Proceedings of DETC '00:
ASME Design Engineering Technical Conferences
and Computers and Information in Engineering Conference
September 10-13, 2000 – Baltimore, Maryland

DETC2000/DTM-14565

MODULAR PRODUCT ARCHITECTURE

Jeffrey B. Dahmus
Graduate Research Assistant

Javier P. Gonzalez-Zugasti
Graduate Research Assistant

Kevin N. Otto
Associate Professor

Center for Innovation in Product Development
Massachusetts Institute of Technology
Cambridge, MA 02139

ABSTRACT

This paper presents an approach to architecting a family of products that share inter-changeable modules. Rather than a fixed product platform upon which derivative products are created through substitution of various add-on modules, the approach here permits the platform itself to be one of several possible sizes or types. Thus, the system is a collection of modules, each of which can be one of several types. We begin by developing function structures of each product in the portfolio, where each embodies a specific physical principle underlying the common technology. Different function structure systems can be used for each physical principle under consideration. These function structures are then compared to determine common and unique modules. Product modularity rules (i.e. dominant flow, branching, and conversion) are then applied to determine further possible modules. Application of any consistent set of modularity rules defines a feasible portfolio architecture. Each portfolio architecture is represented using a modularity matrix of functions versus products, with shared/unique function levels indicated in the matrix. Possible product modules are indicated with boxes while possible portfolio modules are indicated with shading. This method provides a systematic approach to generating possible portfolio architectures and serves as a communication aid for design team deliberations.

INTRODUCTION

Determining product architecture is one of the key activities of any industrial product development activity. Volkswagen claims to save $1.7 billion annually on development and production costs through effective product architecture (Bremner, 1999). Volkswagen is able to take advantage of platform and component commonality by sharing between its four major brands, namely VW, Audi, Skoda, and Seat. These different automobiles share car platforms, which in Volkswagen's case includes front axles, rear axles, front ends, rear ends, exhaust systems, brake systems, and numerous other elements (Bremner, 1999). However, Volkswagen also claims that all vehicles on this shared common platform can be effectively differentiated in the eyes of the customer. Interestingly, Ford Motor Company has similar shared platform ambitions within its new Generic Architecture Process program. However, Ford defines its platform, which will be shared between several car models, to consist of common welding lines, suspension systems, and drivetrains (Bremner, 1999). Ford has similar expectations of large monetary savings in development and production costs while maintaining the ability to effectively differentiate the platformed cars in price and performance. Suprisingly, Volkswagen and Ford's definitions as to what constitutes a platform are vastly different.

This example highlights that the product architecting process, despite being a key determining factor in both cost savings and in the ability to offer product variety, is not well understood. System engineering and architecting remains an activity relegated to heuristics. Such activities are often only completed by experienced systems engineers who have gained an understanding of the various objectives that must be considered when architecting a product line. In this paper, we will develop a systematic methodology to architecting a product portfolio.

System architecting involves clustering various components in a product such that the resulting modules are effective for the company. An ideal architecture is one that partitions the product into practical and useful modules. Some successfully designed modules can be easily updated on regular time cycles, some can be made in multiple levels to offer wide market variety, some can be easily removed as they wear, and some can be easily swapped to gain added functionality. These virtues of effective product modularity are multiplied when identical modules are used in various different products. For example, the VersaPak™ rechargeable battery system is used across dozens of Black & Decker® products.

As elsewhere, we define product modules as sub-systems within a product that are bundled as a unit, and which serve identifiable functions. The product module is the pair, both the subsystem and the functions. We define portfolio modules as product modules that are used in multiple products. Deciding over what makes effective modules, both for each product and for the portfolio as a whole, is the topic of this paper. Modularization decisions can be made after restricting the portfolio to a physical principle (such as DC battery powered, AC electrically powered, compressed air powered, etc.). At this point, much freedom remains in determining how a family should be constructed. Modularity decisions can also be made at the technology research and development phase, when decisions are reached about which physical principles should be explored. This paper focuses on the former, where modularity decisions are made after selecting a physical principle.

We find there are four main influences on a system engineer when determining product partitioning modules that will be used across a product family. These four general types of objectives must be considered when making up-front system architecting decisions. The first is traditional market variance - how variety is needed on each customer concern, as measured by the variance from customer to customer. The second is usage variance - how a product purchaser needs variety after the purchase is made. This variance, typically neglected in market science literature and research, is critical to understanding what product offerings are needed, be it multiple fixed product offerings, swappable modules on a standard interface, or an easily adjustable platform. The third influence is technology change - how fast the various modules change before a product design update is required. The last type of influence we call Design for X - how design, production, supply and lifecycle criteria factor into consideration when determining product partitioning.

RELATED WORK

The development of product families built on product platforms and shared modules has been the subject of much recent research. Meyer and Lehnerd (1997) have done extensive case studies on platforms, pointing out their advantages and challenges, and demonstrating their ability to save costs. Other researchers such as Sanderson and Uzumeri (1995) and Henderson and Clark (1990) have also shown that the use of platforms has given companies an edge on the number of products they can offer and on their profitability over their competitors. Other management research has shown different approaches on managing the planning and use of platforms (Wheelwright and Clark, 1992; Erens and Verhulst, 1996; Robertson and Ulrich, 1998; Pedersen, 1999; Pulkkinen et al., 1999).

