# Chapter 13
# Defining Modules for Platforms: An Overview of the Architecting Process

**Katja Hölttä-Otto, Kevin N. Otto, and Timothy W. Simpson**

**Abstract** Product platforms have shown to provide significant cost and time savings while still allowing companies to offer a variety of products. As a result, a multitude of methods have been developed to design product platforms. These methods, however, have been developed independent of one another, and it can be daunting to try to compare the methods and understand which approach might be suitable when or how the methods might interlink, if at all. In this chapter, we review the platform architecting process and tie together several approaches introduced both in this book and the existing literature. A family of unmanned ground vehicles (UGVs) is used as an illustrative example to demonstrate several of these approaches and their integration.

## 13.1 Introduction

A product platform is a collection of common assets that are shared across a product family, often over product generations. These assets can be anything from a manufacturing line to common system models. A modular platform is a type of platform where the common shared assets are modules. Modules, on the other hand, are typically subassemblies of products that have an easily identifiable function.

K. Hölttä-Otto (✉)
Engineering Product Development, Singapore University of Technology and Design, Singapore 138682, Singapore
e-mail: Katja_Otto@sutd.edu.sg

K.N. Otto
Singapore University of Technology and Design, Singapore, Singapore

T.W. Simpson
Mechanical and Nuclear Engineering, Penn State University, University Park, PA 16802, USA

Industrial and Manufacturing Engineering, Penn State University, University Park, PA 16802, USA

T.W. Simpson et al. (eds.), *Advances in Product Family and Product Platform Design: Methods & Applications*, DOI 10.1007/978-1-4614-7937-6_13,

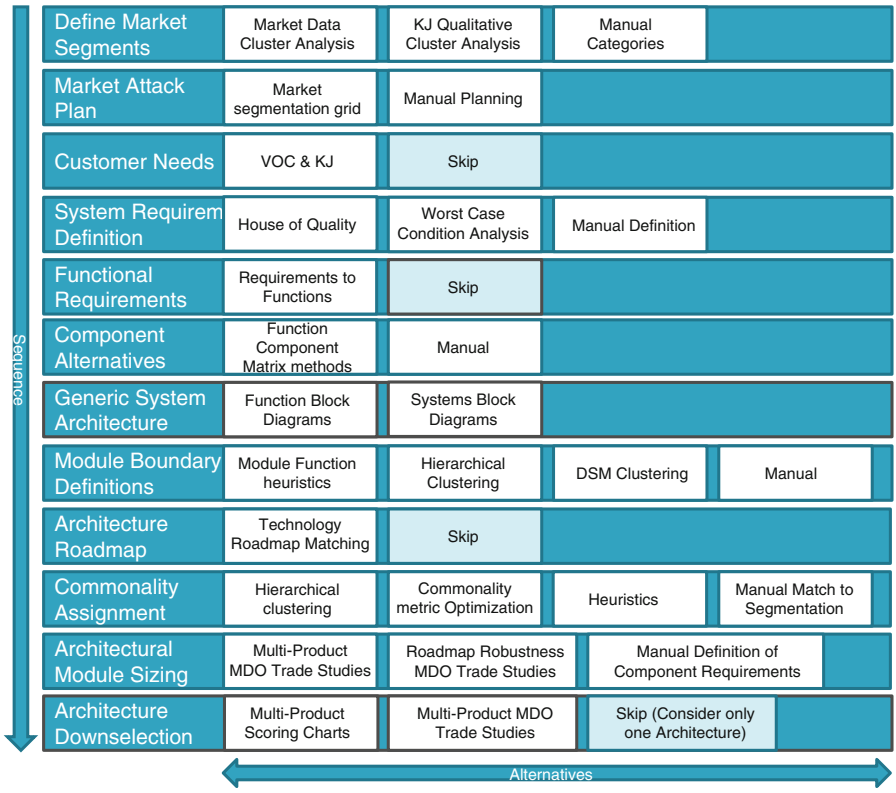


**Fig. 13.1** Architecting steps and possible methods and techniques

Such modules are ideally tightly connected within the module but loosely connected to the rest of the product.

Product platforming has become a common approach to develop product families, and generations of products, in a cost-effective manner. The sharing of assets across the platform products brings cost savings and many operational benefits in manufacturing, logistics, and quality control, for example. Furthermore, since each product is based on a set of common modules, developing additional variants or a new version of past products is easier, since only the variant or updated module will need to be designed.

Over the years there has been active work in developing methods to define these modules for product families: methods to map requirements of a family to a set of products; methods to define the common or unique platform modules; methods to optimize variety, cost, commonality, or other parameters; methods to evaluate platforms; etc. Figure 13.1 shows multiple types of methods and how they can be situated in the overall platform development process from defining customer needs and market segments down to architectural module sizing or architectural down selection. Each method has typically been developed independently of other methods, and it is not clear if and how these methods could be used jointly.

In this chapter we link multiple methods discussed in this book and in the broader literature into a logical, structured process. We do this by presenting a step-by-step approach to transition requirements to a final platform using and referring to relevant methods along the way. The goal in this chapter is to show how the various alternative approaches and methods to platforming can contribute to the overall goal of developing a successful product family.

The rest of this chapter discusses a platform development process in the order shown in Fig. 13.1. This is a prescription for a logical sequence of steps to apply in order, with multiple alternatives at each step as indicated in Fig. 13.1. We do not assert this is the only sequence one can take through these steps; in general, one can find design problems where alternative orders are appropriate. However, Fig. 13.1 is logical and a useful starting point for any firm or stakeholder new to the platform-based development. In each of the next sections, we discuss the various methods suitable for the steps as well as show an illustrative example. We end with a chapter summary and list of implications for platform design.

