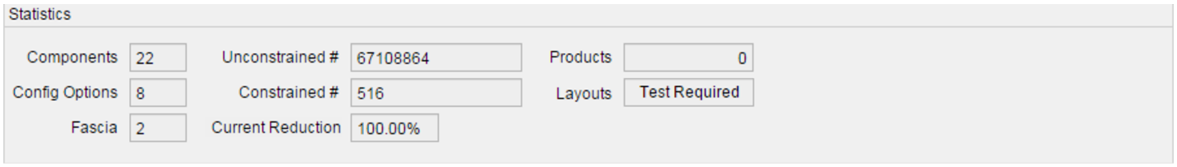
The final configuration constraint is the Disallowed Pairs which are pairs of objects that may not be present within the same product. Table 5.6 presents the Disallowed Pairs where each row represents a pair of objects.

Table 5.6: Center Console Disallowed Pairs

Disallowed Pairs		
Favorite Stations Btns	-	Touch LCD w/ Integrated CD slot
AM/FM Btn	-	Touch LCD w/ Integrated CD slot
Insert/Eject Button	-	Touch LCD w/ Integrated CD slot
Basic Radio LCD	-	Touch LCD w/ Integrated CD slot
Display Btn	-	Touch LCD w/ Integrated CD slot
Temperature Dial	-	Touch LCD w/ Integrated CD slot
CD Btn	-	Touch LCD w/ Integrated CD slot
CD Slot	-	Touch LCD w/ Integrated CD slot
w/ Backup Cam	-	w/o Thermostat
w/o Backup Cam	-	w/ Thermostat
w/ Backup Cam	-	w/o Sat Radio
w/ Backup Cam	-	w/o Sat Radio + Bluetooth
Menu Button	-	LCD w/ Integrated CD slot
Touch LCD w/ Integrated CD slot	-	Max AC Btn
w/ Thermostat	-	Temperature Dial
w/ Thermostat	-	A/C Power Dial

The 16 Disallowed Pairs help to reduce the number of combinations. The Touch LCD does not require as many secondary input devices as the Basic LCD since the user is able to interact with the touch screen directly. For this reason the Touch LCD is listed with a number of components that the LCD will be able to just display buttons for. This will free up room in the Entertainment Region of the Fascia since the Touch LCD has a larger foot print than the Basic LCD. Also notice there are a number of constraints containing only configuration options. These are created by the designer to achieve desired levels of features when the products are defined. This creates a great reduction in possible combinations and allows the designer to better defined the desired variety.

The results of the constraints defined for the product family reveal a large reduction in the total number of combinations. Figure 5.2 displays the statistics panel after the application of the constraints presented above.



Statistics

Components	22	Unconstrained #	67108864	Products	0
Config Options	8	Constrained #	516	Layouts	Test Required
Fascia	2	Current Reduction	100.00%		

Figure 5.2: Statistics after Configuration Constraints have been defined for the Center Console product family

The reduction from 67,000,00 to 516 combination is the result. This results in a much more manageable set of possible products and only allows products that meet the designers requirements to be produced. This section presented the configuration constraints defined for the product family, the reasons behind them and their effects on the total number of possible combinations available. This show the users ability to greatly reduce a large number of combinations with basic constraints. The next section will present the configuration hierarchy for the product family.

5.1.3 Center Console Hierarchy

This section will present the Hierarchy for the Configuration Categories for the Center Console product family. The hierarchy will be presented and the designers possible reason discussed. The hierarchy for this product family has 3 levels and while the hierarchy order may not greatly effect the outcome of the product family it may allow the designer to better organize the developed products and better illustrate the constrained variety in configuration options. The different Configuration Categories were presented in Table 5.2. Figure 5.3 presents the hierarchy that was defined for the Center Console product family.

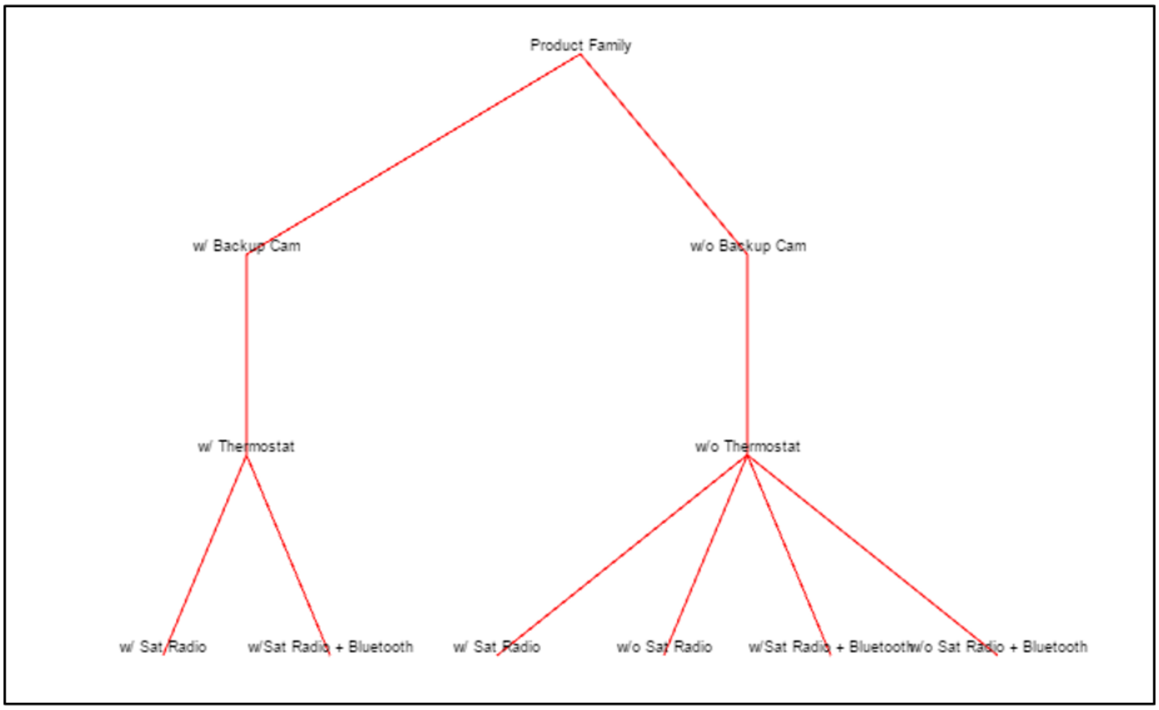


Figure 5.3: Hierarchy of Configuration options for Center Console Product Family

Notice that while the Backup Camera and Thermostat Category have the same effect on the plot of the hierarchy there was a reason behind the ordering. In this case a designer may have decided that the Backup Camera will rank higher than the thermostat because the inclusion of the backup camera means that the Touch LCD must be included in a product and will have a larger effect on the Entertainment region than the Climate Control Region. The lowest level of the hierarchy could have been decided because the configuration options at this level will not effect the overall user interface and have little effect on the physical design. While the hierarchy for these configuration categories only contains 3 levels there can be very specific reasons for their ordering. The next section will define feasible products from the components and configuration options based on the defined constraints.

5.1.4 Define Products

In this section a set of products will be defined from the components and configurations remaining after constraints are applied. The product definition process will include the Name of the product, its components and its available configurations. Each product will also be given a brief summary to explain where it fits into the product family from the perspective of a theoretical Designer. The designers decisions will be based on defining products to reach different different levels of drivers of a 2-door sports car as previously mentioned. For this product family 6 total products were defined and 4 base sets of components. The products are named ”Basic”, ”Premium”, ”Grand Touring”, and ”Speed” models. The ”Basic” and ”Grand Touring” models were given 2 configurations. Table 5.7 presents components lists for the products and Table 5.8 presents the selected configurations for each model, fully defining all 6 products. the columns of each Table represent a product and the rows represent a component or configuration.

Table 5.7: Center Console Product Definition

Component	Basic	Premium	Grand Touring	Speed
Volume Dial	X	X	X	X
Channel Dial	X	X	X	X
AM/FM Btn	X	X		X
Sound Btn	X	X		X
CD Slot	X	X		X
Insert/Eject Btn	X	X		X
LCD w/ Integrated CD Slot			X	
Favorite Stations Btns		X		
CD Btn	X	X		
Temperature Dial	X	X		
A/C on/off	X	X	X	X
A/C Vent Selector	X	X	X	X
Rear Window Defroster Btn	X	X	X	X
Recirculate Air Btn	X	X	X	X
Max A/C Btn		X		
Display Btn		X		
Seek Btn	X	X		X
Track Btn	X	X		X
Basic Radio LCD	X	X		X
A/C Power Dial	X	X		
Auxilary Audio Port		X	X	
Menu Btn	X	X		X

Notice in the tables 5.7 that each model has a unique combination of components. Each model was used to reach a different type of user and driver to fit in with different overall vehicle configurations. The ”Basic” Model would be fitted into the base version of a vehicle, with the goal to reach a lower price point by minimizing the Radio and A/C features. This model would likely also feature a basic cloth interior, less powerful engine, and basic exterior trim. The ”Premium” model would be a slightly upgraded version of the ”Basic” model, aimed at consumers with a slightly larger budget and has more Radio and A/C features. This model would likely include a leather interior and additional exterior trim, but the same engine as the ”Basic” model. The ”Grand Touring” model would be a fully upgraded version of the car, featuring the most advanced radio and climate control features. This model would likely include a better looking leather interior, upgraded sound system, more powerful engine, and ”faster” looking exterior trim. Providing the user with a more comfortable and exciting drive. Finally the ”Speed” model would likely be a stripped down version of the car meant for speed and the track. The purpose of this model is to minimize weight to maximize speed. This means removing as many non-necessary features as possible. This would include light racing seats, a nearly bare interior, the most powerful engine in the line, and exterior trim designed for the best aerodynamics. Each of the models developed in this product family is meant to reach distinct market sects and ideal users, from the budget concerned to the comfort and ride concerned to drivers concerned about the cars track day performance. The different component combinations are parts of the entire vehicle that allow the manufacturer reach specific markets.

