# Chapter 10
# Modular Design

**Sa'Ed M. Salhieh [1] and Ali K. Kamrani [2]**

**Abstract** Modular design aims to subdivide a complex product into smaller parts (modules) that are easily used interchangeably. Examples of modularly designed items are vehicles, computers, and high-rise buildings. Modular design is an attempt at getting both the gains of standardization (high volume normally equals low manufacturing costs) and the gains of customization. The concept of modularity can provide the necessary foundation for organizations to design products that can respond rapidly to market needs and allow the changes in product design to happen in a cost-effective manner. Modularity can be applied to the design processes to build modular products and modular manufacturing processes.

## 10.1 Modularity

Modularity aims to identify the independent, standardized, or interchangeable units to satisfy a variety of functions. Modularity can be applied in the areas of product design, design problems, production systems, or all three. It is preferable to use modular design in all three types at the same time; this can be done by using a modular design process to design modular products and to produce them using a modular production system or modular manufacturing processes.

### *10.1.1 Modularity in Products*

Modular products are products that fulfill various overall functions through the combination of distinct building blocks or modules, in the sense that the overall function performed by the product can be divided into subfunctions implemented

[1] Assistant Professor, Department of Industrial Engineering, University of Jordan, Amman, 11942, Jordan

[2] Associate Professor, Department of Industrial Engineering, University of Houston Houston, Texas 77004, USA

A.K. Kamrani, E.S. Abouel Nasr (eds.) *Collaborative Engineering: Theory and Practice,*
doi: 10.2007/ 978-0-387-47321-5, 

by different modules or components [13]. Product modularity has been analyzed as a form of product architecture that allows a one-to-one correspondence between physical structures and functional structures, as opposed to integral architectures where the functional elements map to a single or very small number of physical elements [15]. Hand tools are considered a good example of integral products, where several functions are mapped to a single physical structure, that is, the tool itself. A personal computer exemplifies a modular product in which a wide range of functions are fulfilled by utilizing a wide range of interchangeable physical structures such as hard drives, CD-ROMs, and motherboards.

The term modularity in products is used to describe the use of common units to create product variants. That is, modularity in products is based on the idea that a complex product could be decomposed into a set of independent components. This decomposition allows the standardization of components and the creation of product variants. Components used to create modular products have functional, spatial, and other interface characteristics that fall within the range of variations allowed by the specified standardized interfaces of a modular product. Mixing and matching different modular components creates a large number of modular products, where each product would have a distinct combination of the modular components, resulting in the creation of products with distinctive functionalities, features, and performance levels.

### *10.1.2 Modularity in Design Problems*

Most design problems can be broken down into a set of easy-to-manage simpler subproblems. Sometimes complex problems are reduced into easier subproblems, where a small change in the solution of one subproblem can lead to a change in other subproblems' solutions. This means that the decomposition has resulted in functionally dependent subproblems. Modularity focuses on decomposing the overall problem into functionally independent subproblems, in which interaction or interdependence between subproblems is minimized. Thus, a change in the solution of one problem may lead to a minor modification in other problems, or it may have no effect on other subproblems.

### *10.1.3 Modularity in Production Systems*

Modularity in production systems aims at building production systems from standardized modular machines. The fact that a wide diversity of production requirements exists has led to the introduction of a variety of production machinery and a lack of agreement on what the building blocks should be. This means that there are no standards for modular machinery. In order to build a modular production system, production machinery must be classified into functional groups from which a selection of a modular production system can be made to respond to different

production requirements. Rogers [11] classifies production machinery into four basic groups of "primitive" production elements. These are process machine primitives, motion units, modular fixtures, and configurable control units. It is argued that if a selection is made from these four categories, it will be possible to build a diverse range of efficient, automated, and integrated production systems.

## 10.2 Modular Systems Characteristics

### *10.2.1 Modules Types*

Modular systems are built from independent units or modules. Two major categories of modules are identified, namely, *function modules* and *production modules* [12]. Function modules are designed to accomplish technical functions independently or in combination with other modules. Production modules are designed on the basis of production considerations alone and are independent of their function. Function modules can be classified on the basis of various types of functions reoccurring in a modular system that can be combined as subfunctions to implement the different overall function (Fig. 10.1). These functions are basic, auxiliary, special, adaptive, and customer specific [9].

- *Basic Functions.* These are functions that can fulfill the overall function simply or in combination with other functions. Basic functions are not variable in principle and they are implemented in *basic modules*.