In the product design literature, one can find several design and manufacturing strategies for offering variety that begin with commonality metrics (Martin and Ishii, 1997; Kota and Sethuraman, 1998). There are also several model-based approaches to designing different kinds of product platforms. Simpson et al. (1999) and Conner et al. (1999) use a Decision Support Problem formulation to design families of products based on scalable platforms. A similar approach is used by Ortega et al. (1999) to show tradeoffs among multiple life-cycle objectives for a product family. Krishnan et al. (1998) developed network models to design families of products that are measured along a single performance criterion. Siddique and Rosen (1999) use a graph-grammar approach to design commonality into a family of products. Finally, optimization approaches have been developed by Gonzalez-Zugasti et al. (1998) and Nelson et al. (1999) to design product platforms and families of variants. Another optimization approach is used by Fujita et al. (1999) for designing a family of products from catalogs of existing swappable modules. These optimization formulations require system equations relating performance to configuration variables, thereby yielding an architecture. We explore here upstream work to identify effective architectures that these aforementioned methods can then evaluate among.

Less work has been done to develop tools to help the system engineer partition systems into common modules. Rechtin and Maier (1997) have developed many architecting heuristics and checks for system engineers to consider when partitioning systems. We have found that these rules of thumb are all often true and all often conflict, making their use limited. Nonetheless, the ideas are sound and influential in practice. In this paper, we present a method for partitioning a set of products into shared and individual modules based upon functional modeling. Function structures developed by Pahl and Beitz (1996) are used to model the products in the family. Starting with these functional models, we then make use of single product architecting rules as first developed by Stone et al. (1998) and then further apply portfolio architecting rules as developed by Zamirowski and Otto (1999). The result is a set of possible product portfolio architectures. These fit into the categorization of portfolio architectures shown by Yu et al. (1999). The method shown here provides a means to select among the possible architectures.

APPROACH

Our approach to architecting systems, which follows on works of Stone et al. (1998) and Zamirowski and Otto (1999), is outlined in Figure 1. Each step will be extensively covered in the subsequent sections. We begin the process by determining what underlying technologies should be utilized, and by establishing the limits of the product family that must share common modules. Each of these products is then developed as relatively independent conceptual designs. If multiple forms are possible for a product application, then one has multiple concepts to consider in this process. The next step is to develop function structures for each of the product concepts. These function structures for each concept are then unioned into a large family function structure. The family function structure indicates the interrelationships of functions for all the products in the family. We next introduce the idea of the modularity matrix, which lists the functions in the family versus the products in the family. Key specifications for the functions, as used in each product, are entered into the matrix. Together, the matrix elements form the architecting decision space. For example, should identical targets be established across products in the family? Should shared modules be formed between different products in the family? Modules within products are indicated with boxes, while modules shared across products are indicated with shading. By grouping different entries in the matrix, different architectures are formed. The matrix thus provides a clear means to describe each alternative portfolio architecture. Architectures so generated can then be compared and contrasted using a selection approach such as Pugh concept selection (Pugh, 1991), more numerical forms such as decision analysis (Thurston, 1990), or numerical metrics such as profit to the firm.

We next present each of these steps in greater detail, in the context of the Black & Decker® VersaPak™ product portfolio. We review function structure modeling and the rules for product and portfolio architecting. We then introduce the modularity matrix in greater detail and present an example of portfolio architecting using this tool. As an example, we will develop an alternative architecture for the Black & Decker® VersaPak™ line.

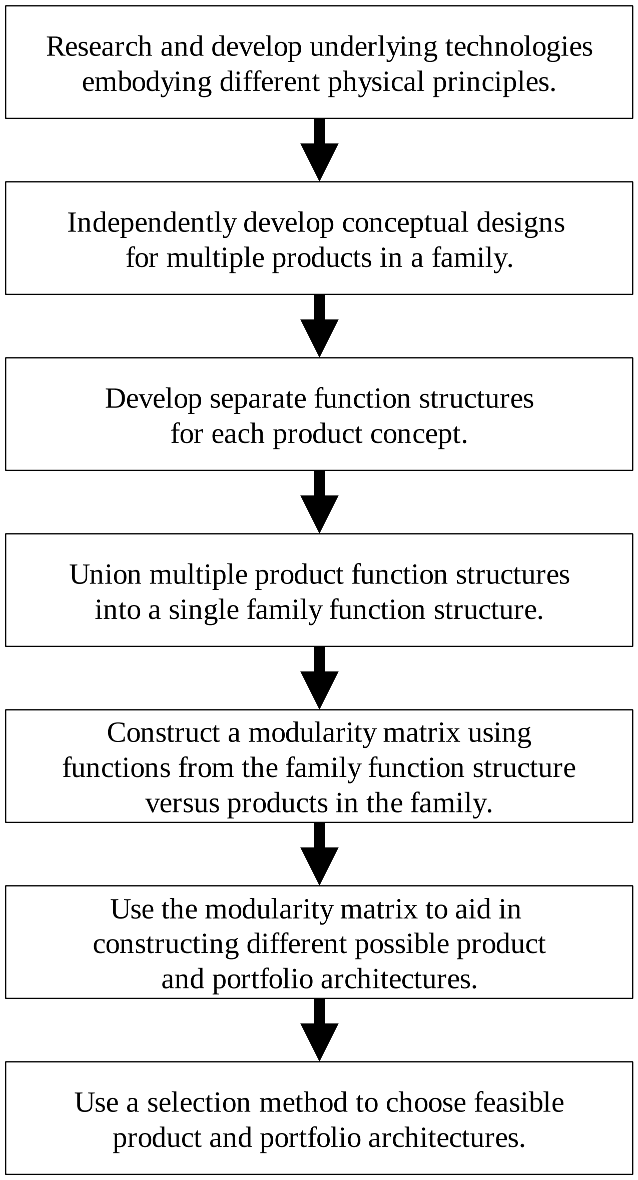

Research and develop underlying technologies embodying different physical principles.
Independently develop conceptual designs for multiple products in a family.
Develop separate function structures for each product concept.
Union multiple product function structures into a single family function structure.
Construct a modularity matrix using functions from the family function structure versus products in the family.
Use the modularity matrix to aid in constructing different possible product and portfolio architectures.
Use a selection method to choose feasible product and portfolio architectures.

