

Modular product architecture

Jeffrey B. Dahmus, Javier P. Gonzalez-Zugasti and Kevin N. Otto,
Center for Innovation in Product Development, Massachusetts Institute
of Technology, Cambridge, MA 02139, USA

This paper presents an approach to architecting a product family that shares inter-changeable modules. Rather than a fixed product platform upon which derivative products are created through substitution of add-on modules, the approach here permits the platform itself to be one of several possible options. We first develop function structures for each product. After comparing function structures for common and unique functions, rules are applied to determine possible modules. This process defines possible architectures. Each architecture is represented using a matrix of functions versus products, with shared/unique function levels indicated. This provides a systematic approach to generating architectures. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: design methodology, product development, conceptual design, product platforms

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 architecture¹. 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 elements¹. 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 drivetrains¹. 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.

1 Bremner, R 'Cutting edge platforms' *Financial Times Automotive World* September (1999) 30-38



Surprisingly, Volkswagen and Ford's definitions as to what constitutes a platform are vastly different.

This example highlights the fact 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 pack module is used across dozens of Black and 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 mar-

- 2 Meyer, M and Lehnerd, A** *The Power of Product Platforms* The Free Press, New York (1997)
- 3 Sanderson, S and Uzumeri, M** 'Managing product families: the case of the Sony Walkman' *Research Policy* Vol 24 (1995) 761–782
- 4 Henderson, R and Clark, K** 'Architectural innovation: the reconfiguration of existing product technologies and the failure of established firms' *Administrative Science Quarterly* Vol 35 (1990) 9–30
- 5 Wheelwright, S and Clark, K** 'Creating project plans to focus product development' *Harvard Business Review* Mar–Apr (1992) 70–82
- 6 Erens, F and Verhulst, K** 'Architectures for product families', in *WDK workshop on Product Structuring* Delft University of Technology, Delft, The Netherlands (1996)
- 7 Robertson, D and Ulrich, K** 'Planning for product platforms' *Sloan Management Review* Vol 39 No 4 (1998) 19–31
- 8 Pedersen, P** 'Organisational impacts of platform based product development' *International Conference on Engineering Design* Munich, Germany Vol 3 (1999) 1507–1512
- 9 Pulkkinen, A, Lehtonen, T and Riitahuhta, A** 'Design for configuration—methodology for product family development' *International Conference on Engineering Design* Munich, Germany Vol 3 (1999) 1495–1500
- 10 Martin, M and Ishii, K** 'Design for variety: development of complexity indices and design charts,' in *ASME Design Engineering Technical Conferences* Sacramento, California, USA (1997) DFM-4359
- 11 Kota, S and Sethuraman, K** 'Managing variety in product families through design for commonality,' in *ASME Design Engineering Technical Conferences* Atlanta, Georgia, USA (1998) DTM-5651
- 12 Simpson, T, Maier, J and Mistree, F** 'A product platform concept exploration method for product family design,' in *ASME Design Engineering Technical Conferences* Las Vegas, Nevada, USA (1999) DTM-8761
- 13 Conner, C, De Kroon, J and Mistree, F** 'A product variety tradeoff evaluation method

ket variance—how variety is needed on each customer concern, as measured by the variance from customer to customer, segment to segment, or brand to brand, for example. 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.

I Related work

The development of product families built on product platforms and shared modules has been the subject of much recent research. Meyer and Lehnerd² 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 Uzumeri³ and Henderson and Clark⁴ 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^{5–9}.

In the product design literature, one can find several design and manufacturing strategies for offering variety that begin with commonality metrics^{10,11}. There are also several model-based approaches to designing different kinds of product platforms. Simpson et al.¹² and Conner et al.¹³ use a Decision Support Problem formulation to design families of products based on scalable platforms. A similar approach is used by Ortega et al.¹⁴ to show tradeoffs among multiple life-cycle objectives for a product family. Krishnan et al.¹⁵ developed network models to design families of products that are measured along a single performance criterion. Siddique and Rosen¹⁶ use a graph-grammar approach to design commonality into a family of products. Finally, optimization approaches have been developed by Gonzalez-Zugasti et al.¹⁷ and Nelson et al.¹⁸ to design product platforms and families of variants. Another optimization approach is used by Fujita et al.¹⁹ 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

for a family of cordless drill transmissions,' in *ASME Design Engineering Technical Conferences* Las Vegas, Nevada, USA (1999) DAC-8625

14 Ortega, R, Kalyan-Seshu, U and Bras, B 'A decision support model for the life-cycle design of a family of oil filters,' in *ASME Design Engineering Technical Conferences* Las Vegas, Nevada, USA (1999) DAC-8612