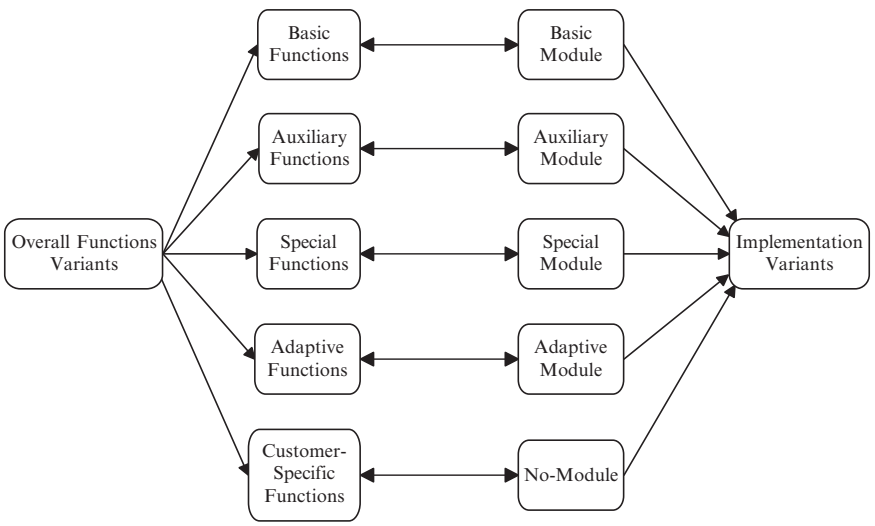


**Fig. 10.1** Function and module types

- *Auxiliary Functions.* These are implemented using auxiliary modules in accordance with basic modules.
- *Special Functions.* These are task-specific subfunctions that may not appear in all overall function variants and are implemented by *special modules*.
- *Adaptive Functions.* These are the functions that permit the adaptation of a part or a system to other products or systems. They are implemented by *adaptive modules* that allow for unpredictable circumstances.
- *Customer-Specific Functions.* These are functions that are not provided by the modular system, and they are implemented by *non-modules* which must be designed individually. If they are used, the result is a mixed system that combines modules and non-modules.

## *10.2.2 Modularity Types*

Product modularity depends on the similarity between the physical and the functional architecture of a design, and on the minimization of the incidental interactions between the physical components that comprise the modules. The nature of the interactions between the modules has been used to categorize product modularity into two major categories of modularity [6, 15, 16]:

- Function-based modularity is used to partition the functionalities of a product and describe how these functions are distributed.
- Manufacturing-based modularity relates to the manufacturing processes and the assembly operations associated with a product.

### 10.2.2.1 Function-Based Modularity

Four classifications of function-based modularity are defined as follows.

1. *Component-Swapping Modularity:* Different product variants belonging to the same product family are created by combining two or more alternative types of components with the same basic component or product. Figure 10.2 illustrates the swapping modularity in which two alternative components (the small rectangular block and the triangular) are combined with the same basic component (the big block), forming product variants belonging to the same product family.

An example of component-swapping modularity in the computer industry is illustrated by matching different types of CD-ROMs, monitors, and keyboards with the same motherboard. This allows for different models of computers to be implemented.

2. *Component-Sharing Modularity*: In this category, different product variants belonging to different product families are created by combining different modules sharing the same basic component. Component-sharing is considered

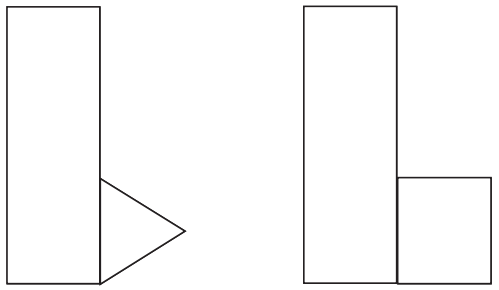

**Fig. 10.2** Component-swapping modularity

the complementary case to component-swapping. Component-sharing and component-swapping modularity are identical except that swapping involves the same basic product using different components and sharing involves different basic products using the same component. The difference between them lies in how the basic product and components are defined in a particular situation. Figure 10.3 shows two different basic components (the block and the triangular) sharing the same component (the circle). Component-sharing modularity in the computer industry is represented by the use of the same power cord, monitor, or microprocessor in different product (computer) families.

3. *Fabricate-to-Fit Modularity:* One or more standard components are used with one or more infinitely variable additional components. Variation is usually associated with physical dimensions that can be modified. Figure 10.4 illustrates a component with variable length (the block) that can be combined with two standard components (the triangular) forming product variants. A common example of this kind of modularity is cable assemblies in which two standard connectors can be used with an arbitrary length of cable.
4. *Bus Modularity:* This type of modularity occurs when a module can be matched with any number of basic components. Bus modularity allows the number and location of basic components in a product to vary. Bus modularity is illustrated in Fig. 10.5. An example of bus modularity is a computer where different input and output units, in addition to different types of mice, RAMs, and hard drives, can exist and vary in both their location and their number.

#### 10.2.2.2 Manufacturing-Based Modularity

Four manufacturing-based modularity classes can be defined as follows.