## 13.2 Market Segment Definition and Market Attack Plan

The start of a product development effort should entail defining the population of customers and applications for the product(s). Typically this starts with an observation of a perceived need in an application domain and extension into exploring the range of different market geographies and demographics that have potential similarities in needs. Market segmentation helps identify potential clusters of customers with similar needs, which enables designing products for each market segment, rather than one product to meet all the vastly different needs.

Initially, one can define a wide range of characteristics upon which to subdivide a population of potential customers, such as applications, geographic boundaries, and demographics. However, one will also find that many such distinctions are artificial and that there is no real difference in customer needs between several over-partitioned divisions of 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. Other options could be use of standard clustering methods, such as hierarchical clustering, or more advanced methods such as fuzzy clustering (Moon et al. 2006; Zhang et al. 2007). While we recommend more rigorous approaches in both this and further steps, a simpler and quicker method is sometimes desired. Regardless of the clustering method chosen, no differences in results can indicate

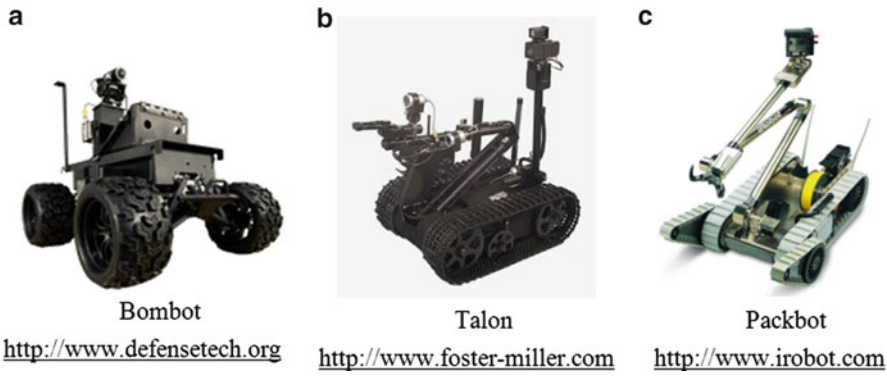


**Fig. 13.2** Examples of existing unmanned ground vehicles

over-partitioning of the market. Unfortunately this is not always obvious, and establishing differences in markets and targeted products is a profitable area of study.

Throughout this chapter, a family of *unmanned ground vehicles* (UGVs) for explosive ordnance disposal is used as an example. UGVs have civilian and military applications, and many systems exist in the market (see Fig. 13.2). Smaller UGVs may provide one-time use to detonate an explosive remotely once it has been located in the field (i.e., the UGV may be destroyed along with the ordnance) while larger UGVs may perform additional functions such as sensing, defusing, or disposing the ordnance, allowing for repeated use and multiple missions.

The primary benefit of UGVs is reducing casualty risk of ordnance disposal regardless of whether it is civilian or military; however, there are many potential applications for UGVs. Partitioning these potential applications can lead to a form of market segmentation. For example, one could partition the UGV user population into the following applications:

1. Explosive abatement—ordnance detection, disabling, and disposal
2. Hazardous sites—hazard sensing and locating in unhealthy environments
3. Combat—providing attack capability for high-risk missions
4. Reconnaissance—providing site observations and intelligence
5. Target and decoy—providing a target that simulates an enemy vehicle
6. 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:

1. Micro, line of sight
2. Small remote operated, 2-mi. range
3. Tactical remote operated, 50-mi. range
4. Long endurance autonomous, 100-mi. range

The resulting matrix of these two categories (application versus range) defines a set of different potential market segments for UGVs. This follows the product

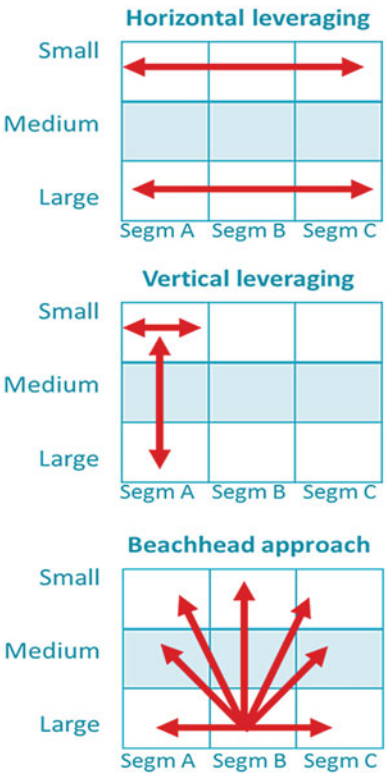


**Fig. 13.3** Platform leveraging strategies [adapted from Meyer and Lehnerd (2000)]

market matrix approach advocated by Kotler and Keller (2009) and Meyer and Lehnerd (2000). In their approach the rows of the matrices correspond to market segments and the columns to product segments, which helps designers identify how platform components may be leveraged across multiple segments (see Fig. 13.3). Several of these segments may be identical in terms of specific customer needs; however, this can be determined later when customer interviews and analyses are conducted.

Given a market segmentation, a decision must be made about how to "attack" the market over time. The corresponding *market attack plan* determines whether new products are offered simultaneously to all segments in parallel or rolled out sequentially over time to different market segments. Key considerations are the addressable market size of each segment and the degree of difference among the segments [see Chaps. 2 and 3 in Simpson et al. (2005)].