Table 5.8: Product Configuration Selection

Configurations	Basic	Premium	Grand Touring	Speed
w/ Backup Cam - w/ Thermostat - w/ Sat Radio			X	
w/ Backup Cam - w/ Thermostat - w/Sat Radio + Bluetooth			X	
w/o Backup Cam - w/o Thermostat - w/ Sat Radio		X		
w/o Backup Cam - w/o Thermostat - w/o Sat Radio	X			
w/o Backup Cam - w/o Thermostat - w/Sat Radio + Bluetooth				
w/o Backup Cam - w/o Thermostat - w/o Sat Radio + Bluetooth	X			X

Table 5.8 shows the selected configurations for each model. Notice that the ”Grand Touring” model is the only model with a backup camera because it is the only model with the LCD Touch Screen. Notice that the rest of the models have selected configurations based on the different market described above. The products defined here will make up the entire product family. The next step of the process is to define layout constraints for the products and layout the components.

5.1.5 Center Console Component Layout

In this section constraints for the layout of the components for the center console product family. The fascia each component will appear in was defined in the Define stage. The constraints are defined from the set of layout constraints defined in Table 3.6. Since this is for the layout of user interface components for a vehicle much of the layout will be for user to easily interact with the desired systems while minimizing distraction from the road. This means a logical component layout must be created so the user can easily find components. To provide a basic layout for all components in the product family 60 constraints were defined. These are presented in Table 5.9. Each constraint cannot be discussed individually due to the large number of them. However, a few of the constraints will be discussed to provide insight into the reasons behind some of the layout constraints.

Table 5.9: Center Console Layout Constraints

Object A	Constraint	Object B/ Location
Basic Radio LCD	- X centered in fascia -	
LCD w/ Integrated CD slot	- X centered in fascia -	
Volume Dial	- below -	LCD w/ Integrated CD slot
Volume Dial	- below -	Basic Radio LCD
Channel Dial	- below -	Basic Radio LCD
Channel Dial	- below -	LCD w/ Integrated CD slot
Channel Dial	- right-aligned -	LCD w/ Integrated CD slot
Volume Dial	- left-aligned -	LCD w/ Integrated CD slot
Volume Dial	- left-aligned -	Basic Radio LCD
Channel Dial	- right-aligned -	Basic Radio LCD
Basic Radio LCD	- Y-center aligned -	LCD w/ Integrated CD slot
CD Slot	- above -	Basic Radio LCD
CD Slot	- left-aligned -	Basic Radio LCD
CD Slot	- Y-center aligned -	Insert/Eject Button
CD Slot	- left-of -	Insert/Eject Button
Basic Radio LCD	- right-aligned -	Insert/Eject Button
Favorite Stations Btns	- X-center aligned -	Basic Radio LCD
Favorite Stations Btns	- above -	Volume Dial
Favorite Stations Btns	- above -	Channel Dial
Favorite Stations Btns	- below -	Basic Radio LCD

Table Continued on Next Page

Continuation of Table 5.9

Object A	Constraint	Object B/ Location
AM/FM Btn	- right-of -	Volume Dial
AM/FM Btn	- left-of -	Channel Dial
AM/FM Btn	- below -	Favorite Stations Btns
Soundbutton	- below -	Favorite Stations Btns
Soundbutton	- right-of -	AM/FM Btn
Soundbutton	- left-of -	Channel Dial
Soundbutton	- right-of -	Volume Dial
Display Btn	- right-aligned -	Soundbutton
Display Btn	- below -	Soundbutton
CD Btn	- left-aligned -	AM/FM Btn
CD Btn	- below -	AM/FM Btn
LCD w/ Integrated CD slot	- Vertical Center at Y -	170 mm
A/C Power Dial	- left-aligned -	Volume Dial
Temperature Dial	- right-aligned -	Channel Dial
Temperature Dial	- Y-center aligned -	A/C Power Dial
Temperature Dial	- Y centered in fascia -	
Max AC Btn	- right-of -	A/C Power Dial
Recirculate Air Btn	- above -	Max AC Btn
Recirculate Air Btn	- right-aligned -	Max AC Btn
Recirculate Air Btn	- right-of -	A/C Power Dial
Rear Window Defroster Btn	- left-of -	Max AC Btn
Rear Window Defroster Btn	- right-of -	A/C Power Dial
AC Vent selector	- above -	Rear Window Defroster Btn
AC Vent selector	- left-aligned -	Rear Window Defroster Btn
A/C Power Dial	- bottom-aligned -	Rear Window Defroster Btn
A/C Power Dial	- bottom-aligned -	Max AC Btn
AC on/off	- X centered in fascia -	
AC on/off	- below -	Rear Window Defroster Btn
AC on/off	- below -	Max AC Btn
Seek Btn	- left-aligned -	Soundbutton
TrackBtn	- right-aligned -	AM/FM Btn
TrackBtn	- above -	Favorite Stations Btns
Seek Btn	- above -	Favorite Stations Btns
TrackBtn	- below -	Basic Radio LCD
Seek Btn	- below -	Basic Radio LCD
Auxiliary Audio Port	- top-aligned -	Basic Radio LCD
Menu Button	- right-of -	Seek Btn
Menu Button	- below -	Basic Radio LCD
Menu Button	- above -	Favorite Stations Btns
Menu Button	- right-aligned -	Favorite Stations Btns

End of Table

With there being 22 components and 60 constraints, that is about more than enough to well constrain each component since most constraints have 2 components that is 4-5 constraints per component. This is enough to constrain each component in all four directions and provide some additional fine tuning to the layout. Notice that LCD types being centered in the X direction axis of the fascia. This is to keep the layout symmetrical and provide a more aesthetically pleasing layout. The Channel and Volume dials are Directly below the LCD with the Volume dial to the left of the Channel dial. This is to provide an easy to understand interface and follow trends of other radio component layouts so the user can easily learn the layout of this interface. The next step is to layout the components based on the constraints. The software presented in the previous chapter was used to layout the components. Figure 5.4 presents the layout of all of the components in the product family. Notice the Entertainment Region on the top and the Climate Control Region below. While there are components that overlap in this plot, the overlapping components will never appear in the same product together. The next section will look at each product individually.

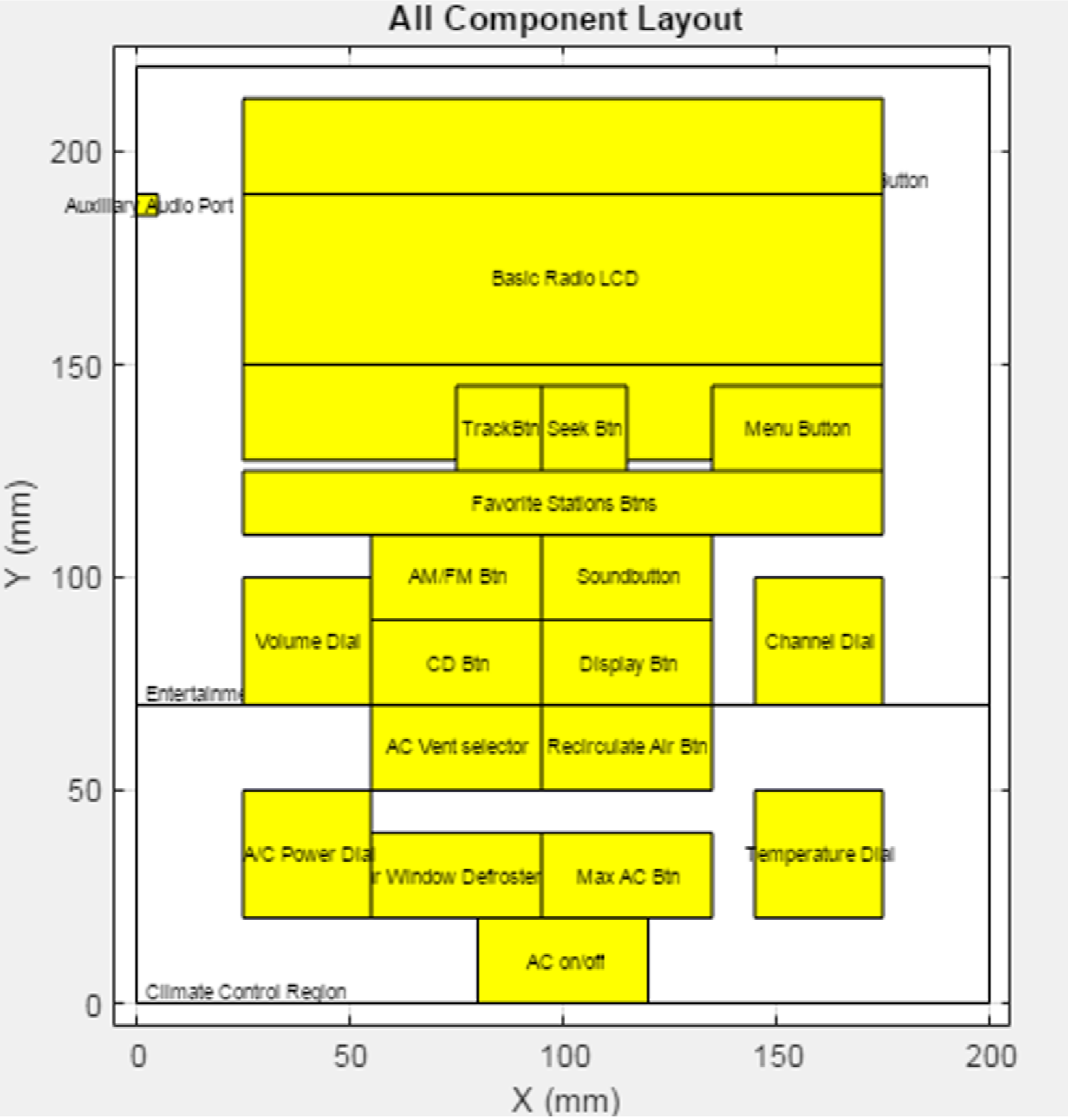


Figure 5.4: Center Console Product Family Full Layout

In this section layout constraints were defined and the components laid out. From this the location of each component for the entire product family was found. The overlap check found that this product family is feasible. The next section will examine the individual products and their layouts.

5.1.6 Review Products

In this section the products and their final layouts will be presented. The previous sections of this case review have been the steps to create these products. This section will just present the results. Tables 5.7 and 5.8 have already presented the product compositions and product definitions. This review will present the component layouts and a table of the final component locations. Figure 5.5 presents the layout of components for the "Base" model of the product family.

