International Perspectives on Modular Design Research: Connecting Different Approaches to Assist in the Development of Modular Product Family Concepts

Marcelo A .Department of Mechanical Engineering, São Carlos School of Engineering, University of São Paulo, Av. Trabalhador Sao-Carlense, 400, São Carlos, SP, 13566-590, Brazil

Research Article

Date of Submission: 10-04-2026 Date of Acceptance: 19-05-2026 Date of Publication: 1-06-2026

ABSTRACT

It has been shown that modular product platforms enable businesses to provide a range of goods while offering significant time and cost benefits. As a consequence, the design research community has created a wide range of product platform techniques during the last ten years. However, since the approaches have mostly been created independently of one another, it is challenging to compare and integrate appropriate methodologies. We build a general set of thirteen platform design processes for creating a platform idea after analyzing the literature on modularity and product platforms. After that, we look at a number of product platform idea development procedures that are used by various businesses, and we get a general flow of the stages from this. The many modular and platform design techniques created by the community may then be chained together by associating them with the sequence.

Keywords: Design techniques, platform architecture, modular and platform design, Product family design and product development procedure

INTRODUCTION

Products that stand alone are uncommon. The majority of items belong to a family of products, which are collections of product variations built on a same platform. Using common assets to allow cost-effectiveness, product platforms facilitate the creation of product families and generations of goods. For instance, cost reductions and other operational advantages in components procurement, production, and quality control are brought about by these shared assets. Product platform design techniques have been actively developed over the years, including ways to map a family's requirements to a collection of products, define common and unique platform modules, optimize variety, cost, commonality, or other parameters, assess platforms, and more [1, 2]. The apparently wide range of information without a logical framework to compare development process tasks and the related accessible methods and tools is now impeding the transfer of these methods, algorithms, and techniques to industrial practice. Each approach has usually been created separately from the others in the design research community, and it is unclear if and how different approaches may be utilized in tandem. As a result, alignment in design processes and a study of the research literature are required. To show how the various areas of emphasis might be connected into one of many design pathways for platform idea creation, we examine the research in the literature. The objective is to demonstrate how different platforming strategies and techniques may support the overarching objective of creating a profitable product family. The early stages of design, including system principles, component selection, and component geometric arrangement, are the main topics of this article. This document does not cover specific design processes like product and component testing and validation. Additionally, we don't talk about supply chain development or manufacturing systems. Modularity and platform research still has to be finished and investigated in these areas.
Thus, the rest of this work is structured as follows. Our strategy for defining different sequences and chains of the platforming techniques that are accessible from the research community is presented in the next section. In Section 3, we provide a case study featuring a family of unmanned ground vehicles and outline the thirteen phases utilizing the different approaches to achieve the suggested design processes. Next, by choosing a few specific processes in Session 4, we provide a guidance for using the matrix connecting possibilities. In Section 5, we examine the difficulties in creating platforming strategies and practices as well as the limits of the suggested matrix and related techniques.

APPROACH AND MODULAR PLATFORM DEFINITION PROCESS

Three tasks were completed in order to facilitate the selection of techniques for industrial application and the connection of the methods for platform concept development. In order to determine an efficient order for platform

development activities, we first examined and contrasted a number of product development procedures from big businesses. The various design techniques were then examined, categorized, and mapped into these tasks. Lastly, each step was now sequentially linked to the methods themselves. As a result, a set of alternate approaches to each phase and a set of design flows that might be used in a modular platform design process were created. After reviewing the literature, we categorized techniques into thirteen general processes that are carried out using different techniques or tools for platform idea creation. Table 1 displays this, arranged alphabetically. However, we observed conflicts over which chores should be completed first. For instance, one may predefine modular designs as modularity permutations and then create optimization formulations for each permutation, or certain optimization techniques can be set up to look for ideal module borders in specific issue formulations. In a similar vein, voice-of-the-customer analysis may be performed for every product version or perhaps for many product variants. Generally speaking, there isn't a single optimal overall response to these questions, and it might vary greatly depending on the situation. As a result, rather than advocating for a single solution, we aim to comprehend and provide suggestions as a starting point.

To find a possible common order of development phases, one might, for instance, compare platforming development processes and modularity sequences encountered in industry. The usage of industrial businesses in practice suggests a practical and effective character, even if it is unclear whether they have any more power in defining the proper sequence of stages. Furthermore, the formulation of industrial design procedures is the result of many person-years of deliberation and work.

Therefore, we gathered a list of industrial design procedures (see Table 1) from a sample of 10 companies that often create modular product platforms in order to offer a notional proposal. Software, automotive, industrial, and electronics industries were chosen to provide a range of applications. ABB [3-5], Airbus [6], Carrier [7-9], Cummins [10, 11], Danfoss [12-14], Ford [15, 16], IBM [17, 18], ITT [19, 20], LG [21], and Motorola [22] were among them. Standard-work technical review requirements—the data that each design project must provide at various stages of the development process—were the focus of our data study. Since we discovered that the amount of specification of standard-work descriptions of process stages differs across the firms and the degree to which they are followed internally varies, we did not directly examine these descriptions. In particular, we did not specifically search standard work documents for the platform or modular processes mentioned in Table 1. While some descriptions lacked significant depth, others provided far more.

Instead, we concentrated on the descriptions associated with each technical review and contrasted them with Table 1's stages. The gate review in each corporate development process for each phase in Table 1 was identified and documented based on our investigation using this collection of data from many businesses. The outcomes are shown in Table 2, where the stages from Table 1 have been rearranged to remove gate review checkpoint ordering disputes across these industrial organizations. In order for the column's process to be consistent with the specified sequence, each vertical item in Table 2 would be monotonically rising along the column. When many rows have the same number, it means that the firm does not specify the order in which these tasks are finished for review—as long as they are finished before going on to the next set of activities. As a result, it is clear that these product development procedures usually enable project leaders to choose whether activities should be completed earlier or concurrently with other tasks. The final recording of the findings for evaluation is all that is needed. Every job that is permitted in Table 2 has the same gate review number per column. A nominal design flow is represented by the resultant sequence in Table 2. Assigning step outcomes to corporate standard task descriptions is challenging for a number of reasons. First, standard work is evolving, and based on descriptions found in the literature, what each firm is doing now is an evolved condition. Second, there are distinctions between the deadlines for completion and the availability of the findings for evaluations. As a result, Table 2's sequence represents our assessment of the most reliable data. Lastly, rather than coming from observational studies of initiatives inside each organization, the data in Table 2 are those that have been reported in the literature. Our experiences, however, are typically in line with what is reported in the literature since we have worked with several of these businesses. As a result, Table 2 shows how the modular platform design processes reported in the study literature may be effectively sequenced. It is evident from Table 2 that the suggested general sequence of modularity techniques fits a suitable sequence for any company. No vertical sequences are out of order. In light of this finding, we suggest the sequence as a helpful place to start for anybody researching