To develop the market attack plan for our UGV product family example, we consider the weight, speed, range, and lift capacity requirements for different potential segments within a *product market matrix* defined by applications versus range. 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. (Simpson et al. 2012). The UGV market segmentation matrix was then

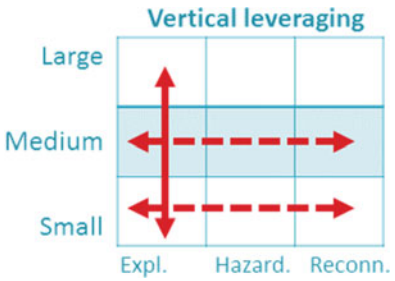


**Fig. 13.4** Planned market strategy for the UGVs

manually clustered into three homogeneous segments to be consistent with current systems in the market. Three "performance tiers" were identified corresponding to small, medium, and large UGVs based on weight. Therefore, the columns are segmented into small, medium, and large UGVs in the market segmentation matrix for the UGV example, and the rows are divided into explosion abatement, hazardous sites, and reconnaissance applications.

Based on this segmentation, a vertical leveraging strategy may be possible as illustrated by the solid vertical arrow in Fig. 13.4. 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 small, medium, and large UGVs. In the future, the platform could be extended by leveraging these modules horizontally to other market segments as shown by the dashed arrows in Fig. 13.4.

We note that one can often associate vertical leveraging to *scalable platforms* where different sizes of the same modules and components are used. Meanwhile, horizontal leveraging can be generally associated with *swappable modules* where a common subset of core modules is reused across different products that are differentiated by integrating the core with swappable unique modules thereby providing different functionality. Successful platforms often utilize a combination of scaling and modularity to attack the different market segments in a strategic and cost-effective manner.

## 13.3 Customer Needs Gathering

Given the list of market segments and a strategy for attack based on the addressable market and their differences, a set of testable, measurable requirements is needed to "design to." The requirements must be based on what the customer seeks in the product. In platform development project, the objective is to create several products for different segments, each with different needs that drive the individual product differences. Therefore, customer needs may be gathered for each individual product separately. In other words the intended variety in the product family is designed first before defining what the common modules in a modular platform should be.

There are well-established methods to gather customer needs such as *voice of the customer* (VOC) (Griffin and Hauser 1993; Churchill and Iacobucci 2004).

Here, interviews are conducted with randomly sampled people from each market segment and questioned over how they use the product. This elicits qualitative and quantitative need statements which they seek from the product.

For the UGV example, the customer requirements were gathered from multiple requirements working groups that met over a period of about 2 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.

For each "segment" in the market segmentation grid in Fig. 13.4, requirements were defined (e.g., weight, range, speed, manipulator length), and threshold and objective values were identified for each requirement. The threshold value is the minimum value that must be met in order to satisfy a requirement while the objective value provides a target that users would like to achieve. Threshold and objective values were defined for the small, medium, and large UGVs, which were used to define the system requirements as discussed next.

## 13.4 System Requirements Definition

Once the customer needs for each market segment and corresponding product variant are clearly defined, the next step is to define quantitative verifiable system requirements for each product variant. Again, there are well-established methods for this step, including the House of Quality (Hauser and Clausing 1988). Another option is to define the worst-case operating conditions and define the system specifications for those cases. Perhaps the simplest option for this step is to convert the customer needs into system requirements similar to target value setting as described by Ulrich and Eppinger (2004) or requirement definition following the INCOSE guidelines (INCOSE 2010). Regardless of what method is used, at the end of this step, quantified system requirements have been identified to which provide design targets to ensure each variant is capable of satisfying the customer needs in its corresponding market segment.

Table 13.1 gives an example of the system requirements that were defined for the UGV example following the customer needs analysis. Threshold and objective values for each requirement are defined for each weight class. This information can be used to help identify common, variant, and unique requirements, i.e., those that are the same (identical) among all three UGVs, those that are similar but vary slightly from one UGV to the next, and those that are unique to a specific UGV. For example, some of the maneuvering, sensing, and communication requirements are the same for each UGV; however, the range and payload requirements vary for each weight class. Finally, the manipulator, reach, drag/roll/push, and large object requirements are unique to each UGV, with no requirements defined for the small UGV when that capability is not present (e.g., large object pickup).

Once a specification list is completed for each product variant, it is a key milestone as it defines what the engineers should design to. Therefore, it should be

**Table 13.1** System requirement areas and corresponding UGV activities and functions

| Requirement | Activities | Functions |
|---|---|---|
| Range (ft) | Travel to desired location | Provide propulsion |
| | Travel home | Store power |
| Slope climb (°) | Travel to desired location | Support weight |
| | Travel home | Support loads |
| Maneuver width (in.) | | (Constraint) |
| On-board volume ($in^3$) | Carry payload | Store payload |
| On-board weight (lb) | | Support loads |
| Drag/roll/push (lb) | Move object | Provide propulsion |
| | Move obstacle | |
| Horizontal reach (in.) | Reach for object | Allow DoF (manip) |
| Vertical high reach (in.) | Reach for object | Allow DoF (manip) |
| Sensing type | Observe surroundings | Sense environment |
| Video vert. high reach (in.) | Capture object | Allow DoF (location) |
| Video horiz. reach (°) | Capture object | Allow DoF (orientation) |
| Large obj pickup length (in.) | Grasp object | Couple sample |
| Large obj pickup width (in.) | Grasp object | Couple sample |
| Large obj pickup height (in.) | Grasp object | Couple sample |
| Lift capacity (lb) | List object | Support loads |
| Tool precision | Use tool | |
| Tool size ($in^3$) | | (Constraint) |
| Tool weight (lb) | | |
| Communication range (ft) | Follow instructions | Transmit/process inform |
| | Send data | Control operation |
| | | Transmit commands/data |
| | | Remote control operation |

checked for various considerations. One important factor is completeness against the customer need list. Each customer need should be covered by an associated requirement—meeting the requirement will ensure satisfaction on the customer need. A second important consideration is conflict management among requirements. Customers will want high performance and low cost. Definition of the target cost and the minimum performance level defines how these conflicting goals are to be simultaneously met by the design team. Such conflicts in the requirements should be highlighted and tracked as risk items, particularly when they cross platform considerations. That is, some product variants are low cost and low performance and might be in the same platform as a high-cost high-performance variant. Platforms across conflicting requirements in this way can be difficult to engineer, and exclusion of one or the other variant from the platform may be warranted.

