Product Family Architecture Modularity Using Function and Variety Heuristics

by

Erik J. Zamirowski

B. S., Mechanical Engineering (1995)

University of Michigan, Ann Arbor

Submitted to the Department of Mechanical Engineering
In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Mechanical Engineering

at the

Massachusetts Institute of Technology

June 1999

©1999 Massachusetts Institute of Technology
All rights reserved

Signature of Author .................................................. Department of Mechanical Engineering
May 7, 1999

Certified by ............................................................ Kevin N. Otto
Robert Noyce Career Development Associate Professor of Mechanical Engineering
Thesis Supervisor

Accepted by ............................................................ Ain A. Semin
Chairman, Department Committee on Graduate Students
PRODUCT FAMILY ARCHITECTURE MODULARITY USING FUNCTION AND VARIETY HEURISTICS

by

ERIK J. ZAMIROWSKI

Submitted to the Department of Mechanical Engineering on May 7, 1999 in partial fulfillment of the requirements for the Degree of Master of Science in Mechanica Engineering

ABSTRACT

This thesis examines the market and functional aspects of developing a product portfolio architecture. Product use-based market analysis of customer needs and product features is performed to determine the set of portfolio target values. The variation in these target values among and within customers is interpreted to yield a portfolio architecture recommendation for each feature. Correlation analysis is used to reduce the combinatorial complexity of target values according to market trends.

Functional representation of a product portfolio is achieved through a monolithic function structure created by combining individual product use function structures. This function structure is partitioned into function clusters, representing potential module options, using a set of function modularity and variety modularity heuristics. The function modularity heuristics reveal relationships among the product functions according to the interconnection and causality of the functions. The variety heuristic, through which product features are mapped to the functions that deliver them, reveals the potential for variety-dependent modularity.

The set of function clusters is compared to determine the degree of interaction among them. When variety function clusters are not orthogonal to each other, the portfolio architecture rules must be modified to compensate for this coupling, or portfolio variety will be compromised. An algorithm is used to reconcile interactions among the function clusters to maximize portfolio variety. The result is a set of function clusters for candidate product modules that achieves the best product variety given functional constraints. The process is exemplified using a portfolio of xerographic products.

Thesis Supervisor: Kevin N. Otto

Title: Assistant Professor of Mechanical Engineering
Acknowledgements

I'd like to thank several individuals for their help and contributions to this work. I thank Kevin Otto, research advisor, for his inspiring optimism, his insights and advice, and the chance to work in a well-equipped lab.

I owe a great debt to Steve Hoover, Naveen Sharma, and Dave Kamprath at Xerox for a good summer experience, as well as their insights into Xerox, Xerox technology, product architecture, and for giving me the opportunity to witness firsthand the challenges of architecting a complex system. Special thanks to Steve Hoover for helping me get to Xerox for the summer and for his efforts on my behalf, and to Naveen Sharma for coming to MIT and contributing at the industry-research CIPD meetings. Thanks to Lee Anne Boswell for her assistance over the summer. I’d also like to thank the rest of the Xerox staff working on the architecture project, and to Dave Zawadski for his thoughts about product development.

Thanks to the CIPD and the NSF for funding my research.

I’d also like to thank the MIT contingent: Javier Gonzalez-Zugasti, for always being a model of patient determination and uncompromising integrity, and for teaching me overhand squash serves; Kan Ota for his thoughtfulness, friendliness, and our discourses on international affairs and foreign policy; Janet Yu, for conversations and occasional venting of frustrations; JoRuëta Roberson, Erick Rios, Ebad Jahangir, and Rajiv Suri.

Thanks to my family: my parents, my two grandmothers, and my brothers for supporting me during this rewarding but trying experience, and lastly and especially to Nicole Woltschlaeger for her love, faithfulness, and inspiration. It’s finally over.
# Table of Contents

1.0 Overview ............................................................................. 5
2.0 Background-nomenclature and terminology ........................................... 7
  2.1 Product architecture .......................................................... 7
    2.1.1 Function decomposition ............................................... 7
    2.1.2 Function expression .................................................... 9
    2.1.3 Interface considerations .............................................. 10
    2.1.4 Modular and integral architecture ................................. 10
    2.1.5 Architecture and systems engineering ......................... 11
  2.2 Product portfolios and families .............................................. 11
  2.3 Product platform ................................................................ 12
  2.4 Variety and variants .......................................................... 12
    2.4.1 Variety as different system inputs and outputs ............... 12
    2.4.2 Variety as variation and robustness ............................. 13
    2.4.3 Market view of product variety ................................... 13
    2.4.4 Product variants ....................................................... 14
  2.5 Modularity .................................................................. 14
  2.6 Benefits .................................................................. 14
  2.7 Limitations ................................................................ 15
3.0 Industrial context .................................................................. 17
  3.1 Xerox corporation .......................................................... 17
  3.2 Electrophotography/xerography ........................................... 18
  3.3 Products .................................................................. 18
4.0 Market factors ..................................................................... 20
  4.1 Customer needs .............................................................. 20
    4.1.1 Core product needs .................................................... 20
    4.1.2 Variant needs ........................................................... 20
    4.1.3 Latent needs ............................................................ 21
    4.1.4 Kano interpretation ................................................... 21
  4.2 Customer needs to product characteristics and target values ........... 22
  4.3 Target values to product uses .............................................. 23
  4.4 Market characterization ..................................................... 25
  4.5 Market characterization - correlation analysis ............................ 28
  4.6 Xerox example ................................................................ 32
    4.6.1 Customer needs to product characteristics .................... 32
    4.6.2 Target values ........................................................... 32
    4.6.3 Market segmentation and applications ......................... 34
    4.6.4 Target value distributions and characterization .......... 35
  4.6.5 Elimination of unwarranted variety ................................. 42
    4.6.6 Identification of integral platform features ................. 43
    4.6.7 Identification of necessarily adjustable features .......... 43
    4.6.8 Correlation of remaining characteristics ..................... 43
  4.6.9 Resulting product family .............................................. 45
5.0 Function analysis for an entire market ......................................... 47
  5.1 Functional representation – function structure .............................. 47
  5.2 Use-case function structures ................................................. 48
  5.3 Monolithic function structure ................................................ 50
6.0 Portfolio architecture modularity ................................................ 53
  6.1 Heuristics and their use for design and modularity ........................ 53
  6.2 Function heuristics ............................................................ 54
    6.2.1 Dominant flow ........................................................ 55
    6.2.2 Branching flow ......................................................... 55
    6.2.3 Transformation/conversion ....................................... 57
    6.2.4 Causally-linked functions ......................................... 57
    6.2.5 Similarity/repetition ............................................... 59
  6.3 Function clusters ............................................................... 62
  6.4 Variety heuristics .............................................................. 62
    6.4.1 Isolation of variety ................................................... 63
    6.4.2 Function structure modification for variety reduction .... 65
  6.5 Variety function clusters ..................................................... 66
7.0 Selection of candidate function clusters for modules ...................... 67
  7.1 Function cluster interactions ................................................. 67
  7.2 Extension of portfolio architecture decision tree ........................ 69
  7.3 Module precedence and selection .......................................... 70
  7.4 Xerox example ................................................................ 71
  7.5 Technology selection ......................................................... 71
8.0 Conclusions ................................................................... 80
9.0 Bibliography .................................................................... 83
Appendices ........................................................................... 85
1.0 Overview

Defining the architecture of a complex system to support a family of products requires the consideration of many views and influences. Market forces (customers and competitors' products), functional requirements (the actual functions that the products must perform), technological limitations (choice of technology), and enterprise factors (organization, existing technology) all play important roles.

The goal of this thesis is to demonstrate a framework for developing the function architecture of a product portfolio. It incorporates analysis of market factors and the functional requirements of the set of products. Market information is used to define the products that are most applicable for the market. The variation in customer requirements for their desired products may be analyzed to select the best set of features and performance target values. This set mitigates differences in customer preference by analyzing the statistical parameters of the distributions of customer requirements and incorporating weight factors as indications of market potential.

While some architecture implications arise from these distributions, customers are solely interested in the satisfaction of their needs, and are disinterested (in most cases) in the form that the solution takes. The product functions necessary to produce the desired results must be understood by the development team to achieve product success. Function-logic diagrams such as function structures are useful in explicitly representing the relationships among the system functions.

Once a system function structure has been developed by merging the individual product use function structures, rules are applied to indicate suitable partitioning of the "product" into potential modules. These heuristics use patterns observed in successful single-product modules to define modules for the monolithic function structure. New heuristics are proposed to identify modules that group functions that provide product variety, which will be used to develop components that may be swapped, included, or excluded from the product to produce necessary variants.

Finally, an algorithm is proposed to select the function clusters to use from among those found via the heuristics. The beneficial interactions among the important modules, those that provide variety, are maximized, while the detrimental interactions are minimized. The result is a set of candidate modules that serve as the basis of a portfolio architecture.
The organization of the thesis is as follows:

Chapter 2 begins with the definition of the key terminology for discussing products, and concludes with why modularity is a powerful way to meet the needs of a diverse customer base.

Chapter 3 briefly examines Xerox Corporation and its relation to this research.

Chapter 4 demonstrates how market analysis can be used to propose a set of products.

Chapter 5 reveals how functional requirements from individual customers can be combined into an artifact that captures functional and logical relationships.

Chapter 6 illustrates how a product may be partitioned into modules according to the relationships among functions and how functions relate to market requirements.

Chapter 7 presents a method for choosing, once opportunities for modules have been identified, which modules to build to support product variety and partitioning efficiency.

Chapter 8 summarizes the proposed process and its implications.

Other works related to the major topics explored in this thesis are the following:


2.0 Background-Nomenclature and Terminology
Companies have been developing products for decades. In comparison, the study of product development has much more recently become a research interest. While industry has been focused on bringing products to market, researchers are now reflecting on the process and its elements, and attempting to resolve the discrepancies in terminology found between different segments of industry. This section provides the definitions for much of the terminology used in this field that relates to this research.

2.1 Product Architecture
*Product architecture* relates to the physical manifestation of functionality. Several researchers have developed definitions for this relationship. Alexander [2.1] describes product architecture as “fitting” the design form to the surrounding context so as to minimize misfits, or functional discrepancies. Ulrich [2.2] posits that product architecture “is the scheme by which the function of a product is allocated to physical components.”

Architecture is the basic conceptual system description. It is the definition stage before specification and detail design. It is necessary for the subdivision of the development task into manageable portions. Architecture is the framework from which all other activities lead, and the foundation upon which all depends. Product architecture is the first fusion of the fuzziness of front-end development. It allows estimation and modeling efforts to predict system performance. Selection of product architecture commits resources and constrains design, limiting the scope and capabilities of the endeavor. Since so much is fixed after this decision point, specifying an architecture that has the capability of meeting all of the anticipated market needs is critical. Product architecture definition is the last chance to ensure this capability without considerable penalty for modification, including dedication of resources and the resistance to change of corporate culture.

Ulrich further refines his definition of product architecture into three regimes. He identifies functional elements, the mapping of elements into physical components, and the specification of interfaces as the key factors in describing any product’s architecture.

2.1.1 Function Decomposition
A *function* is the form-independent description of an event whose input and output states differ. They are usually verb-noun pairings indicating some action performed. Functions are commonly classified according to their relation to the main task of the system. For systems that have a
single or relatively few tasks, the *main functions* perform them. *Auxiliary functions* relate to the main function indirectly. As functions move from generality to specificity, a hierarchical relationship is sometimes created. When this is the case, the decomposed functions are said to be *subfunctions* of the parent function.

If the main function of a cassette player (playing a tape) were decomposed into individual elements, those elements would include tape movement, signal recovery, signal playing, and a host of other related categories. Using the function structure [2.3] modeling technique, a method which traces flows of energy, material, and information through the product system, a representation of the main function appears in Figure 2.1. Tape and batteries are material inputs to the system. They are design assumptions and the adoption of exogenous dominant design product architectures for energy and sound data storage. The subfunctions are decomposed in Figure 2.2.

Figure 2.1. Main Function for a cassette player

Figure 2.2. Decomposition of Sony Walkman Main Function
2.1.2 Function Expression
The next aspect of product architecture involves relating the established product functions to the physical domain. It can be thought of as relating the function, the "what?" of the product, to the form, or "how?" of the product. For the tape movement function of the tape player, a power source, motor, transmission, and method for selection by the user are all incorporated as physical components. Obviously, alternative methods are conceivable, such as using air flow, magnetism, or other physical systems to move the tape. At this stage of product definition, morphological component alternatives can be compared for satisfying functional requirements. The main components of the Sony Walkman are the power, mechanical, electronic, and structural. Since structural considerations are form-dependent, they do not appear as functions in the function structure, but may be added after other functional elements have been given form. Structural elements relate form elements to each other, much like an automobile unibody provides a framework for many other modules. Figure 2.3 reveals a further decomposition of the functions, as well as the grouping of functions into physical components.

Figure 2.3. Sony Walkman function structure and components.
2.1.3 Interface Considerations
The final aspect of architecture concerns interfaces among components. How the development team handles their specification will affect many of the aspects we consider important to understanding product families. Interfaces are the chemical, mechanical, electromagnetic, thermal, or other relationships among the components. Function structures explicitly render the interfaces between functions as flows, and when physical components are matched to functions, these flows at the component boundaries become important interface consideration for further specification. In the case of the tape player, the power source (battery) must pass electricity to the motor. Depending upon the method of transmission, other interface constraints apply, such as physical connection, shielding and insulation, and heat dissipation. Thus, function flows give an impression of the basic interface, but additional clarification is necessary to design the elements that provide the functions that act on the flows.