1. OEM (Original Equipment Manufacturer) modules are group of components that are grouped together because a supplier can provide them at a less expense than if they were to be developed in-house. For example, tires in cars are OEM modules.
2. Assembly modules are groups of components that are grouped together because they solve related functions and are bundled together to ease assembly. For

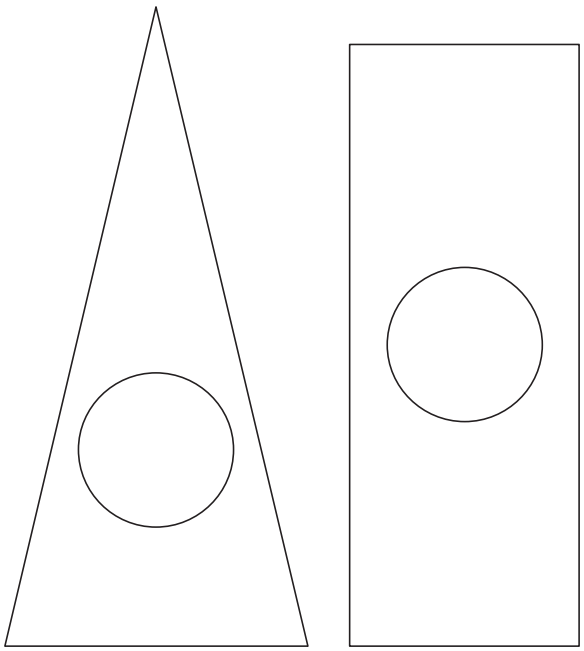

**Fig. 10.3** Component-sharing modularity

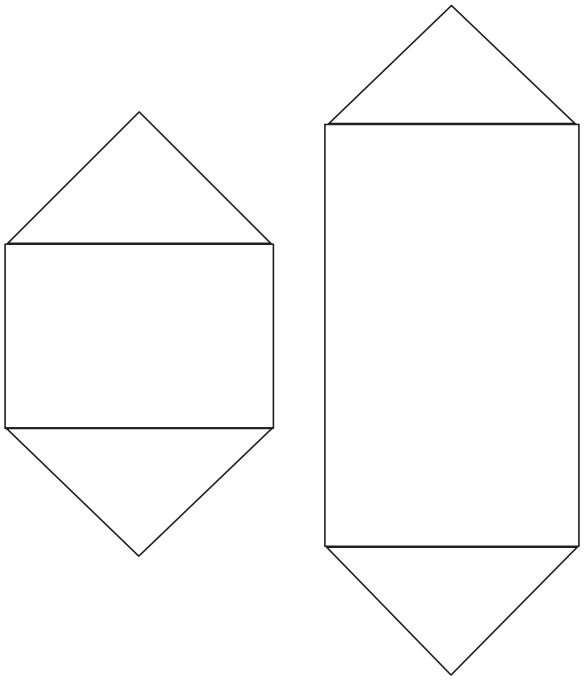

**Fig. 10.4** Fabricate-to-fit modularity

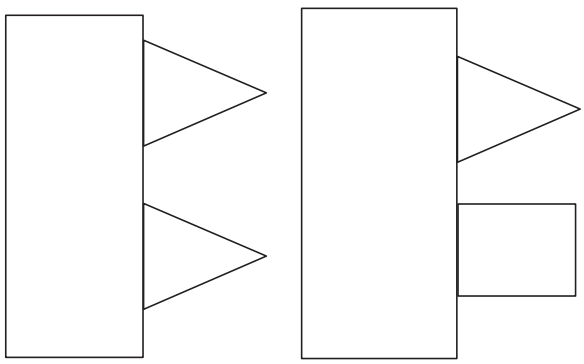

**Fig. 10.5** Bus modularity

example, the dial buttons and the associated electronic circuit in a telephone are all bundled together as a subassembly when making telephones.

3. Sizable modules are components that are exactly the same except for their physical scale. Sizable modules are manufactured using the same exact operations and machine. Lawn mower blades are an example of sizeable modules.
4. Conceptual modules are modules that deliver the same functions but have different physical embodiments. Conceptual modules can lead to a significant change in the manufacturing operations without affecting the functionality of the product. For example, designers may use a gearbox to reduce the speed delivered from a motor to a pump, or they could use a chain-sprocket system to deliver the same function.

## 10.3 Modular Systems Development

In general, modular systems can be developed by decomposing a system into its basic functional elements, mapping these elements into basic physical components, then integrating the basic components into a modular system capable of achieving the intended functions. This approach faces two important challenges [10]: (1) Decomposition: Finding the most suitable set of subproblems may be difficult. (2) Integration: Combining the separate subsystems into an overall solution may also be difficult. To fully comprehend the underlying foundations of modular systems development, decomposition categories are further discussed.