Next, a function-based approach or a component-based approach can be used to start to develop the platform architecture for the family. To define a platform, the function-based approach takes a more general view to define common functions independent of the form. The component-based approach, on the other hand, makes use of a priori knowledge, however acquired or assumed, of the most relevant components needed. If the function-based approach is used, that approach is followed by mapping function to form, i.e., defining the components as part of

embodiment design. Therefore, in the end both approaches lead to the same result though the function-based approach perhaps considers a broader set of alternatives. In this chapter we introduce and demonstrate both approaches.

## 13.5 Functional Requirements Definition

A *functional requirement* is a requirement to exhibit a certain functional behavior, i.e., to do something. A *constraint* is a nonfunctional requirement such as weight requirement. Defining functional requirements is discussed extensively in Otto and Wood (2001). In this chapter, we follow their suggested process of taking the customer needs and separating those into functions and constraints, then listing the customer activities of the desired system, using those activities to form a functional model, and finally comparing the resulting functional model with the original customer needs. This process results in understanding *what*, as opposed to how, the system should do to achieve the customer needs.

In the interest of space, we have listed the activities next to the corresponding customer requirements in Table 13.1. On a typical mission, a UGV would travel to a desired location, observe and record the surroundings, and collect objects or act on objects as instructed and communicated back home. The supporting functions for these basic customer activities include moving the camera and manipulators up, down, and horizontally. Since this is an existing family of robots, our choice of functions was influenced by the existing configuration of the robot family. This will result in a similarity between the embodiment of this functional model and the design created using the component-based approach. The resulting functional model is shown in Fig. 13.5. Table 13.1 shows how this functional model meets the customer requirements.

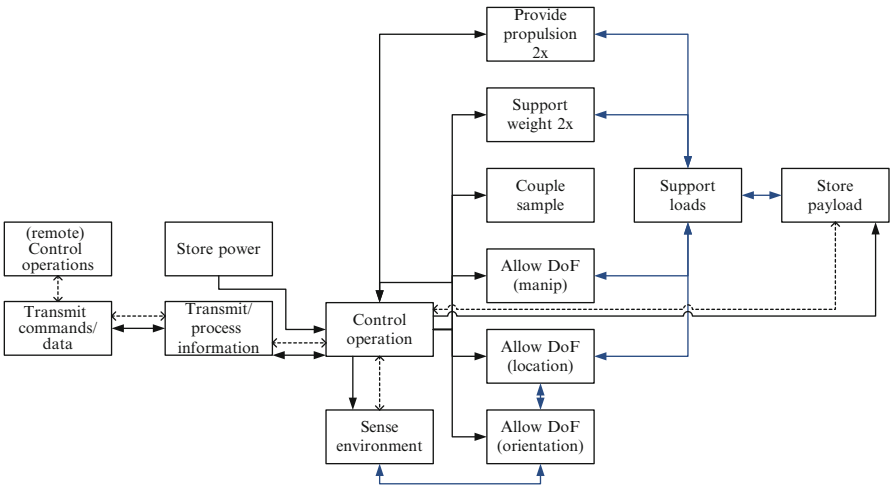


**Fig. 13.5** Functional model of a UGV

## 13.6 Component-Based Approach

Components can be defined either directly from system requirements or by mapping functions, as defined in the previous step, to form part of the system embodiment phase (Pahl and Beitz 1996). There are at least two well-established methods that have incorporated the mapping of requirements to components. In Martin and Ishii's (2000) *Design for Variety* method, they extend the House of Quality to create a function-component matrix for calculating the *Generational Variety Index* (GVI) for each subsystem. A key step in this matrix is to map the engineering requirements into subsystems, or components, in order to identify how much effort is needed to redesign a component if there was a change in a specific requirement. Subsystems with a high GVI value will undergo significant redesign in order to satisfy the range of customer needs while low GVI values indicate components that will remain relatively stable across the family. Figure 13.6 shows the GVI values for the UGV example (Simpson et al. 2012). As seen, the manipulator and chassis will vary substantially within the family (i.e., high GVI values) while cameras and OCU (Operator Control Unit) will have little variation.