2.1.4 Modular and Integral Architecture
Architectures can be classified according to the mapping of functions to the adopted form. A continuum between one-to-one mapping of functions to elements ad a many-to-one mapping exists. Modular architecture incorporates individual or very few functions into components. Integral architecture involves combining functions within components, which leads to the creation of unique interfaces. Generally, components that exhibit greater modularity have a one-to-one mapping with functions, while integral components satisfy many functions with a single hardware element. More modular designs can allow an exchange of modules for different performance, while integrated functions optimize performance at a single point in design space.

In the Sony Walkman case, product variety is achieved through modularity. Components that provide specific functions are either present or absent, and performance levels relate to the quality of components used. Sony offers variety primarily through improving the performance levels of its functions or by rearranging its function-set into a new product. Sanderson and Uzumeri [2.4] identify these as incremental and topological design changes. They note that the establishment of the platform allowed Sony to enter new market niches with limited development expenditures.
The development of product families is strongly influenced by architecture decisions concerning modularity and integration. The extent of integration directly affects the ability to provide variety to the market.

2.1.5 Architecture and Systems Engineering
Product architecture concepts relate to recent systems engineering research. Maier [2.5] describes four system “views,” or perspectives: functional, physical, performance, and management/implementation. The first two regimes echo the definition of an architecture, where the functional product description is manifest in the physical form domain. The latter two views concern specification and execution of the design.

Products like automobiles are dominant designs with mature architectures. Most of the subsystems that comprise them have existed for years and have evolved individually through performance and reliability improvements and cost reduction. For these mature products, the goals of future product development efforts are to allow the addition of new, innovative features that address unmet customer needs, and to improve the design and development of the existing architecture. For mature products, the architecture is a constraint.

New product architectures revolutionize the system-level approach to the design problems addressed by the product, either through new system elements or a reconfiguration of existing ones.

2.2 Product Portfolios and Families
A product portfolio is a set of offerings by a company to a market. There is not necessarily any relationship among the products, except that they have a common source. The method by which features are provided through components and other hardware within products defines the portfolio architecture. There are many ways of delivering features in a company’s set of offerings. Frequently, products are individually designed and targeted at different market segments of consumers. This type of portfolio architecture is fixed and does not share components.

A product family is another class of product portfolios. Product families are distinct in that they incorporate a set of shared resources. Most often, the common resource shared within a product family is hardware-based; that is, components, parts, or some other physical subsystem is common for multiple products. As Ulrich and Robertson [2.6] point out, this commonality is
not necessarily restricted to hardware components and tooling for component fabrication that are shared, but can also include manufacturing processes and assembly systems, product development processes, and the people and relationships that realize the products.

The product family is an attempt to reduce total product cost by taking advantage of commonization. Since it is often impractical to offer a single product capable of satisfying all customer requirements simultaneously, or prohibitive to include elements that satisfy all customer requirements and can be selected by the individual user, companies offer separate products. These products are intended to appeal to customer niches or satisfy clusters of customer requirements.

2.3 Product Platform
This set of resources shared within a product family is defined as the product platform. Meyer and Leonard [2.7] describe the product platform as the "set of subsystems and interfaces that form a common structure from which a stream of derivative products can be efficiently developed and produced." Examples of platformable attributes fit into several domains:

Design: modularity, scalability, interface specification
Manufacturing: process, datum assignment, assembly, materials
Distribution/Service: supply channels, vendors
Platforms are a logical extension of mass production toward mass customization, since at least a portion of the product is common and can be produced more economically in higher volumes.

2.4 Variety and variants
Variation in the expectations and requirements of different customers leads to a need for product variety. In order to fully understand the ramifications of variety on an architecture, we must establish a definition of variety. There are two key types: variety as differences in system inputs and outputs, and variety as variation within common system inputs and outputs.

2.4.1 Variety as different system inputs and outputs
This addresses the notion of performance of different functions. A specific set of inputs delivers a specific system output, which varies according to the inputs. For example, a copier that also automatically staples the documents it creates illustrates the principle. The input is a stack of paper, and the output is either a stapled or unstapled document. Hence, the basic functionality of
the system, that of copying, is enhanced or augmented by the addition of a stapling feature. It is easy to see that while the need to staple might put some constraints on the copier (maximum number of pages per document, variation in the alignment of the stacked pages, etc.). The basic function can be provided without the added feature. This variety relates to the expectations of the customer using the product, and the set of customer needs the product is meant to address. This variety arises from differences in customer preferences. We can also expect explicit input variety in terms of user input to the system in order to effect the desired results. A system, to some extent, needs to be told or have an algorithm to determine the desired initial state in order to reach the desired final state. Extending our stapler example, some user input is required in order to tell the system to perform the enhanced function of stapling, whether that input originate from the touch of a button, a default setting, or a data block in a digital job file.

2.4.2 Variety as variation and robustness
In another sense, a product must be able to tolerate variation caused by variation in system inputs. Taguchi’s notion of robust design [2.8] addresses this by advocating systems that are virtually insensitive to variation in input parameters. For example, a copier should ideally be able to perform all its basic and enhanced functions on paper of different weight, surface texture, thickness, stiffness, etc. This can be largely impractical, so some specifications must be developed to limit the necessary system robustness.

2.4.3 Market view of product variety
We can consider a customer to be satisfied with a product if it meets all of his or her expressed and latent needs. The mere mention of each function that a product performs creates different expectations in the minds of the customers. The more functions and features a product seeks to incorporate, the greater the variation in the target values of its customers. In order to render a product attractive to as many customers as possible, it would be best to offer a product for each distinct set of customer needs. Maximum customer satisfaction would be achieved if each customer received a custom-made offering that perfectly addressed all of that customer’s needs, but this is not necessarily the profit-maximizing strategy from the company’s perspective. In order to mitigate the potentially prohibitive cost of mass customization, a strategy must be imposed. One strategy to reduce manufacturing cost and also support product variety involves the development of modules to support a product family. As Ehrens[2.9] notes, a “solution for
coping with the problem of mass-customisation lies in the development of product families, which offer a large variety from a small set of modules that can be easily combined.”

2.4.4 Product variants
A product variant, also sometimes called a derivative product, is any product that incorporates a shared platform. For example, the Pontiac Firebird and the Chevrolet Camaro models both result from a common General Motors F-car platform and are variants. Product variants use a scheme to offer multiple products while implementing some shared systems.

2.5 Modularity
Modularity, the tendency of components to provide one or a few main functions, is an important way to deliver product variety. Ulrich and Tung [2.10] identify several classes of modularity observed in products:

Component Swapping: a set of components may be substituted for one another via a common interface.

Component Sharing: a single component is used in many different products via a common interface.

Fabricate-to-fit: a set of components that share standard interfaces but may be scaled along non-interface dimensions or attributes to produce variants.

Bus: multiple component-swapping via a multi-port standardized interface.

Sectional: multiple components that share common, repeated interfaces so that each component may connect arbitrarily with additional components.

Modularity makes product variety possible through the use of modules as differential elements of product variety. Different products are possible through the inclusion, exclusion, or substitution of modules with different features or performance characteristics. It is this aspect of product modularity which will be exploited in the partitioning of the product into modules to deliver variants.

2.6 Benefits
Using modularity as a means of offering product variety has many benefits. The following list summarizes major reasons many researchers cite for adopting the modularity principle.
Lower fixed cost: modules spread their fixed costs over all the products that use them, thus lowering the burden in each product.

Reduced development time: since multiple products are developed concurrently through the use of common modules, the development time of follow-on variants that incorporate the previously designed modules take less time to develop.

Component economies of scale: Where mass production reduced cost through standardization, mass customization accomplishes the same through standardization at the component level.

Greater manufacturing efficiency: modules with well-defined interfaces are usually easier to assemble into products, and common assembly lines can be changed to manufacture variants at a lower cost than unique designs and independent assembly systems.

Ease of product diagnosis, testing, maintenance, repair, and disposal: the components and modules of modular products are isolated entities that may be controlled independently before and after assembly.

2.7 Limitations
Although there are important benefits from adopting modular product architecture, there are some drawbacks that are often cited and must be considered and addressed by the development team.

Static product architecture: Once an architecture is selected, it is difficult to modify it, except for the substitution of alternative modules at the standard interfaces. Planning before implementation is crucial to ensure that the configuration supports all desired product variants.

Performance optimization: When platform performance targets are selected, these values will deviate from some customers’ optimal values. The design will be sub-optimal for some portion of the market.

Ease of reverse engineering: A product that is highly modular is easier for competitors to understand through benchmarking activities, while more integrated functions are much more subtle and obscure the underlying design relationships.

Increased unit variable costs: Since components are not integrated, part count is higher than for individually-designed products. Hence, the variable costs of modular products will be higher.

Excessive product similarity: Using common modules leads to products that “look” very similar, except when the variety occurs at the user interface. Automobile interiors are examples of
modules that can appear to the customer to be very different when in fact they can attach to common frame components through a common mechanical interface.

Need for clear technological understanding/mature technology: Without a thorough understanding of the technology that will comprise the products, or some structured method like the one proposed in this research, it will be difficult to select the appropriate module boundaries to achieve proper product variety.
3.0 Industrial Context
The Center for Innovation in Product Development, an NSF Engineering Research Center, encourages the performance of research in conjunction with one of its industrial partners. The goal is to provide a practical example of the research, as well as to ensure that the research achieves a suitable level of industrial applicability to meet the needs of the sponsors. This research was conducted with the participation and support of the Xerox Corporation.

3.1 Xerox Corporation
Xerox Corporation [3.1] began as the Haloid Company of Rochester, New York. Before its licensing of the Battelle Memorial Institute-refined Chester Carlson 1938 invention, Haloid manufactured photographic paper. In 1948, Haloid marketed the first product to incorporate the electrophotographic/xerographic process, and with the success of the first plain-paper office copier in 1961, the company changed its name to Xerox.

Today, Xerox is an international corporation that continues to shift its focus toward document technology, an idea encompassing the entire process of creating, handling, and storing documents. Xerox is also firmly committed to further integration of digital technologies into its products.

3.2 Electrophotography/Xerography
Most Xerox products share a common process rooted in Chester Carlson’s original discovery. This basic electrophotographic process [3.2] consists of creating a document image and transferring that image to a substrate, the media upon which print marks are made. An intermediate photoreceptive media is developed with a charge pattern that attracts minute, pigmented toner particles to it. These particles are then transferred and bonded to the substrate to produce a finished document. Figure 3.1 shows the arrangement of the process steps around the central media, or “drum.” To limit the scope of the analysis, we constrained our view to the substrate handling subsystem, which consists of components related to transferring the substrate through the system boundary.
3.3 Products
Considerable product variety exists in the copier/printer industry. Xerox divides its product offerings into categories that focus on particular customer groups or different product features. Figure 3.2 shows the general categories into which the products in their overall portfolio fall. Xerox also offers service products such as maintenance and supply contracts, and software products like their web-based document-organizing system called DocuShare.

Figure 3.1. A schematic of the major steps in the xerographic process (from Pai and Springett)

Figure 3.2. Examples from the Xerox hardware product portfolio.
Xerox offers multiple products in each segment of their portfolio. Common components are shared not only within segments, but also across them.

For document creation products, examples of major vectors of product variety are color, productivity, and usable types of substrate. Dominant designs, such as continuous-feed web printers like the one in Figure 3.3, and sheet-fed machines form the basis of product families for the major market competitors.

Figure 3.3. An example of a printer/print engine.
4.0 Market Factors
A crucial aspect of the design of a product architecture is careful consideration of existing and anticipated market factors. In this work we will use customer-gathered information, in the form of customer needs as the basis for a set of product characteristics or features to be proposed. These features are classified by the nature of their respective attribute levels or target values. These target values are combined to form the basis of uses, representing possible products, which customers evaluate in terms of their individual preferences.

4.1 Customer Needs
A customer need is a solution-indeterminate statement that expresses the key underlying aspects of customer consideration. Griffin and Hauser define it as “a description, in the customer’s own words, of the benefit to be fulfilled by the product or service.”[4.1] The underlying values that customers have is emphasized, as is the clear requirement that the product satisfy them. Customer needs are important for understanding what a customer wants from a product, and serve as the evaluative criteria against which any product will be measured. Customer needs [4.2] are discovered through customer interrogation. Needs are either directly captured in the language used by the customer, or can be interpreted from it. The preferred technique for gathering this data ranges from interview, questionnaire, focus group, monitored product use, and other means, all of which are intended to determine the Voice of the Customer. The needs have relative importance differences between customers which can be exploited through design optimization or a range of product offerings. Customer needs can be grouped according to their universality and their ability to be expressed.

4.1.1 Core product needs
From the set of identified customer needs, a distinction may be made in terms of universality of acceptance. Needs which are important to all customers are classified as core product needs. This set encompasses the basic functionality of a product, those functions that any customer expects from any product in the given category.

4.1.2 Variant needs
Whenever customers differ on the importance of the satisfaction of a need by a product, or when the exact value of the need differs from customer to customer, that need is classified as a variant need. An example of an importance variant need is coolness on a summer day for an automobile
customer. This need may not be very important for someone in Alaska, but may be critical for one in the South. A value variant need could be color, since different people might prefer different frequencies of reflected light from a product. Variant needs, as the name implies, may be addressed through separate products or variant modules that may be substituted, included, or excluded from any given product offering.