Figure 1: Overview of the Portfolio Architecting Process.

BACKGROUND

Functional decomposition of products can be completed through various methods. FAST (Value Analysis Incorporated, 1993), Hatley and Pirbhai (1987), and other function-logic diagramming methods all attempt to accurately describe what a product is to do by mapping a system of functions for the product. The end result of these function-logic diagrams is a clearer understanding of the set of related functions necessary to allow the product to fulfill its overall intended function. Ideally, such a diagram is form-independent and identifies discrete, indivisible functions.

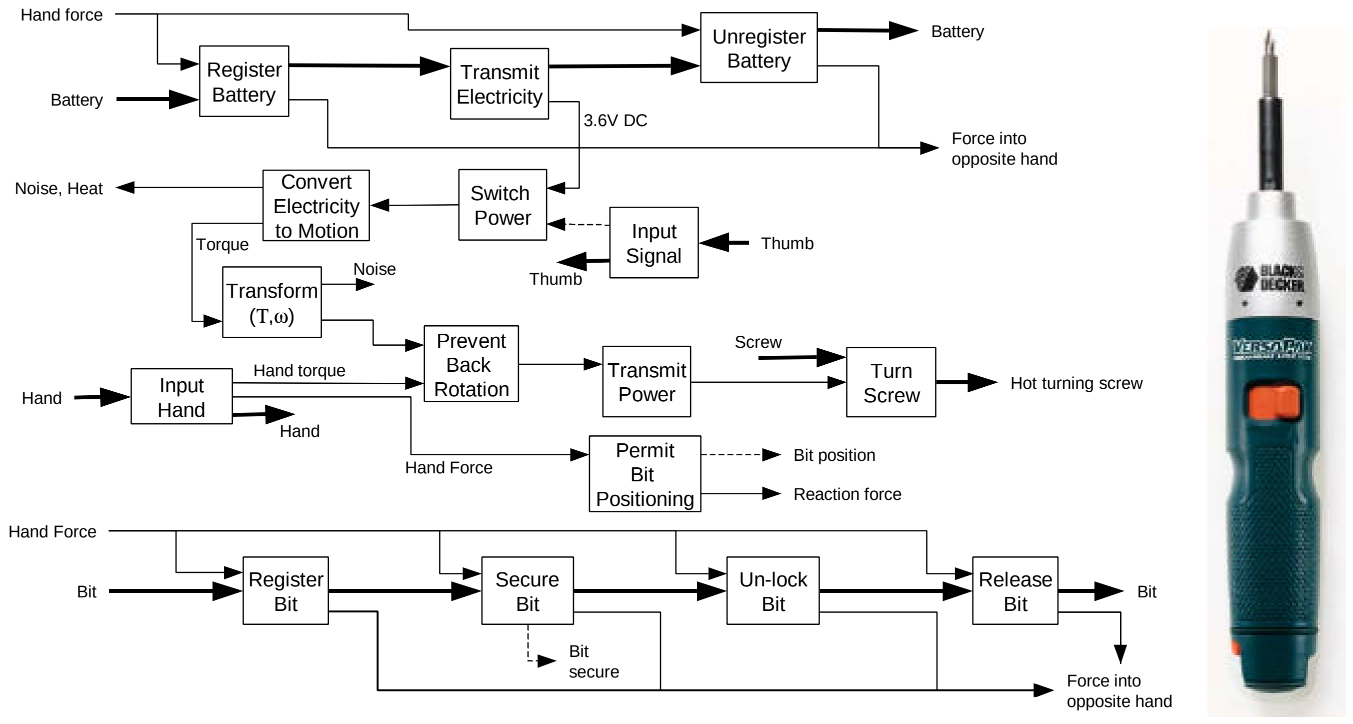


Figure 2: Function Structure for a VersaPak™ Cordless Screwdriver

For electromechanical systems, a technique to create diagrams known as function structures is widely employed (Pahl and Beitz, 1996). A function structure is a set of sub-functions interconnected by flows. Identifying these flows proves effective for helping to partition products into modules. For example, sub-functions with large sets of inter-connecting flows are not good candidates for separation into individual modules. The flows connecting sub-functions are identified as information, material, or energy flows. An example of a function structure for a VersaPak™ cordless screwdriver is shown in Figure 2. Note that only product functions are shown; human and other systems used as a part of inserting or removing screws are not shown. Only the things the device actually interfaces with are part of the function structure. Things outside this system are shown as flows into and out of the structure.