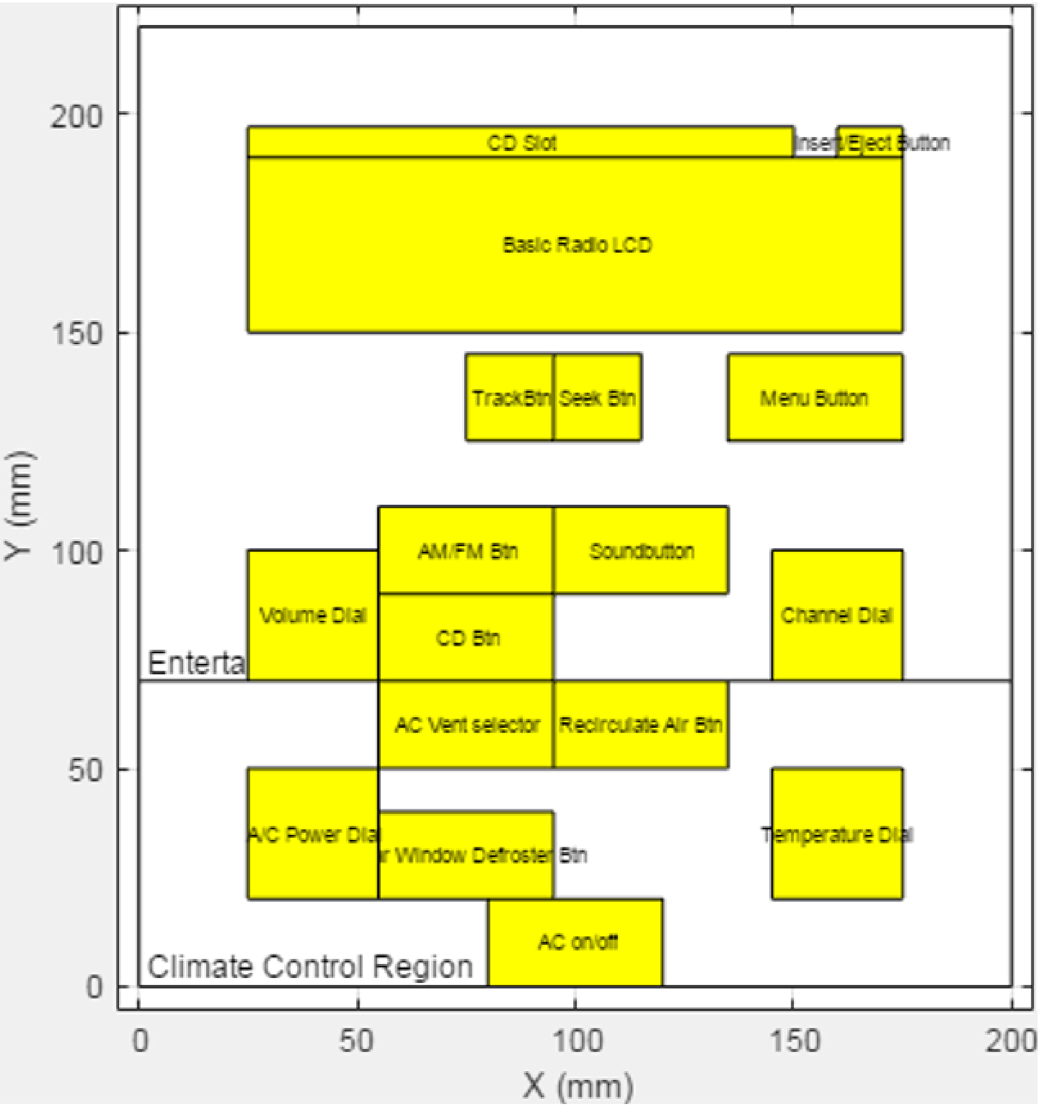


Figure 5.5: Center Console "Base" Product Layout

Notice that all of the components listed for the product are included in the plot and that all of the constraints are satisfied. Figure 5.6 presents the ”Premium” model layout. Figure 5.7 presents the ”Grand Touring” model and Figure 5.8 presents the ”Speed” model. These layouts reflect the component composition of the products and layout constraints of all of the components in the product family. They also reflect the market each model is aimed towards.