4.1.3 Latent needs
Another type of customer need is the latent need, a need that is difficult for customers to express since current products do not address them. Because they are so ephemeral, satisfying these needs can lead to dramatically successful products that address desires that customers often don’t realize that they have since they are not offered products that invoke them.

4.1.4 Kano Interpretation
The Kano diagram [4.3] describes the link between product performance and customer satisfaction. The Kano diagram can be used to understand different types of customer needs. Kano identifies three classes of customer needs. The first set are the needs that must be met to satisfy all customers. Performance below some definite level (although this level may be different for different customers) is unacceptable and will result in a product without a market. This group is known as the dissatisfiers. An example of a dissatisfier customer need is the ability to use 110V AC to power a U.S.-intended coffee maker. For the satisfiers group, customers will be increasingly satisfied by increases in performance, and the correlation is roughly linear. We can also consider these to be the needs that, through technological injection or improvement, will continue to improve over time, as will customer expectations of them. Pages per minute for a copier is such a need. Productivity must continue to improve in order to satisfy customers with future product offerings. The third case is the delighters, where any improvement (or inclusion) is met by exponential growth in satisfaction. These are customer needs that surprise the customer, and supplying them in any amount will thrill the market. Figure 4.1 illustrates these classes graphically.
By thinking about customer needs in these terms, it is possible to better define the specifications and future scope of a product platform. The needs that are the dissatisfiers must be included in the platform or distributed throughout all applicable modules. The needs that are the linear satisfiers must be considered as modules that will be continually improved in successive product offerings. The delighters should be modules that can be either implemented or excluded from the product by virtue of the flexibility inherent in the architecture.

It is important to note that although the Kano diagram provides a method of relating needs to the progression of product performance, it trends toward increasing performance, proliferation of features, and higher cost for individual products, which are the observed trends for mature architectures mentioned earlier.

4.2 Customer needs to product characteristics and target values
Since customer needs are the underlying desires of the market, they are not inherent solution concepts. A product characteristic or feature (used interchangeably here) is an aspect or attribute of a product that either partially or completely addresses a customer need. They are the manner in which a product responds to the challenges customer needs present. Each need will have a set of related features. For simplification of the industrial example, a single product
characteristic fulfills a customer need. For each product characteristic \( c \), a set of possible target values exists. This set depends upon the type of feature, which can be classified into one of three groups. Binary features are attributes that are either present or absent in the product from the customer’s perspective, and typically assume a single value when they appear in a product. The set of target values for product characteristics of this group is (4.1). Discrete-value features are attributes that can adopt any point-value selected from a set of options, but don’t sensibly exist at multiple levels. For this group, the attribute value \( v \) is selected from the set of possible values described by (4.2). Discrete-value features are a subset of binary features, in that they are desired, but only at specified values. Multiple-value features, an extension of discrete-value features, are robust attributes that exist as a subset of the design space. This subset can be continuous, multiple disparate continuous ranges, or discrete values that ignore uncertainties or tolerance bands. These features are typically the aggregates of individual use case requirements. Target values for adjustable features fit into this classification and are of the general form (4.3).

\[
\begin{align*}
  v_b & \in C_b = \begin{cases} 0 : \text{feature absent}, \\ 1 : \text{feature present} \end{cases} & (4.1) \\
  v_d & \in C_d = \{v_1, v_2, v_3, \ldots, v_m\} & (4.2) \\
  v_M & \subseteq C_M = \{[a_1, b_1], [a_2, b_2], \ldots, [a_l, b_l]\} & (4.3)
\end{align*}
\]

All target values are organized in rank order determined by quantification or some linearized qualitative relationship where appropriate. For example, power from lowest to highest, length from shortest to longest, and so on. Target values that cannot be ordered except for binary or two-level features will be difficult to interpret through a correlation analysis, since trends in the variation of utility will not necessarily be relevant.

4.3 Target values to product uses

When they purchase a product, customers desire it to accomplish something. A product use (4.4) is the value of the product characteristic vector for a customer’s usage or particular application.

\[
U = \left[\left( v_1 \in c_1 \right) \land \left( v_2 \in c_2 \right) \land \ldots \land \left( v_k \in c_k \right)\right] \quad (4.4)
\]
For example, if a subset of product characteristics for a toaster is [Appealing color, Multiple slices], and the product characteristic vector for all the potential products is [(White, Black, Chrome), (Two, Four, Six)], a use case could be [White, Two].

A use in the conventional sense, such as an activity, may only apply to a subset of the elements in the product characteristic vector. When applied to the previous example, the use/activity “Toast a bagel” would only relate to the “Multiple slices” product characteristic, not necessarily to the “Appealing color” attribute.

Use cases are the characteristics necessary for products that are the entire set or a subset of potential uses for a customer. It is evident that the most basic marketable product must be capable of fulfilling a single use case.

While all possible combinations of products can be defined from this vector, we are at most interested in all the use cases revealed through the analysis of market data. Since they are generated for specific applications, use case target values are assumed to be point values with no variance. Variance will be calculated through summation of the individually importance-weighted use cases across segments or the general market.

For mass customization products, products that are tailored to the needs of individual users, it is possible each use could describe a potential product, since it defines a value for each necessary product characteristic. This would be true if each product were used once, then disposed. For non-single-use products, the different uses must be satisfied by the purchase of a single product, either through a nominal target value that the customer finds acceptable for all the uses, or through an adjustment on the product that the customer can use to meet each usage target value. The relationship among needs, features, target values, and uses is graphically represented in Figure 4.2.
Customers are represented in this context by uses or collections of multiple uses that satisfy both their needs and their requirements during product action.

4.4 Market characterization
Many techniques have been developed and applied to the analysis of markets for the proper placement of products. Lancaster [4.4] considers the economic factors of product placement for monopolistic and other forms of competition. Moore et al [4.5] use conjoint analysis to select product target values that will optimize either market share, profit, or contribution. These all ignore use variance, and treat customers as averages over their uses. They also do not indicate how to design products to arrive at these targets, but merely suggest what the most successful targets will be.

In Yu et al [4.6], customer need levels are evaluated at different uses of the product through surveying techniques. Figure 4.3 reviews the analytical steps. The segment and population means and deviations are calculated for each need, and these derived values are compared to arrive at a preferred architecture style given the observed relationship.
Customer needs and product attribute levels are evaluated instantaneously and throughout the uses of the product by the customers through surveying techniques. The segment and population means and deviations are calculated for each need, and these derived values are compared to arrive at a preferred portfolio architecture style given the observed relationship. Figure 4.4 illustrates the decision tree for such an analysis.
If the population mean is not fixed in time, then customer requirements are evolving. By developing a portfolio of products around a platform that is refreshed in successive generations, the portfolio average can follow the population average. If the population deviation is not large, then the implied solution concept is to offer an integral architecture that offers a common form solution for all products. When the segment deviations match the overall market deviations, this implies that all customers require the specified feature at all target values throughout the product uses. One way to accomplish this is to create an architecture that adjusts according to the customers’ use. Finally, if the deviations of the segments do not match the population deviation, then a modular product family organized around an interchangeable module or set of modules is appropriate.

The technique relies upon data fitting the normal model, which is acceptable for single use cases. Creating a suitable representation of a customer's requirements must incorporate several applicable use cases, but often this aggregate use case information is multimodal, and the normal model cannot be completely justified. This is readily apparent for binary features, whose distributions at the use case level resemble Bernoulli trials, or discrete value product characteristics that have two or more target levels. Through the selection of multiple portfolio targets, market average performance, which might not satisfy any customers [4.7] may be avoided.

The statistical parameters mean and standard deviation, when computed for the feature target value distributions across the intended customer base can be used to justify some portfolio architecture decisions. These distributions shed light on how variety in performance can map to variety in form across a portfolio, but do not necessarily define the architecture of the individual products. Furthermore, the distributions can be used to segment the overall market and seek
correlations to significantly reduce the number of products in the product portfolio from the set implied by the total number of possible combinations of target values regardless of the form by which the features are manifest.

4.5 Market characterization – correlation analysis
While the preceding method interprets the use-varying features of each customer need/product characteristic, it does not retain the links among the individual needs at the customer or segment level. The result is a set of architecture recommendations that supports product variety in excess of the actual requirements of the market if the needs are not orthogonal. This is to say, if there is a relationship between the form that two provided features take, and if the target values of the needs are correlated, then the features may be lumped together into a common element at the given target values. When two features are functionally unrelated, and thus potentially share no common components, then correlation does not aid the design effort, but merely identifies the product combinations that are most likely to succeed in the marketplace.

As an example, suppose that a survey of automobile customer needs reveals that two levels of available power and two levels of interior comfort are important. Even if large available power and high interior comfort correlate (i.e. customers want either high power and luxurious interiors, or low power and Spartan conditions) this does not necessarily imply that two systems are sufficient. Since available power relates to the engine/powerplant, and interior comfort corresponds with seating, dashboard, and door padding features, and these elements are normally separate, all four levels must be designed, even if only two of them are important. From a manufacturing perspective, unless the elements that provide the features can be integrated, the will have to be produced individually and assembled.

Only if there is no correlation among the needs and their target values will this technique reveal the necessary number of variants. In many cases, the actual number of variants is substantially less, as is the number of variants that the company can effectively manage and manufacture.

Well-defined and demonstrated segments or sub-markets within the intended market are identified if they exist. A market segment is defined as a set of customers who share common needs. Often, demographics like sex, nationality, age, and other classifications [4.8] are used to group customers, and real data or stereotypes are used to determine their needs and preferences. provided that the sample size is sufficiently large and random, the market segments should
materialize from the data through cluster analysis [4.9] techniques. The method of the previous section is modified to treat segments and use cases, and also to reduce the number of products according to correlation analysis.

Each segment (or customer, if segmentation is not assumed) is given a normalized importance weight, \( \omega_s \), such that the sum of all importances across the segments equals 1 (4.5). The disposition of this weight factor across the segments can be on the basis of the importance of satisfying the specified group, the expected profitability of the segment, the size of the segment relative to the others, or other strategic factors.

\[
\sum_{i=1}^{m \text{ segments}} \omega_s (i) = 1
\]

(4.5)

Within each of these segments, customer need/product characteristic information is gathered to determine the relevant product uses. A product use vector, \( u_i \), is constructed for each use. A normalized importance weight, \( \omega_i \), is given to the use relative to all the other uses for that segment such that sum of the use importances for each segment equals 1 (4.6). Customers place differential importance on their uses. The importance weight \( \omega_i \) is the product usage time proportion spent in the use case. For a customer’s/segment’s set of \( p \) use cases,

\[
\sum_{i=1}^{p \text{ uses}} \omega_i (u_i) = 1
\]

(4.6)

The use vector containing target value information for each segment’s/customer’s \( p \) uses is constructed. The same is done for all \( m \) segments to produce a set of all the possible product uses for the entire market and respective \( \omega_i \) importance weights.

We need to consider each segment/customer relative to all the possible product options available or requested in order to construct a distribution for each product characteristic. In order to do this, we construct a product family use vector \( U_f \), consisting of all the target values revealed from the market survey for each product characteristic. Each use is then a subset of this product family use vector. We must now determine an appropriately normalized contribution for each use to the overall market distribution across the target values for each product characteristic.
We first characterize each segment/customer. We construct the segment use vector, \( U_s \), which is analogous to \( U_T \) except it will only aggregate the uses of that particular segment. We determine the total \( \omega_v \) for each target value \( v \) of each product characteristic. Within the segment use vector, any product characteristic comprised of \( q \) target values has a distribution of the form (4.7) where (4.8) and (4.9).

\[
S=[\omega_1, \omega_2, \ldots, \omega_q] \quad (4.7)
\]

\[
\omega_{v,i} = \sum_{i=1}^{p} \beta \omega_i \quad (4.8)
\]

\[
\beta = \begin{cases} 
0 : v(u_i) \neq v_j \\
1 : v(u_i) = v_j 
\end{cases} \quad (4.9)
\]

From these constructed distributions, assuming the normal model, we can calculate estimates for the mean and standard deviation of each product characteristic, and then consider an acceptable range for customer preference. For example, target values within the range \([\mu-\sigma, \mu+\sigma]\) may be the portfolio-critical values, and target values outside this range may be excluded to a 66% confidence level.

This method will only be helpful provided intracustomer variance (variance among the target values of a customer’s uses) is less than intercustomer variance (variance between customers). As the two approach each other, then meaningful conjecture is obscured by the lack of product definition, and the architectural suggestion in this case is adjustability or to offer modular solutions through products that only satisfy individual uses. The tradeoff then becomes whether or not a customer will purchase a product that will not perform all of their uses. Particular market segments with high use case variance can possibly be subdivided into smaller segments with lower individual variances.

As an illustrative example, we have a product family consisting of two product characteristics, A and B. We survey two customers, C1 and C2, for their important uses and find the following information.

<table>
<thead>
<tr>
<th>Customer</th>
<th>( \omega_1 )</th>
<th>Use</th>
<th>( \omega_2 )</th>
<th>( A )</th>
<th>( B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>0.7</td>
<td>1</td>
<td>0.8</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>0.2</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>0.3</td>
<td>3</td>
<td>0.4</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>0.6</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
Organized as \( U_s \) vectors in a matrix, the target values are written as

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>UC2</td>
<td>5</td>
<td>10</td>
</tr>
</tbody>
</table>

The corresponding importance weight contributions from the uses at the defined values of the characteristics are

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC1</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>UC2</td>
<td>0</td>
<td>0.6</td>
</tr>
</tbody>
</table>

And the aggregate across the customers (the whole market) using the \( \omega_s \) term is

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>0.56</td>
<td>0.32</td>
</tr>
</tbody>
</table>

This M vector forms the basis of a probability distribution function for each product characteristic. For example, the A distribution appears as \( N(3.36, 4.64) \), while the B distribution is \( N(6.76, 7.53) \).