As compared with general function structure use as in Pahl and Beitz (1984), we restrict the function structure to only product functions (and not human functions, etc. required to turn screws). We also generally consider a particular phase of product development. Function structures can be used in very preliminary research and development efforts to conceive and develop new technologies that are not currently physically embodied in the current product lines or in any competing product lines. The ability of function models to be form independent and thereby allow a team to conceive alternative forms, is often used as a reason for functional modeling. We will take a different view on this statement, though, that requires explanation.

As is the case in most modern product development efforts at major corporations, we split concept generation into two phases, a phase involving research and development of technology, and a phase involving technology deployment into product lines. For example, with the Black & Decker® VersaPak™ line, one could also generate new technology concepts that use a different physical principle, such as compressed air powered devices. Functional modeling can be used to describe requirements at a level of detail generic to both physical principles. We are not concerned with this problem here.

Rather, we focus upon the deployment of technology most effectively into product lines: how to make modules such that they should or should not be shared across products. This is a different problem for which functional modeling, and function structures in particular, is ideally suited. A technology concept has been selected and developed to the point of feasibility, and now the question is over how to deploy this into several products.

The function structure in Figure 2 shows various distinct product sub-functions, each in its own box. Arrows between the boxes show the various flows. The system boundaries, where the user interacts with the product, exist where these flows enter and exit the system.

To develop a function structure for any given product, Otto and Wood (1998) present an approach based upon tracing flows. For every customer need, a flow is identified. This flow is then traced through the product, as it would flow during use, as a sequence of sub-functions that change the flow. The point of view of the product is always considered, hence hands, batteries, and tooling are passed through the product, not vice versa. These independent chains are then merged into a complete function structure network that is minimally comprehensive of the customer needs.

Function structures are so generated for each product concept. Later, we will consider how to cluster sub-functions into modules for any one of the products. These modules form the modular architecture of an individual product.

Once the function structures for each product in the family have been developed, they must be unioned into the family function structure. The union of the function structures yields a single diagram that has every function of every product on it, complete with their flow interactions. For a VersaPak™ family of products consisting of a cordless screwdriver, multipurpose saw, ScumBuster™ scrubber, 2-speed drill, and Wizard™ rotary tool, the family function structure is shown in Figure 3.

Note that different information is attained if one considers the intersection of the function structure diagrams. For example, in Figure 3, the intersection is shown as the un-shaded functions. These are the candidate functions to possibly modularize into a common platform. On the other hand, this is incomplete as not all functions in the family are represented. Often, a function is performed by all of the variants, but with drastically different flows. For example, the "Transmit Power" function conveys rotational or linear motion, depending on which variants are considered. Therefore, this function could not be platformed across the entire portfolio. However, the same function might be platformed across a subset of the products, such as the drill, ScumBuster™, and Wizard™. Thus, a more sophisticated