15 Krishnan, V, Singh, R and Tirupati, D 'A model-based approach for developing a family of technology-based products' *The University of Texas at Austin Management Department Working Paper* (1998)

16 Siddique, Z and Rosen, D 'Product platform design: a graph grammar approach,' in *ASME Design Engineering Technical Conferences* Las Vegas, Nevada, USA (1999) DTM-8762

17 Gonzalez-Zugasti, J, Otto, K and Baker, J 'A method for architecting product platforms with an application to the design of interplanetary spacecraft,' in *ASME Design Engineering Technical Conferences* Atlanta, Georgia, USA (1998) DAC-5608

18 Nelson, S, Parkinson, M and Papalambros, P 'Multicriteria optimization in product platform design,' in *ASME Design Engineering Technical Conferences* Las Vegas, Nevada, USA (1999) DAC-8676

19 Fujita, K, Sakaguchi, H and Akagi, S 'Product variety deployment and its optimization under modular architecture and module commonalization,' in *ASME Design Engineering Technical Conferences* Las Vegas, Nevada, USA (1999) DFM-8923

20 Rehtin, E and Maier, M *The Art of Systems Architecting* CRC Press, Boca Raton, FL (1997)

21 Pahl, G and Beitz, W *Engineering Design: A Systematic Approach* Springer Verlag, London (1996)

22 Bischoff, W and Hansen, F *Rationelles Konstruieren Konstruktionsbücher Bd 5* VEB-Verlag Technik, Berlin (1953)

23 Roth, K *Konstruieren mit Konstruktionskatalogen* Springer Verlag, Berlin (1982)

24 Rodenacker, W *Methodisches Konstruieren Konstruktionsbücher Bd 27* Springer Verlag, Berlin (1970)

partition systems into common modules. Rehtin and Maier²⁰ 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, as described by Pahl and Beitz²¹ and based upon the rich history of functional modeling, are used to model the products in the family²²⁻²⁴. Starting with these functional models, we then make use of single product architecting rules as first developed by Stone et al.²⁵ and then further apply portfolio architecting rules as developed by Zamirowski and Otto²⁶. The result is a set of possible product portfolio architectures. These fit into the categorization of portfolio architectures shown by Yu et al.²⁷ The method shown here builds on the above works to provide a means to select among the possible architectures.

2 Approach

Our approach to architecting systems is outlined in Figure 1. Each step will be extensively covered in the subsequent sections. A design team developing a completely new product portfolio, or perhaps augmenting an existing product portfolio with additional product variants, would implement the process outlined here. This would occur after possible products are selected for inclusion in the portfolio product line and after the basic physical principles of each product are established. A design team can then apply this architecting process. Obviously, exploratory iteration is required to decide if some loosely related products should be included in the portfolio, or perhaps rejected upon completion of the exploration as outlined here.

As outlined in Figure 1, 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

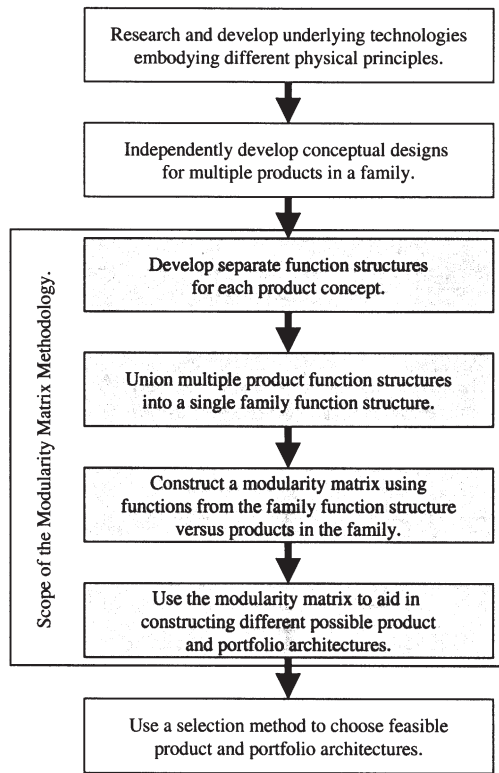


Figure 1 Overview of the portfolio architecting process

- 25 Stone, R, Wood, K and Crawford, R** 'A heuristic method to identify modules from a functional description of a product,' in *ASME Design Engineering Technical Conferences Atlanta, Georgia, USA (1998) DTM-5642*
- 26 Zamirowski, E and Otto, K** 'Identifying product portfolio architecture modularity using function and variety heuristics,' in *ASME Design Engineering Technical Conferences Las Vegas, Nevada, USA (1999) DTM-8760*
- 27 Yu, J, Gonzalez-Zugasti, J and Otto, K** 'Product architecture definition based upon customer demands' *Journal of Mechanical Design* Vol 121 No 3 (1999) 329–335

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.