modularity tools and methodology in design research literature. Additionally, we provide it as a realistic beginning point for integrating various techniques and technologies into beneficial design processes.
In the context of the platform development process shown in Table 2, previous research is discussed in the next section. This is a prescription for applying the processes in a logical order, with options for each step shown in Table 3 and Figure 1. Additionally, we provide a list of suggested modularity deliverables that should be reviewed at each stage, regardless of the particular techniques. This is not the only way to go through these processes or the only approaches that may be used in a particular step; generally speaking, there are design challenges where other ordering are suitable. For any company or stakeholder unfamiliar with platform-based product development, Table 3 offers a sensible and practical place to start. Alternative techniques are used in each phase, including one simple way (such a manual procedure) and one or more more sophisticated methods.
There is often an option over whether or not to use a functional modeling approach, and some of the mid-sequence phases are interdependent. The extended "function" and "component" columns in the "alternatives" columns are used in Table 3 to illustrate this. In Section 3.5, this will be covered in more depth. We also add "skip" as an alternative, meaning that, while there is a loss of information involved, skipping that specific process step is one option to think about. For instance, after establishing module boundaries, one may decide to forego the architectural roadmap and go straight to commonality assignment, but doing so would not guarantee that the platform is adaptable to any unexpected future changes in the market and technology.

In general there is always a trade-off between time allocated and used for each step and the expected benefits. Sometimes a choice is made to use a simpler approach with lower expected benefits to expedite the process. The baseline flow depicted in Table 3 represents a minimal set of steps necessary to generate a modular platform concept. It provides a simplified approach that starts with a set of existing products and makes them share common modules. This is the manually listing out of market segments and the product offers for each segment as an attack plan. Against this baseline, all additional methods are discussed in the next section for the added benefits provided over this minimal approach. An illustrative example is also provided in the next section to tie the various methods and approaches together.

ALTERNATIVE METHODS AND APPROACHES

Step 1: Market Segment Definition

The first step in the sequence is to define the market segments. Modular product platforms exist to more easily enable multiple products for multiple markets. Thus, the platform development effort ought to entail defining the population, or populations, of customers and product applications. Typically this starts with an observation of a perceived need in an application domain and extension into exploring the range of different geographies and demographics that may have similar needs. Market segmentation methods are well established [23] and will not be discussed here. The methods help in identifying potential clusters of customers with similar needs. This, in turn, enables designing a product variant on a per-market-segment basis, rather than a product that is trying to meet all the vastly different needs.
There are many approaches to clustering the market base into these segments. One can, for example, define a wide range of characteristics upon which to subdivide a population of potential customers, such as applications, geographic boundaries, behaviors, and demographics [24]. However, one will also find that many such distinctions are artificial and that there is no real difference in customer needs and there is a risk of over-partitioning the population. For each smallest partition of the population one can conduct surveys of customer demographics, use applications, or even customer needs analysis to establish clear distinctions amongst the population. Clustering these results into internally homogeneous and externally heterogeneous groups helps form clear *market segments*.
The simplest approach to identifying these clusters of similar customers is to manually cluster on characteristics such as use or business application as market segments. For example, defense-related applications form a different segment than commercial applications, and geographic/regional variations may lead to distinct segmentation. Another option could be use of standard clustering methods such as hierarchical clustering. Within the design research community, clustering methods have been developed to establish common needs among refined segment characteristics including

socio-economic attributes [25, 26]. Mathematical optimization models can also be used to determine demands for uncertain markets and estimate the revenue and profit of the defined market segments [27]. For example, Zhang et al. [26] used a fuzzy clustering approach to determine market segments for a product family. One issue that can arise is over-partitioning of the market, i.e., offering too many product variants. Unfortunately this is not always obvious until sales figures become apparent and do not match the sizes of the original clustering. Establishing differences in markets and targeted products remains a fruitful area of study.
We use a family of *unmanned ground vehicles* (UGVs) for explosive ordnance disposal as an example to discuss the various methods in this paper (see Figure 2). In this example, the market segments were manually defined against perceived business categories with lead users already exploring UGVs on their own accord. UGVs are being explored in both civilian and military applications. Smaller UGVs are often used to detonate an explosive remotely. These UGVs may be destroyed along with the ordnance. Larger UGVs, on the other hand, may perform additional functions such as sensing, defusing, or disposing the ordnance, allowing for repeated use and multiple missions.

UGVs are used primarily to reduce casualty risk of ordnance disposal regardless of whether it is civilian or military; however, they can be used for many other types of functions as well. Partitioning these potential applications can lead to a form of market segmentation. As an example, we segment the UGV "market" into the following applications:
Explosive abatement – ordnance detection, disabling, and disposal
Hazardous sites – hazard sensing and locating in unhealthy environments
Combat – providing attack capability for high-risk missions
Reconnaissance – providing site observations and intelligence
Target and decoy – providing a target that simulates an enemy vehicle
Civil and commercial – UGVs for commercial transport

Meanwhile, the range and autonomy of the UGVs can provide a second axis to categorize potential customer segments:

Micro, line of sight
Small, remote operated, 2 mile range
Tactical, remote operated, 50 mile range
Long endurance, autonomous, 100 mile range

Step 2: Market Attack Plan
Following the product market matrix approach advocated by Kotler and Keller [28] and Meyer and Lehnerd [29], we can create a matrix of these two categories (application versus range) as shown in Figure 3. In their approach the rows of the matrices correspond to market segments and the columns to products segments, which helps designers identify how platform elements may be leveraged across multiple segments. The segments may share specific customer needs with multiple other segments, but this can be determined later during customer needs identification.
This matrix helps in creating a *market attack plan* that determines whether new products are offered simultaneously to all segments in parallel or rolled out sequentially over time to different market segments, Step 2 in the sequence of Figure 1. While the baseline process of market planning involves assessment of many factors include the sales potential, competitive offers, marketing and distribution, we consider here the mutual impact of modularity and the leveraging potential it offers. In uncertain market environments, scenario based Monte Carlo simulation and models could be applied to validate the market planning [30].
Notice that vertical leveraging (solid vertical line in Figure 3) can be associated with *scalable platforms* where different sizes of the same modules and components are used, while horizontal leveraging (dashed horizontal lines in Figure 3) can generally be associated with *swappable modules* where a common subset of core modules are used across different products. Successful platforms often utilize a combination of scaling and modularity to attack the market in a strategic and cost-effective manner [27]. Lei and Moon [31] develop a decision support system to identify new market segments and product positioning for product family design based on market data and product properties.