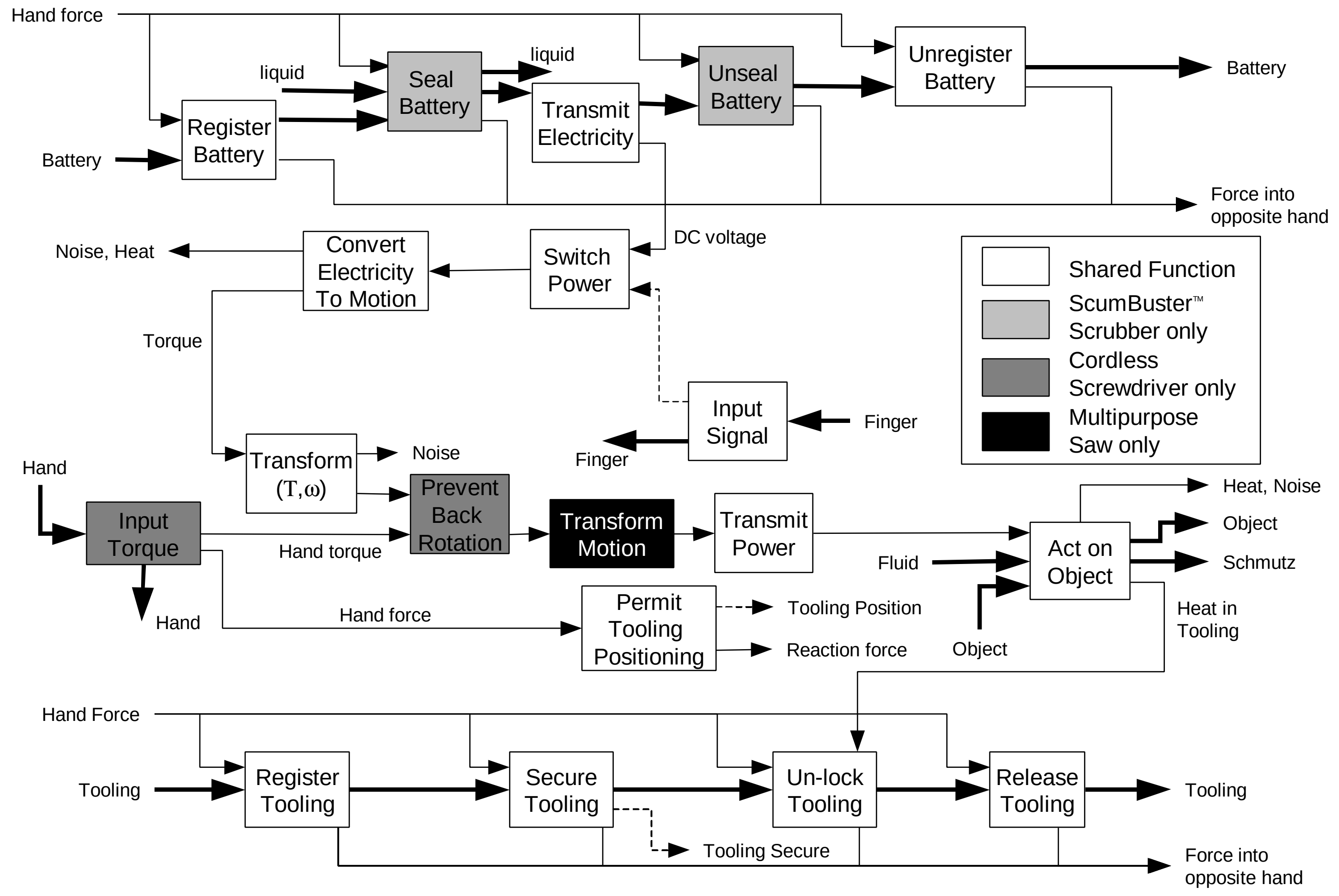


Figure 3: Family function structure for a VersaPak™ portfolio of products.

view of platforming must be employed.

MODULARITY RULES

When considering a single product, Stone et al. (1998) identified a set of three heuristics that can be used to identify product modules on a function structure. The heuristic methods applied to modularize product function structures are divided into three types: dominant flow, branching flows, and conversion-transmission.

The dominant flow heuristic examines flows through a function structure, following flows until they either exit from the system or are transformed into another flow. The sub-functions through which a flow can be traced, define a module. More succinctly, a set of sub-functions through which a flow passes, from initial entry or formation of the flow in the system, through final exit or conversion of the flow within the system, define a module.

The branching flow heuristic examines flows that branch into or converge from parallel function chains. Each branch of a flow can become a module. Each of these modules interface with the product through the point at which the flow branches or converges.

The conversion-transmission module examines flows which are converted from one type of flow to another. A conversion-transmission module converts an energy or material into another form, then transmits that new form of energy or material. In many instances, this conversion-transmission module is already housed as a module, as in the case of an electric motor.

When considering the entire portfolio, an additional set of rules can be applied to aid in module identification (Zamirowski and Otto, 1999). The heuristic methods applied to modularize portfolio function structures are divided into two types: shared functions and unique functions.

Shared functions can be used as a means to define portfolio modules. Functional groups that share similar flows and functions, and that appear multiple times in a comprehensive portfolio function structure, should be grouped into a single module. This module can then be reused throughout the portfolio of products.

Variant functions are those functions that are unique to a single product or subset of products. Such functions should be grouped into a module. Isolating variety in this way echoes the idea of delayed differentiation in design for variety (Martin and Ishii, 1997).

MODULARITY MATRIX

The family function structure is an effective tool to visualize the interactions of flows and candidate modular partitions. For example, partitioning lines that cross many flows will define a module that may be difficult to design since it will require much communication with the groups developing the adjoining modules. On the other hand, it is difficult to simultaneously visualize the partitions for multiple products in a family. A further difficulty occurs when labeling different sized modules. That is, while some functions are

Function	Cordless Screwdriver	Multipurpose Saw	ScumBuster™ Scrubber	2-Speed Drill	Wizard™ Rotary Tool
Register Battery	1	2	1	2	1
Unregister Battery	1	2	1	2	1
Transmit Electricity	1	2	1	2	1
Seal Battery	-	-	yes	-	-
Unseal Battery	-	-	yes	-	-
Input Signal	thumb	finger	finger	finger	thumb
Switch Power	forward/reverse/off	on/off/lock	on/off	forward/reverse/off/lock	low speed/high speed/off
Convert Electricity to Motion	motor A	motor B	motor C	motor D	motor E
Transform (T, ω)	transmission A	transmission B	transmission C	transmission D	-
Transmit Power	rotating shaft	translational blade	rotating shaft	rotating shaft	rotating shaft
Input Hand Torque	yes	-	-	-	-
Prevent Back Rotation	yes	-	-	-	-
Transform Motion	-	yes	-	-	-
Register Tooling	hexagonal hole	blade carriage	triangular hole	three-prong chuck	collet
Secure Tooling	retaining clip	set screw	snap fit	chuck housing	collet nut
Permit Tool Positioning	product shape	handle	handle	trigger handle	trigger handle
Act on Object	rotate	cut	scrub	drill/rotate	grind

Figure 4: Modularity matrix for a VersaPak™ portfolio of products.

grouped into a module, there will be several sizes offered of that module to create multiple product variants. Thus, a single platform concept, shared across all product variants, is not sufficient to think about platforming.

To deal with this complexity, we define here the modularity matrix, which aids in the application of the modularity rules, both for products and for product portfolios. A modularity matrix lists the possible functions from a family function structure as rows in the matrix, then lists the possible products from that family as columns. Each matrix element contains a value that represents the function specification level required. Ideally, a single value is used, though some functions are sufficiently complex that multiple specifications may be required. A modularity matrix for a VersaPak™ family of products is shown in Figure 4.

The specification values entered in the matrix represent targets for the functions of each product. These various values form the architecting space that will define possible product and portfolio architectures. That is, a design team must select specification values for each function in each product. The extent to which a product's set of specifications is compatible defines how well the individual product will work. The extent to which a function has the same targets established across products defines how well functions can be satisfied through shared modules. Thus, the architecting process for each product and for the product family is laid clear. Completing such activities is critical to the development process.