This modularity matrix approach provides a design team with a uniform representation of multiple product and portfolio architectures, each with different constitutive modules, yet each represented by a modularity matrix. These matrices are a visual representation of possible modularity schemes, and provide an indication of the design requirements for each module. That is, each product may place slightly more stringent requirements on the modules along different specification dimensions. The matrix serves as one of the first indicators of this. Architectures generated by this method can then be compared and contrasted using a selection approach such as Pugh

concept selection²⁸, more numerical forms such as decision analysis²⁹, 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 and 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 and Decker® VersaPak™ line.

2.1 Individual product function structures

Functional decomposition of products can be completed through various methods. FAST³⁰, Hatley and Pirbhai³¹, 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.

For electromechanical systems, a technique to create diagrams known as function structures is widely employed²¹. 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 Beitz²¹, we restrict the function structure to only product functions. 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 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.

28 Pugh, S *Total Design: Integrated Methods for Successful Product Engineering* Addison-Wesley, Wokingham, UK (1991)

29 Thurston, D 'A formal method for subjective design evaluation with multiple attributes' *Research in Engineering Design* Vol 3 No 2 (1990) 105–122

30 *Value Analysis, Value Engineering, and Value Management*, Value Analysis Incorporated Clifton Park, NY (1993)

31 Hatley, D and Pirbhai, I *Strategies for Real-Time System Specification* Dorset House Publishing Company, New York (1987)

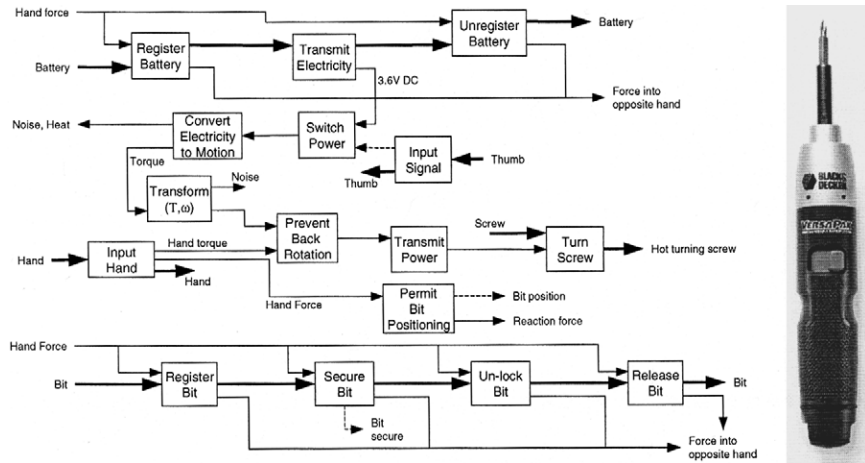


Figure 2 Function structure for a VersaPak™ cordless screwdriver

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 and 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. We address 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 centers on 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 Wood³²

32 Otto, K and Wood, K 'Product evolution: a reverse engineering and redesign methodology' *Research in Engineering Design* Vol 10 No 4 (1998) 226–243

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, through 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.