### *10.3.1 Decomposition Categories*

System decomposition is expected to result in two benefits [10]: (1) Simplification: Decomposing large systems into smaller ones will lead to a reduction in the size of

the problem that needs to be solved, which will make it easier to manage. (2) Speed: Solving smaller problems concurrently (parallel solutions) will reduce the time needed to solve the overall problem. Decomposition methods can be categorized according to the area into which they are being applied, namely, product decomposition, problem decomposition, and process decomposition [*8*].

#### 10.3.1.1 Product Decomposition

Product decomposition can be performed at various stages of the design process and can be defined as the process of breaking the product down into physical elements from which a complete description of the product can be obtained. Two approaches are used in product decomposition, *product modularity* and *structural decomposition*.

1. *Product Modularity*

   Product modularity is the identification of independent physical components that can be designed concurrently or replaced by predesigned components that have similar functional and physical characteristics. Product modularity relies on the lack of dependency between the physical components. The computer industry provides an excellent example of modular products, where the major components of the computer are manufactured by many different suppliers allowing the manufacturers of microprocessors to choose from a wide library of products.

2. *Structural Decomposition*

   The system is decomposed into subsystems, and those are further decomposed into components leading to products, assemblies, subassemblies, and parts at the detailed design stage. The decomposition is represented in a hierarchy structure that captures the dependencies between subsystems.

#### 10.3.1.2 Problem Decomposition

For centuries, complex design problems were handled by breaking them into simpler, easy-to-handle subproblems. Problem decomposition should continue until basic independent products or units are reached. The interaction between the basic products should be identified and introduced as constraints imposed by higher subproblems. Problem decomposition is divided into *requirements decomposition*, *constraint–parameter decomposition*, and *decomposition-based design optimization*.

1. *Requirements Decomposition*

   Requirements represent an abstraction of the design problem, starting with the overall requirement (general demand) and ending with the specific requirements (specific demands). The ability to meet a requirement is given by a design function. The requirements decompositions and their relationships to the corresponding functions are represented in a tree diagram (Fig. 10.6), where specific requirements are mapped into specific functions.

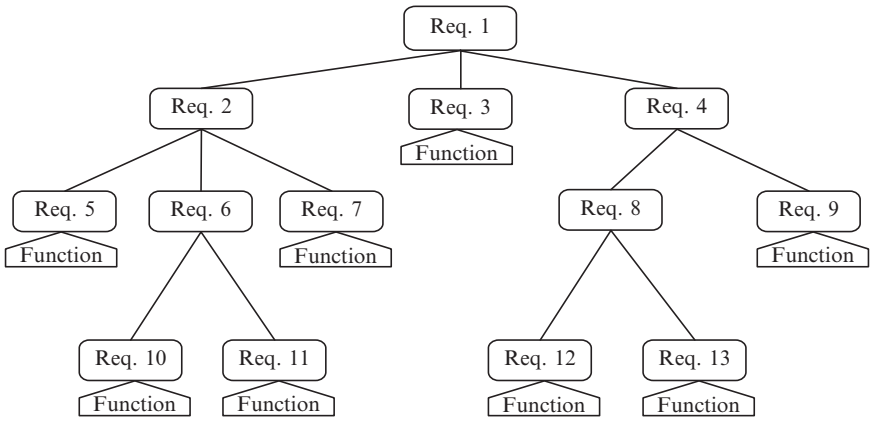


**Fig. 10.6** Requirements decomposition

2. *Constraint–Parameter Decomposition*

The parameters describe the features (quantitative or qualitative data) of the product, while the constraints define the ranges of values assigned to parameters that are defined by product requirements. The problem structure is represented in an incidence matrix [8]. The incidence matrix is decomposed by grouping all nonempty elements in blocks at the diagonal. It is preferable that the blocks be mutually separable (independent). In some cases, overlapping between variables or constraints may occur.

The design of a ball bearing is used to illustrate the decomposition [8]. The parameters are listed in Table 10.1 and the constraints are shown in Table 10.2. The constraint–parameter incidence matrix is shown in Fig. 10.7. The decomposed matrix is shown in Fig. 10.8.

3. *Decomposition-Based Design Optimization*

The decomposition of a large complex design problem into smaller independent subproblems facilitates the use of mathematical programming techniques to solve and optimize the subproblems [3, 4]. The solutions are integrated to provide an overall solution. The objective is to decompose a complex system into multilevel subsystems in a hierarchical form (Fig. 10.9), in which a higher-level subsystem controls or coordinates the subsystems at the lower level. The subsystems are solved independently at the lower level. The objective at the higher level is to coordinate the action of the first level to ensure that the overall solution is obtained.