To develop the market attack plan for our UGV product family example using the *product market matrix*, we consider the following performance characteristics: weight, speed, range, and lift capacity. Over 50 different potential applications were identified in the matrix based on type of ordnance, functionality (e.g., dig, detonate, diffuse), location of operation, etc. [32]). We then manually clustered the UGV market segmentation matrix into three segments that are, for the sake of this example, consistent with current systems in the market. We identified three "performance tiers" based on weight: (1) small, (2) medium, and (3) large. Mapping this into the matrix, we segmented the columns into small, medium and large UGVs, and the rows into explosion abatement, hazardous sites, and reconnaissance.
Based on this segmentation, a vertical leveraging strategy may be possible as illustrated by the solid vertical arrow in Figure 3. This would enable a new family of UGVs to be developed and scaled up for this initial segment, with platform modules created to satisfy requirements for the three sizes of UGVs. The dashed lines in Figure 3 illustrate the potential to later extend the platform by leveraging these modules horizontally to other market segments.

Step 3: Customer Needs Gathering
Once isolated as individual market segments, another step is to establish the customer needs for the each targeted product variant, Step 3 of Figure 1. This is a segment by segment step. Therefore, customer needs may be gathered for each individual product separately. In other words, *the intended variety in the product family should be designed prior to defining what the common modules should be in the platform*.
Hauser and Griffin [33] provide seminal work on gathering voice of the customer. Here, interviews are conducted with randomly sampled people from each market segment, and questioned about how they use the product. This elicits qualitative and quantitative need statements which they seek from the product. Zhang et al. [26] use conjoint analysis and discrete choice models to evaluate customer perceived utilities. Yu et al. [34] provide a method to group common modules based on common needs across segments.
The customer needs can be categorized into groups according to the similarity of attributes or the extent of customer satisfaction. For example, Kano [35, 36] introduced a conceptual model to analyze customer preferences. In the Kano's model, customer requirements are classified into three different types of needs: basic, performance, and attractive needs.
The defined customer-needs should be managed and updated continuously to develop a new product for the next generation. Schuh and Tanner [37], Harlou [38], and Eilmus et al. [39] utilize customer-needs variety diagrams to manage customer-needs variety. The link between the variants of a product family and the customer needs can be visualized using the tool *Tree of External Variety* as shown in Figure 4 [40].

For the UGV example, the customer requirements were gathered from multiple Requirements Working Groups that met over a period of about two years. Each group was comprised of representatives from different branches of the military and included senior personnel, ordnance disposal experts, and technicians involved with logistics, maintenance, and support.

Step 4: Systems Requirements Definition
The next stage is to specify quantifiable verifiable system requirements for each product variation once the customer demands for each market category and associated product variant have been clearly identified (see Figure 1). Once again, there are tried-and-true techniques at this stage, such as the House of Quality [41]. Determining the worst-case operating circumstances and the system requirements for them is an additional choice. Converting the customer's demands into system requirements, akin to target value setting as explained by Ulrich and Eppinger [42] or utilizing INCOSE principles for requirement formulation [43], may be the most straightforward choice for this phase. Whichever approach is used, quantifiable system requirements have been determined at the conclusion of this process, which give design objectives to guarantee that each version can meet the demands of customers in its respective market segment. Sensitivity analysis may be used to estimate the ideal system needs and resources by combining Quality Function Deployment (QFD) with mathematical models [44]. Customers' preferences may differ depending on particular functional needs, design, manufacturing process, and new technologies, making it challenging to

precisely identify and disseminate the system requirements in the early phases of design. Change propagation analysis (CPA) may be used to discover product variants and forecast the evolving impacts of the need in order to reduce design and requirement variations throughout product development. In order to identify the dangerous region and comprehend the relationships between components based on design modifications, Clarkson et al. [45] developed a change prediction approach. To measure the effect of altering the requirements and behaviors, change propagation behaviors [46] and a change propagation index [30] were suggested. For the UGV example, specifications (such as weight, range, speed, and manipulator length) were established for every market segmentation grid in Figure 3. For every need, threshold and objective values were determined. While the goal value is a target that consumers would want to reach, the threshold value is the minimal value that must be attained in order to meet a requirement. system specifications that were established during the customer needs study for the UGV example. For every weight class, threshold and goal values were established for every need. UGVs are required with varying system requirements for specific targets found in commercial or combat contexts, depending on the uses. For instance, high-resolution cameras with zooming capabilities are used in military-grade systems to identify obstacles and particular targets at great distances. Customers may decide on the systems and specifications of the distinct goals, which are regarded as the particular needs of a UGV family. Donaldson [47] provides many instances of system needs. Common, variant, and unique requirements—that is, those that are the same (same) across all three UGVs, those that are similar but vary slightly amongst UGVs, and those that are exclusive to a particular UGV—can be identified with the use of this information. For instance, each UGV has similar needs for maneuvering, sensing, and communication, but each weight class has different requirements for range and payload. Lastly, the needs for the manipulator, reach, drag/roll/push, and big object are specific to each UGV; the tiny UGV does not have any requirements since it does not require that capacity (such as large item pickup). Once again, Donaldson [47] provides particular instances.

Step 5: Functional Requirements Definition

Functional needs and limitations are two common categories for requirements [48, 49]. Usually, a functional requirement demonstrates a certain functional action, such as doing something. Conversely, a restriction is a condition that cannot be fulfilled by a single function. Two common examples of restrictions are weight and size. A lesson on creating functions vs constraints may be found in Otto and Wood [49]. We adhere to their recommended procedure, which begins with client demands that are divided into functional requirements. After that, we keep listing the intended system's customer activities and use those activities together with the functional requirements to create a functional model. Next, we compare that functional model to the initial requirements of the client. As a consequence of this method, designers are better able to comprehend what the system should do to meet consumer wants rather than how. A function-based method or a component-based approach may be used to build a platform architecture after the system requirements have been established, as shown in Figure 1. Although a more popular alternative technique is described here, a sequential route from function-based approaches in the early conceptual stages to more sophisticated component-based methods is also feasible. In order to describe common functions regardless of specific embodiment choices, the function-based approach adopts a broader perspective. The component-based method, on the other hand, uses a priori knowledge of the most relevant components required, regardless of whether it is obtained or assumed. Mapping function to form, or specifying the components as part of embodiment design, comes after using the function-based method. Therefore, although the function-based method may take into account a wider range of options, both techniques ultimately provide the same outcome. We provide a quick demonstration of both methods in this article.