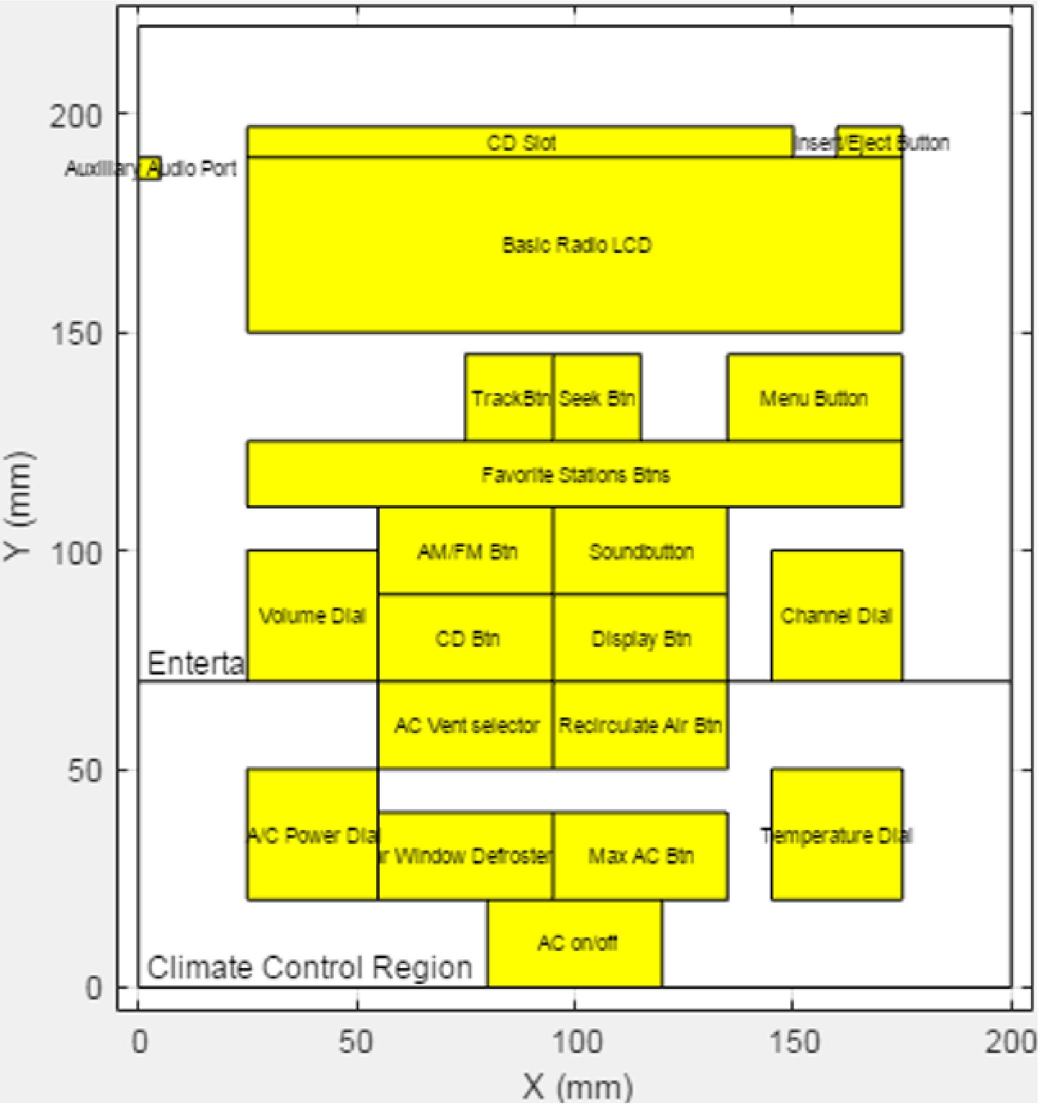


Figure 5.6: Center Console ”Premium” Product Layout

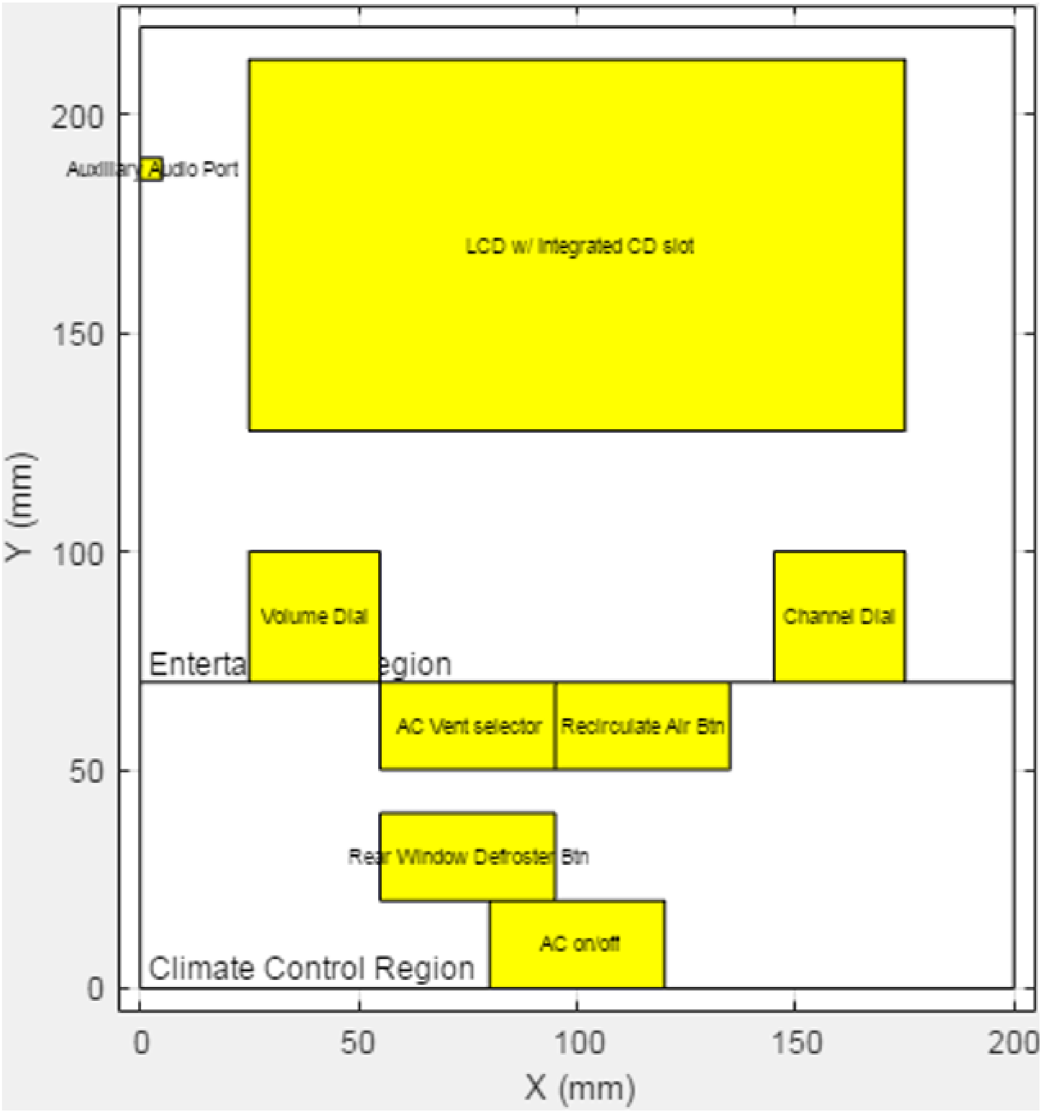


Figure 5.7: Center Console ”Grand Touring” Product Layout

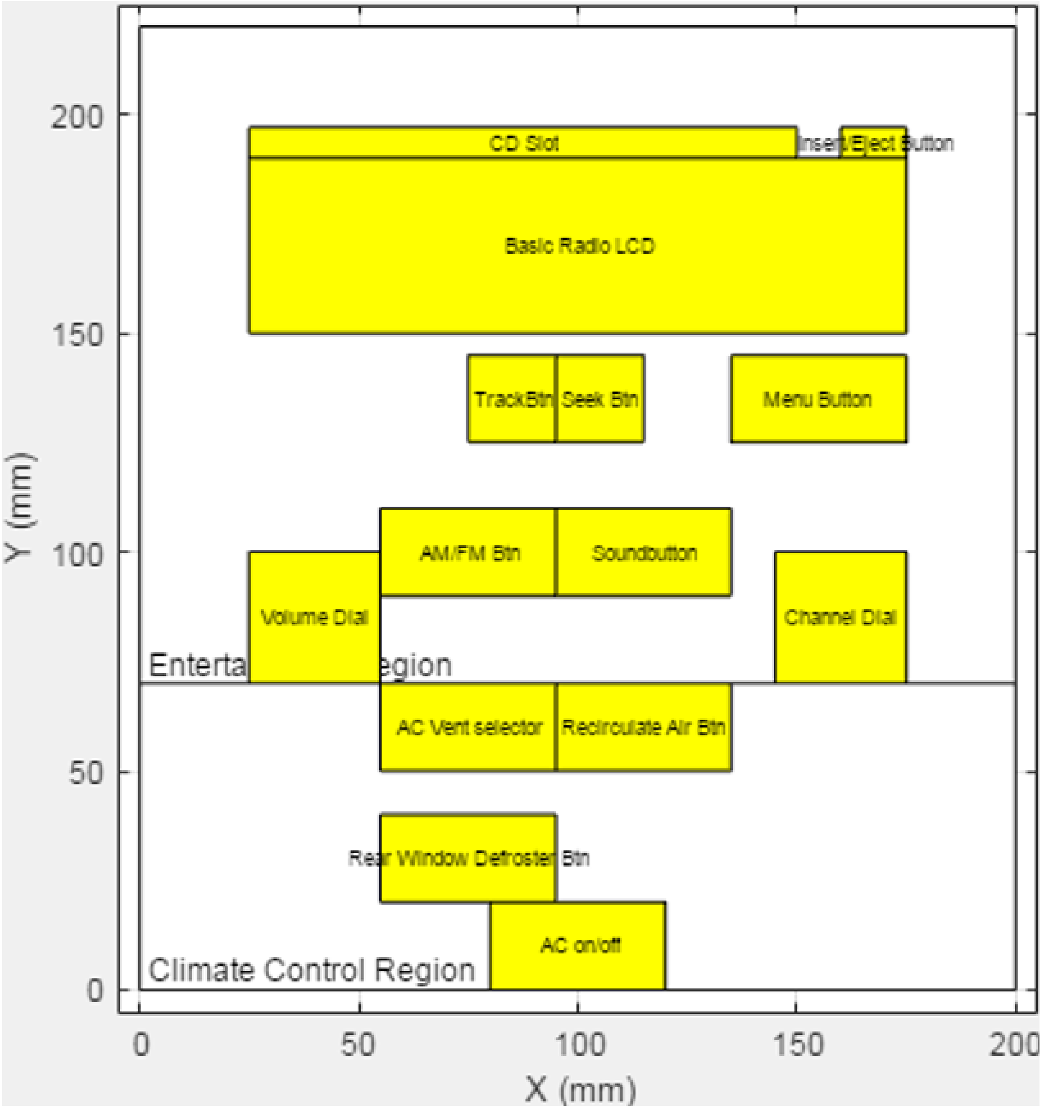


Figure 5.8: Center Console ”Speed” Product Layout

These plots show the layouts of each of the final products. Notice that for each product a component appears in, it is in the same location. This will allow for a simplified desight process with fewer unique components since components can easily be interchanged or not used. Finally Table 5.10 presents the calculated final X and Y coordinates for the components along with their height and width.

Table 5.10: Center Console Product Results

Component	Width mm	Height mm	X mm	Y mm
Volume Dial	30	30	25	70
Channel Dial	30	30	145	70
AM/FM Btn	40	20	55	90
Soundbutton	40	20	95	90
CD Slot	125	7	25	190
Insert/Eject Button	15	7	160	190
LCD w/ Integrated CD slot	150	85	25	127.5
Favorite Stations Btns	150	15	25	110
CD Btn	40	20	55	70
Temperature Dial	30	30	145	20
AC on/off	40	20	80	0
AC Vent selector	40	20	55	50
Rear Window Defroster Btn	40	20	55	20
Recirculate Air Btn	40	20	95	50
Max AC Btn	40	20	95	20
Display Btn	40	20	95	70
Seek Btn	20	20	95	125
TrackBtn	20	20	75	125
Basic Radio LCD	150	40	25	150
A/C Power Dial	30	30	25	20
Auxiliary Audio Port	5	5	0	185
Menu Button	40	20	135	125

These coordinates would allow the user to take the coordinates from the software and either directly into a their CAD program of choice or edit them to fit more detailed aesthetic requirements. This case study allowed for the creation of 6 products and their component layouts. The products were produced from a large number of components and an even larger number of unconstrained components. From the other available component combinations more products could possibly be created without having to re determine component layouts.

This case study has presented the configuration and layout process for a user interface that relies heavily on aesthetics and the users ability to interact with it, without causing a major distraction. Most of the components could have been defined as optional since most of the components are features to aid in the users comfort and enjoyment of the vehicle. This allowed for more variation in the layout of the components. For the next case study most of the components will be required and the layout will be more strict as the Instrument panel is used to display critical vehicle information to the driver and must be able to be read and understood at a glance.

5.2 Instrument Panel

In this case study the Instrument panel for a family of vehicles will be configured and laid out. The Instrument panel houses a large number of components that provide the driver with vital vehicle information, including vehicles speed and diagnostic information. Due to the type of information and the amount of information the dashboard displays, it is important that the driver be able to easily see and comprehend the information being displayed. Variety in the instrument panel product family is created mostly through configuration options. Instrument panels also pack a fairly large number of components into a tight space creating an interesting layout challenge as well. The first step of the configuration process is to define the components and configuration options for the product family.

5.2.1 Define Instrument Panel

For this case, a basic list of common instrument panel components and configuration types have been listed. This component list is based on observations from a few different instrument panels and observations of current trends. Table 5.11 presents the set of components for this product family. Notice that all of the components that commonly appear in Instrument panel including various gauges and indicator lights. This results in a total of 20 components. The measurements of these were averages based on the observed instrument panels. A couple things to note are that the Multifunction LCD is a small LCD screen that displays the Odometer, Trip information, mpg, and gear selected. While the

Odometer, Trip Odometer, and Gear Indicator are only seven-segment displays. This can replace the regular odometer, trip odometer and gear selection components in a product.

Table 5.11: Dimensions of Instrument Panel Components

Component	Dimensions (mm)	Fascia Choice
Speedometer	(100,100)	Primary
Tachometer	(100,100)	Primary
Odometer	(25 ,7)	Primary
Trip Odometer	(15 ,7)	Primary
Fuel Gauge	(40 ,40)	Primary
Left Turn Signal	(10 ,10)	Primary
Right Turn Signal	(10 ,10)	Primary
Engine Temp Guage	(40 ,40)	Primary
Battery Gauge	(40 ,40)	Primary
Gear Indicator	(15 ,15)	Primary
Low Fuel Indicator	(10 ,7)	Primary
High Beam Indicator	(10 ,10)	Primary
Seat Belt Indicator	(7 ,10)	Primary
Door Ajar Indicator	(10 ,7)	Primary
Oil Warning light	(10 ,7)	Primary
Battery Warning Light	(10 ,7)	Primary
Parking Break Light	(10 ,10)	Primary
Check Engine Light	(10 ,9)	Primary
ABS Light	(10 ,10)	Primary
Multifunction LCD	(60 ,30)	Primary

Since the instrument panel is used to display such important information there is little room for variety in the form of components but variety can be created through the configuration options selected. For this product family 2 configuration categories were created. Table 5.12 presents the configuration categories and options. The 'Backlight' category is used to select the type of backlight the Instrument panel will have for night driving. The ”White” option is just a plain white light. The ”RGB LED” option gives the user the ability to customize the backlight color and ”None” is no backlight. The ”Gauges” category is for the selection for the type gauges used in the products. The Analog gauges are the standard needle gauges, and the Digital gauges are LCD versions of the gauges. This will allow for a level of variety and customization even in the Instrument panel.

Table 5.12: Configuration Options for Instrument Panel Product Family

Configuration Category	Configurations Options
Backlight	White
	RGB LED
	None
Gauges	Analog
	Digital

The last required definition is the fascia for the product family. For this set of products there will only be a single defined layout region. This is just labeled as ”Primary”. Table 5.13 presents the fascia for the insturment panel.