Laying out this information in a matrix allows commonalties to be easily identified. These commonalties can in turn lead to possible modules. First, we can form groupings of functions column-wise, which incorporate multiple functions all within one product. This highlights possible product modules. These modules can be selected on the family function structure using the rules of dominant flow, branching flow, and conversion-transmission.

Second, we can form groupings of functions row-wise, which incorporate the same functions into multiple products as a single module. This highlights possible portfolio modules that can be shared among multiple products. These modules can be selected on the family function structure using the rules of common and unique modules.

PORTFOLIO ARCHITECTING

The method described above was used to analyze modules for the Black & Decker® VersaPak™ portfolio of products. Once separate function structures were developed for each individual product, a family function structure, as shown in Figure 3, was created. This family function structure takes into account all functions that the portfolio products must satisfy. Some functions clearly overlap, while others are unique to individual products.

Function	Cordless Screwdriver	Multipurpose Saw	ScumBuster™ Scrubber	2-Speed Drill	Wizard™ Rotary Tool
Register Battery	1	2	1	2	1
Unregister Battery	1	2	1	2	1
Transmit Electricity	1	2	1	2	1
Seal Battery	-	-	yes	-	-
Unseal Battery	-	-	yes	-	-
Input Signal	thumb	finger	finger	finger	thumb
Switch Power	forward/reverse/off	on/off/lock	on/off	forward/reverse/off/lock	low speed/high speed/off
Convert Electricity to Motion	motor A	motor B	motor C	motor D	motor E
Transform (T, ω)	transmission A	transmission B	transmission C	transmission D	-
Transmit Power	rotating shaft	translational blade	rotating shaft	rotating shaft	rotating shaft
Input Hand Torque	yes	-	-	-	-
Prevent Back Rotation	yes	-	-	-	-
Transform Motion	-	yes	-	-	-
Register Tooling	hexagonal hole	blade carriage	triangular hole	three-prong chuck	collet
Secure Tooling	retaining clip	set screw	snap fit	chuck housing	collet nut
Permit Tool Positioning	product shape	handle	handle	trigger handle	trigger handle
Act on Object	rotate	cut	scrub	drill/rotate	grind

Figure 5: Modularity matrix showing current product modules for a VersaPak™ portfolio of products.

ESTABLISHING THE MODULARITY MATRIX

Once the family function structure for the portfolio of products is created, various architectures can be abstracted utilizing the modularity rules. First, grouping functions column-wise identifies possible product modules. We do this with boxes, as shown in Figure 5. This modularity matrix shows the current product modules as designed by Black & Decker®.

Next, we identify the portfolio architecture by identifying shared functions across the product line. By examining similarities across columns for a single function in the modularity matrix, possible portfolio architectures can be explored. When different products share specification levels for a given function or module, module sharing across the family is possible. If specification levels are not shared between multiple products, shared modules may not be feasible.

For the VersaPak™ portfolio, one can see that there are currently few shared modules, as highlighted in Figure 6. Basically, only the battery system functions are shared across products in the family. As an alternative, we might try and define a single battery sufficient for all the variant products, thereby increasing commonality. We might also try to use only two sizes of motors, rather than having each product have a unique motor.

Notice that with a modularity matrix, one can define shared modules for a subset of the entire portfolio. For example, in the VersaPak™ portfolio, the battery system functions are shared by the cordless screwdriver, ScumBuster™ scrubber, and Wizard™ rotary tool as a one-battery module. While the multipurpose saw and 2-speed drill share the same battery system functions, these products utilize a two-battery module. Sharing across some but not all of the products in a portfolio can be easily represented using the modularity matrix, as shown below.

SIMULTANEOUS ARCHITECTING

Using the modularity matrix, alternate product and portfolio architectures can be described. We can make new choices for target specifications such that more groupings are possible. If we choose the same target values across all or some products for a given function, then that product module can be shared across the portfolio.

For example, one could develop portfolio architecture for the VersaPak™ line that maximally shares modules. One such portfolio architecture concept is shown in Figure 7. Here, only two motor sizes are used across the portfolio of products, the smaller powered products all use one size of motor, and the larger powered products all use another common motor. The ScumBuster™ scrubber, 2-speed drill, and Wizard™ rotary tool share a common means of registering and securing tooling. The other two products of the family, namely the cordless screwdriver and multipurpose saw, do not share the same methods for registering and securing tooling. However, they do modularize these functions in a way similar to the modularization shown by the other family members.

Note however, that determining a final "correct"

Function	Cordless Screwdriver	Multipurpose Saw	ScumBuster™ Scrubber	2-Speed Drill	Wizard™ Rotary Tool
Register Battery	1	2	1	2	1
Unregister Battery	1	2	1	2	1
Transmit Electricity	1	2	1	2	1
Seal Battery	-	-	yes	-	-
Unseal Battery	-	-	yes	-	-
Input Signal	thumb	finger	finger	finger	thumb
Switch Power	forward/reverse/off	on/off/lock	on/off	forward/reverse/off/lock	low speed/high speed/off
Convert Electricity to Motion	motor A	motor B	motor C	motor D	motor E
Transform (T, ω)	transmission A	transmission B	transmission C	transmission D	-
Transmit Power	rotating shaft	translational blade	rotating shaft	rotating shaft	rotating shaft
Input Hand Torque	yes	-	-	-	-
Prevent Back Rotation	yes	-	-	-	-
Transform Motion	-	yes	-	-	-
Register Tooling	hexagonal hole	blade carriage	triangular hole	three-prong chuck	collet
Secure Tooling	retaining clip	set screw	snap fit	chuck housing	collet nut
Permit Tool Positioning	product shape	handle	handle	trigger handle	trigger handle
Act on Object	rotate	cut	scrub	drill/rotate	grind

Figure 6: Modularity matrix showing current portfolio modules for a VersaPak™ portfolio of products.

modularity matrix is not trivial. One must consider both portfolio sharing concerns and individual product concerns. Generally, these two can conflict. The portfolio considerations leads one to specify average components that work across multiple products. Product considerations leads one to specify individually tailored components that will only work on one specific product.