Function-Based Methodology

A functional requirement cluster analysis is another study pertaining to functional needs. In order to translate consumer wants into functional requirements, Jiao et al. [50] devised a method based on fuzzy clustering and association rule mining. Excessive functional variation for customer pleasure might lead to a sharp rise in expenses, even while the functional requirements are aligned with consumer expectations [50]. Schuh et al. [51] created variety diagrams, such as a hierarchical function structure, to manage and update the variety of functional needs for the next generation in a product family. The normal goal of a UGV is to move to a pre-planned place, observe and record the environment, gather things or samples, or operate on objects as directed and transmitted from a distant point. Moving

the camera and manipulators up, down, and horizontally is one of the supporting functions for these simple tasks. Our selection of functions was impacted by the robot family's current setup since this is an established family of robots. As a consequence, the design produced utilizing the component-based method will resemble the implementation of this functional model (see Figure 5).

Component-Based Methodology

Without initially using the function-based method, components may be defined straight from the requirements. As an alternative, the functions may be used to specify the components during the system embodiment phase [49, 52]. Martin and Ishii's [53] Design for Variety approach, which extends the House of Quality to produce a function-component matrix to determine the Generational Variety Index (GVI) for each subsystem, is one example of component alternative research. The mapping of requirements to components has been integrated into these approaches. In order to determine the amount of work required to redesign a component in the event that a need changes, the GVI matrix assists in mapping engineering requirements into components. While low GVI values suggest components that will stay mostly unchanged across the family, high GVI subsystems will undergo extensive reconfiguration to meet the variety of client wants. The GVI values for the UGV case are shown in Figure 6 [32]. As can be seen, there will be significant diversity in the manipulator and chassis within the family (i.e., high GVI values), but there will be little variation in the cameras and OCU (Operator Control Unit). The GVI is a systematic composite. A multi-criteria method was created by Krause et al. [40] employing the

The components as shown in Figure 7 may be redesigned using the Variety Allocation Model (VAM). By connecting the unique client demands with the variant components, the tool seeks to increase the quantity of standard parts.

(Put Figure 6 Here) (Put Figure 7 Here)

In addition to these two methods, one could use *Modular Function Deployment* (MFD) [54] to create a similar *Product Property Matrix* and identify platform modules. Luo, et al. [55] also offer a QFD-based approach for platform optimization.
Yet another possible component-based approach is to use a DSM, or *Design Structure Matrix* [56-58] to help identify modules. DSM is most appropriate when redesigning existing products. Then a DSM is formed by decomposing those products and mapping their components in those the matrix. A component-based DSM for the UGV family is shown in Figure 8. For the UGV example, the DSM is constructed using a teardown disassembly approach to system decomposition [59]. In this case, four UGVs were disassembled; their subsystems were identified and recorded to create the component-based DSM.
After defining components, introducing methods for managing component variety is known to be an important step for the next generation in a product family [37]. Schuh and Tanner [37], Schuh et al. [51], and Harlou [38] employed component variety diagrams for the variety management.

Step 6: Generic System Platform Architecture Definition
A modular platform forms the base for a set of product variants, possibly variants over multiple product generations. It is thus important that a modular architecture should encompass all the variant architectures that it must support, the next step in Figure 1. A simple way to create such a “generic” architecture for a modular platform is to create an architecture for each variant and merge these results together. The variant architectures can be created using either the functional or component-based approach described above.
A modular platform architecture is typically not generated as a single event transforming all product variants. Even when a modularity plan is developed, products themselves may be designed and developed in a sequence, one variant released at a time into the market. The legacy variants remain while the new platform variants are released. In this environment of constrained development resources, one approach is to consider the legacy products in the platform architecture. That is, to consider modular architectures that only modify portions of the entire architecture. Such approaches are entirely amenable to the methods outlined here. Typically several alternatives are considered, from

radically new architectures to simple derivative architectures of existing systems. To capture this spectrum of possible redesigns, alternative schematic architectures can be used to keep track of the alternative platform concepts. The functional modeling approach is sometimes preferred over matrix-based approaches due to its visually intuitive nature and higher abstraction level. In the functional approach [49, 52], the recommendation is to build each individual functional model and then merge these into a generic platform architectural model. In the UGV example, the functional model in Figure 5 provides a generic functional model as the individual UGVs differ in performance levels, not functionality, in the vertically leveraged scalable platform. While each model may have a different sized payload bay, for example, all of the UGVs have these components to perform common tasks of reaching/grasping objects and observing their surroundings, i.e., the functions are the same.

Harlou [38] provides a system block diagram using the concept of organ from the theory of domains [60], and Bruun et al. [61] extended the Harlou's work to visualize a product system including a set of component variants, interfaces, and modules in a product family.

Matrix-based methods are another option, using the system components as might be described with a system block diagram. The GVI and MFD methods discussed previously can both be used to map functions to components and thereby define modules. The collection of components considered defines the generic platform architecture. Notice that a DSM also represents a generic architecture. Similar to the functional models, the DSM in Figure 8 is already a generic DSM since the UGV family differs in size, etc. of the components but not in generic component types.

Other generic system architecture research includes methods by Blanchard and Fabrycky [62] and heuristic methods by Maier and Rechtin [63]. Crawley et al. [64] make note of system level interactions and architecture. Brière-Côté et al. [65] propose a structure-based approach to the management of product variety in product families.

Step 7: Component Alternatives

Following the definition of the generic system platform architecture, the variety of components within the product family must be specified. The sizes for the family as well as the functions and its attributes have to be defined. Alternatives can be developed manually or using tools such as the function component matrix [53], variety allocation model (VAM) [40], component variety management [37, 38, 51], or modular function deployment (MFD) [54]. The VAM (see Figure 7) supports the development of alternatives by separating the components in standard and variant shares. This results in platform concepts with more components and a higher level of commonality within the product family or fewer components and a lower level of commonality.

In the UGV example, the main circuit board variant depends on all application-relevant requirements and resulting functions and principles (see Figure 7). An alternative concept could be to strive for more components and split the circuit board into a standard component for the standard functions and additional modular circuit boards to provide optional functions. The standard circuit board is built in every UGV variant, and additional boards can be added depending on the specific application for a given variant. An alternative concept could be to reduce components and design a circuit board that fulfills all functions, but this would likely result in costly unused functionality and limited upgradeability.