#### 10.3.1.3 Process Decomposition

Process decomposition is the decomposition of the entire design process, starting with the need recognition and ending with the detail design. The activities in the design process are modeled in a generic manner independent of the specific product being

**Table 10.1** Ball bearing design parameters

| Parameter | Description | Parameter | Description |
|---|---|---|---|
| $d_e$ | Pitch diameter | $\beta_l$ | Free contact angle |
| $d_o$ | Outer-race diameter | $r_o$ | Outer-race curvature |
| $d_i$ | Inner-race diameter | $r_i$ | Inner-race curvature |
| $P_d$ | Diametral clearance | $P_e$ | Free endplay |
| $d$ | Rolling-element diameter | $s$ | Shoulder height |
| $I$ | Race conformity ratio | $\theta$ | Shoulder angle height |
| $r$ | Race curvature radius | $R$ | Curvature sum |
| $B$ | Total conformity | $R_x$ | $x$ direction effective radius |
| $I_o$ | Outer-race conformity | $R_y$ | $y$ direction effective radius |
| $I_i$ | Inner-race conformity | $\Gamma$ | Curvature difference |
| $D$ | Race curvature distance | $\beta$ | Contact angle |

**Table 10.2** Ball bearing design constraints

| | | | |
|---|---|---|---|
| $C_1$ | $d_e = \frac{1}{2}(d_o + d_i)$ | $C_7$ | $P_e = 2DSin\,\beta_f$ |
| $C_2$ | $P_d = d_o - d_i - 2d$ | $C_8$ | $S = r(1-\cos\theta)$ |
| $C_3$ | $f = \frac{r}{d}$ | $C_9$ | $\frac{1}{R} = \frac{1}{R_x} + \frac{1}{R_y}$ |
| $C_4$ | $B = f_o + f_i - 1$ | $C_{10}$ | $\Gamma = R = \left(\frac{1}{R_x} - \frac{1}{R_y}\right)$ |
| $C_5$ | $D = Bd$ | $C_{11}$ | $R_x = d(d_e - d\cos\beta)/2d_e$ |
| $C_6$ | $\beta_f = \arccos\frac{r_o + r_i - \frac{1}{2}(d_o - d_i)}{r_o + r_i - d}$ | $C_{12}$ | $R_y = \frac{f_i d}{(2f_i - 1)}$ |

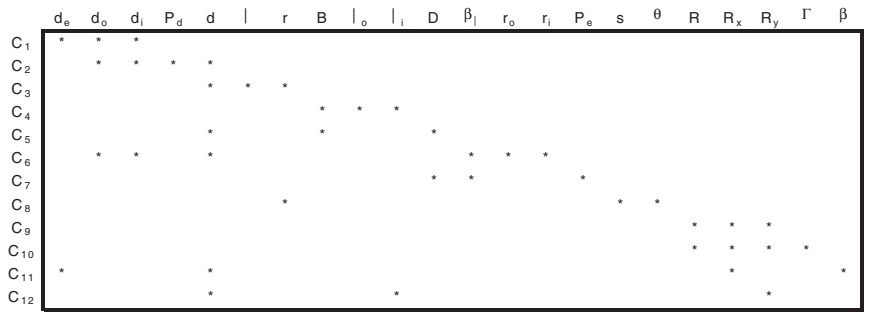


**Fig. 10.7** Ball bearing design constraint–parameter incidence matrix

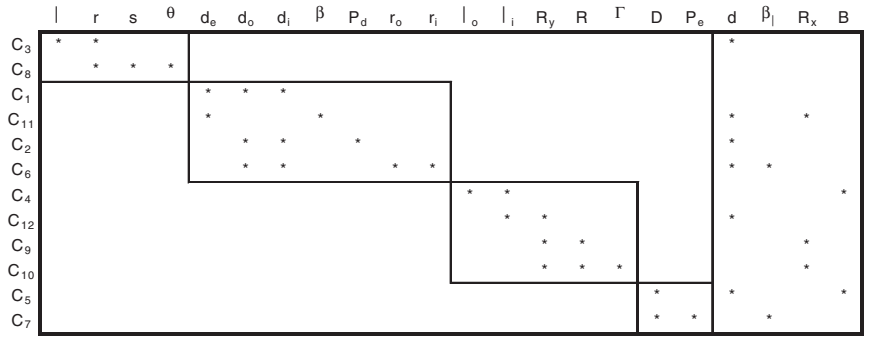


**Fig. 10.8** Decomposed constraint–parameter incidence matrix

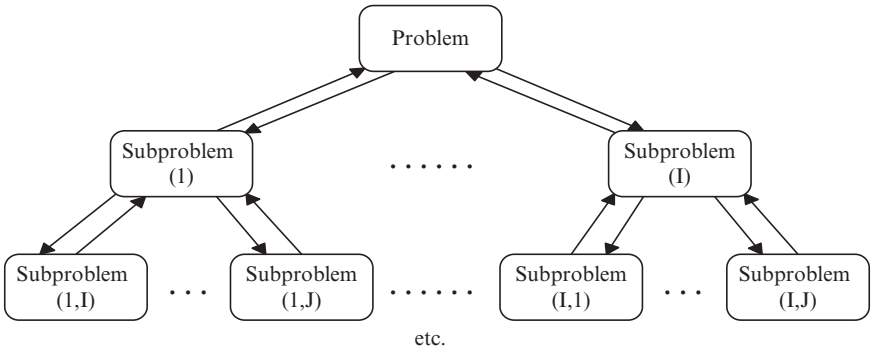


**Fig. 10.9** Hierarchical decomposition of a complex system

designed. Three perspectives of process decomposition were recognized. These are *product flow perspective*, *information flow perspective*, and *resource perspective*.