Arranged into a market matrix, the values can be easily calculated using a spreadsheet. The following figure shows the arrangement of the market information into such a matrix.

[Figure 4.5. Market information matrix organized by needs/target values and segments/uses.]
The columnar major divisions are the identified customer needs/product characteristics. The minor columnar divisions are the target values for these needs. The major row divisions are the customer market segments with their respective segment importances. The individual rows are the product uses for each segment with their respective importance weights. The last row for each market segment is the aggregate segment distribution for each need, and the last row below all segments is the overall aggregate market distribution.

4.6 Xerox example
The process described was applied to a Xerox market to determine the best product variants, in terms of the proper combination of target values, to offer to the market, and to see if there were any architecture implications through the described techniques. The market data included here uses publicly attained information and is not necessarily correct in its detail in order to prevent disclosure of proprietary information. The results do not reflect Xerox current or future product plans or Xerox product development policy or procedure, but is meant as a demonstration of the process on a market opportunity for an industrial partner.

4.6.1 Customer needs to product characteristics
The initial set of customer needs was compiled from a set of data aggregated from public information, interviews, and independent research firm results [4.12-4.27]. From this set of data, a set of ten key customer attributes was selected to use for the research example. We selected these key product characteristics related to the important underlying customer needs in order to address intramarket preference differences and to help define the scope of products for the market. The following list defines the set of product characteristics examined. These general features apply to most products offered by Xerox and their competitors.

1. Number of selectable substrates: This refers to the number of enabled types of substrate that can be used, at any given time, to create a document. It is the number of available feed trays in a conventional copier, for example, since given n trays, only n types of substrate may be loaded at any time, and only documents incorporating n or fewer different types of substrate may be constructed.
2. **Productivity:** Productivity is the measure of process speed, in terms of document pages per minute, adjusted to a standard A4 or Letter stock size. Current products operate between 7 and 180 cut sheets per minute, and significantly higher in the case of continuous web systems.

3. **Stock type:** This is an indication of the broad class of substrate used by the machine. Two types are currently available: continuous web and cut sheet.

4. **Colors:** This refers to the number of base colors used by the system to produce the available color gamut. Current products vary from monochromatic, to highlight, to full process color, to extended gamut full process color. Monochromatic uses a black pigment, while highlight adds an additional one or two colors to this palate. Full process uses four primary pigments (cyan, magenta, yellow, and black) to produce a range of colors, while extended gamut systems add pigments to this set to increase the range of colors that can be created.

5. **Duplex:** Duplex refers to the ability to print to both sides of a substrate. The term for a one-sided document is Simplex.

6. **Substrate caliper:** Caliper in this context refers to the thickness of the substrate used.

7. **Substrate weight:** Weight refers to the average mass per square meter of a sheet of the substrate, and is expressed in grams per square meter.

8. **Substrate sizes:** This refers to the length and width dimensions of the substrate.

9. **Image quality:** This refers to a metric that evaluates the produced document on the basis of several quantitative and qualitative factors. The higher the value on this one to five scale, the higher the overall quality of the produced document.

10. **Variable data applicability:** This refers to whether or not the customer identifies the particular application as being able to use the new technology capability of variable data. Variable data is the creation of a unique digital image for every page of a document, instead of the mechanical reproduction of a single document page. The technique is currently used by direct mail marketers to produce individually-tailored documents for specific customers to increase the probability of success.

### 4.6.2 Target values

For each product characteristic, a suitable set of target values was proposed based upon existing products and VOC feedback. The target values are listed in Figure 4.6.
<table>
<thead>
<tr>
<th>Product Characteristic</th>
<th>Performance Target Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selectable Substrates</td>
<td>1</td>
</tr>
<tr>
<td>Productivity</td>
<td>Low</td>
</tr>
<tr>
<td>Stock Type</td>
<td>Web</td>
</tr>
<tr>
<td>Colors</td>
<td>Black</td>
</tr>
<tr>
<td>Duplex-Capable</td>
<td>Yes</td>
</tr>
<tr>
<td>Substrate Caliper</td>
<td>Thin</td>
</tr>
<tr>
<td>Substrate Weight</td>
<td>Light</td>
</tr>
<tr>
<td>Substrate Size</td>
<td>Extra-Small</td>
</tr>
<tr>
<td>Image Quality</td>
<td>1</td>
</tr>
<tr>
<td>Variable Data</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Figure 4.6. Table of product characteristics and performance target values.

4.6.3 Market segmentation and applications

The use of clustering techniques as discussed earlier is the statistically-valid method of developing market segments. In this example, the target market exists, and thus the product family is intended to be a substitution good. Market research data for each segment was used to evaluate the preferences of the customer base. Ideally, data would be directly provided from the intended customers, but since a wealth of raw data was available it was used in lieu of a formal, independent marketing survey to demonstrate the technique. Careful questioning to determine the importance of the use cases and underlying needs, accompanied by analytical clustering techniques will help to assemble similarly structured information.

For the purposes of efficacy of research and applicability for the industrial sponsor, five customer groups, established and documented in the Xerox-targeted industry, were selected. These market segments (what Xerox calls “environments”) are defined as the following:

1. Central Reprographics Department (CRD): CRDs are corporate-contained resources for creating documents. They typically handle all corporate printing needs of the companies that maintain them.
2. Quick Printer (QP): Quick printers are job shops for off-the-street customers or small corporate accounts. Kinko’s is an example of a quick printer.
3. Commercial Printer (CP): Commercial printers produce the widest range of documents. Their applications include
4. Book Printer (BP): As the name implies, these customers produce books, booklets, and other finished documents.
5. Form Printer (FP): These customers produce single or multiple page forms.

These segments were weighted relative to each other by the number of document pages each produces per year. This was done in order to assign the \( \omega \) weights, and it indicated the relative importance of each environment. Other potential factors to adjust this weight may be included, such as importance of segment, profit, or other strategic attributes.

Each environment/segment produces documents called applications. Figure 4.7 lists these documents. Ideally, an application is a point-value product use. Since no unique definition exists for any given application (i.e. a book may vary in terms of size, substrate, etc.) each application is actually a distribution based upon market information.

<table>
<thead>
<tr>
<th>Advertising</th>
<th>Direct Mail</th>
<th>Magazines</th>
<th>Posters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Books</td>
<td>Directories</td>
<td>Manuals-tech</td>
<td>Packaging</td>
</tr>
<tr>
<td>Booklets</td>
<td>Flyers</td>
<td>Manuals-training</td>
<td>Reports</td>
</tr>
<tr>
<td>Brochures</td>
<td>Forms</td>
<td>Newsletters</td>
<td></td>
</tr>
<tr>
<td>Catalogs</td>
<td>Labels</td>
<td>Newspapers</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.7. Applications considered as product uses.

4.6.4 Target value distributions and characterization
The market data was analyzed according to section 4.5 to create market and segment target value distributions and to calculate mean and standard deviations. The overall matrix is presented as Appendix A.

Each class of product characteristic has an inherent preferred expression in the architecture domain depending upon the nature of the underlying distribution. The implications are considered irrespective of cost.

For discrete-value features, differential preference across segments implies modularity, while uniform preference implies inclusion of the feature in the platform. Differential preference within segments is best suited to adjustable architectures, since it indicates that the segment’s use cases necessitate performance at multiple levels during the product’s useful life.

1. Number of selectable substrates
Figure 4.8 displays the distributions of the five market segments and the total market as cumulative distribution functions. A cumulative distribution function is helpful for determining the portion of the market that would be satisfied with a given level of product performance. We can consider that, since ranges of performance are available through the fixed adjustable type, providing performance up to target value three will satisfy more than 80% of the total market.
We can also use the probability distributions in the correlation analysis to pick the best target values and match them with other feature target values to choose the variants. The portfolio should adopt a modular product family architecture, as current Xerox products demonstrate by offering individual modules for additional substrates.

![Selecteable Substrates Cumulative Distribution](image)

Figure 4.8. Selectable substrates target value cumulative distributions.

2. Productivity

Figure 4.9 displays the results of the segment-based analysis as a set of probability density functions. Four of the five segments are interested in medium to high productivity from their products, while the Form Printer segment only values high productivity. Since no segment appears to value the low target value, and it lies outside of the portfolio-critical range, it can be eliminated from the overall set of target values. The eventual product portfolio will then contain only the medium and high productivity options offered as part of a modular product family. Xerox products currently provide significantly higher productivity (doubling of output) through a pair of identical “print engines,” tantamount to purchasing two products.
3. **Stock type**

Given the possible options of sheet-based and web-based systems, the market data reveals three groups, as displayed in Figure 4.10. One group, containing the Form Printers, is interested in web-based products. The Quick Printers, when polled regarding their currently used products, highly favor the sheet-based models. All other segments use each to varying degrees or depending upon the particular application.

This does not conclusively illustrate that a hybrid concept is preferred, but rather that the other attributes coupled with the web or sheet system are preferred over the alternative. This recalls the preceding discussion of customer needs, since customers are most interested in their own
satisfaction, not necessarily in the method through which they are satisfied. When the segment weights are applied and the market distribution is calculated, the result is centered between the two architectures and has a large deviation. Again, a modular product family is recommended. Current Xerox products separate into integral portfolio architectures according to web and sheet systems and are not the result of variation through module substitution.

4. **Number of colors**
This is an example of a multiple-valued feature expressed as a set of probability density functions. Segment variance is indicative of indistinct segment preferences, except for book printers who prefer black and form printers who favor highlight color for bank statements and other documents. Colors should also be provided through a modular product family according to the market target value distributions. Current Xerox products either offer fixed numbers of colors, or are capable of adjusting depending upon the number of toner colors supplied.

![Colors](image)

**Figure 4.11.** Number of colors target value distributions.

5. **Duplex printing**
Customers uniformly prefer duplex printing products, with few use exceptions like posters, flyers, and labels. Due to the uniform preference, the portfolio architecture indication is to provide duplex capability through a fixed integral element. Xerox products similarly provide duplex capability integrally.
6. **Substrate caliper and weight**
All segments are centered at the medium weight and thickness value, and they vary only in their deviation. Weight and caliper are highly correlated, when the properties of a range of substrates are analyzed. These features are best provided, due to the high variance within each segment and across all segments, through adjustable portfolio solutions in every product.

![Figure 4.12. Substrate caliper and weight target value distributions.](image)

7. **Substrate sizes**
The segments are centered at the medium value, but their variances indicate the inclusion of small and large sizes, from a market share perspective, are warranted. Adjustable products are warranted, if the products can be designed to adjust according to document size across the given range. Current Xerox products have specific maximum length and width that is adjustable within a particular range.

![Figure 4.13. Substrate size target value distributions.](image)
8. Image quality
As is clear in Figure 4.14, the majority of form and book printer uses do not require high image quality. Other segments are distributed over the low and intermediate values to produce the combined market distribution covering values one through three. A modular solution is recommended due to the differences in segment deviations.

![Image Quality PDF](image)

Figure 4.14

9. Variable data
Since none of the segments have a strong preference for the inclusion of this feature in their products, and the preference distribution is uniform across all segments (a small percentage interested in having it), the portfolio architecture recommendation is for an integral architecture without the feature present. If the feature is desired anywhere in the portfolio, then it should be implemented through a modular architecture. Had specific segments been highly interested in the feature, the differential preference (utility) would be best exploited through modularity.
This is an example of how market data taken alone might lead to a problematic or overly complicated solution. When the functionality of a feature like “variable data” is considered, it will be shown that a modular solution is inappropriate because it requires a different data paradigm that has a global effect.