Step 8: Module Boundary Definition

Dividing the system into modules of different sizes or capabilities is a crucial step in both the functional and component-based approaches. Numerous evaluations of modularity techniques exist [66, 67]. Several modularity heuristics, such as those for module identification using a functional model [68, 69] and heuristic detection of commonality using a modularity matrix [70], are examples of techniques. MFD was created by Ericsson and Erixon [54] to separate components into modules based on strategic considerations. Krause et al. employ the Module Process Chart (MPC) [40] (Figure 9) to integrate these product strategy considerations with a functional component-based modularization methodology. By displaying the chosen modules at each stage of life, it helps the module definition. Additionally, a number of clustering methods that use a DSM have been developed with module definition in mind [71, 72]. When the necessary level of modularity is reached, the majority of clustering algorithms include a metric to determine when to end the clustering. A method to reduce connections outside of modules was created by Thebeau [73]. Borjesson and Hölttä-Otto made improvements to this [74]. The goal of Yu et al. [72] is to reduce the amount

of data required to characterize the connection between modules. In contrast, Hölttä-Otto and de Weck [75] assess module independence using singular value decomposition, while Borjesson and Hölttä-Otto presents methods for concurrent clustering based on strategic similarity drivers and module independence [76]. There are reviews of similarity modularity and connection modularity elsewhere [66, 77, 78]. Modules may be defined using supervision and hierarchical clustering in addition to metrics [79]. Moon and McAdams [81] proposed a strategic decision-making method for module sharing in platform design using a game theoretic approach. This can be done either based on the functional requirements, component specifications, or in combination with another method that defines a metric, like MFD. Mathematical models may be used to determine the number of modules and the values of design variables in the modules [82, 83], as well as to assess the interface complexity of the described modules [70]. Using the UGV example, we continue both the functional and component-based techniques for the module boundary definition stage. To find possible modules for the functional method, we first use the module heuristics [68]. The branching flow heuristic (blue dashed lines), a transmission type module, and a dominating flow module (both in red) are used to identify many possible modules, as shown in Figure 5. Because of the functional model's very high abstraction level, modules may only contain one abstracted function. We see that more module candidates, such as all the individual drives (inside the "Allow DoF" routines) as conversion-transmission modules, would be found if the system were further disassembled. We decided to cluster the DSM for the component-based approach. Although there are other clustering methods available [84, 85], we decided to use manual clustering for the UGV example due to its relative simplicity. The clustered DSM for the first UGV design in Figure 6 is shown in Figure 8(b). The partitioning proposes two bus modules (for the chassis and electronics), two 3x3 modules (for communications and observation), and six single component modules (for particular capabilities, such manipulator), as shown in the picture. If the system were further broken down, a bigger DSM with larger modules would be produced to identify modules inside modules, much like the function-based method. Figure 5 and Figure 8(b) for the UGV example show that the outcome is almost identical. For instance, the mast with the camera head was separated as a module using both approaches (functions allow orientation DOF, allow location DOF, and sense environment in Figure 5). Likewise, both approaches left the electronics (control UGV function in Figure 5) and chassis (support loads function) as buses. DSMs may be programmed and automated, and the functional approach is more visible and intuitive.

Step 9: Architecture Roadmap and Future Uncertainty Management

Once a platform architecture and its modularity have been established, it may be necessary to take into account how it will evolve over time as technology advances. In order to prepare further module level improvements and related product upgrades, architecture roadmap research incorporates technology roadmapping [86, 87]. Generally speaking, a technological roadmap is a strategy for creating future goods with unfinished technology. The historical patterns enable the forecasting of anticipated performance in the future, notwithstanding their incompleteness. Moore's law [88, 89] or comparable performance increase curves in fields like power transmission [91], printers [42], and telephony [90] are often used in technological forecasts. To represent upcoming technological developments, technology roadmaps should be developed and updated alongside product family roadmaps. This is due to the fact that a product family roadmap [92–94] describes how the platform, derivatives, and product family will change in accordance with the technological prediction. The implementation of various technology and product family roadmaps was presented by Lee and Park [94]. Such roadmaps are often created at the module level in product platform design, where each module is carefully examined for potential future development and strategically planned updates over time. Even while individual modules change over time, the system architecture and interfaces of those modules stay the same until the modules change so drastically that a whole new platform is needed [30]. In some situations, the platform may last 15–20 years while the modules are updated once a year, however the module update cycle is usually quicker than the platform redesign period. A product development methodology based on module and product roadmaps was presented by Schuh et al. [51, 95]. A platform-oriented strategy is chosen over a product-oriented approach in their study for the long-term establishment of commonality while developing modules and products under the roadmaps. Other approaches, like MFD [54], take into account the possibility of technological advancement when defining module boundaries. For instance, a modularity rule in MFD specifies that a component or sub-assembly should be divided into a module if it is anticipated to change over time. Customers' preferences are

impacted by product diversity because it gives them more freedom to choose a product in markets with intense competition [96]. The value of a product provides decision-making flexibility for creating new goods or revamping current ones in unpredictable market contexts [97]. Customers' tastes, technological advancements, economic conditions, business strategies, competitive actions, and governmental restrictions are just a few examples of how design may be adjusted to changing surroundings. In order to take advantage of future investment possibilities and appropriately react to market trends in unpredictable situations, strategic flexibility and adaptability are crucial [98]. The opportunity to explore more flexibility and strategic value throughout design is made possible by market-based design methodologies. For instance, Jiao et al. [99] created a valuation framework to assess the configuration possibilities inherent in design and applied genuine alternatives to product family design. In the meanwhile, a stochastic dynamic framework was put out by Gamba and Fusari [100] to value the role that modular operations and the modularization process play in the design of systems employing actual alternatives. We may develop a strategy for changing (or not changing) the modules over time based on the general product design and the existing module sizes for the UGV example. For instance, we may anticipate that mobility technologies (such as the Provide Propulsion and Support Weight modules) will stay largely unchanged in the foreseeable future; as a result, there is no intention to develop these modules on a roadmap. Conversely, battery technology (e.g., Store Power module), computer platform, and embedded controls software (e.g., Controls module) are more likely to change over time. As technology advances, each UGV would need to be redesigned or upgraded. For instance, battery energy per unit weight or cost is always improving, and every two years, higher stored energy expectations may be projected. The point at which it is beneficial to update the battery module to greater power is determined by comparing this to the stored energy levels required to accomplish increasing power loads of enhancing other modules (e.g., bigger powertrain motors, improved imaging cameras, longer range communication, etc.). Technology roadmapping may assist in preventing the platform from impeding the adoption of new technologies.

Step 10: Commonality Assignment

As we discussed earlier, the UGV generic architecture and any individual product architecture is identical, since the modules differ in size but not function. In order to define how many different modules are needed to meet the customer requirements for the UGV, or all products in general, the next step in Figure 1 is a commonality analysis. The maximum number of instances for any module is the number of product variants if every instance needs to be unique to every product variant (e.g., the chassis may be unique to the small, medium, and large UGVs). Conversely, the minimum number of instances for any module is one, i.e., one module that is common on all of the variants (e.g., the same camera on every UGV). Reality usually lies in between those two extremes where modules instances may vary slightly across different subsets of products (e.g., the manipulator on the medium UGV may have two articulating segments while the large UGV may have three).