2.2 Family function structure

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

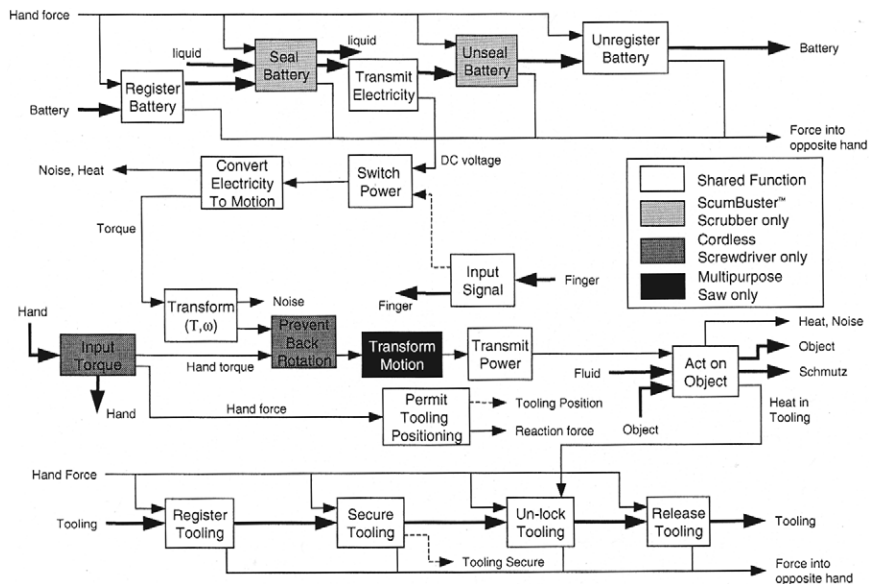


Figure 3 Family function structure for a VersaPak™ portfolio of products

of the function structure diagrams. For example, in Figure 3, the intersection is shown as the unshaded 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 2-speed drill, ScumBuster™ scrubber, and Wizard™ rotary tool. Thus, a more sophisticated view of platforming must be employed.

2.3 Modularity rules

When considering a single product, Stone et al.²⁵ 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 interfaces 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 identification²⁶. 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. Func-

tional 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.

Unique functions are those that are specific 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 variety¹⁰.

2.4 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 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 on 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 modules can be shared. Thus, the architecture for each product and for the product family is laid clear.






					
Function	Cordless Screwdriver	Multipurpose Saw	ScrubBuster™ Scrubber	2-Speed Drill	Wizard™ Rotary Tool
Register / Unregister Battery	1 battery	2 batteries	1 battery	2 batteries	1 battery
Transmit Electricity	1 battery	2 batteries	1 battery	2 batteries	1 battery
Seal / 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
Prevent Back Rotation	yes	-	-	-	-
Input Hand Torque	yes	-	-	-	-
Transform Motion	-	yes	-	-	-
Register / Release Tooling	hexagonal hole	blade carriage	triangular hole	three-prong chuck	collet
Secure / Unlock 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

3 Portfolio architecting

In this section, we describe the how to use the modularity matrix, family function structure, and modularity rules to develop the portfolio architecture. The approach described was used to analyze modules for the Black and Decker® VersaPak™ portfolio of products. Separate function structures were developed for each individual product, and 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.

Once the family function structure for the portfolio was created, various architectures could be developed by changing the targets within the matrix and utilizing the modularity rules. There are two aspects to this: developing the individual product architecture for each product and developing any shared modules within the portfolio architecture.

3.1 Product modules

First, we can consider the architecture of an individual product. This can be done by forming groupings of functions using the architecting rules on the function structure of each individual product. These groupings of functions can then be displayed within the modularity matrix using boxes in the product's column, as shown in Figure 5. Figure 5 shows the modules for the products as currently designed by Black and Decker®.

For example, the 2-speed drill features two product modules, as illustrated by the two boxes. The functions of register/unregister battery and transmit electricity were bundled into a module that makes use of two batteries. The second module in the 2-speed drill bundles the functions of register/release






					
Function	Cordless Screwdriver	Multipurpose Saw	ScumBuster™ Scrubber	2-Speed Drill	Wizard™ Rotary Tool
Register / Unregister Battery	1 battery	2 batteries	1 battery	2 batteries	1 battery
Transmit Electricity	1 battery	2 batteries	1 battery	2 batteries	1 battery
Seal / 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, to)	transmission A	transmission B	transmission C	transmission D	-
Transmit Power	rotating shaft	translational blade	rotating shaft	rotating shaft	rotating shaft
Prevent Back Rotation	yes	-	-	-	-
Input Hand Torque	yes	-	-	-	-
Transform Motion	-	yes	-	-	-
Register / Release Tooling	hexagonal hole retaining clip	blade carriage set screw	triangular hole snap fit	three-prong chuck chuck housing	collet collet nut
Secure / Unlock Tooling	-	-	-	-	-
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

tooling and secure/unlock tooling. In the case of the drill, a three-prong chuck is used. A single module addresses both functions, and is thus shown as a box on the modularity matrix.

To form such product modules for any column (product in the portfolio), one must examine the function structure for the product, and apply the modularity rules. For example, the 2-speed drill's battery module was defined by the dominant flow rule applied to the battery. The three-prong chuck module was defined by the dominant flow rule applied to the bit. Similar application of the modularity rules to the other products in the portfolio (columns in the modularity matrix) forms other product modules.

3.2 Shared modules

Next, we explore the portfolio architecture by identifying shared functions across the portfolio. By examining similarities across columns for a single function in the modularity matrix, possible sharing can be considered. When different products have common specification levels for a given function or module, it can be shared. If specification levels are not common between multiple products, then shared modules may not be feasible.