The overall matrix with obfuscated data of segment averages and standard deviations is provided as Figure 4.16. The actual data has been modified so as not to reveal proprietary information and to indicate the trends of information gathered through publicly available resources. Now that the individual segment and market distributions are known or computed, the methodology may be applied to reduce the number of product variants possible to construct from an architecture that is modularized along the divisions of Figure 4.6 from the combinatorial total of 81000 to a “manageable” quantity of actual products to produce by exploiting the trends they contain. The major reduction methods are the elimination of unwarranted target values, identification of platform and adjustable features, and the correlation of remaining product characteristics not reduced through one of the other methods. The portfolio architecture permits many products, but it does not say that they must necessarily be built. The task is to decide which ones are necessary for market success.
<table>
<thead>
<tr>
<th></th>
<th>Selectable Substrates</th>
<th>Productivity</th>
<th>Stock Type 1</th>
<th>Colors 1</th>
<th>Duplex 1</th>
<th>Substrate Caliper 1</th>
<th>Substrate Weight 1</th>
<th>Substrate Size 1</th>
<th>Image Quality 1</th>
<th>Variable Data 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP</td>
<td>Diff -0.08</td>
<td>-0.09</td>
<td>0.11</td>
<td>0.27</td>
<td>0.00</td>
<td>-0.05</td>
<td>-0.01</td>
<td>0.06</td>
<td>0.08</td>
<td>-0.04</td>
</tr>
<tr>
<td></td>
<td>Mu 2.28</td>
<td>2.36</td>
<td>1.63</td>
<td>2.22</td>
<td>1.00</td>
<td>2.02</td>
<td>2.00</td>
<td>3.25</td>
<td>1.78</td>
<td>1.66</td>
</tr>
<tr>
<td></td>
<td>Var(σ²) 0.19</td>
<td>0.24</td>
<td>0.23</td>
<td>0.33</td>
<td>0.00</td>
<td>0.12</td>
<td>0.13</td>
<td>3.35</td>
<td>0.19</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td>Sigma 0.44</td>
<td>0.49</td>
<td>0.48</td>
<td>0.57</td>
<td>0.00</td>
<td>0.34</td>
<td>0.36</td>
<td>0.60</td>
<td>0.44</td>
<td>0.35</td>
</tr>
<tr>
<td>BP</td>
<td>Diff -0.30</td>
<td>0.12</td>
<td>-0.20</td>
<td>-0.16</td>
<td>0.00</td>
<td>0.11</td>
<td>0.15</td>
<td>0.02</td>
<td>0.30</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>Mu 2.06</td>
<td>2.57</td>
<td>1.32</td>
<td>1.77</td>
<td>1.00</td>
<td>2.19</td>
<td>2.15</td>
<td>3.21</td>
<td>2.00</td>
<td>1.94</td>
</tr>
<tr>
<td></td>
<td>Var(σ²) 0.20</td>
<td>0.22</td>
<td>0.22</td>
<td>0.35</td>
<td>0.00</td>
<td>0.08</td>
<td>0.11</td>
<td>0.24</td>
<td>0.03</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>Sigma 0.45</td>
<td>0.47</td>
<td>0.46</td>
<td>0.60</td>
<td>0.00</td>
<td>0.29</td>
<td>0.33</td>
<td>0.49</td>
<td>0.16</td>
<td>0.23</td>
</tr>
<tr>
<td>FP</td>
<td>Diff 0.69</td>
<td>0.35</td>
<td>-0.49</td>
<td>-0.07</td>
<td>0.00</td>
<td>-0.03</td>
<td>0.05</td>
<td>-0.27</td>
<td>-0.70</td>
<td>0.06</td>
</tr>
<tr>
<td></td>
<td>Mu 3.05</td>
<td>2.80</td>
<td>1.03</td>
<td>1.89</td>
<td>1.00</td>
<td>2.05</td>
<td>2.05</td>
<td>2.92</td>
<td>1.00</td>
<td>1.96</td>
</tr>
<tr>
<td></td>
<td>Var(σ²) 0.16</td>
<td>0.08</td>
<td>0.03</td>
<td>0.06</td>
<td>0.00</td>
<td>0.02</td>
<td>0.02</td>
<td>0.25</td>
<td>0.00</td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td>Sigma 0.40</td>
<td>0.28</td>
<td>0.17</td>
<td>0.24</td>
<td>0.00</td>
<td>0.15</td>
<td>0.15</td>
<td>0.50</td>
<td>0.00</td>
<td>0.20</td>
</tr>
<tr>
<td>QP</td>
<td>Diff 0.03</td>
<td>-0.24</td>
<td>0.48</td>
<td>-0.74</td>
<td>0.00</td>
<td>0.05</td>
<td>-0.33</td>
<td>0.08</td>
<td>-0.22</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Mu 2.39</td>
<td>2.21</td>
<td>2.00</td>
<td>1.22</td>
<td>1.00</td>
<td>2.13</td>
<td>1.68</td>
<td>3.27</td>
<td>1.48</td>
<td>1.89</td>
</tr>
<tr>
<td></td>
<td>Var(σ²) 0.21</td>
<td>0.28</td>
<td>0.00</td>
<td>0.10</td>
<td>0.00</td>
<td>0.06</td>
<td>0.24</td>
<td>0.13</td>
<td>0.15</td>
<td>0.10</td>
</tr>
<tr>
<td></td>
<td>Sigma 0.45</td>
<td>0.53</td>
<td>0.00</td>
<td>0.32</td>
<td>0.00</td>
<td>0.25</td>
<td>0.49</td>
<td>0.36</td>
<td>0.39</td>
<td>0.31</td>
</tr>
<tr>
<td>CRD</td>
<td>Diff 0.13</td>
<td>-0.16</td>
<td>0.31</td>
<td>-0.41</td>
<td>0.00</td>
<td>0.05</td>
<td>-0.18</td>
<td>-0.09</td>
<td>0.13</td>
<td>-0.01</td>
</tr>
<tr>
<td></td>
<td>Mu 2.49</td>
<td>2.29</td>
<td>1.83</td>
<td>1.55</td>
<td>1.00</td>
<td>2.13</td>
<td>1.83</td>
<td>3.10</td>
<td>1.83</td>
<td>1.89</td>
</tr>
<tr>
<td></td>
<td>Var(σ²) 0.22</td>
<td>0.24</td>
<td>0.14</td>
<td>0.28</td>
<td>0.00</td>
<td>0.08</td>
<td>0.22</td>
<td>0.14</td>
<td>0.22</td>
<td>0.10</td>
</tr>
<tr>
<td></td>
<td>Sigma 0.47</td>
<td>0.49</td>
<td>0.38</td>
<td>0.53</td>
<td>0.00</td>
<td>0.28</td>
<td>0.47</td>
<td>0.38</td>
<td>0.47</td>
<td>0.31</td>
</tr>
<tr>
<td>Market</td>
<td>Diff 0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Weighted</td>
<td>Var(σ²) 0.15</td>
<td>0.06</td>
<td>0.16</td>
<td>0.20</td>
<td>0.00</td>
<td>0.01</td>
<td>0.04</td>
<td>0.02</td>
<td>0.16</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Sigma 0.38</td>
<td>0.24</td>
<td>0.39</td>
<td>0.45</td>
<td>0.00</td>
<td>0.07</td>
<td>0.20</td>
<td>0.15</td>
<td>0.40</td>
<td>0.04</td>
</tr>
</tbody>
</table>

Figure 4.16. Matrix of product characteristic distribution data (obfuscated data)

4.6.5 Elimination of unwarranted variety
Upon surveying the market, many of the anticipated target values should be eliminated due to lack of appreciable customer demand. We discovered that several target values were not necessary, thus reducing the number of product combinations. The reduced product characteristic target value matrix is provided as Figure 4.17. Note that some distributions indicate further reductions, but the rationale behind these decisions will be provided in the following sections. The number of possible product variants that the architecture could support is reduced through the combinations of target values to 7776.
<table>
<thead>
<tr>
<th>Product Characteristic</th>
<th>Selectable Substrates</th>
<th>Performance Target Values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Moderate</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>Web</td>
<td>Sheet</td>
</tr>
<tr>
<td></td>
<td>Black</td>
<td>Highlight Full Color</td>
</tr>
<tr>
<td>Duplex-Capable</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Substrate Caliper</td>
<td>Thin</td>
<td>Medium Thick</td>
</tr>
<tr>
<td>Substrate Weight</td>
<td>Light Medium Heavy</td>
<td></td>
</tr>
<tr>
<td>Substrate Size</td>
<td>Small Medium Large</td>
<td></td>
</tr>
<tr>
<td>Image Quality</td>
<td>1</td>
<td>2 3</td>
</tr>
<tr>
<td>Variable Data</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Figure 4.17. Reduced product characteristic target value matrix.

4.6.6 Identification of integral platform features
Uniform acceptance of product features across all market segments, or, alternatively, low customer dissatisfaction and low implementation costs are clear signals for feature inclusion into the product platform. Duplex capability is clearly a platform product characteristic, as evidenced by the distribution graph for all segments and the overall market. Since all market segments desire products that print to both sides of a document, by platforming this feature all products receive it. Through platforming such variety to the high-end value, the number of product alternatives the portfolio architecture is capable of supporting is reduced to 1944.

4.6.7 Identification of necessarily adjustable features
Whenever product features must vary beyond the range supported by a single target value in order to enable a use, an adjustable or robust technology/architecture is preferred. For example, while the population and segment means for substrate thickness and weight are centered within the range of possible materials, the high, similar standard deviations of the mean values for each segment and across the market indicates that an adjustable integral portfolio architecture is necessary. The form element to handle the range of substrate weights and thicknesses must adjust either automatically, through user input, or through substitution of components by the user when the properties exceed the performance limitations of the installed technology.

4.6.8 Correlation of remaining characteristics
At this point, we do not reduce the variety capability of the portfolio architecture, but rather decide which combinations of variant target values on each characteristic to offer as pairs, thereby defining which subset of all possible supported variants will be actually offered as real products. Six of the original ten product characteristics have not been addressed to this point. Characteristically, the market mean target values do not match those of the individual segments, and thus the market variances are appreciable even though the segment variances are not large.
In order to determine if correlation exists among the segments’ preferred values, we construct a matrix of the segment means and variances for each product characteristic, and determine the market means and variances for each characteristic. The variance is weighted by the $\omega_s$ segment importance weight. We calculate $r$, the correlation coefficient, by the pairwise evaluation of the product characteristics using (4.10)

$$
    r = \frac{\sum_{s} \omega_s (\mu_{si} - \mu_{i})(\mu_{sj} - \mu_{j})}{\sigma_i \sigma_j}
$$

where i and j are indices of the compared characteristics. Figure 4.18(a) graphically represents $r$. The $r$ values were arranged into the matrix in Figure 4.18(b). The gray rows and columns indicate the previously addressed characteristics. As $|r|$ decreases, the significance of the correlation is reduced. This analysis used a threshold value for $|r| = 0.6$, above which a correlation was presumed. Six pairwise combinations have $|r|$ values $> 0.6$. Assuming linearity, the attribute target values are combined by matching the order of the values with the sign of the coefficient. These relationships can be graphically arranged into a network composed of generally-valid structures that do not violate the relationships among the other network members.

![Target value relationships as $|r|$ approaches 1](image)

![Correlation matrix](image)

Figure 4.18. Correlation analysis

Figure 4.19 illustrates some valid and invalid structures. The positive and negative signs indicate the sign of the $r$-value. Individual nodes (the product characteristics that are correlated) are added to the network according to the magnitude of $r$, beginning with the largest. Figure 4.20
shows that the graph of the six highly correlated characteristics identified do not violate each other’s dependencies. The seventh characteristic (number of colors), which is not correlated with any of the other characteristics, is external to the correlation network. The number of variants must be increased by a factor of the number of necessary target values of this characteristic to accommodate all possible options.

![Figure 4.19. Valid and invalid structures.](image1)

![Figure 4.20. Structure from correlation matrix.](image2)

4.6.9 Resulting product family
It is easy to see that for a product that has more than ten features and multiple target values, the number of combinations would be difficult to manufacture. Figure 4.21 illustrates how the initial number of combinations can be reduced to a suitable set using what can be gleaned from market information.

<table>
<thead>
<tr>
<th>Method of Reduction</th>
<th>Number of Product Combinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>81000</td>
</tr>
<tr>
<td>Unwarranted variety</td>
<td>7776</td>
</tr>
<tr>
<td>Platforming</td>
<td>1944</td>
</tr>
<tr>
<td>Adjustability</td>
<td>216</td>
</tr>
<tr>
<td>Correlation</td>
<td>12</td>
</tr>
</tbody>
</table>

![Figure 4.21. Reduction of product combinations through each analytical step.](image3)

Using the target values for these characteristics creates a set of twelve product variants, listed in Figure 4.22. The correlated features reveal several interesting properties which can be carried onto further architectural evaluation. Highly correlated target values for different features can be combined through integral architecture, but this deviates from the desire to use modularity for the beneficial reasons listed in Section 2.6 to allow variants. Also, unless the correlated features have some functional commonality, and thereby share common components that may be shared, there is no reduction in design effort from correlation.
<table>
<thead>
<tr>
<th>Variant #</th>
<th>Selectable Substrates</th>
<th>Productivity</th>
<th>Stock Type</th>
<th>Colors</th>
<th>Duplex</th>
<th>Substrate Color</th>
<th>Substrate Weight</th>
<th>Substrate Size</th>
<th>Image Quality</th>
<th>Variable Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>Med</td>
<td>Sheet</td>
<td>K</td>
<td>Yes</td>
<td>Medium</td>
<td>Medium</td>
<td>Large</td>
<td>3</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Med</td>
<td>Sheet</td>
<td>K</td>
<td>Yes</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>High</td>
<td>Web</td>
<td>K</td>
<td>Yes</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>High</td>
<td>Web</td>
<td>K</td>
<td>Yes</td>
<td>Medium</td>
<td>Medium</td>
<td>Small</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>Med</td>
<td>Sheet</td>
<td>HL</td>
<td>Yes</td>
<td>Medium</td>
<td>Medium</td>
<td>Large</td>
<td>3</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Med</td>
<td>Sheet</td>
<td>HL</td>
<td>Yes</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>High</td>
<td>Web</td>
<td>HL</td>
<td>Yes</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>High</td>
<td>Web</td>
<td>HL</td>
<td>Yes</td>
<td>Medium</td>
<td>Medium</td>
<td>Small</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>Med</td>
<td>Sheet</td>
<td>Full</td>
<td>Yes</td>
<td>Medium</td>
<td>Medium</td>
<td>Large</td>
<td>3</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>Med</td>
<td>Sheet</td>
<td>Full</td>
<td>Yes</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>High</td>
<td>Web</td>
<td>Full</td>
<td>Yes</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>High</td>
<td>Web</td>
<td>Full</td>
<td>Yes</td>
<td>Medium</td>
<td>Medium</td>
<td>Small</td>
<td>1</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Figure 4.22 Variants resulting from correlation analysis.

The portfolio architecture should at a minimum support these listed product variants, since they should be the most valuable ones according to this analysis. Without consideration of functionality, we have at a baseline the features and target values that the market is interested in purchasing. We may make some claims about the manner in which the architecture should manifest them, but this is usually done with some notion of the form that the feature will take.