There has been extensive work in determining commonality. Thevenau and Simpson [101] provide a review of multiple commonality indices. GVI provides a useful approach for commonality assignment. A low GVI value indicates a low chance of redesign within the product family; therefore, a single module may suffice for this component. Meanwhile, a high GVI value indicates a significant chance for redesign. In this latter case, multiple modules will be needed to achieve the range of requirements for the family. For the UGV example, a subsequent analysis of each pair of UGVs (e.g., small and medium, medium and large, and large and small) was conducted to translate the GVI values in Figure 6 to parametric variation needed in each subsystem. Through this analysis we sought to identify potential opportunities for scaling, for example, the chassis in one or more dimensions based on the threshold and objective values for each UGV pair even though the chassis will vary across each weight class. Table 4 summarizes the final recommendations for commonality in key subsystems. Here a grey box indicates common settings across two or more UGVs, e.g., chassis height can be common to all three UGVS, but only the small and medium have common chassis length and width based on the requirements. Performance models of the UGV could be combined with optimization algorithms to determine the best settings for each module and each parameter in the family, using these recommendations as a starting point.

Step 11: Architecture Module Sizing

Sizing of the module alternatives from the previous step is a well-researched area. It includes a host of approaches using numerical methods, clustering, trade-off analysis and optimization techniques to determine module sizes. Simpson, et al. [32], Fujita and Yoshisa [102], Khire et al. [103], Yu, et al. [34], Kumar, et al. [104], Dai and Scott [105], Liu. et al. [106], Hernandez, et al. [107], Chowdhury et al. [108, 109], and Moon, et al. [83] provide example formulations. Han and Papalambros [110] and Kokkolaras, et al. [111] discuss *target cascading*, an approach for sizing modules based on higher level system requirements. Given targets on each system response for each product variant, the shared modules can be sized to identify optimal settings using such numerical methods.
Additionally the functional interface has to be specified for each module variant. The functional module boundaries are compared against the range and sizes of the modules resulting in the performance requirements at the interfaces. These requirements are documented in the functional interface specifications; so, all concerned stakeholders know how the modules must match functionally. This can reduce the design effort and enable the rapid configuration of product variants.
This is repeated for each platform architecture alternative, thereby quantifying their performance (on all variants) and allowing a well determined platform down-selection. In some cases also multiple platforms might be needed to best benefit from platform commonality without sacrificing performance or other aspects. Research on platforming extent [112, 113] aims to answer how many platforms there should be within a given family or products.

Step 12: Architecture Concept Layout
Based on the developed modules and defined variants, the first three-dimensional platform concepts can be designed. In this step, the design of the size and type of the module interfaces is important. It has to be assured that each interface with its dimensions, forces, and media flows is suitable for the range of developed modules.
Krause et al. [114, 115] introduced the Module Interface Graph (MIG) (Figure 10) to visualize and analyze the variety of components and connecting flows (i.e., interfaces) in a product family. Particularly, the MIG also provides the information on the simplified spatial view of the real product. It supports the designing of the first three-dimensional platform concepts and interfaces. The modular product structure of the UGV product family is shown in Figure 10 based on the results of the clustered DSM in Figure 8b. For instance, an electrical, information and structural connection are needed for the interface between the main body and the front arm for tools. Between the main body and the battery, an electrical and a structural connection are necessary.

Step 13: Architecture Downselection
Assuming these works result in more than one optimal solution, trade-offs amongst the results must be considered. Khire, et al. [103] present Pareto frontier solution visualization methods. Gonzalez-Zugasti, et al. [116] extend Pugh's process to compare platforms that support multiple products, each evaluated separately. Saari and Sieberg [117] discuss pairwise comparison methods to improve decision-making with alternatives. Saari [118] discusses hierarchical decisions to improve optimal solutions by introducing a generalized inconsistency theorem that could also be extended for platform selection. Frey et al. [119] show that the concept selection process with multiple stakeholders is forced to be inconsistent with a set of assumptions that each appear reasonable at face value. While all of these concerns raised by these authors are true, this means one or more of the group decision-making assumptions must be violated. In industrial practice, this typically means a senior person (e.g., platform manager, program manager, project manager) assumes responsibility for the decision, and it no longer is a group decision.
For the UGV example, we can create equations of the requirements in terms of module sizing variables. For example, we can describe batteries with energy storage capacity, size, weight, etc. Using such variables, we can derive UGV system-level equations of the UGV top speed, overall mass, climbing angle, etc. Table 5 shows the module sizes for the platform, which UGV variant each is used in, and the overall performance of each UGV variant on several criteria. Note that Table 5 is different for each platform architecture, and a concept selection matrix can be used to rank these platform architectures.

DISCUSSION

Each of the various tasks of platform concept development has been discussed along with the various methods available, as outlined in Figure 1. Given this selection of methods for each of the tasks, it is apparent there is a wide array of permutations, each representing an alternative "design flow" through the matrix linking each set of alternatives. We offer thoughts on selection of several particular platform design flows.

The first design flow to discuss is the baseline, the straight flow-down in the first column of the matrix in Table 3. This approach is the most basic and fastest approach to achieving a platform, though it is also limited in assurance of correctness in terms of necessary future platform redesign iterations to fix any likely product or performance gaps on individual products. Nonetheless, the approach is typical for simple two or three variant product platforms, such as a small medical device product family where two or three distinct products serving distinct procedures and market segments are sought to be made from a common platform. The market segmentation and attack planning (Steps 1 and 2) are often obvious and can be done manually in this simple case. Requirement differences (Step 4) are considered, and these are mapped to a basic systems block diagram (Step 6). The variety and unique requirements help partition the diagram into shared and unique blocks (Step 8). With this, specific modules can be sized (Step 11), and then the platform concept is laid out (Step 12). All other steps, such as the customer needs analysis and consideration of future products, while recommended, is not strictly necessary for a platform redesign. For cases where there is a small portfolio of non-changing products sought to be brought into one platform for cost reduction requirements, this approach may be enough.

A second design flow through the available methods is the Modular Function Deployment (MFD) approach used extensively by Erixon et al. [53], who have documented many industrial applications. For companies new to shared modular platform design, this approach may be the most useful starting point, offering modularity design guidance without significant tool complexity. While the approach incorporates all thirteen steps as deemed needed by any application, the basic approach makes use of five basic steps, each with a specific tool. The basic design flow consists of (1) mapping customer needs to requirements with a Quality Function Deployment (QFD) matrix, (2) generating a system block diagram of components, (3) establishing which components are the most critical and partitioning them using the Modular Function Deployment (MFD) matrix, (4) grouping the components into modules either manually or by using clustering algorithms on the MFD matrix, and (5) generating concept layouts using the suggested clustering.