Table 5.13: Dimensions and Location of Instrument Panel Fascia

Fascias	Dimensions (mm)	Location (X, Y)
Primary	(350,100)	(0,0)

In this section the different components, configurations, and the fascia were defined for the instrument panel product family. This helped show the various aspects that make up an Instrument panel and how variety can be developed.

5.2.2 Constrain Instrument Panel Configurations

The next step of the configuration-layout process is to define constraints on combinations of components and configurations. Instrument Panels present an interesting case because many of the components in the product family will be required. This creates difficulty in creating variety across different models. However, Configuration options and some components have been added to create variety in this product family. The first constraint is the Required Components constraint. This constraint has the most entries because since many of the components are required for functional or safety reasons. Table 5.14 presents the required components for this product family.

Table 5.14: Instrument Panel Required Components

Required Components
ABS Light
Battery Gauge
Battery Warning Light
Check Engine Light
Door Ajar Indicator
Engine Temp Gauge
Fuel Gauge
High Beam Indicator
Left Turn Signal
Low Fuel Indicator
Oil Warning Light
Parking Brake Light
Right Turn Signal
Seat Belt Indicator
Speedometer
Tachometer

Each of the required components are either required to fulfill the basic functional requirement or provide vital safety or vehicle information to the driver. The next constraint is the Required Pairs constraint. These will be used to meet the designers requirements for components that must appear in a product together. These provide the designer with the ability to better define some of the variety within this small product family. Table 5.15 presents the Required Pair constraints.

Table 5.15: Instrument Panel Required Pairs

Required Pairs		
Odometer	-	Trip Odometer
Gear Indicator	-	Odometer
None	-	Digital

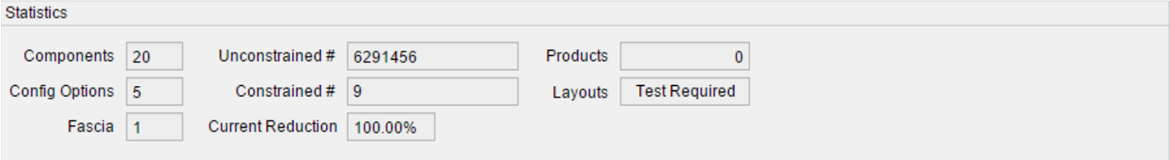
Notice that the Odometer, Trip Odometer, and Gear Indicator are required to appear together. This is because each performs a specific function required for the Instrument Panel to function properly. However the set of them can also be replaced by the Multifunction LCD. The Digital Gauges are required to go with None since the digital gauges are LCD screens and will have built in backlights and can already be customized.

The final constraint to be defined is the disallowed pairs constraint. Table 5.16 presents the Disallowed Pairs for the instrument panel product family.

Table 5.16: Instrument Panel Disallowed Pairs

Disallowed Pairs		
Odometer	-	Multifunction LCD
Gear Indicator	-	Multifunction LCD
None	-	Analog

Notice that The Odometer and Multifunction LCD are disallowed. This is because these components will perform the same in whatever product they appear in. Also notice that the Analog gauges are disallowed with None for the backlight. This is because the gauges must have a backlight of some form. While the Digital gauges will not require a backlight, the Analog gauges do. All of the constraints applied create a large effect on the number of constrained configurations available. The reduction is from 6.2 million to 9 possible products. While much of this reduction is due to the Required constraints the Required and Disallowed pairs still played a significant role. Figure 5.9 presents the statistics panel for of the product family after constraints have been applied.



Statistics

Components 20

Config Options 5

Fascia 1

Unconstrained # 6291456

Constrained # 9

Current Reduction 100.00%

Products 0

Layouts Test Required

Figure 5.9: Statistics after Configuration Constraints have been defined for the Instruements Panel product family

This section presented the configuration constraints for this product family. While the Required constraint create the largest effect on the possible products, the other constraints help the designer create the desired variety within the product family. The next section will examine the configuration hierarchy and explain the hierarchy ordering.

5.2.3 Instrument Panel Hierarchy

In this section the hierarchy for the Instrument Panel. There are only 2 configuration categories for this product family but the order of them can produce very different results. Figure 5.10 presents the hierarchy.

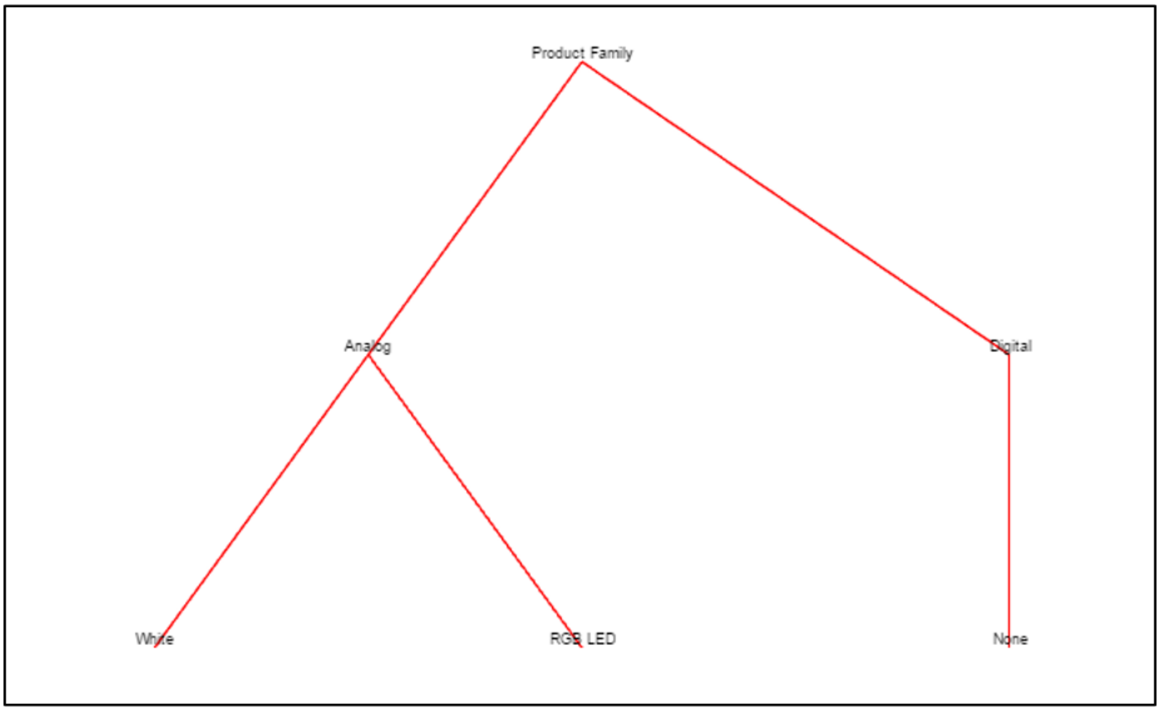


Figure 5.10: Hierarchy of Configuration options for Instrument Product Family

Notice that for this hierarchy the gauge types were given priority over the backlight. This decisions was made because the Gauge types directly effect what type of backlights are available to the instrument panel. For this case, the difference between the Digital and Analog gauges was large enough that it should be given priority over the backlights. This section presented the configuration hierarchy for the Instrument Panel product family. This provides insight into how the designer would like to organize the products with in this family and aids in visualizing the differences in the products within the family.

5.2.4 Instrument Panel Product Definitions

In this section the products for the instrument panel product family will be be defined. Since most of the components were defined as Required, there will be little difference between the component composition of the products. The only difference in the component composition will be whether a product has the Odometer, Trip Odometer, and Gear indicator or the Multifunction LCD. Table 5.17 presents the products defined for this case.

Table 5.17: Instrument Panel Product Definition

Component	Basic	Premium	Grand Touring
Speedometer	X	X	X
Tachometer	X	X	X
Odometer	X		
Trip Odometer	X		
Fuel Gauge	X	X	X
Left Turn Signal	X	X	X
Right Turn Signal	X	X	X
Engine Temp Guage	X	X	X
Battery Gauge	X	X	X
Gear Indicator	X		
Low Fuel Indicator	X	X	X
High Beam Indicator	X	X	X
Seat Belt Indicator	X	X	X
Door Ajar Indicator	X	X	X
Oil Warning light	X	X	X
Battery Warning Light	X	X	X
Parking Break Light	X	X	X
Check Engine Light	X	X	X
ABS Light	X	X	X
Multifunction LCD		X	X

Notice that the ”Base” product has the Odometer, Trip Odometer, and Gear Indicator trifecta. While the ”Premium” and ”Grand Touring” models have the Multifunction LCD instead. These differences along with the differences in Configuration Options selcted in

Table 5.18, allow the designer to create variety among the products.

Table 5.18: Instrument Panel Product Configuration Selection

Configurations	Basic	Premium	Grand Touring
Analog - White	X		
Analog - RGB LED		X	
Digital - None			X

Notice in Table 5.18 that each product has a unique set of configuration options selected. The ”Base” model has the simplest options from each configuration category. This creates a unique product with the lowest cost. Where the ”Premium” and ”Grand Touring” models have increasing features to create variety and models to different market sects. The ”Grand Touring” being able to fit into the ’luxury’ form of a vehicle. The product defined in this section will make up the Instrument Panel product family. This ended up with 3 distinct products being developed. The next section will present the component layout process for the instrument panels.

5.2.5 Instrument Panel Layout

The layout of components in the fascia is the next step of the configuration-layout process. While the product configuration process was restricted by the requirement that many components be required to exist in every product, the layout process is much less restricted and has a large number components being fit into a single fascia. Even though there are less components in this product than the Center Console it ended up taking more, 61 in total, constraints to achieve the desired layout. Table 5.19 presents the layout constraints defined for the set of components in the Instrument Panel.