| | Subsystem | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Requirement** | Chassis | Battery | Tracks | Communication box | Electronics box | Manipulator | Gripper | Cameras | Payload bay | Antennae | OCU |
| Range (feet) | | | | 6 | | | | | | 6 | |
| Slope Climb (deg) | 3 | | 3 | | | 1 | | | | | |
| Maneuver width (in) | 3 | | 3 | | | | | | | | 3 |
| On board vol (in^3) | 6 | | | | 3 | | | | 6 | | |
| On board wt (lb) | 6 | | | | 3 | | | | 6 | | |
| Drag/Roll/Push (lb) | 6 | 3 | 6 | | | 6 | 6 | | | | |
| Horiz reach (in) | 6 | 3 | | | | 9 | 1 | | | | |
| Vert high reach (in) | 1 | | | | | 9 | 1 | | | | |
| Sensing type | | | | | | | | | 3 | | |
| Video vert high reach (in) | 1 | | | | | 9 | 1 | | | | |
| Large Obj Pickup (length) | | | | | | 3 | 6 | | | | |
| Large Obj Pickup (width) | | | | | | 3 | 6 | | | | |
| Large Obj Pickup (height) | | | | | | 3 | 6 | | | | |
| Lift capac (lb) | 6 | | | | | 9 | 3 | | | | |
| Tool precision | | | | 1 | | | 6 | 1 | | | 1 |
| Tool size (in^3) | | | | | | 3 | 6 | | | | |
| Tool wt (lb) | | | | | | 3 | 6 | | | | |
| Comm range (ft) | | 3 | | 6 | | | | | | 6 | |
| **GVI Values** | **38** | **9** | **12** | **13** | **6** | **58** | **48** | **1** | **15** | **12** | **4** |

**Fig. 13.6** Generational variety index for UGV example (Simpson et al. 2012)

An alternative method is *Modular Function Deployment* (MFD) (Ericsson and Erixon 1999), which creates a similar *Product Property Matrix* to identify platform modules (see Chaps. 4 and 24 for details and examples of MFD). Chapters 14 and 15 discuss QFD approaches to support platform optimization.

If existing products are available for analysis, an alternative approach for creating this matrix is to decompose the products into subsystems and components and then use a DSM, or *Design Structure Matrix* (Steward 1981; Eppinger et al. 1994; Browning 2001) to help identify modules. A component-based DSM for the UGV family is presented next in reference to the defining of a generic system architecture.

## 13.7 Generic System Platform Architecture Definition

A platform forms the base for a set of product variants; thus, the platform architecture should encompass all the variant architectures that intend to support. A simple way to create such a "generic" architecture for the platform is to create an architecture for each of the variants and merge them together. The variant architectures can be created using functional modeling or a component-based approach.

The functional modeling approach has benefit of being intuitive and visual, particularly for more complex block diagrams. In the functional approach (Pahl and Beitz 1996; Otto and Wood 2001), the recommended approach is to build each individual functional model and then merge these models into a generic platform architectural model. In the UGV example, the functional model in Fig. 13.5 is already a generic functional model since the individual UGVs differ in performance levels, not functionality, in the vertically leveraged scalable platform. For example, while each model may have a different size manipulator or camera, 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.

Rather than the functional block diagram approach, another possibility is to work with matrix methods and functions to extend the Generational Variety Index and Modular Function Deployment methods introduced in the previous section. Both methods can be used to map functions to components and thereby define modules, without explicitly defining a generic platform architecture. Module definition is discussed in more detail in the next section.

Finally, one can forgo the functional approach altogether and work directly with components, using the component-based Design Structure Matrix (DSM) approach. This avoids difficulties of thinking functionally but thereby also limits the space of design solutions to the components at hand. For demonstration, this component-based DSM approach is used for the UGV example (see Fig. 13.7). For the UGV example, the DSM is constructed using a teardown disassembly approach to system decomposition (Chiriac et al. 2011). In this case, four UGVs were disassembled;

**Fig. 13.7** Unclustered DSM for the UGV platform

| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Chassis | | | x | x | | | x | x | | | | x | | |
| 2 | Battery | | | | | | | | | | | | | | |
| 3 | Flipper | x | | | | | x | | | | | | | | |
| 4 | Main track | x | | | | | x | | | | | | | | |
| 5 | Communication box | | | | | | x | | | | | | | x | |
| 6 | Electronics box | | x | x | x | x | | x | x | x | x | x | x | | |
| 7 | Manipulator | x | | | | | x | | | | | | | | |
| 8 | Mast | x | | | | | x | | | x | | | | | |
| 9 | Head | | | | | | x | | x | | | x | | | |
| 10 | Gripper | | | | | | x | | | | | | | | |
| 11 | Cameras | | | | | | x | | | x | | | | | |
| 12 | Payload Bay | x | | | | | x | | | | | | | | |
| 13 | Antenna | | | | | x | | | | | | | | | x |
| 14 | OCU | | | | | | | | | | | | | x | |

their subsystems or components were identified and then recorded to create the component-based DSM shown in Fig. 13.7. This DSM represents the "generic" platform architecture for the UGV family, which can now be used for module definition as discussed next.

## 13.8 Module Boundary Definition

There are many methods to define modules (Gershenson et al. 2004; Simpson 2004; Simpson et al. 2006). Some common methods include modularity heuristics (Stone et al. 2000; Zamirowski and Otto 1999), heuristic identification of commonalities using a modularity matrix (Dahmus et al. 2001), Modular Function Deployment (Ericsson and Erixon 1999) to identify modules by importance of components, and finally various numerical heuristic methods that operate on a DSM and cluster its elements into modules (Helmer et al. 2010; Yu et al. 2005; Yu et al. 2007). Most clustering algorithms include a measure to decide when to conclude the clustering, i.e., when the desired degree of modularity is achieved. Thebeau (2001) developed a measure to minimize connections outside modules whereas Yu et al. (2005) use an information theoretic approach to minimize the information needed to describe the connectivity between modules. Excellent reviews of both coupling and similarity modularity can be found elsewhere (Gershenson et al. 2004; Guo and Gershenson 2003). Hierarchical clustering can also be used to define modules, either based on the product requirements, component specifications (Hölttä-Otto et al. 2008), or in conjunction with another method that defines a metric, such as Modular Function Deployment (see Chaps. 4 and 24).