This forms the set of desired products for the market, whether they are designed individually or separately. The design task, and the exploration of the rest of the research, is how to partition the system distributed across the product portfolio into modules to deliver products that meet these performance and market requirements.
5.0 Function Analysis for an Entire Market
Identification of a product portfolio’s functional requirements and subsequent coordination with customer needs contributes to its overall success. QFD [5.1, 5.2] has been demonstrated to provide an accounting for this relationship by explicitly and hierarchically stipulating the customer needs and their relationship to product specifications. Other research [5.3] has also demonstrated that customer preference/utility estimation for monotonically increasing technology-based products can be useful in identifying performance targets for platformed products. This exercise is more complicated when the products are intended for a diverse market with varying customer needs, and also when the product architecture has not been assumed a priori.

Explicitly representing these functions schematically is accomplished through various methods, depending upon the nature of the system to be described. FAST [5.4], Hatley-Pirbhai [5.5], and other function-logic diagramming methods are attempts to illustrate the links among the sub-functions necessarily performed by products. Function logic diagrams explicitly represent the system and force the development team to clearly understand the relationships among functional elements. Function-logic diagrams also serve as artifacts of the development process and a common format for representing systems. Designers and developers may use them as a way to share their ideas and encourage discussion. Regardless of the method used, the result is a function-logic diagram that captures the set of interconnected functions deemed necessary for accomplishing the product use. Ideally, it is form-independent, and has sufficient granularity to describe the system at a level of appropriate detail. The goal is to produce a function view of the product portfolio through the assembly of product use function views.

5.1 Functional representation - function structure
For electromechanical systems, a widely used diagramming method is the function structure [5.6, 5.7], which connects sub-functions with flows of energy, material, and information. An example of a function structure for an ink jet printer is included as Figure 5.1.
The function structure explicitly defines relations among the functions through the flows. Bold arrows indicate material, like ink, which flows from its entrance into the system to be stored, until it exits as filtered ink and as a part of the printout. Dashed lines represent information flows, such as the input data, and energy flows are represented using thin lines. It is also important to note that the interface with the user occurs at the flow entrances and exits, the left and right sides of the diagram that are the system boundaries.

5.2 Use-case function structures
Product uses are helpful in developing function structures since they address the important aspects of the product from the customer perspective. The underlying customer needs must be satisfied by the product, but often these needs are insufficient to describe all the required product functionality, and the development team must supplement the needs with its own knowledge of the system. Many functions are either assumed or not considered by the customer, but are vitally important to the satisfaction of customer needs and system requirements. In addition, not all customers desire the same set of features for an ideal product. Thus, use case function structures represent individually tailored products.

All of the product uses for the Xerox substrate path were individually examined, and individual function structures were developed. Starting with the main and auxiliary functions,
the various flows of energy, material, and information were connected through them to yield a system. Next, the functions were decomposed into sub-functions approaching the level of basic functions described by [5.8].

Figure 5.2 exemplifies a portion of the main functions function structure. These main functions are part of the substrate flow, which was selected for the detailed analysis from all possible alternatives in the Xerox product. These functions are form-independent abstractions requiring further refinement to create the basic function representation in Figure 5.3, which identifies the sub-functions that are decomposed from the prior function structure.

![Diagram](image)

**Figure 5.2. Portion of a product use system-level function structure**

![Diagram](image)

**Figure 5.3. Decomposition of the function structure in Figure 5.2 into its basic functions**

The function “introduce substrate,” meant to represent the method by which substrate is set in motion from a static state, is decomposed into contact, separate, and advance substrate functions, which are further clarifications of the function of introduction. It is clear that these functions lack specificity, but given the desire to remain form-independent, there is a conceptual limit to function decomposition.
5.3 Monolithic function structure
Finally, the set of function structures for the different product uses were compared for similarity and redundancy and then merged into a single function structure that described the entire set of product uses. All possible subfunctions were included, representing an imaginary monolithic concept that completed all the functions for all the uses. It is conceivable that not all product uses can be assembled into a common function structure. Uses that have different functions from those contained in the aggregate structure, provided that they operate on the same system flows, could be connected to the structure through “OR” logic to include the additional functions. The process for creating a single function structure is schematically represented in Figure 5.4.

![Product Use FS's](image)

Figure 5.4. Schematic representation of product use function structure integration to create a monolithic function structure.

The use of web and sheet substrate by segments of the Xerox target market exemplifies the concept of merging use case function structures into the monolithic function structure. The bold function chain and accompanying substrate flow trace the path of the web substrate from its loading into the system to its selection for use in creating a document at the rightmost function in Figure 5.5. The requirements of other use cases (related to substrates) necessitate the inclusion of sheet substrate and multiple substrate options. Figures 5.6 and 5.7 reveal the successful integration of the single sheet and additional sheet flows and functions, respectively, into the monolith.
Figure 5.5. Section of the monolithic function structure, web substrate function chain highlighted.

Figure 5.6. Section of the monolithic function structure, sheet substrate function chain highlighted.
Figure 5.7. Section of the monolithic function structure, web and sheet substrate function chains.

It is evident that additional flows of either web or sheet substrate may be integrated similarly, up to the maximum number required from any of the use cases. With the monolith constructed in such a manner, the product variants can potentially be enabled by making the branches of the function chains into modules that terminate at a common bus.

The monolith is included as Appendix B. Functions are individually numbered and labeled.
6.0 Portfolio Architecture Modularity

In transforming this function structure into a viable product portfolio architecture, the transition from the function to the technology domain must be made, and often it is complicated by the combinatorial complexity of grouping these sub-functions into suitable candidates for components or subassemblies. Depending upon where subsystem boundaries are drawn, modules and components are implemented in different manners. The success of the product portfolio depends upon the proper, efficient partitioning of the system.

Modularity heuristics offer a method of determining which functions, based upon their connectivity to other functions and their relation to meeting customer needs or specifications, should be clustered together. The resulting clusters are irrespective of cost and existing modules within a company, but rather are useful for considering clean-sheet designs or the planning of future product portfolios, they can provide a framework for evaluating these other important aspects of modularity.

6.1 Heuristics and their use for design and modularity

Heuristics serve as a component of engineering design activities. While scientific principles rely upon a methodology to derive proof, heuristics are more empirically-minded, and derive credence from preponderance of evidence. A heuristic program [6.1] incorporating minimal initial information is desired since a detailed, deterministic analysis of all possible product configurations including cost, performance, and other aspects of the product family to select the optimal configuration would likely require more time than selecting any configuration and implementing it.

Recently, heuristics have been developed and applied to these function structures to identify opportunities for modularity. McAdams, Stone, and Wood [6.2] have illustrated that commonalities at the functional level exist among different product types, and that their heuristics exploit these similarities to group functions in the hopes that an isolated physical form can later be designed that provides the grouped set of functions. They propose a set of three heuristics to assist in the partitioning of functions in a function structure into suitable elements based on the paths of the flows of energy, matter, and information through the functions of the system. They applied these heuristics to single products with similar functions. The application of these
heuristics to product families and the addition of several heuristics to propose modules to allow product variety is proposed to extend the research.

The McAdams-Stone-Wood method identifies the functions that correspond to the most important customer needs. For a single product intended for a homogeneous set of customers, or for the redesign of an existing product, this seems appropriate. Product families, however, seek to integrate related products through a common platform and to satisfy differences through modularity schemes. Different products in a portfolio are intended to provide customers the differences in performance they seek. One way of doing this is with a platform architecture; the difficulty in selecting a platform is that it becomes complex in modules and interfaces, to be capable of supporting the demanded variety. It would be beneficial to partition an architecture so that it is capable of targeting points along vectors of product variety within the scope of the product portfolio. For example, this is the intent behind the architecture of some products that incorporate swappable modules. Common requirements would be embodied within the platform components, while features satisfying market segments would be relegated to modules that could be interchanged to achieve different product performance targets.

Using function structures to capture the function-logic view of a product portfolio, a process for identifying sensible modules from a functional and variety perspective is proposed. We then select among the candidate function clusters given technology constraints and market variety demands to produce a product portfolio architecture.

6.2 Function heuristics
Now that a function-logic representation of the overall product family exists in the form of a monolithic function structure, the functions must be grouped to define function clusters for which technology options can be envisioned.

By studying consumer products, a set of heuristics [6.2, 6.3] have been developed to describe observed trends in mechanical products between functions and their corresponding physical solutions. By simply examining the function structure arrangement of the functions and flows using these heuristics, one can identify possible function modules, which can then be used to develop actual modules, a physical sub-assembly that performs the associated set of functions.

Five function heuristics were applied to the monolithic function structure to reveal all sensible function clusters. The first four were first presented in [6.3] and [6.2]. The fifth heuristic is proposed for product portfolios represented in a monolithic function structure.
- Dominant flow
- Branching flow
- Transformation/conversion
- Causally-linked functions
- Similarity/repetition

The function clusters indicated in the following sections by numbers refer to the list of 96 identified modules in Appendix C.

6.2.1 Dominant flow

Modules that are evident from the dominant flow heuristic include the set of sub-functions through which a flow passes or is converted within the system boundaries. Examples of dominant system flows in the targeted product category are the toner, image, controls, energy, and substrate. Making the functions along these flows into modules is often impractical, since they interact with so many other flows through the related functions, but they can serve as logical means of organizing the overall development task for a large team. The dominant flows contained in the system function structure are the substrate, electrical energy, control signals to handle motion of the substrate through the system, and job information (a type of control signal containing the data related to a document that governs substrate type to select, output format, and other user-defined inputs).

6.2.2 Branching flow

Branching flows bifurcate dominant flows to create parallel function chains, and in doing so reveal possible modules from the branch point for each parallel chain. Figure 6.1 shows separate branches of substrate flows entering the system and converging at the “Sense Substrate” function. Each of the three branches is a candidate for eventual modularity through this heuristic.
Gray boxed functions indicate the three modules by the branching flow heuristic

Figure 6.1. Modularity according to the "branching flows" heuristic
This heuristic usually produces bus modularity, since the common flow that is divided converges at a single module. All of the previously referenced flows from the dominant flow heuristic (substrate, control information, electrical energy, job information) divide into branches. Current Xerox high-productivity products use a similar bus architecture method to the one depicted in figure 6.1 to offer flexibility in the number of kinds of substrate available at a given time for assembly into a document.

6.2.3 Transformation/conversion
Functions that convert or convert and transmit flows constitute possible modules according to the transformation/conversion heuristic. Any function that converts a flow from one form to another, schematically represented in Figure 6.2, is a candidate for delivery as a module. Functions such as “Transform electrical energy” are explicit candidates for modularity under this heuristic. This rule could be considered more liberally to include functions that relate to a change in state of one of the dominant flows. In this case, those functions related to the heating of the substrate, the transfer of toner onto the substrate, the fusing of toner to the substrate, and the conditioning of the substrate would all be individually-indicated modules.

![Transformation/Conversion](image)

Figure 6.2. Transformation/conversion schematic representation.

Transformation/conversion modules generally include only one or two functions, and are thus not very interesting to consider as portfolio modules, since the goal is to group as many functions together as possible to create the minimum number of modules necessary for adequate product variety.

6.2.4 Causally-Linked Functions
Functions or function groups that are causally-linked, or whose group existence depends solely on the presence of one of the functions, form the basis of modules.
An example of causal linkage can be seen in the pair of “invert substrate” functions, depicted schematically in Figure 6.3. In order to maintain a common orientation of all document pages, the second occurrence is necessitated by the first, according to this function structure. Note that other concepts may not require this second occurrence, such as a concept that had opposing print heads to simultaneously print on both sides of a sheet. These alternatives would have a different function structure.

![Figure 6.3. Example of the causality modularity heuristic for flipping a sheet of paper.](image)

Causality functions act along a single flow, but are not always sequential. Assuming that proximity between functions correlates with proximity between modules, the rearrangement of the order of functions can allow the causally-linked functions to form clusters and will result in modules which are readily isolated. Since in current copiers fusing the toner to the substrate releases moisture, functions related to removing the negative effects of desiccation of the substrate are linked to functions related to fusing through a causal relationship. As a hypothetical example, if these functions were consolidated into a single module, then the insertion of a new fusing technology that did not adversely affect the substrate (i.e. one that did not liberate water from the media) could be accomplished by replacing this single module within the product instead of removing separate modules or components and interfering with other system flows and existing components.

The causal linkage heuristic suggests that functions correcting process inefficiencies (removal of untransferred toner and stripped paper fibers from a present-day copier, for example) should be contained within the same modules as the functions that produce the inefficiencies (transfer toner to substrate). Future technology insertion to correct for the inadequacies of implemented technologies would leave no residual artifacts with this consolidation.
6.2.5 Similarity/Repetition

Function groups that share similar types of inputs, outputs, and functions that appear multiple times in the same function structure should be consolidated into a module that will then be used at multiple locations simultaneously within the product, as illustrated in Figure 6.4. Single products do not typically have repeated functions, since they perform single uses for customers. Since the use cases have been assembled into the monolithic function structure, redundant function chains may be present. These function chains may be engendered as modules, or consolidated by rerouting some of the system flows to eliminate the repeated functions.