1. *Product Flow Perspective*

   Design activities required to translate customer requirements into a detailed design of products are the focus of this perspective. The design activities are modeled as blocks with identified inputs and outputs (the output of one activity becomes the input of another activity). The decomposition tries to eliminate redundant activities and reorganize other activities to be performed concurrently, which will eventually reduce the product development time.

2. *Information Flow Perspective*

   Analysis of the precedence constraints between the design activities is the main concern of this perspective. Precedence constraints are utilized to generate the required information needed to build supporting databases and communication networks and to schedule design activities, all concurrently.

3. *Resource Perspective*

The resources provide activities with a mechanism for transforming inputs to outputs. In this perspective two types of constraints are considered:

- *External resource constraints*, in which the resource used by the activity is generated by an activity or resource that is external to the design process.
- *Internal resource precedence constraints*, in which the resource is developed in the design process and used by other activities.

### *10.3.2 Component Grouping into Modules*

After decomposing the system into its basic components or elements, a modular system should be constructed by integrating the basic similar elements based on a criteria set by the product design team. A modular system can be thought of as an integration of several functional elements that, when combined, perform a different function than their individual one. The similarity between the physical and functional architecture of the design must be used as a criteria for developing modular systems. Another criterion that must be used is the minimization of the degree of interaction between physical components. The degree of interaction between physical elements is an important aspect of modularity, which must be identified, minimized, or eliminated. The strength of a modular system design can be measured by the weakness of the interactions or the interfaces between its components.

Grouping objects (i.e., components, parts, or systems) into groups based on the object features has been done using Group Technology (GT) approaches. GT is defined as the realization that many problems are similar, and that by grouping similar problems, a single solution can be found to a set of problems, thus saving time and effort [1, 5, 7]. Similar components can be grouped into design families and new designs can be created by modifying an existing component design from the same family. Objects grouping or cluster analysis in GT is concerned with grouping parts into part families and machines into machine cells [2]. A number of algorithms and methods are available for clustering parts and machines such as the following [14]:

- The rank order clustering algorithm
- The modified rank order clustering algorithm
- The bond energy algorithm
- The cluster identification algorithm
- The extended cluster identification algorithm
- Similarity coefficient-based clustering
- Mathematical programming-based clustering

## 10.4 Modular Product Design

Modular product design is an important form of strategic flexibility, that is, flexible product designs that allow a company to respond to changing markets and technologies by rapidly and inexpensively creating product variants derived from different

combinations of existing or new modular components. Kamrani and Salhieh (2002) [6] developed a four-step methodology for the development of modular products as follows.

### *10.4.1 Needs Analysis*

This step includes gathering the market information required to identify customer needs, and arranging the identified needs into groups and finally prioritizing the needs according to their importance.

### *10.4.2 Product Requirements Analysis*

Product requirements are identified on the basis of the results of needs analysis. The product requirements are classified into three classes as follows.

- *Functional objectives* needed to meet the customer's primary needs.
- *Operational functional* requirements that impose both functional and physical constraints on the design.
- *General functional requirements* (*GFRs*) that satisfy customers' secondary needs, which could form a critical factor for the customer when comparing different competitive products that accomplish the same function. GFRs should be weighted with respect to their importance.

### *10.4.3 Product/Concept Analysis*

Product/concept analysis is the decomposition of the product into its basic functional and physical elements. These elements must be capable of achieving the product's functions. Functional elements are defined as the individual operations and transformations that contribute to the overall performance of the product. Physical elements are defined as the parts, components, and subassemblies that ultimately implement the product's function. Product concept analysis consists of product physical decomposition and product functional decomposition. In product physical decomposition, the product is decomposed into its basic physical components which, when assembled together, will accomplish the product function. Physical decomposition should result in the identification of basic components that must be designed or selected to perform the product function. Product functional decomposition describes the product's overall functions and identifies components' functions. Also, the interfaces between functional components are identified.

- *Product Physical Decomposition.* The product is decomposed into subsystems and/or subassemblies capable of achieving the product function. The decomposition process should continue until basic physical components are reached.
- *Product Functional Decomposition.* Functional decomposition should aim at representing the intended behavior (the functions) of a product and its parts. A function could be implemented by a single physical element (component) or by a combination of elements arranged in a specific manner. Functional components are arranged according to several logical considerations that will ensure the accomplishment of their intended combined function. The logical arrangement is called a working principle which defines the mode of action that the product/system will perform on the inputs to reach the output state. To analyze the product function, the overall function of the product should be conceptualized into an action statement (verb–noun form). Then, the overall function is broken into subfunctions, and those are further decomposed into lower-level functions. This functional breakdown is continued until a set of functions that could be achieved by available components is reached. At this point, functions are mapped into components, and components are arranged forming subassemblies leading to an overall assembly that will ultimately accomplish the overall function.