Several other design flows are possible making use of the Design Structure Matrix (DSM) methods. The range of project complexity and tool chain automation can vary substantially with DSM methods. For example, as a simplified method, in some projects it might be valuable to do a fast rough estimate for potential modules, jumping straight to the final steps early. As we can see in Figure 1 and Table 3, one of the shorter paths to a platform from the baseline approach is to simply build a DSM reflecting a system block diagram directly from off-the-shelf components and link these to customer requirements using QFD, for example, similar to the MFD approach. One could then simply cluster the component DSM to form modules using the available surrogate cost minimization clustering algorithms, and then proceed with the module sizing and layout. This is again most likely suitable for a redesign of an existing family for more efficient variety and commonality ratio, where modules likely vary in type rather than size.

The choice of using a DSM clustering algorithm approach versus an MFD clustering approach is one of choice on available algorithms. The basic MFD approach uses manual clustering, Stake [120] applied multi-objective K-means clustering to align common specifications, and Stake and Blackenfeldt [121], Borjeson and Hölttä-Otto [76] and others combined the multi-objectives of MFD with the DSM clustering algorithms. The span from MFD to DSMs is well researched and marked by the degree of manual versus automated clustering desired.

A more fundamental overhaul of a product platform, or a "clean sheet" design of a new product family, would likely require other combinations of methods. For example, the use of function-based methods can help abstract the problem and can lead to better solutions than simple reconfiguration of components. Function-based methods are used in more complex, hierarchical systems such as building systems including complex HVAC systems and elevator systems [122], or even larger systems including ship building [123] and aircraft [124]. In such systems, the system functionality must be allocated across subsystems. For example, the electrical controls of an elevator might be a functional module that is shared across many sizes of elevators. This module consists of all wires and software connecting the pressed buttons and sensors to the drive motor ultimately. Yet, the button switches must be on each floor and in each cab, whereas the motor must be in the hoistway. This functional module has shared components, yet it also is dispersed

across different subsystems that must be grouped for location reasons. In such hierarchical modular platform designs, there must be both dispersed functional modules (e.g., electrical controls) as well as localized component modules (e.g., a cab module or floor button module). The key difference with systems requiring functional modularity is that the system functionality must be allocated to different component modules, and there are degrees of freedom in how this is accomplished. For example, it is perhaps functionally irrelevant where to locate the computational dispatching algorithms in an elevator system, whether on the local controller moving with cab, on a floor level button controller, on a controller next to the motor controller, or even distributed over all of these, or even none of the above and entirely separate. To consider these questions of complex hierarchical systems, functional block diagrams become necessary along with partitioning of the functionality.

Another lens to consider alternative design flows using the various methods is to consider the level of automation desired. This can span from purely manual methodologies to algorithmic search and trade-offs at each step. For example, Becz [124] presents an automated concept design exploration tool to search over several alternative functional modularity schemes with associated alternative components for the conceptual design of F35 aircraft. Trade-space comparisons of thousands of different modular design alternatives are possible, to establish and explore the Pareto frontier.

Finally, while vastly different in complexity of actions, notice the distinction between the functional and component approaches are overall not that different. Only 4 of 13 steps are necessarily different; much of the early work involving customer needs and specification requirements are similar. Also, the work needed to ascertain the pool of available components and design options to consider is the same. Similarly, the design work on concept layout is the same. Overall, we find the sequence of methods in Table 3 to be logically consistent and supported by what we have found and observed in practice.

SUMMARY AND CONCLUSIONS

In order to connect the many platform research strands into logical sequences that can be used realistically for product platform development, we tried to evaluate the literature in this study. We do not claim that this sequence is the best or even the only one. However, we do assert that it is a suitable place to start since, as Table 2 illustrates, it is in line with real platform development procedures at businesses that often create product platforms. Businesses often have questions about whether or not to follow all of the instructions. We contend that there are dependencies and that it is not always necessary or justified. For example, static/mature industrial markets need more attention to cost models and less technology predictions. The selection of stages to use is often problem-specific and contingent upon the knowledge and resources at hand. Comparing and analyzing the various industrial processes across the columns in Table 2 is an intriguing area for future research. Future research might examine how a company's platforming strategy and practices have changed over time, for instance, in terms of the sequence, quantity, and substance of the different gate evaluations.

REFERENCES

[1] Product Platform and Product Family Design: Methods and Applications, edited by T.W. Simpson, Z. Siddique, and J. Jiao. Springer, New York, 2005.

[2] Product Family Design and Platform-Based Product Development: A State-of-the-Art Review, Jiao, J., T.W. Simpson, and Z. Siddique. Journal of Intelligent Manufacturing, 18(1), pages 5–29, 2007.

[3] Greening the Gate Model, Swanstrom, L. ABB Review, 2009, 2: 23–24.

[4] Leup, P., and C. Rytoft, eds. IEC61850 Special Report. 2010, ABB Review. 62 pages.

[5] Anderstig, S., D. Eklund, and M.T., Development of Production Concept, Mälardalen University, 2012. Mälardalen University, Västerås, Sweden, 2012.

[6] Altfeld, H.-H., Managing the development of very complicated items in commercial aviation projects. Ashgate Publishing, Ltd., 2010.

[7] Vavoski, S., Durable Tooling: An International Sourcing Approach. Cambridge, MA: MIT, 2002.

[8] Hutton, T., Six Sigma against ACE. Cambridge, MA: MIT, 2004.

[9] LeBlanc, A., Using Value Methodologies in Pratt & Whitney Canada's Project Management and Product Development Processes. Value World, 2006, 29(2), pp. 2–7.