To establish shared modules, one can consider a function in the matrix, and determine whether different products have similar specification targets. If they are sufficiently close such that making them the same does not overly impact performance, then the targets can be made the same. For example, the convert electricity to motion function in the Versapak™ modularity matrix has very similar targets for the cordless screwdriver, Wizard™ rotary tool, and ScumBuster™ scrubber, and could reasonably be made identical. Making them the same would permit all to use the same

motor. Similar examination of other rows could permit other shared modules to be considered.

For the current VersaPak™ portfolio, one can see that there are few shared modules, as highlighted in Figure 6. Only the battery system functions share functional specification values across the products. Therefore, it is only the battery pack module that is currently shared across products in the family. For example, we might try and define a single battery sufficient for all the variant products, thereby increasing commonality. Rather than having each product have a unique motor, we might also try to use only two sizes of motor for the convert electricity to motion function.

3.3 Simultaneous architecting

It should be clear that one cannot consider an individual product architecture nor shared modules independently. They are coupled; configuring a shared module impacts all products and changing a product architecture impacts shared modules. 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 a 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 while 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/releasing and securing/unlocking tooling.






					
Function	Cordless Screwdriver	Multipurpose Saw	ScumBuster™ Scrubber	2-Speed Drill	Wizard™ Rotary Tool
Register / Unregister Battery	1 battery	2 batteries	1 battery	2 batteries	1 battery
Transmit Electricity	1 battery	2 batteries	1 battery	2 batteries	1 battery
Seal / 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
Prevent Back Rotation	yes	-	-	-	-
Input Hand Torque	yes	-	-	-	-
Transform Motion	-	yes	-	-	-
Register / Release Tooling	hexagonal hole	blade carriage	triangular hole	three-prong chuck	collet
Secure / Unlock 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 shared modules for a VersaPak™ portfolio of products






					
Function	Cordless Screwdriver	Multipurpose Saw	ScumBuster* Scrubber	2-Speed Drill	Wizard™ Rotary Tool
Register / Unregister Battery	1 battery	2 batteries	1 battery	2 batteries	1 battery
Transmit Electricity	1 battery	2 batteries	1 battery	2 batteries	1 battery
Seal / 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
Prevent Back Rotation	yes	-	-	-	-
Input Hand Torque	yes	-	-	-	-
Transform Motion	-	yes	-	-	-
Register / Release Tooling	hexagonal hole	blade carriage	three-prong chuck	three-prong chuck	three-prong chuck
Secure / Unlock 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 shared modules for a VersaPak™ portfolio of products

The other two products of the family, namely the cordless screwdriver and multipurpose saw, do not share the same methods for these functions. 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’ modularity matrix is not trivial. One must consider both portfolio sharing concerns and individual product concerns. Generally, these two can conflict. The portfolio considerations lead one to specify average components that work across multiple products. Product considerations lead 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/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/unseal battery function. 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.

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.

The multipurpose saw and 2-speed drill share the same battery system functions at the same level of power, size, and attachment, and thus both utilize the same two-battery module. Sharing across some but not all of the products in a portfolio can also be easily represented using the modularity matrix, as shown in Figure 7.

Making use of the modularity matrix therefore provides structure to the very difficult task of developing both the functional exclusion/inclusion boundaries and specifications for modules that are shared across products. The approach is to first develop possible product modularizations, then compare these across products, seeking common scope of functions and common specifications for shared modules.

3.4 Architecture evaluation

The work above is the heart of the approach outlined here, and results in the generation of several candidate modularizations, each expressed individually by a modularity matrix. It is a concept generation exercise. As the next step not discussed here, 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 Otto³³. 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 optimizations completed to better estimate performance^{12,19,34}. In real world applications, business cases of each platforming scenario would also have to then be developed³⁴.

4 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 possible modules in the product family. To clarify amongst possible architectures, we then arrange the identified functional modules into modularity matrices, which display the desired functions and their levels for all products in each possible family. The possible modules identified through product and portfolio

33 Gonzalez-Zugasti, J and Otto, K 'Platform-based spacecraft design: a formulation and implementation procedure', in *IEEE Aerospace Conference Big Sky*, Montana, USA, Vol 1 (2000) pp 455–463

34 Gonzalez-Zugasti, J, Otto, K and Baker, J 'Assessing value for product family design and selection', in *ASME Design Engineering Technical Conferences Las Vegas, Nevada, USA* (1999) DAC-8613

modularity rules can 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.