### *10.4.4 Product Concept Integration*

Basic components resulting from the decomposition process are arranged in modules and integrated into a functional system. The manner by which components are arranged in modules will affect the product design. The resulting modules can be used to structure the development teams needed. Following are the steps associated with product integration.

#### 10.4.4.1 Identify System-Level Specifications

System-level specifications (SLS) are the one-to-one relationship between components with respect to their functional and physical characteristics. Functional characteristics are a result of the operations and transformations that components perform in order to contribute to the overall performance of the product. Physical characteristics are a result of the components' arrangements, assemblies, and geometry that implement the product function. A general guideline for identifying the relationships can be presented as follows:

1. *Functional characteristics*
   (a) Identify the main function(s), based on the functional decomposition.
   (b) Identify the required operations and transformations that must be performed in order to achieve the function based on the function flow diagram.
   (c) Document the operations and transformations.
   (d) Categorize operations and transformations into a hierarchy structure.

2. *Physical characteristics*

(a) Identify any physical constraints imposed on the product based on the requirement analysis.
(b) Identify possible arrangements and/or assemblies of the components, based on previous experiences, previous designs, engineering knowledge, or innovative designs/concepts.
(c) Document possible arrangements and/or assemblies.
(d) Categorize arrangements and assemblies into a hierarchy structure.

Physical and functional characteristics, forming the SLS, are arranged into a hierarchy of descriptions that begins by the component at the top level and ends with the detailed descriptions at the bottom level. Bottom-level descriptions (detailed descriptions) are used to determine the relationships between components, 1 if the relationship exists and 0 otherwise. This binary relationship between components is arranged in a vector form, "System-Level Specifications Vector" (SLSV). Figure 10.10 illustrates the hierarchical structure of the physical and functional characteristics.

#### 10.4.4.2 Identify the Impact of the System-Level Specifications on the General Functional Requirements

SLS identified in the previous step affect the GFRs in the sense that some specifications may help satisfy some GFRs, while other specifications might prevent the implementation of some desired GFRs. The impact of the SLS on GFRs should be clearly identified. This will help in developing products that will meet, up to a satisfactory degree, the GFRs stated earlier. The impact will be determined on the basis of the following:

| | | |
|---|---|---|
| -1 | : | Negative impact |
| 0 | : | None |
| +1 | : | Positive impact |

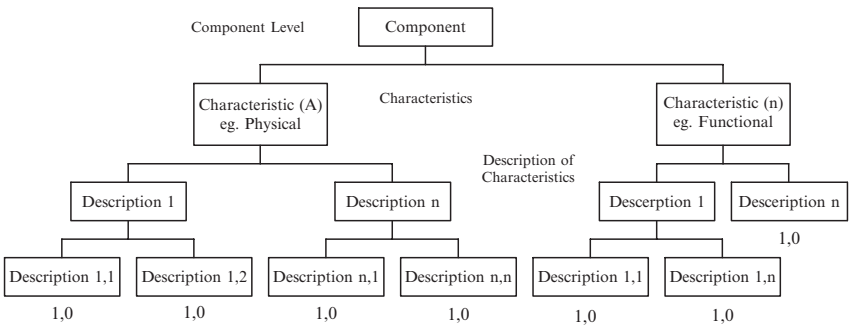


**Fig. 10.10** System-level specification decomposition hierarchy

**Table 10.3** GFR versus SLS

| System-level specifications | General functional requirements | | |
|---|---|---|---|
| | FR (1) | FR (2) | FR (*m*) |
| SLS (1) | –1 | 1 | 0 |
| SLS (2) | . | . | . |
| SLS (*n*) | 1 | 0 | 1 |

A negative impact represents an undesired effect on the GFRs such as limiting the degree to which the product will meet the general requirement, or preventing the product from implementing the general requirement. While a positive impact represents a desired effect that the SLS will have on the general requirements, such SLS will ensure that the product will satisfy the requirements and result in customer satisfaction. An SLS is said to have no impact if it neither prevents the implementation of the GFR nor helps satisfying the GFR. An example of the SLS impact on the GFRs is shown in Table 10.3.

For example, the SLS (1) have a negative impact on the FR (1), positive impact on the FR (2), and no impact on the FR (*m*).