![Figure 6.4. Schematic representation of the similarity/repetition modularity heuristic.](image)

From the example of the analysis of a document creation product family, repeated function blocks appear related to the “Synchronize and Register Substrate,” “Transfer Toner to Substrate,” and “Fuse Toner to Substrate.” For example, Figure 6.5 illustrates repeated function group that could be performed by multiple similar but separate instantiations of the same module. The sub-function groups occur several times in the monolithic function structure (Figure 6.6) due to black and white versus multiple color printing. Each of the repeated groups, one for each color, forms a set of sub-functions that is a candidate module by this heuristic. For the use case of a customer that would like to produce a black and white document, only a single function chain is necessary. If the customer requires multiple colors, a repeated function chain is needed.
Figure 6.5. Functions meeting the similarity/repetition modularity criterion.
Figure 6.6. Reoccurrence of the functions meeting the similarity/repetition criterion.
6.3 Function clusters
Appendix C lists the function clusters according to the heuristic used to identify them. The number codes following the module number and heuristic represent the individual functions from Appendix B that comprise the module. In all, 83 function modules were identified.

Modules present in current Xerox products that are indicated by the function clusters are the branches of substrate flow (substrate feeder modules) and transformation of electrical energy. Branches of electrical energy and control signal flow are also achieved through components separated from the rest of the substrate handling system. Causally-linked modules of the type described above are not present in current products.

6.4 Variety heuristics
The function heuristics partition the product portfolio function structure according to its functions and flows. The indicated modules suggest interfaces based on lessons learned from the analysis of single product architectures, but do not consider the impact of using these modules across a product portfolio.

The challenge of product architecture for a portfolio is providing different sets of features or target values to different customers. Since in some cases only some of the functions are affected by differing customer preferences, they can be consolidated into alternative modules. Provided we understand the relationship between the product characteristics and product functions, the market preference distributions can be mapped onto the monolithic product portfolio function structure. The carrier or influence functions, or the functions directly related to the customer needs, will be linked according to the distributions of target values of the various customer needs/product characteristics. When the customer need is not attributable to a distinct set of functions, an alternative function structure must be developed to support the product portfolio, or separate product families must be developed based on disparate function structures.

This group of heuristics examines the market requirements and indicates a preferred modularity for enabling a product portfolio capable of product variants that meet sub-market/niche requirements. The system impact of each requirement can be viewed independently as a first pass. Requirements are normally not orthogonal, so the architecture indications generated by these heuristics should be reviewed relative to all system requirements so the best candidates for system partitioning can be selected.
The underlying philosophy behind the variety modularization is to minimize the system impact of variety. This heuristic is loosely formulated as, “separate functions whose performance must vary across the intended market by consolidating them into modules.” Two ways of protecting platform elements from variety are isolation of variety and reducing variety through an alternative function structure.

6.4.1 Isolation of variety

One can group functions into clusters by separating functions affected by market variety requirements from those that are not affected. As depicted in Figure 6.7, a function that relates to variety can be isolated in a module so that the related feature can be altered, included, or excluded from any product. For example, the top-level functions “Select Substrate Source” and “Introduce Substrate” must manage the material flow of substrate, which contains significant variety in terms of the number enabled at any given time, a requirement that is estimated to vary across the market. Since these are the carrier/influence functions for this requirement, and other functions are not associated with the satisfaction of this requirement, these two functions (and their requisite sub-functions) constitute a candidate variety module. It is indicated in Figure 6.8.

Isolation of Variety
Group functions related to individual features

Figure 6.7. Schematic representation of isolation of variety heuristic.
Figure 6.8. Example of grouping functions related to a feature into a candidate variety module.
6.4.2 Function structure modification for variety reduction

Once all the product characteristic requirements have been mapped onto the function structure, the remaining functions can be grouped into the core platform module, since variation due to changing customer needs does not impact the remaining functions.

While the first variety heuristic relates to product variety mapping, the second concerns modification of the function structure to reduce the impact of variety. By eliminating variety adjacent to common functional elements or clusters of functions, an increase in the amount of product variant commonality is possible. If the sub-functions adjacent to a group of common sub-functions can be made the same across all product variants, then they can be incorporated into the common group. Typically, this means moving the variety to the edges of a function structure by redesigning the function structure of a few product variants by considering alternate functions that produce the same net effect.

For example, by converting an input substrate web to a sheet, the impact of the stock type requirement for handling web and sheet is reduced. The products no longer have to handle both webs and sheets, and now can be standardized to deal with only sheets. After the conversion at the “Cut Web” functions, the function chains are identical, since the web-handling requirement has disappeared. The modified function structure is shown in Figure 6.9.

The results of this phase are function structure variants and variety modules, consisting of carrier functions, for product characteristics with varying target values.

![Diagram of function structure modification](image)

This notion echoes the idea of delayed differentiation [6.4] in design for variety, but that is from the design-the-assembly-process point of view. Here we translate it to the product family architecture design point of view. Appendix C lists the variety modules, thirteen of which were identified. Modules for number of selectable substrates, stock type, number of colors, duplexing, caliper and weight, image quality, and variable data features were found. The related functions
could potentially be isolated into a module. In the case of the productivity and substrate size, virtually all functions are affected, and thus these modules are not helpful in defining product architecture. In fact, in order to provide increased productivity, Xerox bundles multiple print engines, the part of the product that puts marks on the substrate, into single products. The customer achieves a doubling of productivity by purchasing nearly two machines. It is postulated that there is a set of features or product characteristics that can be isolated into potential modules, and a set that cannot. The features that cannot easily be isolated, like productivity within the context of the substrate sub-system, may form the basis of separate product platforms at distinct fixed target values in the portfolio. Closely-related product families emerge separately by choosing a fixed architecture for this feature. Substrate size is another such feature that may be fixed at specific values for the portfolio, but adjustable about these values within the individual products.

The function structure modification heuristic offers two alternatives for reducing the system variety: reducing web and sheet to web only (by cutting the sheets free as a final step) or sheet (by cutting the web into sheets at the beginning), and keeping a constant document width but cutting smaller width documents as a last step. Both solutions pose other system problems, and are only mentioned to illustrate the heuristic.

6.5 Variety function clusters
Of the thirteen variety clusters identified, four (one for productivity, two for number of colors, and one for duplex capability) were found in the modules of current Xerox products. Those clusters that could not be isolated to a few functions indicate that perhaps the substrate handling system is not the appropriate place to address the specified variety. For example, since productivity impacts virtually all functions, the system productivity is not highly adjustable within the substrate system through modularity.

From the 81 functions of the monolithic function structure, 96 function clusters were identified. One factor in defining the architecture of the product family is to understand the complexity of the interactions among the modules. An algorithm to decide upon the proper modules to use in the product portfolio is explained in the next section.
7.0 Selection of Candidate Function Clusters for Modules
Given a function structure representation of a system partitioned into candidate function clusters, the task of the development group is now to determine which are the most appropriate to use in constructing their product portfolio. This also entails adopting form for the function clusters that comprise the intended modules.

An algorithm is necessary for the selection of the best modules for the product family. In order to do this, the module interactions must be resolved. Huang and Kusiak [7.1] present an algorithm for forming modules from identified components which relies upon the knowledge of pairwise interactions between components and the suitability of creating a module from the given pair. The goal of this algorithm is the minimization of undesired interactions and the maximization of desired interactions between components. If this method were modified to consider modules as clusters of functions (components), then the result would be a set of modules that maximize desired interactions and minimize undesired interactions. This is explored in section 7.1.

The next task is to decide if such modules can be developed that are sufficiently robust to support the variety of intended applications. When we associate requirements with the various flows of energy, material, and information, we determine the requirements at each function boundary. By extension, when we group the functions into candidate modules, we can specify their requirements. By selecting technologies that satisfy the basic functions contained in the modules, we can use the input and output requirements to drive the selection of design parameter values. We can also determine if a particular technology is capable of satisfying the requirements and estimate module cost for future comparisons.

7.1 Function cluster interactions
Function clusters of the monolithic product portfolio function structure relate to each other through three distinct interactions, schematically represented in Figure 7.1.
1. *Isolated clusters* share no common functions and are independent of one another. There is no barrier to creating form modules based on either one (or both) since they are functionally orthogonal.

2. *Nested clusters* share common functions, and the functions of one module are a subset of the functions of the other. Again, there is no direct barrier to creating two modules. An example of nested modules are a CPU and a motherboard. Each is an independent module, but the processor functions must work in concert with the functions contained on the mother board. One reason that the CPU is isolated is so that its performance characteristics may vary while the board remains the same.

3. *Intersecting clusters* share a common set of functions, but have additional functions that are not shared. In order to create these modules, it is necessary to divide one or both of the clusters, as illustrated in Figure 7.2. The pair may be split into three sub-clusters, namely $A'$, $B'$, and $A \cap B$, or one cluster may be preferred over another by deciding upon modules $A$ and $A'$, or $B$ and $B'$.
Particular attention must be paid to the intersecting variety modules. The requirements for each of these modules must vary relative to each other, otherwise they would be lumped into a single common module. Without selecting platform design parameters for all variants by deciding upon modules A and A', or B and B', one must decompose the modules into three partitions, namely A', B', and A∩B. This will allow retention of product variety flexibility.

At this stage, the product portfolio function structure has alternative groupings identified, each sensible from either a functional or variety requirement perspective. Deciding upon which of the modules to select is influenced by the interactions among modules and the variants that are supported. Module selection begins with reconciling overlap among the variety modules. These subdivisions increase the number of modules, but also the number of product variants possible. In order to limit variety, or to support specific variants at the exclusion of others, one could opt for one of the two alternative arrangements.

The critical problem with dividing modules is that the resultant sub-modules may deviate from the heuristic guidelines governing creation of modules. The rationale for creating modules in the first place, that of preserving the flow relationships, would be contradicted if a module interface was placed at an inopportune location. Flows would be separated at undesired points, leading to increasing interface complexity.

7.2 Extension of portfolio architecture decision tree
The decision tree presented in section 4.4 treats all product features as orthogonal, or as if they could be delivered individually irrespective of each other. When the features are mapped according to their supporting functions, this orthogonality is not necessarily preserved. In the cases of nested and intersecting function clusters, this is especially true. When the functions related to the feature are able to be isolated, then the feature can always be delivered through a module of the type specified in Figure 4.4. When the features are not orthogonal in function-space, they must be carefully considered relative to each other in order to determine how to deliver the desired target values across the product portfolio.

For features whose market distributions indicate integral portfolio architectures (uniform market preference with a very small standard deviation), no alteration in the strategy is necessary for function-coupled features. The portfolio is only required to perform at a fixed target value for the specified need, for which it can, in most cases, be designed.
For features where the portfolio recommendation is for a modular product family (segment variation is different from the population variation) and the related functions cannot be isolated, the product portfolio must incorporate the feature through distributed adjustment (adjustable integral) instead of a swappable module. The functions will not allow an entire module to be created that carries the feature, so the variety that the market desires in that feature must be derived through the distributed elements or sacrificed completely.

7.3 Module precedence and selection
It is clear from previous discussion that any modularity may be selected. However, some modules are more important than others. Modules that incorporate more than one or two functions are interesting, since they have the potential to minimize the number of necessary modules. Single-function modules are not interesting, since they may always be implemented. A single-function module will always be nested within other modules.

This algorithm begins with the assumption that good product architectures enable desired product variants from the market analysis when one ignores other selection criteria. By preferring the product variety modules, the product architecture supports the product variants through substitution, inclusion, or exclusion of these modules. Thereafter, modules may be selected that provide additional benefit unrelated to product variety, such as causally-linked function modules, repetition function modules, branching flows, and so on.

Given this variety modularity criteria, a module selection algorithm is proposed.

1. Select all possible variety modules that do not intersect with any other possible variety modules (i.e. nested or isolated variety modules).
2. Reject all possible modules that intersect with the variety modules in step 1.
3. Next, identify possible modules that nest or are isolated from the accepted variety modules and do not intersect with other possible modules. These are the *complementary modules* (ITALIC), which have no penalty for inclusion. They may be accepted without introducing interference effects or entanglements with the other modules.
4. The set of possible modules that remains does not conflict with the accepted modules, but they intersect each other. These possible modules must be examined to determine if partitioning some of them will relieve the intersections and thereby allow them to be accepted into the scheme.

70
These guidelines provide constraints on defining the modularity partitioning for a platform and its supported variants, but even within this framework trade-off decisions over equivalent partitions will arise (such as in Steps 1, 3 and 4). Relative importance of variety modules can serve as a means of module selection. For example, two possible variety modules, A and B, may intersect so both cannot be implemented unaltered. If it is more important to offer product variety along the feature related to module A than it is for B (A has a wider distribution of target values than B, for example), then module A may be implemented.

At the conclusion of this algorithm, the architecture that maximizes the benefit of product variety, and includes as many other modularity benefits as possible, is proposed. Alternatively, module value, a metric considering cost, variant enabling, and other factors may be used to prioritize the benefit of each module, then they can be selected according to this metric.

7.4 Xerox example
The 81 functions in the monolithic function structure and the 96 indicated function and variety modules were assigned numbers and arranged into a matrix, which is included as Appendix C. First, all interactions among interesting modules (those containing more than one function) were identified. Recall an interaction is either being nested or overlapping in the monolithic function structure for the platformed portfolio. In the copier family study, there were 35 modules that comprise the rows and columns of the square interactions matrix provided as Figure 7.3.
Figure 7.3. “Interesting” modules interaction matrix.