This ability of the modularity matrix to highlight both product and portfolio concerns is well shown by the ScumBuster™ scrubber. The ScumBuster™ can share the register battery, unregister battery, and transmit electricity functions with the cordless screwdriver and Wizard™ rotary tool. If one were to look solely at portfolio architecture, sharing these ScumBuster™ modules across the portfolio would be an obvious choice. However, from the modularity matrix, it is also apparent that the ScumBuster™ product module could include these functions along with seal battery and unseal battery. In this case, a decision must be made to follow either product or portfolio modularity for the ScumBuster™. Knowing how an individual product fits into a family of products on a modular level is critical to determining successful product and portfolio architectures.

ARCHITECTURE EVALUATION

The work above all results in the generation of several candidate modularizations, each expressed individually by a modularity matrix. It is a concept generation exercise. Next, a single modularity matrix must be selected. The most basic approach is to use a Pugh concept selection approach, as discussed in Gonzalez-Zugasti and Otto (2000). Even when simplified to a Pugh style selection, the evaluation process is difficult, as several product concepts must be compared on multiple criteria, rather than just one product concept. The result of a Pugh process is a subset of the platform alternatives, each as represented with a modularity matrix. These selected platforms should be developed further. For example, more complex performance models might be developed of these architectures, and numerical optimization completed to better estimate performance (Gonzalez-Zugasti et al., 1999, Fujita et al., 1999, Simpson et al., 1999). In real world applications, business cases of each platforming scenario would also have to then be developed (Gonzalez-Zugasti et al., 1999).

CONCLUSIONS

In this paper we have shown an approach towards architecting a portfolio of products to take advantage of possible commonality through the reuse of modules across the family of products. The method is based on functional modeling of the products using function structures. We begin by creating the function structure for each desired individual product. These function structures should embody a specific physical principle. We then combine these individual structures to construct a family function structure. This family function structure is an effective tool to identify common and unique functional modules in the product family. We then arrange the identified functional modules into a modularity matrix, which displays the desired functions and their levels

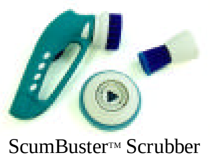
Function	Cordless Screwdriver	Multipurpose Saw	 ScumBuster™ Scrubber	2-Speed Drill	Wizard™ Rotary Tool
Register Battery	1	2	1	2	1
Unregister Battery	1	2	1	2	1
Transmit Electricity	1	2	1	2	1
Seal Battery	-	-	yes	-	-
Unseal Battery	-	-	yes	-	-
Input Signal	thumb	finger	finger	finger	thumb
Switch Power	forward/reverse/off	on/off/lock	on/off	forward/reverse/off/lock	low speed/high speed/off
Convert Electricity to Motion	motor A	motor B	motor A	motor B	motor A
Transform (T, ω)	transmission A	transmission B	transmission A	transmission D	-
Transmit Power	rotating shaft	translational blade	rotating shaft	rotating shaft	rotating shaft
Input Hand Torque	yes	-	-	-	-
Prevent Back Rotation	yes	-	-	-	-
Transform Motion	-	yes	-	-	-
Register Tooling	hexagonal hole	blade carriage	three-prong chuck	three-prong chuck	three-prong chuck
Secure Tooling	retaining clip	set screw	chuck housing	chuck housing	chuck housing
Permit Tool Positioning	product shape	handle	handle	trigger handle	trigger handle
Act on Object	rotate	cut	scrub	drill/rotate	grind

Figure 7: Modularity matrix showing possible product and portfolio modules for a VersaPak™ portfolio of products.

for each product in the family. The possible modules identified through product and portfolio modularity rules can then be visually displayed within the modularity matrix. The matrix allows a design team to consider different partitioning schemes for each product (product architecture) and for the portfolio as a whole (portfolio architecture). Design teams can then use the modularity matrix as a tool to select an architecture that incorporates an appropriate amount of commonality. An example showing the application of this method to the design of a family of battery-powered hand tools has been presented. The results show possible architectures for the individual tools and for the tool family as a whole.

ACKNOWLEDGEMENTS

The research reported in this document was made possible in part by the MIT Center for Innovation in Product Development under NSF Cooperative Agreement Number EEC-9529140. Any opinions, findings, or recommendations are those of the authors and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] Bremner, R. (1999). "Cutting Edge Platforms." Financial Times Automotive World. September 1999: 30-38

[2] Conner, C., De Kroon, J. and Mistree, F. (1999). "A Product Variety Tradeoff Evaluation Method for a Family of Cordless Drill Transmissions." *1999 ASME Design Engineering Technical Conferences*, Las Vegas, Nevada. DAC-8625.