Table 5.19: Instrument Panel Layout Constraints

Object A		Constraint		Object B/ Location
Speedometer	-	left-of	-	Tachometer
Multifunction LCD	-	X centered in fascia	-	
Odometer	-	X centered in fascia	-	
Trip Odometer	-	X centered in fascia	-	
Trip Odometer	-	above	-	Odometer
Fuel Gauge	-	right-of	-	Speedometer
Engine Temp Gauge	-	right-of	-	Speedometer
Battery Gauge	-	right-of	-	Speedometer
Battery Gauge	-	right-of	-	Fuel Gauge
Battery Gauge	-	left-of	-	Tachometer
Battery Gauge	-	above	-	Odometer
Engine Temp Gauge	-	right-of	-	Battery Gauge
Engine Temp Gauge	-	left-of	-	Tachometer
Fuel Gauge	-	left-of	-	Multifunction LCD
Fuel Gauge	-	above	-	Multifunction LCD
Engine Temp Gauge	-	above	-	Multifunction LCD
Trip Odometer	-	below	-	Fuel Gauge
Battery Gauge	-	X centered in fascia	-	
Left Turn Signal	-	left-of	-	Right Turn Signal
Left Turn Signal	-	left-of	-	Multifunction LCD
Right Turn Signal	-	right-of	-	Multifunction LCD
Right Turn Signal	-	bottom-aligned	-	Left Turn Signal
Right Turn Signal	-	bottom-aligned	-	Multifunction LCD
Right Turn Signal	-	left-of	-	Tachometer
Left Turn Signal	-	right-of	-	Speedometer
Right Turn Signal	-	right-aligned	-	Engine Temp Gauge
Gear Indicator	-	X-center aligned	-	Odometer
Gear Indicator	-	above	-	Trip Odometer
Gear Indicator	-	below	-	Battery Gauge
Engine Temp Gauge	-	right-of	-	Multifunction LCD
Low Fuel Indicator	-	above	-	Left Turn Signal
Low Fuel Indicator	-	below	-	Fuel Gauge
Low Fuel Indicator	-	above	-	Multifunction LCD
Oil Warning Light	-	right-aligned	-	Engine Temp Gauge
Oil Warning Light	-	below	-	Engine Temp Gauge
Oil Warning Light	-	above	-	Right Turn Signal
Oil Warning Light	-	above	-	Multifunction LCD
High Beam Indicator	-	left-aligned	-	Battery Gauge
High Beam Indicator	-	above	-	Multifunction LCD

Table Continued on Next Page

Continuation of Table 5.19

Object A	Constraint	Object B/ Location
High Beam Indicator	- below -	Battery Gauge
Seat Belt Indicator	- X centered in fascia -	
Door Ajar Indicator	- bottom-aligned -	Seat Belt Indicator
Door Ajar Indicator	- right-of -	Seat Belt Indicator
Door Ajar Indicator	- bottom-aligned -	High Beam Indicator
Door Ajar Indicator	- left-of -	Oil Warning Light
Battery Warning Light	- X centered in fascia -	
Battery Warning Light	- below -	Battery Gauge
Battery Warning Light	- above -	Seat Belt Indicator
Check Engine Light	- left-of -	Multifunction LCD
Check Engine Light	- right-of -	Left Turn Signal
ABS Light	- right-of -	Multifunction LCD
ABS Light	- above -	Multifunction LCD
ABS Light	- bottom-aligned -	Oil Warning Light
ABS Light	- below -	Engine Temp Gauge
Parking Brake Light	- right-of -	Multifunction LCD
Parking Brake Light	- left-of -	Tachometer
Check Engine Light	- Y-center aligned -	Multifunction LCD
Parking Brake Light	- Y-center aligned -	Multifunction LCD
Low Fuel Indicator	- X-center aligned -	Fuel Gauge
Low Fuel Indicator	- above -	Battery Warning Light
ABS Light	- left-of -	Tachometer

End of Table

The 61 constraints to layout the 20 components into a single layout region help illustrate how complex the layout process can be. Partial ordering was used on various components to aid in positioning them within the fascia and achieve the desired layout. The use of the layout grammar constraints defined in Chapter 3 allows for the layout to quickly be generated based on a small set of constraints. Figure 5.11 presents the layout for all of the components in the Instrument Panel.

Notice that the Speedometer and the Tachometer are on the extremes of the fascia. This is because it was observed that this is often a design choice in instrument panels. The Speedometer is on the left because it has been observed that a Speedometer is commonly on the Left. The Multifunction Display and the Odometer occupy the same space because they will never appear in the same product and perform identical tasks. This standardizes

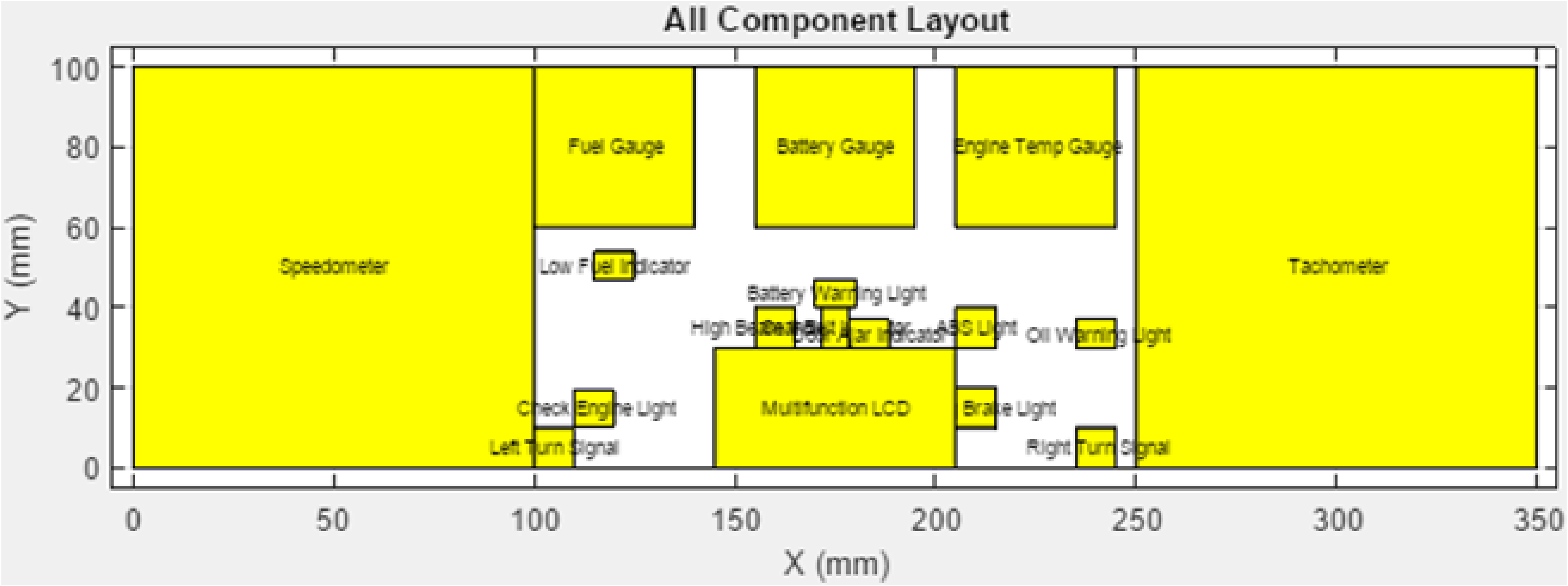


Figure 5.11: Instrument Panel Product Family Full Layout

the layout of similar components,to make different models appear similar and easier to find the desired information.

This section presented the constraints and plot of the layout of all components in the product family. While this product family had less variety in components among the products, a large number of constraints were still required to layout all of the components.

5.2.6 Review Instrument Panel Products

The final step in the Configuration Layout process is to review the results of the process. Tables 5.17 and 5.18 have already presented the Component adn Configurations selected for each product. Figure 5.11 presented the layout of all of the components in the product family. However the individual product layouts have yet to be reviewed. Figure 5.12 presents the layout for the ”Base” model of the product family. Notice that the Odometer, Trip Odometer and, the Gear Indicator are present in the center of the layout. The same location that the Multifunction LCD is also placed. Since the ”Premium” and ”Grand Touring” versions of the instrument panel have the same component composition Figure 5.13 presents the layout for both of these models. Notice that the Multifunction LCD takes the place of the Odometer and its other associated components, while the rest of the layout is identical to the ”Basic” model.