The letters in the matrix correspond to the following relationships:
I: The modules are identical (i.e., main diagonal)
A: The module in the row is nested within the module in the column
B: The module in the column is nested within the module in the row
N: The modules in the rows and columns intersect each other

The subset of variety modules is first considered. Figure 7.4 highlights these modules. Since no relative importance of variety modules via their related features and product characteristics was assumed, maximization of the number of product variety modules was sought. Figure 7.5 identifies the interactions among the variety modules.
Since our only preference is to have as many variety modules as possible, modules 89, 93, and 94 are excluded from this set. All other variety modules are acceptable since they do not intersect with other modules. No modules have been partitioned, but the rejected modules may be considered for partitioning if providing this product variety is critical. The modules represent functions related to one method of duplexing (opposed printing rather than physically inverting paper), image quality, and variable data. This is actually good in the case of duplexing and
variable data, since these items are to be part of the platform, based on the customer needs analysis of Chapter 4. We do not need separate modules, since customer preference is uniform.

The accepted modules carry the selectable substrates, stock type, number of colors, duplex, and caliper/weight. All of these variations in customer preference can be offered through modular solutions.

Next, the other “interesting” modules that negatively intersect with the selected modules are eliminated. Figure 7.6 shows that eleven module candidates are eliminated from the set. The modules are some of those formed according to the dominant flow heuristic, the causality heuristic, and the repetition heuristic.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| N | N | B | D | S | B | B | B | N | N | N | N | N | N | N | N | N | N | B | B |
| N | N | I | B | B | B | B | B | N | N | N | N | N | N | N | N | N | N | N | N |
| A | A | I | I | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A |
| A | A | I | I | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A |
| I | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N |
| N | N | B | N | N | N | I | N | N | N | N | N | N | N | N | N | N | N | N | N |
| N | N | B | N | I | B | I | B | I | B | I | B | I | B | I | B | I | B | I | B |
| N | N | B | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N |
| N | N | N | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B |
| N | N | N | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B |
| N | N | N | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B |
| N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N |
| N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N |
| A | A | B | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N |

**Figure 7.6.** Modules that intersect accepted variety modules are shaded.

Next, the modules that do not negatively intersect with the accepted modules or any of the remaining modules may be immediately added to the set, since there is no need to decompose them. Figure 7.7 reveals the modules that fall into this category.
Figure 7.7. Complementary modules in light gray have no negative intersections with either accepted modules (dark gray) or any of the remaining modules.

The intersections indicated are with modules that have already been eliminated from the set. For example, module 39 (double invert causality) does intersect with modules 51 and 52, but they were eliminated due to their intersections with variety modules 86 and 90.

Only a small set of “interesting” modules remain. They share the common characteristic that they do not conflict with the accepted modules, but they do conflict with each other. Figure 7.8 shows the interaction matrix for the remaining eight modules. If we maximize the “good” interactions (A’s, B’s, and no interaction) and minimize the “bad” interactions, we would accept 7 and 44, and reject 47.

Figure 7.8. Remaining modules interaction matrix. Light gray are accepted modules, dark gray are excluded.
Modules 8, 9, 10, 43, and 46 have no beneficial interactions with each other, but they do with the accepted modules. Closer examination of functions in modules 8, 9, and 10 (modules relating to control signal flow) reveals that if functions 14, 15, 16, and 20 were separated from each to form module 97, the resulting sub-modules formed 8', 9', and 10' (as well as 97) would be acceptable. Figure 7.9 shows the portion of the function and module matrix (Appendix C) containing these modules, and the progression of the decomposition.

<table>
<thead>
<tr>
<th>8</th>
<th>9</th>
<th>10</th>
<th>43 X</th>
<th>46 X</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>12</td>
<td>18</td>
<td>27</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>18</td>
<td>27</td>
<td>28</td>
</tr>
</tbody>
</table>

By partitioning functions 14, 15, 16, and 20, the remaining sub-modules 8', 9', and 10' may be accepted.

<table>
<thead>
<tr>
<th>97</th>
<th>8'</th>
<th>9'</th>
<th>10'</th>
<th>43 X</th>
<th>46 X</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>12</td>
<td>18</td>
<td>27</td>
<td>28</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>18</td>
<td>27</td>
<td>28</td>
<td>29</td>
</tr>
</tbody>
</table>

Figure 7.9. Resolution of module interactions through decomposition.

Modules 43 and 46 are rejected since they do not conform to any of the accepted modules. They are both modules according to the repetition heuristic.

Finally, Figures 7.10 and 7.11 list the modules that are selected and rejected according to the algorithm. Note that uninteresting modules (transform electricity, for example) may still be added. These results indicate that the product family can use modules to enable six of the ten product features. Figure 7.12 summarizes the general architecture implication of each product characteristic.
The initial portfolio architecture indication of Figure 4.4 is presented as the “CN Implication” column. The portfolio modularity indication with respect to product functions (as viewed in the monolithic function structure for the entire platformed family) is listed in the “Modularity Implication” column.

Productivity and stock type carrier functions are not isolated within the function structure, so even though the customer need analysis indicates a modular product family, they are mutually exclusive and thus must be divided into separate product families. The function relationships simply will not allow the definition of an easily swappable module to provide the necessary product variety.

Within each platform are the integral and adjustable features; namely, duplex, caliper, weight, size, image quality, and variable data. The carrier functions for these features are distributed throughout the system (except for duplex) and cannot be formed into modules.
Duplex can be assembled as a module even though the portfolio recommendation is for fixed integral.

Since the functions related to the number of selectable substrates and number of colors can be isolated, the recommendation of a modular product family is tenable through a variant module within each sub-platform.

Given this insight, it is clear that the product portfolio is best divided into two families: a high productivity family and a medium productivity family. Common elements are shared within these two families but not between them, and productivity is the dividing feature because it affects virtually all functions in the substrate handling subsystem. Figure 7.13 gives a depiction of the overall product portfolio according to how each feature is realized.

**Product Portfolio**

<table>
<thead>
<tr>
<th>High Productivity Product Family</th>
<th>Medium Productivity Product Family</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Product Family</td>
<td>Web Product Family</td>
</tr>
<tr>
<td>Swappable Modules</td>
<td>Swappable Modules</td>
</tr>
<tr>
<td>Colors</td>
<td>Colors</td>
</tr>
<tr>
<td>Selectable Substrates</td>
<td>Selectable Substrates</td>
</tr>
<tr>
<td>Fixed Integral</td>
<td>Fixed Integral</td>
</tr>
<tr>
<td>High Productivity</td>
<td>High Productivity</td>
</tr>
<tr>
<td>Variable Data</td>
<td>Variable Data</td>
</tr>
<tr>
<td>Duplex</td>
<td>Duplex</td>
</tr>
<tr>
<td>Web</td>
<td>Sheet</td>
</tr>
<tr>
<td>Adjustable Integral</td>
<td>Adjustable Integral</td>
</tr>
<tr>
<td>Image Quality</td>
<td>Image Quality</td>
</tr>
<tr>
<td>Caliper</td>
<td>Caliper</td>
</tr>
<tr>
<td>Weight</td>
<td>Weight</td>
</tr>
<tr>
<td>Size</td>
<td>Size</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sheet Product Family</th>
<th>Sheet Product Family</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swappable Modules</td>
<td>Swappable Modules</td>
</tr>
<tr>
<td>Colors</td>
<td>Colors</td>
</tr>
<tr>
<td>Selectable Substrates</td>
<td>Selectable Substrates</td>
</tr>
<tr>
<td>Fixed Integral</td>
<td>Fixed Integral</td>
</tr>
<tr>
<td>Medium Productivity</td>
<td>Medium Productivity</td>
</tr>
<tr>
<td>Variable Data</td>
<td>Variable Data</td>
</tr>
<tr>
<td>Duplex</td>
<td>Duplex</td>
</tr>
<tr>
<td>Web</td>
<td>Sheet</td>
</tr>
<tr>
<td>Adjustable Integral</td>
<td>Adjustable Integral</td>
</tr>
<tr>
<td>Image Quality</td>
<td>Image Quality</td>
</tr>
<tr>
<td>Caliper</td>
<td>Caliper</td>
</tr>
<tr>
<td>Weight</td>
<td>Weight</td>
</tr>
<tr>
<td>Size</td>
<td>Size</td>
</tr>
</tbody>
</table>

Figure 7.13. Product portfolio resulting from market and function analyses.

A template architecture for each product within the portfolio is provided as Figure 7.14. Connected elements are common within each platform, while the modules governing substrates and colors are interchangeable. Additional function-based assembly modularity exists within the platform according to the recommendations in Figure 7.10.
7.5 Technology selection

Once the module boundaries have been determined, suitable technologies are applied. Design catalogs, morphological analysis, and other structured methods may be used to determine acceptable forms to provide the given functions. Most often, development teams have solution concepts in mind or available, and can use them once the module boundaries have been established. Requirements can be transferred to the module boundaries, and technology elements can be modeled to determine if the element will support them. The framework for all other detail-design activities has been established, and division of the design task can now begin.
8.0 Conclusions

Successful portfolio architectures are defined based on many factors. This research has attempted to merge the market and functional views into a product representation, then determine a set of function clusters for modules that would provide the set of indicated variants.

It is beneficial to consider customers’ uses instead of simply the average value of their needs, since customers can require different levels of performance throughout the useful life cycle of the product they purchase.

Customer needs analysis reveals important market requirements for the features of the product portfolio, but the interactions among the functions that carry these features must be understood in order to arrive an appropriate portfolio architecture. When strong function coupling exists, more features must be integrated into the common product platform at fixed target values, which sacrifices product variety, or the functions must be incorporated through adjustable elements, which increases design complexity. Considering features independently of their related functionality may lead to concepts that cannot be implemented. Some product features do not map to specific areas of a function structure, and so are not easily integrated into a single module. Other characteristics will lend themselves readily to such a mapping and will form the basis of swappable modules designed to deliver differential product performance.

In the presence of function coupling, correlation analysis can help to reduce design complexity, provided market feature correlation corresponds with product function correlation. When groups of features have observable correlation for customers, but that trend is not present in the function view of the portfolio, then the correlation is not helpful for reducing development tasks and costs.

This method reveals the clustering of functions that can serve as a roadmap for future technology development. Modules that would be applicable in many products may be defined and mass produced to be part of a massively-customizable product technology strategy. These core modules may be arranged into a host of products, since they would embody the core features of the product family. Typically, this identification comes with significant understanding of the technology that comprises the products, so mature product families, which benefit from this kind of platform organization effort, are good candidates. Modular solutions can protect a design against uncertainty in market analysis regarding variation in customer preference.
The methodology offers a justification for the success of some modularity decisions. Modules that exist because of their isolation according to functional and variety heuristics belong within modules. Several examples within current Xerox products became obvious as the function and variety function clusters were identified on the monolithic function structure. Although these modules were determined for other reasons ostensibly, their manifestation reinforces the validity of the heuristics. The methodology provides opportunities for new modules and product partitioning that might not have been evident to the development team based upon their experiences and existing processes.

This activity is best suited for a new product venture that begins with a clean-sheet design. Products leveraging existing technology can also benefit from this technique, but dependency on existing modules could constrain the possible architectures. Careful consideration of the potential products served by common modules must be made to decrease the chance that an accepted module will limit the number of possible products.

Since alternative concepts can produce the same effects, function structures are not unique, and thus at each stage of the process several concepts were considered. For example, the product use of marking both sides of the substrate is currently accomplished in many ways (inverting the substrate, adding extra transfer positions oriented toward the top and bottom of the substrate, etc.) and should be captured on a parallel function structure for eventual comparison. In creating a consolidated product family function structure, it may not be possible to identify a unique function-logic description. Again, parallel options will perpetuate novel concepts through the process and might yield powerful hybrid concepts.

Since we normally think of the technology required to perform specific functions, it is difficult to constrain thinking to the form-independent domain. This was obvious from the discussions with Xerox developers to create function structures for the product uses. If the development team thinks only in a conceptual manner, they might struggle to create innovative concepts that are not dependent upon existing technology. If they neglect their own corporate technology context, they might miss promising opportunities for new research, or may ignore resources and system expertise that already exists. Thus the development team must balance these factors during the deliberations.
The application of the structured method proposed here was advantageous for several reasons. The function structure captures the architecture concept. The development team has an artifact of their deliberations and a vehicle for expressing concepts. They also have a common standard for describing concepts, and a means for smaller teams to develop subsystem diagrams and attempt to interface them with the other groups' work.

The identification of potential modules yields an exhaustive search for potential partitioning. While this plethora of possibilities may create additional complexity, it can produce unanticipated modules.

The method lends itself to automation. Once a function-logic diagram is created, an algorithm can find many of the function modules, and an expert system designed to query the user about causality and variety dependencies can facilitate the identification of the rest. Also, the selection algorithm to determine the set of modules for the architecture can be implemented via computer.

The portfolio roadmap, complete with target values, function clusters for modules, and a notion of the portfolio architecture, can now be used to select technology elements to model for cost and performance. It is likely that the limitations of available technology will cause a rethinking of the structure of the product portfolio just as the consideration of function relationships did for the customer needs market analysis. Ideally, the technology elements can be directly mapped to the function structure, and given the partitioning of the portfolio, the development tasks can be subdivided to teams or individuals who can use other techniques to bring the platform rapidly to market.
9.0 Bibliography


Appendix B: Monolithic Function Structure and Numbered Functions
Appendix C: Indicative Function Clusters from Function and Variety Heuristics

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
<td>K</td>
<td>L</td>
<td>M</td>
<td>N</td>
<td>O</td>
<td>P</td>
<td>Q</td>
<td>R</td>
<td>S</td>
<td>T</td>
<td>U</td>
<td>V</td>
<td>W</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>AA</td>
<td>BB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>