[10] Gokal, R., New Product Development: Imagining the Future While Analyzing the Art of the Possible. HTI Cummins Turbo Technology, 14: 7-8 (2010).
[11] Hoffman and Maher, "How Cognition Cockpit Enhances Development Procedures at Cummins, Inc." Cognition Corporation, 2010.
[12] Hauksdottir, D., Vermehren, A., and Savolainen, J. Requirements Reuse at Danfoss. IEEE Requirements Engineering Conference. [13] Kvist, M., Product Family Assessment, IEEE, 2012. Danish Technical University, Copenhagen, Denmark, 2010.
[14] Matzen, D., A Methodical Approach to Service-Oriented Product Development, in Management. Danish Technical University, Copenhagen, Denmark, 2009.
[15] Soderborg, N., Ford's Six Sigma Design. Six Sigma Forum Magazine, pages 15–22, 2004.
[16] Yurinová, Ford's Development Time-Related Cost Reduction System, Institute of Quality Engineering. Slovak University of Technology, 2009.
[17] Bauer, R., et al., The Silverlake Project: Transformation at IBM. Oxford University Press, New York, NY, 1992.
[18] Competitive Dominance: Beyond Strategic Advantage and Quality Management, Tang, V. and R. Bauer. John Wiley & Sons, New York, NY, 1995.
[19] ITT Industries, VBPD: Adding Sure Winners to the Product Pipeline in Our Hands Extra. [20] Reed, T., Integrating Design for Assembly and Manufacturing Into ITT Industries' New Product Development Process (2002, ITT Industries: http://www.itt.com/iohextra/rel13/text-article1.html). Edition 2. DFMA Forum, Newport, RI, 2004.
[21] Kim, S., Software & System Product Line Seminar 2012: KOSTA, Seoul, South Korea, SSPL Strategy in Electronics Industry.
[22] Tegel, D. and R. Kriva, Motorola: Using Six Sigma in New Product Development. PDMA Visions Magazine, 28(4), pages 14–16, 2004.
[23] Churchill, G. and D. Iacobucci, Methodological Foundations of Marketing Research. South-Western College Pub, Cincinnati, OH, 2004, 9th Ed.
[24] Cleveland, M., N. Papadopoulos, and M. Laroche, Consumer behavior, demographics, and identity. 2011 International Marketing Review, 28(3), pp. 244-266.
[25] Moon, S.K., S.R.T. Kumara, and T.W. Simpson. Data Mining and Fuzzy Clustering to Support Product Family Design. ASME Design Engineering Technical Conferences-Design Automation Conference. 2006. ASME, Philadelphia, PA.
[26] Zhang, Y., J. Jiao, and Y. Ma, Market segmentation using fuzzy clustering for product family placement. Engineering Design Journal, 18(3), p. 227-241, 2007.
[27] A market-driven approach to product family creation by D. Kumar, W. Chen, and T.W. Simpson. International Journal of Production Research, 47(1), pp. 71-104, 2009.
[28] Marketing Management, Kotler, P. and K. Keller. Prentice Hall, Upper Saddle River, NJ, 2009.
[29] The Power of Product Platforms: Creating Value and Cost Leadership, Meyer, M.H. and A.P. Lehnerd. The Free Press, New York, 1997.
[30] Suh, E., O. de Weck, and D. Chang, Framework and case study for flexible product platforms. Engineering Design Research, 2007, 18(2), pp. 67-89.
[31] A Decision Support System for market-driven product positioning and design by Lei, N. and S.K. Moon. Systems for Decision Support, 2015. 69: pages 82–91.
[32] Simpson, T.W., et al., An Integrated Approach to Product Family Design: From User Requirements to Commonality Specifications. Engineering Design Research, 2012, 23(2), pp. 141-153.
[33] The Voice of the Customer, Hauser, J., and Griffin, A. Marketing Science, 12(1), pages 1–27, 1993.
[34] Yu, J.S., J.P. Gonzalez-Zugasti, and K.N. Otto, Customer Demand-Based Product Architecture Definition. ASME Journal of Mechanical Design, 121(3), pages 329-335, 1999.
[35] Kano, N., et al., Must-be and attractive qualities. 39–48 in The Journal of the Japanese Society for Quality Control, 14(2), 1984.
[36] Wang, T. and P. Ji, Quantitative investigation of Kano's model for understanding consumer demands. International Journal of Quality & Reliability Management, 27(2), pp. 173-184, 2010.
[37] Schuh, G. and H. Tanner. Using the Variant Mode and Effects Analysis to Master Variant Variety. ASME Design

Engineering Technology Conference, Paper No.: DETC98/DFM-5736. 1998. Atlanta, Georgia.
[38] Harlou, U., Contribution to a theory of product families: Creating product families based on architectures, Department of Mechanical Engineering. Technical University of Denmark, 2006.
[39] S., E., et al. in DESIGN2012, the 12th International Design Conference. 2012. Design Societies, Dubrovnik, Croatia.
[40] Krause, D., et al., Integrated Development of Modular Product Families: A Methods Toolkit, T.W. Simpson, et al., Editors, Advances in Product Family and Product Platform Design. Springer New York, 2014, pp. 245-269.
[41] The House of Quality by J.R. Hauser and D. Clausing. Harvard Business Review, 66(3), pp. 63-73, 1988.
[42] Ulrich, K.T., and S.D. Eppinger, Product Development and Design. Third Edition, ed. McGraw-Hill/Irwin, New York, 2004.
[43] INCOSE, INCOSE Systems Engineering Handbook v3.2. 2010, International Council on Systems Engineering, www.incose.org.
[44] Advances in Product Family and Product Platform Design, Simpson, T.W., et al. Springer-Verlag New York, 2014.
[45]Predicting Change Propagation in Complex Design, Clarkson, P.J., Simons, C., and Eckert, C. 788-797 in Journal of Mechanical Design, 126(5), 2004.
[46] Change and customization in complex engineering domains, Eckert, C., P.J. Clarkson, and W. Zanker. Engineering Design Research, 15(1), pp. 1–21, 2004.
[47] Donaldson, B., Product Family Design Tools for Unmanned Ground Vehicles, Mechanical & Nuclear Engineering. [48] Suh, N.P., The Principles of Design, Pennsylvaina State University, University Park, PA, 2010. Oxford University Press, New York, NY, 1990.
[49] Product Design: Methods in Reverse Engineering and New Product Development, Otto, K.N. and Wood, K.L. Prentice Hall, Upper Saddle River, NJ, 2001.
[50] Jiao, J., and Y. Zhang, Association rule mining-based product portfolio identification. Computer-Aided Design, 37(2), pages 149–172, 2005.
[51] Schuh, G., S. Rudolf, and J. Arnoscht. creation of modular product platforms in an integrated manner. 2010 Proceedings of Technology Management for Global Economic Growth (PICMET), IEEE, 2010.
[52] Engineering Design: A Systematic Approach, G. Pahl and W. Beitz. K. Wallace, ed., 2nd Revised Edition. Springer-Verlag, New York, 1996.
[53] Design for variety: Creating standardized and modularized product platform architectures, Martin, M.V. and K. Ishii. Engineering Design Research 2002, 13(4), p. 213:235.
[54] Controlling Design Variants: Modular Product Platforms, Ericsson, A. and G. Erixon. ASME, New York, 1999.
[55] Luo, X., J. Tang, and C. Kwong, An Optimization Method for a Scalable Product Platform Based on QFD. 141-156 in Engineering Optimization, 42(2), 2010.
[56] Steward, A.D., The Design Structure System: An Approach to Complex System Design Management. IEEE Transaction on Software Engineering, 28(3), pp. 71-74, 1981.
[57] Eppinger, S.D., et al., A Model-Based Approach to Product Development Task Organization. Engineering Design Research, 6(1), 1–13, 1994.
[58] Browning, T.R., Using the Design Structure Matrix for System Integration and Decomposition Issues: An Overview and Future Prospects. IEEE Transactions on Engineering Management, 48(3), pp. 292-306, 2001.