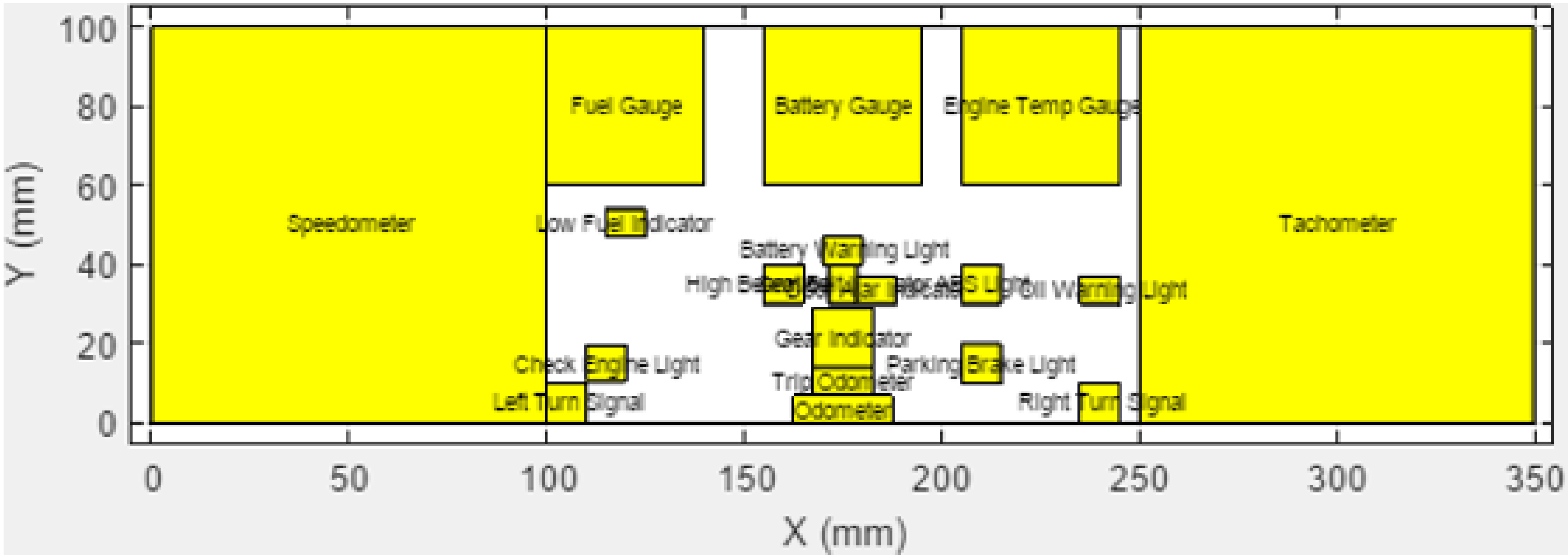


Figure 5.12: Instrument Panel ”Base” Product Layout

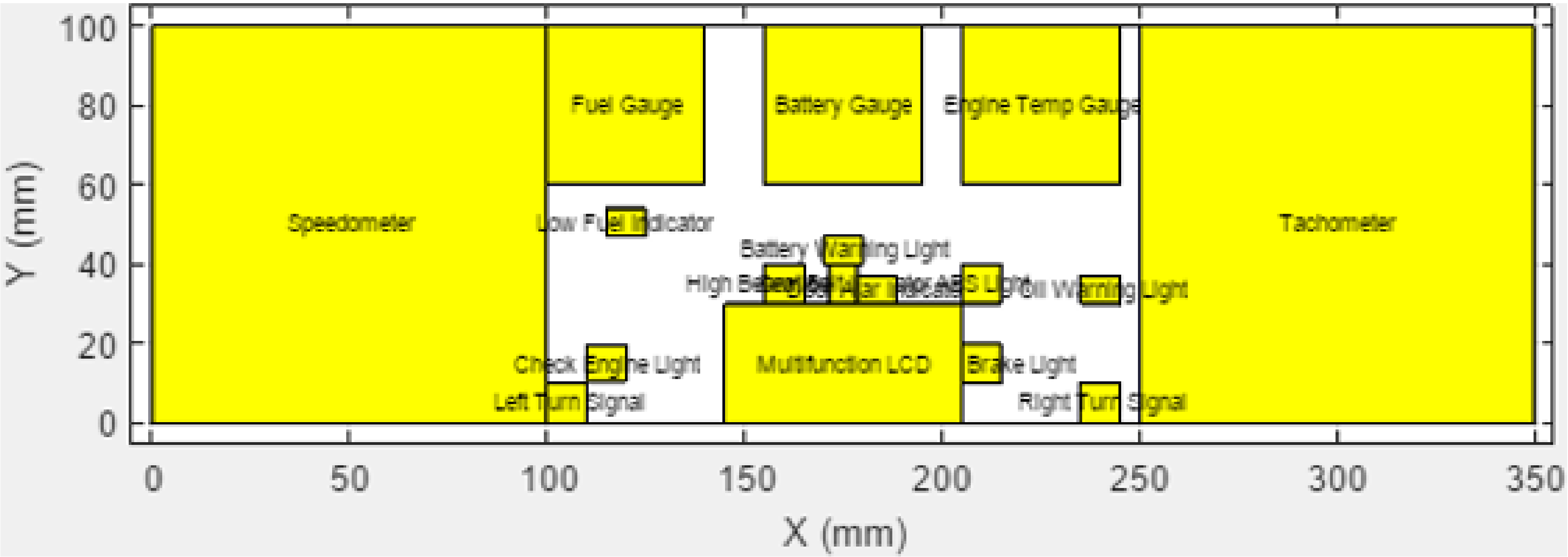


Figure 5.13: Instrument Panel ”Premium” and ”Grand Touring” Product Layout

Table 5.20: Instrument Panel Layout Results

Component	Width mm	Height mm	X mm	Y mm
Speedometer	100	100	0	0
Tachometer	100	100	250	0
Odometer	25	7	162.5	0
Trip Odometer	15	7	167.5	7
Fuel Gauge	40	40	100	60
Left Turn Signal	10	10	100	0
Right Turn Signal	10	10	235	0
Engine Temp Gauge	40	40	205	60
Battery Gauge	40	40	155	60
Gear Indicator	15	15	167.5	14
Low Fuel Indicator	10	7	115	47
High Beam Indicator	10	10	155	30
Seat Belt Indicator	7	10	171.5	30
Door Ajar Indicator	10	7	178.5	30
Oil Warning Light	10	7	235	30
Battery Warning Light	10	7	170	40
Parking Brake Light	10	10	205	10
Check Engine Light	10	9	110	10.5
ABS Light	10	10	205	30
Multifunction LCD	60	30	145	0

Finally the numerical results of the layout process must be reviewed. Table 5.20 presents the coordinates for each of the components in the product family. The software ensured there were no component overlaps. Making the product family entirely feasible.

5.3 Summary

This chapter presented 2 case studies for the application of the Configuration-Layout process presented in this thesis. These case studies presented two different sub-systems of a vehicle, the complexity of these design spaces, and the ease of configuring these design spaces with the aid of the Configuration-Layout process presented in this thesis. While each of these case studies only produced a handful of products, as a set of subsystems different combinations of the products could create more variety to the overall vehicle. The end results of these case studies is two sets of products and a basic physical layout of their user interface components. Through the use of the software presented in the previous chapter each case study was able to be completed within an hour. While this seems like a long time, this process includes the conceptual development of multiple products, feasibility calculations and the start of the embodiement design process. The next chapter will present the answers to the research questions of this thesis and the closing statements.

CHAPTER 6
CLOSURE

The Primary goal of this thesis is to present a method for developing product families with constraints of the layout of components. Specifically, to constrain the combintorial generation of products and layout components of the user interface. The research as presented this was thesis focused on 1) the modeling of spatial relationships of components prior to embodiment design and 2) the determination of design decisions during conceptual design on later design stages. These research developments were supported by formulas, algorithms, and a software implementation. This chapter summarizes the findings presented in this thesis. Section 6.1 presents the answers to the initial research questions presented in Section 1.2. The contributions made in this thesis are presented in Section 6.2. Section 6.4 discusses the limitations of the work presented in this thesis and avenues of future work. Final remarks are given in Section 6.5.

6.1 Answers to Research Questions

The research questions for this thesis presented in Section 1.2 are revisited and the hypotheses tested in this section.

Research Question 1

How can spatial relationships within products be simply captured and represented prior to embodiment design?

Hypothesis 1

A spatial grammar can be used to represent spatial relationships with a mathematical representation

Testing Hypothesis 1: The first research question is addressed by the development of the spatial layout grammar presented in 3.3.3. The layout grammar allows for basic component arrangements to be defined with only basic physical design information about the product family. The use of linear relationships for spatial constraints allows for simple translations from the layout grammar to mathematical representations. The Coffeemaker example presented in Chapter 3 and 4, along with the automobile user interface examples presented in chapter 5 illustrate the application of these layout constraints into the development of the user interface for product family. The constraints allow for the designer to start initial physical design and determine if their desired physical layout is feasible. The software application presented in Chapter 4 demonstrates the ease of implementing the layout grammar and constraints into an easy to a quick and simple application. Allowing designers to quickly produce basic layouts and check for overlap and feasibility.

Research Question 2

How can configuration design methods be augmented to determine the effects of design decisions during conceptual design on embodiment design?

Hypothesis 2

Constraints and optimization methods using both discrete and continuous mathematics can be used to determine the effects of configuration design decisions on physical design spaces.

Testing Hypothesis 2: The Configuration-Layout method presented in Section 3.3 uses a combination of discrete and continuous mathematics to generate feasible products in configuration design and create component layouts. The discrete methods allow for products to be developed by combinatorially generating combinations of components and configuration options. Section 3.3.1 presented the Configuration-Design space which introduced a set of constraints on the combinatorial generation process, allowing for the designer to shape the possible combinations to reach market sects or create a desired variety. The examples presented throughout this thesis showed that these combinatorial constraints allowed for a large effect on the number of available combinations. The effects of these constraints can be calculated prior to full enumeration of possible products.

The transition from discrete to continuous mathematics utilized matrices to properly index and keep track of components. This allowed for the layout constraints and simplex optimization method to easily be run and transition back and forth from configuration and layout methods. The effects of the layout constraints are able to be easily determined by performing the optimization process. The linear optimization method provides a quick and simple method. These methods allow for designers to easily determine the feasibility of the desired product family, in both product composition and component layout.

6.2 Contributions

The areas of product family configuration and object packing have been researched extensively. However research on the combined configuration-layout problem has not been found. The primary contribution of this work is the method for addressing the combined Configuration-Layout Design problem. The method allows the designer to plan out and develop a full family of products starting from only a set of components and configuration options. The combinatorial constraints developed allow for the designer to shape the variety of the product family to meet their needs. The layout grammar and constraints allow for the designer to quickly and easily place components relative to other components or the designated fascia.

A secondary contribution of this research is the algorithms and data structures developed for implementing the Configuration-Design method. The software presented in Chapter 4 demonstrates these. The algorithms for Constraining and generating feasible component combinations account for a large portion of the Configuration-Layout method implementation. The ability to transition back and forth between the configuration and layout phases of the process allow designers to continually make changes and easily determine the effects of changes on the product family.

6.3 Limitations

The methods and implementation presented in this thesis can be applied to a wide variety of product families but does have limitations. The layout process of the Configuration-Layout method currently has a couple of main limitations. The first of which is the shape of the objects used. The layout process is currently limited to square and rectangular bounding boxes. This is due to the linear simplex algorithm being used to layout components. The current layout constraints do not consider non-rectangular objects. Changes would need to made to the layout constraints and simplex optimization method to account for non-rectangular components. Along with the shape limitations, the methods do not currently allow for the rotation of components. Future work will discuss the addition of rotational degrees of freedom in the expansion from 2D to 3D.

There are also currently software limitations. The limits on the number of components, configuration options, and fascias before the software crashes are currently unknown. However test cases with upto 25 components, 3 configuration categories with 3 configuration options each, and 3 fascias have been tested without software crashes. The limits on constraints are also unknown, but tests with upto 10 required, 18 required pairs, and 18 disallowed pairs has been tested. The limits on configuration constraints will depend on the number of components connected via constraints. This will effect the size of the smaller sets of combinations being generated during the process. Early iterations of the software found that generating sets of combinations greater than 2^{21} starts taking up to much of the computers memory and can cause the computers to crash. In the current version of the software that would mean sets of components connected by constraints containing more than 20 elements may cause the program and computer to crash.