[3] Erens, F. J. and Verhulst, K. (1996). "Architectures for product families." *WDK workshop on Product Structuring*, Delft University of Technology

[4] Fujita, K., Sakaguchi, H. and Akagi, S. (1999). "Product Variety Deployment and its Optimization under Modular Architecture and Module Commonalization." *1999 ASME Design Engineering Technical Conferences*, Las Vegas, Nevada. DFM-8923.

[5] Gonzalez-Zugasti, J. P. and Otto, K. N. (2000). "Platform-Based Spacecraft Design: A Formulation and Implementation Procedure." *2000 IEEE Aerospace Conference*, Big Sky, Montana

[6] Gonzalez-Zugasti, J. P., Otto, K. N. and Baker, J. D. (1998). "A Method for Architecting Product Platforms with an Application to the Design of Interplanetary Spacecraft." *1998 ASME Design Engineering Technical Conferences*, Atlanta, Georgia. DAC-5608.

[7] Gonzalez-Zugasti, J. P., Otto, K. N. and Baker, J. D. (1999). "Assessing Value for Product Family Design and Selection." *1999 ASME Design Engineering Technical Conferences*, Las Vegas, Nevada. DAC-8613.

[8] Hatley, D. and Pirbhai, I. (1987). Strategies for Real-Time System Specification, Dorset House Publishing Company.

[9] Henderson, R. M. and Clark, K. B. (1990). " Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms." *Administrative Science Quarterly* 35.

[10] Kota, S. and Sethuraman, K. (1998). "Managing Variety in Product Families through Design for Commonality." *1998 ASME Design Engineering Technical Conferences*, Atlanta, Georgia. DTM-5651.

[11] Krishnan, V., Singh, R. and Tirupati, D. (1998). "A Model-Based Approach for Planning and Developing A Family of Technology-Based Products.". Austin, The University of Texas at Austin Management Department. April 1998. Working Paper

[12] Martin, M. and Ishii, K. (1997). "Design for Variety: Development of Complexity Indices and Design Charts." *1997 ASME Design Engineering Technical Conferences*, Sacramento, California. DFM-4359.

[13] Meyer, M. H. and Lehnerd, A. P. (1997). The Power of Product Platforms. New York, The Free Press.

[14] Nelson, S., Parkinson, M. and Papalambros, P. (1999). "Multicriteria Optimization in Product Platform Design." *1999 ASME Design Engineering Technical Conferences*, Las Vegas, Nevada. DAC-8676.

[15] Ortega, R., Kalyan-Seshu, U. and Bras, B. (1999). "A Decision Support Model for the Life-Cycle Design of a Family of Oil Filters." *1999 ASME Design Engineering Technical Conferences*, Las Vegas, Nevada. DAC-8612.

[16] Otto, K. N. and Wood, K. (1998). "Product Evolution: A Reverse Engineering and Redesign Methodology." *Research in Engineering Design* Volume 10(Number 4): pp. 226.

[17] Pahl, G. and Beitz, W. (1996). Engineering design : a systematic approach. London ; New York, Springer.

[18] Pedersen, P. E. (1999). "Organisational Impacts of Platform Based Product Development." *International Conference on Enginnering Design*, Munich

[19] Pugh, S. (1991). Total design : Integrated methods for successful product engineering. Wokingham, England ; Reading, Mass., Addison-Wesley Pub. Co.

[20] Pulkkinen, A., Lehtonen, T. and Riitahuhta, A. (1999). "Design for Configuration - Methodology for Product Family Development." *International Conference on Engineering Design*, Munich

[21] Rechtin, E. and Maier, M. (1997). The art of systems architecting. Boca Raton, CRC Press.

[22] Robertson, D. and Ulrich, K. (1998). "Planning for product platforms." *Sloan Management Review* 39(4): 19-31.

[23] Sanderson, S. and Uzumeri, M. (1995). "Managing Product Families: The Case of the Sony Walkman." *Research Policy* 24.

[24] Siddique, Z. and Rosen, D. (1999). "Product Platform Design: A Graph Grammar Approach." *1999 ASME Design Engineering Technical Conferences*, Las Vegas, Nevada. DTM-8762.

[25] Simpson, T., Maier, J. and Mistree, F. (1999). "A Product Platform Concept Exploration Method for Product Family Design." *1999 ASME Design Engineering Technical Conferences*, Las Vegas, Nevada. DTM-8761.

[26] Stone, R., Wood, K. and Crawford, R. (1998). "A Heuristic Method to Identify Modules from a Functional Description of a Product." *1998 ASME Design Engineering Technical Conferences*, Atlanta, Georgia. DTM-5642.

[27] Thurston, D. (1990). “A Formal Method for Subjective Design Evaluation with Multiple Attributes.” *Research in Engineering Design* 3(2): 105-122.

[28] Value Analysis Incorporated, Ed. (1993). Value Analysis, Value Engineering, and Value Management. Clifton Park, NY.

[29] Wheelwright, S. C. and Clark, K. B. (1992). “Creating Project Plans to Focus Product Development.” *Harvard Business Review*(March-April 1992).

[30] Yu, J., Gonzalez-Zugasti, J. P. and Otto, K. N. (1999). “Product Architecture Definition Based Upon Customer Demands.” *Journal of Mechanical Design* 121(3): 329.

[31] Zamirowski, E. and Otto, K. (1999). "Identifying Product Portfolio Architecture Modularity Using Function and Variety Heuristics." *1999 ASME Design Engineering Technical Conferences*, Las Vegas, Nevada. DTM-8760.