For the UGV example, we continue with both the functional model and the DSM to demonstrate alternative approaches to module boundary definition. First, using the functional model as a starting point, we use the module heuristics (Stone et al. 2000) to identify potential modules in the system architecture. As seen in Fig. 13.8,

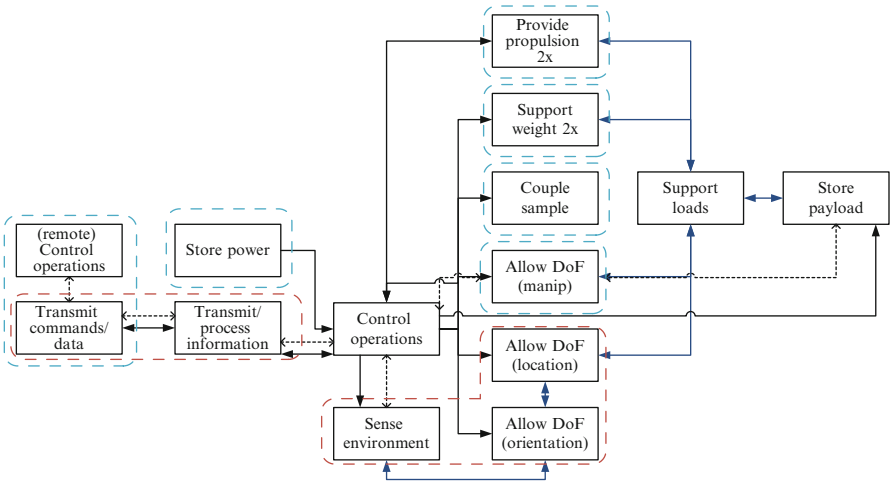


**Fig. 13.8** Module definition using module heuristics

we identify multiple potential modules using the branching flow heuristic (blue dashed lines) as well as a transmission type module and a dominant flow module (both in red). The function structure is at a relatively high abstraction level, and thus the modules, for most part, include only one abstracted function. The modules defined in this approach are sensible and occur in current UGVs on the market. We note that if the system were to be decomposed to a further level of granularity, we would identify additional module candidates—such as all the separate drives (within the "Allow DoF" function) as conversion-transmission modules. We refer the reader to Chap. 9 for more discussion on the level of granularity and its impact on module identification.

Another approach to define module boundaries is to cluster the component DSM developed in Fig. 13.7. There are many clustering algorithms available (see http://www.dsmweb.org), but manual clustering is sufficient for the UGV example. Figure 13.9 shows the clustered DSM for the original UGV architecture in Fig. 13.7. This partitioning results in two bus modules (for the chassis and electronics), two $3 \times 3$ modules (for communications and observation), and multiple modules that consist of a single component (for specific functionality, e.g., manipulator). Again, if the system was decomposed further, a larger DSM with larger modules would be created to identify modules within modules. For this system, one can compare Figs. 13.8 and 13.9 and see there is essentially the same result obtained. Both methods resulted in compounded components into a module, and both methods identified the chassis and electronics as buses (isolated modules with many parallel connections).

**Fig. 13.9** Clustered DSM for the UGV family

| | | 1 | 6 | 2 | 13 | 5 | 14 | 3 | 4 | 7 | 8 | 9 | 11 | 10 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Chassis | | | | | | | x | x | x | x | | | | x |
| 6 | Electronics box | | | x | | x | | x | x | x | x | x | x | x | x |
| 2 | Battery | | | | | | | | | | | | | | |
| 13 | Antenna | | | | | x | x | | | | | | | | |
| 5 | Communication box | | x | | x | | | | | | | | | | |
| 14 | OCU | | | | x | | | | | | | | | | |
| 3 | Flipper | x | x | | | | | | | | | | | | |
| 4 | Main track | x | x | | | | | | | | | | | | |
| 7 | Manipulator | x | x | | | | | | | | | | | | |
| 8 | Mast | x | x | | | | | | | | | x | | | |
| 9 | Head | | x | | | | | | | | x | | x | | |
| 11 | Cameras | | x | | | | | | | | | x | | | |
| 10 | Gripper | | x | | | | | | | | | | | | |
| 12 | Payload Bay | x | x | | | | | | | | | | | | |

## 13.9 Architecture Roadmap

A platform should also support multiple product generations over time. A good way to design for future product variants is to plan for technology evolution. This can be done in several ways. In some methods such as MFD (Ericsson and Erixon 1999), this is taken into the account during module boundary definition. For example, in MFD a modularity rule states that if a component or subassembly is expected to evolve over time, it should be separated into a module. This concept generalizes into the idea of *technology roadmapping* (Albright 2002; Suh et al. 2010). In general, a *technology roadmap* is a plan to develop products in the future using technology that is not yet fully developed. Despite the incompleteness, the past trends allow for future prediction of expected performance. A common technology forecast is Moore's law (Allan et al. 2002; Edenfeld et al. 2004) or similar performance improvement curves in, for example, telecommunication (Willyard and McClees 1987), printers (Ulrich and Eppinger 2004), and power transmission (Daima and Oliverb 2008).

In product platform design, such roadmaps are done at the module level, where each module is scrutinized for future evolution and strategically roadmapped. The system architecture of such modules now and into the future remains fixed, until the modules change so radically an entirely new platform is required (Suh et al. 2010). The module update cycle is usually faster than the platform redesign cycle.