6.4 Future Work

While the research presented in this thesis enables new forms of configuration design problems to be addressed, there are opportunities for future research. One direction for future research is for the inclusion of configuration hierarchies in the evaluation of the developed product family and possible extension of combinatorial constraints to provide designers with more control over the configuration process. The final area of future work is expanding the layout process from 2D to 3D and increase the number layout constraints and the inclusion of connections between components in the layout of components and evaluation of feasibility.

Configuration hierarchies and component connections were included as sections of the software presented in Chapter 4, but these elements are not used as factors in the determining product feasibility or evaluating results of the product family development process. A configuration hierarchy allows the designer to select which configuration category is more important. Augmenting the methods presented in this thesis with the ability to take configuration hierarchy into account would allow the designer to better judge the results of the combinatorial generation process and evaluate the product family to determine if the desired variety has been met.

The combinatorial generation process introduced three configuration constraints (Required, Required Pairs, Disallowed Pairs) and their effects were characterized. These constraints are able to effectively reduce the unconstrained number of combinations but do not provide the designer with a large number of ways to shape the variety of the product family. Developing additional configuration constraints would provide the designer more control in the type of product varieties that are able to be developed.

The layout methods in this research was limited to 2D layout and presented a limited number of constraint. Expanding the layout process into 3D space would require modification of the current simplex model to account for the additional dimension. Other optimization methods may also require consideration since the additional dimension start to create a more complex problem. 3D projection is a possible method to expand from 2D to 3D. The additional axis would also require the development of more layout constraints would likely need to account for the additional axis of movement but also start considering rotational degrees of freedom. Additional constraints could also provide the designer to develop more detailed layouts to achieve their desired design.

Component connections were presented in the software in Chapter 4 but are not used to evaluate the feasibility of the product family. Expanding the layout process to consider the effects of connections between components would provide the designer with more feasibility information based on any requirements the connections may have. Other avenues for expansion include the detection of collision between components, and providing a warning if the total area of the components is larger than the space they are to be laid out in. Additional future work would include adding weights or hierarchies to the layout constraints. This would allow designers to give priority to constraints and provide greater control over the layout process.

6.5 Closing Remarks

Product family design research is aimed at providing designers and engineering methods to more effectively and efficiently develop product families. The work of this thesis intends to augment current product family design methods by providing designers the ability to combinatorially generate entire families of products and determine the effects of design decisions at an early stage. The development of both configuration and layout constraints will allow designers shape product families to their desire and determine if their desired variety is feasible. While the methods presented in this these will aid designers in the development of product families, it does not replace a designer. These methods still require a knowledgeable designer to supervise the process and results, just like any method. It is the goal for designers to embrace this method as a means for determining product feasibility and increasing the speed of the product development cycle.

REFERENCES

[1] G. Pahl, W. Beitz, J. Feldhusen, and K.-H. Grote, *Engineering design: a systematic approach*, 3rd. London: Springer-Verlag, 2007, p. 629, ISBN: 978-1-84628-318-5.

[2] Z. Siddique, “Common Platform development: designing for product variety,” Ph.D. Dissertation, Georgia Institute of Technology, 2000, p. 501.

[3] B. P. Corbett and D. W. Rosen, “A configuration design based method for platform commonization for product families,” *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 18, no. 01, 2004.

[4] Z. Siddique and D. W. Rosen, “On combinatorial design spaces for the configuration design of product families,” *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 15, no. July 2001, pp. 91–108, 2001.

[5] K. Ulrich, “The role of product architecture in the manufacturing firm,” *Research Policy*, vol. 24, no. 3, pp. 419–440, 1995.

[6] M. V. Martin and K. Ishii, “Design for Variety: Development of Complexity Indices and Design Charts,” in *ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, paper # DETC97/DFM-4359*, Sacramento, CA, 1997, pp. 1–9.

[7] R. B. Stone, K. L. Wood, and R. H. Crawford, “A Heuristic Method to Identify modules from a Functional Description of a Product,” in *Proceedings of ASME Design Theory and Methodology Conference, paper # DETC98/DTM - 5642*, Atlanta, GA, 1998, pp. 1–21, ISBN: 4755588561.

[8] J. Yu, J. P. Gonzalez-Zugasti, and K. N. Otto, “Product architecture definition based upon customer demands,” *Journal of Mechanical Design*, vol. 121, no. 3, pp. 329 –335, 1999.

[9] K. N. Otto and A. Sudjianto, “Modularization to Support Multiple Brand Platforms,” in *Design Engineering*, Pittsburgh, PA, 2001, pp. 1–14.

[10] P. J. Newcomb, B. Bras, and D. W. Rosen, “Implications of Modularity on Product Design for the Life Cycle,” *Journal of Mechanical Design*, vol. 120, no. 3, pp. 483–490, 1998.

[11] S. Kota, K. Sethuraman, and R. Miller, “A Metric for Evaluating Design Commonality in Product Families,” *Journal of Mechanical Design*, vol. 122, no. 4, p. 403, 2000.

[12] H. J. Thevenot and T. W. Simpson, “Commonality indices for product family design: a detailed\rcomparison,” *Journal of Engineering Design*, vol. 17, no. 2, pp. 99–119, 2006.

[13] D. W. Rosen, “Design of Modular Product Architectures in Discrete Design Spaces Subject to Life Cycle Issues,” in *Proceedings of ASME Design Engineering Technical Conferences, paper # 96-DETC/DAC-1485*, Irvine, Ca, 1996, pp. 1–13.

[14] S. Finger and J. R. Dixon, “A review of research in mechanical engineering design. Part I: Descriptive, prescriptive, and computer-based models of design processes,” *Research in Engineering Design*, vol. 1, no. 1, pp. 51–67, 1989.

[15] D. W. Rosen and J. R. Dixon, “Languages for Feature-Based Design and Manufacturability Evaluation,” *International Journal of Systems Automation: Research and Applications*, vol. 2, no. 4, pp. 353–373, 1992.

[16] J. Cagan, “Engineering Shape Grammars: where we have been and where we are going,” in *Formal Engineering Design Synthesis*, E. K. Antonsson and J. Cagan, Eds., 1st, Cambridge: Cambridge University Press, pp. 65–92, ISBN: 0-521-79247-9.

[17] G. Reddy and J. Cagan, “An Improved Shape Annealing Algorithm For Truss Topology Generation,” *Journal of Mechanical Design*, vol. 117, p. 315, 1994.

[18] B. P. Corbett and D. W. Rosen, “Platform commonization with discrete design spaces: Introduction of the flow design space,” in *Proceedings of the ASME Design Engineering Technical Conference, Paper # DETC2003/DTM-48678*, vol. 3, 2003, pp. 835–846.

[19] J. Cagan, K. Shimada, and S. Yin, “A survey of computational approaches to three-dimensional layout problems,” *Computer-Aided Design*, vol. 34, no. 8, pp. 597–611, 2002.

[20] Z. Gürbüz, S. Akyoku, b. Emirolu, and A. Güran, “An Efficient Algorithm for 3D Rectangular Box Packing Placed,” in *Applied Automatic Systems: Proceedings of Selected AAS 2009 Papers*, Ohrid, 2009, pp. 131 –134, ISBN: 5054834699.

[21] F. G. Ortmann, N. Ntene, and J. H. van Vuuren, “New and improved level heuristics for the rectangular strip packing and variable-sized bin packing problems,” *European Journal of Operational Research*, vol. 203, no. 2, pp. 306–315, 2010.

[22] G Scheithauer and J Terno, “A branch-and-bound algorithm for solving one-dimensional cutting stock problems exactly,” *Applicationes Mathematicae*, vol. 23, no. 2, pp. 151–167, 1995.

[23] T. Fanslau and A. Bortfeldt, “A Tree Search Algorithm for Solving the Container Loading Problem,” *INFORMS Journal on Computing*, vol. 22, no. 2, pp. 222–235, 2010.

[24] J. J. Kim and D. C. Gossard, “Reasoning on the Location of Components for Assembly Packaging,” *Journal of Mechanical Design*, vol. 113, no. 4, pp. 402–407, 1991.

[25] C. Aladahalli, J. Cagan, and K. Shimada, “Objective Function Effect Based Pattern SearchAn Implementation for 3D Component Layout,” *Journal of Mechanical Design*, vol. 129, no. 3, p. 255, 2007.

[26] S. Szykman and J. Cagan, “Constrained Three-Dimensional Component Layout Using Simulated Annealing,” *Journal of Mechanical Design*, vol. 119, no. March 1997, p. 28, 1997.

[27] L. Hanna Landry and J. Cagan, “Search Strategies in Evolutionary Multi-Agent Systems: The Effect of Cooperation and Reward on Solution Quality,” *Journal of Mechanical Design*, vol. 133, no. 6, p. 061 005, 2011.

[28] L. M. Weitzman and K. Wittenburg, “Grammar-based articulation for multimedia document design,” *Multimedia Systems*, vol. 4, no. 3, pp. 99–111, 1996.

[29] A. Borning, R. Lin, and K. Marriott, “Constraints for the web,” in *Proceedings of the fifth ACM international conference on Multimedia - MULTIMEDIA ’97*, New York, New York, USA: ACM Press, 1997, pp. 173–182, ISBN: 0897919912.

[30] G. J. Badros, A. Borning, and P. J. Stuckey, “The Cassowary linear arithmetic constraint solving algorithm,” *ACM Transactions on Computer-Human Interaction*, vol. 8, no. 4, pp. 267–306, 2001.

[31] J. Clausen, “Branch and bound algorithms-principles and examples,” *Department of Computer Science, University of Copenhagen*, pp. 1–30, 1999.

[32] J. E. Beasley, “An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure,” *Operations Research*, vol. 33, no. 1, pp. 49–64, 1985. arXiv: `opre.33.1.49` [10.1287].

[33] S. Szykman and J. Cagan, “A Simulated Annealing-Based Approach to Three-Dimensional Component Packing,” vol. 117, no. June 1995, pp. 308–314, 1995.

[34] Y.-C. Xu, R.-B. Xiao, and M. Amos, “Simulated annealing for weighted polygon packing,” 2008. arXiv: `0809.5005`.

[35] M. Agarwal and J. Cagan, “A Blend of Different Tastes: The Language of Coffeemakers,” *Environment and Planning B: Planning and Design*, vol. 25, no. 2, pp. 205–226, 1998.

[36] G. B. Dantzig, *Linear programming and extensions*. Princeton, N.J.: Princeton University Press, 1998, ISBN: 0691059136.