For example for our UGV case, we consider the generic product architecture and the current modules sizes, and we can create a plan for evolving (or not evolving) the modules over time based on this. For example, we expect mobility technology (i.e., provide propulsion, support weight modules) to remain relatively constant in the near future, and thus there is no plan to evolve this module on a roadmap. The battery technology (i.e., store power module) and computing platform and embedded controls software (e.g., controls module), on the other hand, are expected to improve over time, and each UGV will need to be redesigned and upgraded as the

technology evolves. For example, battery energy per unit cost or per unit weight continually improves, and one can project increased stored energy expectations every 2 years. Comparing this against stored energy levels needed to achieve increased power loads of improving other modules (e.g., larger power train motors, increased imaging cameras, longer range communication) defines the point in time where it is useful to upgrade the battery module to higher power. Having such a roadmap will prevent the platform for hindering evolution and improvements as technology changes and new technologies are developed.

## 13.10 Commonality Assignment

The next step is to decide how many sizes of each module are needed, given the number of product variants and the modularity boundaries identified for the components. 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 module 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).

The Generational Variety Index (see Sect. 13.6) provides a starting point for commonality assignment. Low GVI values will not require much redesign within the product family; therefore, a single module may suffice for this component. Meanwhile, high GVI values require significant redesign indicating that 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 recommendations in Fig. 13.6 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. The final recommendations for commonality in key subsystems are listed in Table 13.2 where a “c” indicates where common settings may be used 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 threshold and objective requirements.

In the approach outlined here, the determination of how many module sizes are needed is separated from the task of actually sizing each module. In Table 13.2, different sizes are shown for each of the product variants (small, medium, large), though the actual sizes of the modules is not yet determined. All that has been defined is the commonality. Further, for each architecture alternative, the allocation of Table 13.2 will change. Choice of module boundaries affects the choices of

**Table 13.2** Recommendations for subsystem commonality (Simpson et al. 2012)

| Subsystem | Design parameters | Small | Medium | Large |
|---|---|---|---|---|
| Chassis | Length | c | c | |
| | Width | c | c | |
| | Height | c | c | c |
| Mobility | Wheels/tracks | c | c | c |
| | Wheel/track diameter | | | |
| | Wheel/track width | | | |
| Batteries | Length | c | c | c |
| | Width | c | c | c |
| | Mass | c | c | c |
| Manipulators | Outer arm radius | c | c | |
| | Arm segment length | c | c | |
| | Number of links | c | c | |

number of module sizes. Mathematical models of the UGV performance 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. This is discussed in the next section; meanwhile, in Chap. 18 Khire et al. discuss how optimization can be used for commonality selection in more detail, and Simpson et al. (2012) provide additional details on the UGV example.

## 13.11 Architectural Module Sizing and Down Selection

At this point, there is a small set of alternative architecture platform concepts, each complete with a modularity scheme to implement the set of product variants. However, the modules have not been sized nor assessed for performance capability to correctly operate in each supported product variant. Since the module sizes are not yet known, the performance of each product variant as instantiated in each platform concept is not known. These must be computed or estimated before a down selection can be made amongst the platform concepts.

To do this, we now 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. We term the module variables with $x$, UGV system responses with $y$, and the equations with $f$.

Given targets on each system responses $y$ for each product variant, the shared modules can be sized for best sizes $x$. This is repeated for each platform architecture alternative, thereby quantifying their performance (on all variants) and allowing a well determined platform down selection.

**Table 13.3** Module sizes for a UGV family (Simpson et al. 2012)

| Subsystem | Design parameters | Small | Medium | Large |
|---|---|---|---|---|
| Chassis | Length | 0.56 | 0.59 | 0.66 |
| | Width | 0.23 | 0.22 | 0.30 |
| | Height | 0.32 | 0.33 | 0.34 |
| Mobility | Wheels/tracks | Tracks | Tracks | Tracks |
| | Wheel/track diameter | 0.26 | 0.26 | 0.26 |
| | Wheel/track width | 0.03 | 0.03 | 0.13 |
| Batteries | Length | 0.11 | 0.11 | 0.11 |
| | Width | 0.06 | 0.06 | 0.06 |
| | Mass | 1.40 | 1.40 | 1.40 |
| Manipulators | Outer arm radius | 0.02 | 0.02 | 0.02 |
| | Arm segment length | 0.57 | 0.52 | 0.31 |
| | Number of links | 3 | 3 | 3 |

For the UGV example, Table 13.3 shows the module sizes for one instance of the platform to achieve at least 80 % effectiveness across all of the requirements in each market segment. Table 13.3 is different for each alternative platform architecture, and a concept selection matrix can be used to rank each. The actual sizing of modules is, however, generally a nontrivial analysis. Typically, trade-off analyses are done and visualized for the non-dominated combinations of module variables that are for the Pareto frontier. Amongst this set, combinations can be sorted using judgment or a value-based gauge of the performance criteria. Methods to accomplish this are discussed by Khire et al. in Chap. 18, and more details on the trade-offs in the UGV family are discussed by Simpson et al. (2012).

## 13.12 Summary

Methods and tools to support product platform design have advanced remarkably over the past decade, and there is no single path to architecting a product platform. Some methods are more suitable for an approximate approach to get fast results, while others are better suited for more sophisticated analysis; however, most fall somewhere in between these two extremes.

Looking back at Fig. 13.1, we can see how one of the shortest paths to a platform is to build a DSM directly from components of existing products or from customer requirements using quality function deployment, for example, and then simply clustering the DSM to form modules. This is most likely suitable for a redesign of an existing family for more efficient variety and commonality ratio. For a more fundamental overhaul of a product platform, or a clean sheet design of a new product platform, other combinations of methods are likely more productive. Use of function-based methods helps abstract the problem and can lead to better solutions than simple reconfiguration of components.

Overall, we do not recommend or promote any one particular method over another nor even espouse the sequence through the methods as shown. Rather our goal has been to provide a reference for linking all of the different modular product platform development methods out there and in this book, along with where to find more information on these methods.

## References

Albright RE (2002) Roadmapping for global platform products: product development and management association. Visions Magazine 26(4):19–22

Allan A, Edenfeld D, Joyner WJ, Kahng A, Rodgers M, Zorian Y (2002) 2001 technology roadmap for semiconductors. Computer 35(1):42–53

Browning TR (2001) Applying the design structure matrix to system decomposition and integration problems: a review and new directions. IEEE Trans Eng Manag 48(3):292–306

Chiriac N, Hölttä-Otto K, Suh E, Lysy D (2011) Three approaches to complex system decomposition. In: 13th international dependency and structure modelling conference, DSM'11, Cambridge, MA

Churchill GA, Iacobucci D (2004) Marketing research: methodological foundations, 9th edn. South-Western College Publishing, ISBN: 0324201605

Dahmus JB, Gonzalez-Zugasti JP, Otto KN (2001) Modular product architecture. Des Stud 22(5):409–424

Daima TU, Oliverb T (2008) Implementing technology roadmap process in the energy services sector: a case study of a government agency. Technol Forecast Soc Change 75(5):687–720

Edenfeld D, Kahng A, Rodgers M, Zorian Y (2004) 2003 technology roadmap for semiconductors. Computer 37(1):47–56

Eppinger S, Whitney D, Smith R, Gebala D (1994) A model-based method for organizing tasks in product development. Res Eng Des 6(1):1–13

Ericsson A, Erixon G (1999) Controlling design variants: modular product platforms. ASME Press, New York

Gershenson JK, Prasad JK, Zhang Y (2004) Product modularity: measures and design methods. J Eng Des 15(1):33–51

Guo F, Gershenson JK (2003) Comparison of modular measurement methods based on consistency analysis and sensitivity analysis. In: ASME 2003 design engineering technical conferences, Chicago, IL. ASME, Paper No. DETC2003/DTM-48634

Griffin A, Hauser JR (1993) The voice of the customer. Market Sci 12(1):1–27

Hauser JR, Clausing D (1988) The house of quality. Harv Bus Rev 66(3):63–73

Helmer R, Yassine A, Meier C (2010) Systematic module and interface definition using component design structure matrix. J Eng Des 21(6):647–675

Hölttä-Otto K, Tang V, Otto K (2008) Analyzing module commonality for platform design using dendrograms. Res Eng Des 19(2):127–141

INCOSE (2010) INCOSE systems engineering handbook v3.2. International Council on Systems Engineering. http://www.incose.org/

Kotler P, Keller KL (2009) Marketing management. Prentice Hall, Upper Saddle River

Martin M, Ishii K (2000) Design for variety: developing standardized and modularized product platform architecture. Res Eng Des 13(4):213–235

Meyer MH, Lehnerd AP (2000) The power of product platforms. The Free Press, New York, NY

Moon SK, Kumara SR, Simpson TW (2006) Data mining and fuzzy clustering to support product family design. In: ASME design engineering technical conferences – design automation conference, Philadelphia, PA. ASME, Paper No. DETC2006/DAC-99287

Otto K, Wood K (2001) Product design: techniques in reverse engineering, systematic design, and new product development. Prentice-Hall, New York, NY

Pahl G, Beitz W (1996) Engineering design: a systematic approach. Springer, New York, NY

Simpson TW (2004) Product platform design and customization: status and promise. Artif Intell Eng Des Anal Manuf 10(1):3–20

Simpson TW, Siddique Z, Jiao J (2005) Product platform and product family design: methods and applications. Springer, New York, NY

Simpson TW, Marion T, de Weck O, Holtta-Otto K, Shooter SB (2006) Platform-based design and development: current trends and needs in industry. In: ASME 2006 international design engineering technical conferences, Philadelphia, PA. Paper No. DETC2006/DAC-99229

Simpson TW, Brennan S, Slingerland LA, Bobuk A, Logan D, Reichard K (2012) From user requirements to commonality specifications: an integrated approach to product family design. Res Eng Des 23(2):141–153

Steward DT (1981) The design structure system: a method for managing the design of complex systems. IEEE Trans Eng Manag 28(3):71–74

Stone RB, Wood KL, Crawford RH (2000) A heuristic method for identifying modules in product architectures. Des Stud 21(1):5–31

Suh ES, Furst MR, Mihalyov KJ, de Weck O (2010) Technology infusion for complex systems: a framework and case Study. Syst Eng 13(2):186–203

Thebeau RE (2001) Knowledge management of system interfaces and interactions for product development process. System design & management program. M.S. Thesis, Massachusetts Institute of Technology, Cambridge, MA

Ulrich KT, Eppinger SD (2004) Product design and development, 3rd edn. McGrw-Hill, New York, NY

Willyard CH, McClees CW (1987) Motorola's technology roadmap process. Res Manag 30(5):13–19

Yu TL, Yassine A, Goldberg DE (2005) An information theoretic method for developing modular architectures using genetic algorithms. University of Illinois, Department of General Engineering. Urbana-Champlain: Illinois Genetic Algorithms Laboratory IlliGAL

Yu TL, Yassine AA, Goldberg DE (2007) An information theoretic method for developing modular architectures using genetic algorithms. Res Eng Des 18(2):91–109

Zamirowski EJ, Otto KN (1999) Identifying product family architecture modularity using function and variety heuristics. In: ASME design engineering technical conferences – 11th international conference on design theory and methodology, Las Vegas, NV. ASME, Paper No. DETC99/DTM-8760

Zhang Y, Jiao J, Ma Y (2007) Market segmentation for product family positioning based on fuzzy clustering. J Eng Des 18(3):227–241