#### 10.4.4.3 Calculate Similarity Index

The degree of association between components should be measured and used in grouping components into modules. This can be done by incorporating the GFR weights, in addition to the SLSVs and their impacts on the GFRs to provide a similarity index between components. The general form of the similarity index is as follows:

$$(S)_{1\times 1} = (1 \overset{\text{SLSV}}{\underset{(C_1 \& C_2)}{0}} \ . \ . \ a_n)_{1\times n} \times \overset{\text{SLS \& FR}_s}{\begin{pmatrix} 1 & . & . & . & b_{1,m} \\ 0 & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ b_{n,1} & . & . & . & b_{n,m} \end{pmatrix}_{n\times m}} \times \overset{\text{Weights for Fl}}{\begin{pmatrix} 1 \\ 0.9 \\ . \\ . \\ c_{m\times 1} \end{pmatrix}_{m\times 1}}$$

The similarity indexes associated with components are arranged in a component versus component matrix as shown below:

| | C1 | C2 | C3 | – | C*n* |
|---|---|---|---|---|---|
| C1 | X | $S_{1\times 2}$ | $S_{1\times 3}$ | – | $S_{1\times n}$ |
| C2 | | X | $S_{2\times 3}$ | – | $S_{2\times n}$ |
| C3 | | | X | – | $S_{3\times n}$ |
| – | | | | X | – |
| C*n* | | | | | X |

#### 10.4.4.4 Group Components into Modules

Components with high degree of association should be grouped together in design modules. This can be accomplished by using an optimization model that maximizes the sum of the similarities. The optimization model will identify independent modules that can be designed simultaneously. The model is Np-Hard.

Heuristic algorithms have been used as an alternative technique for solving Np-Hard problems. Modularity decomposition problem could benefit from these algorithms for finding solution in less time. This proposed method for decomposing similarity matrices in modularity is based on Network Flows and Optimization. The main reason for the application of network optimization as base structure is its ability in coupling deep intellectual contents with a remarkable range of applicability. Many combinatorial problems which are innately hard can be solved by transforming to network concepts.

This specification makes us to define our problem as a graph acceptable in Network Flows rules to have the opportunity to solve this decomposition by Network algorithms. After calculation of similarity measure, nodes and edges are defined as follows:

- *Node*: Each component represents a node in our proposed graph.
- *Edge*: Relationship between any two components represents an edge such that if component *i* has similarity index more than 0 with component *j*, there exists an edge between them.
- *Flow of each edge*: The similarity index between each two components is the flow of the edge between them (see Fig 10.11).

The objective function associated with this network is to find modules such that each module has the maximum amount of collective similarity indexes. It means that it is necessary to find modules created by group of components which are connected with each other by the largest similarity indexes. Each component can be assigned to just one module and each module must contain most similar components to each other. Based on this objective, Max Flow algorithms are used for solving this problem. In a capacitated network, it is required to send as much flow between start and end node. This concept has been applied in other engineering applications such as matrix rounding problem and feasible flow problems.

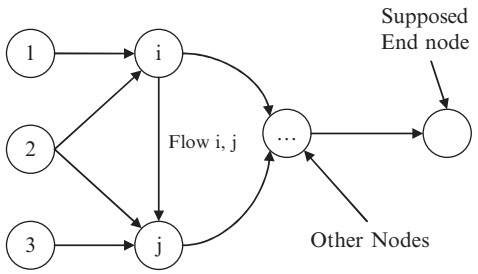


**Fig. 10.11** Graph for a decomposed product

In this problem, the maximum flow for the main graph is determined. This establishes the solution for the first module that contains components (nodes) in maximum flow. Components which are assigned to the first module will be removed and a new graph will be created and this graph for maximum flow is solved to find the second module. The algorithm will continue until all possible modules are assigned. For the max flow algorithm, augmenting path algorithm or preflow-push algorithm will be used on the basis of network specifications.

The preflow-push algorithm is one of the most powerful and flexible algorithms for max flow problems. The preflow-push algorithms maintain a preflow at each intermediate stage. Active nodes in this algorithm are nodes that have positive excess. Because this algorithm attempts to achieve feasibility and in a preflow-push algorithm, the presence of active nodes indicates that the solution is infeasible, the basic operation of this algorithm is to select an active node and try to remove its excess by pushing flow to its excess.

A maximum preflow is defined as a preflow with the maximum possible flow into the sink. This algorithm is a polynomial algorithm. The sample of the pseudocodes for the solution methodology is as follows:

*Network Set Begin*

*x = 1*
*n = number of components*
**While** *each component has assigned to a module*
*Design graph of components.*
*Use preflow-push algorithm to find the maximum flow from each node to node t.*
*Assign components in the maximum flow with similarity more than 1 to a module named module x.*
*Similarity of module = Sum (similarity of components)*
*x = x + 1.*
**End while**.
*Use selecting Procedure*
**End** *Procedure.*
*Selecting Procedure Begin*
*y = 1*
**While** *n < > 0*
*Choose the module with biggest Similarity*
*Eliminate components in that module from total components.*
*n = n – Eliminated components*
*y = y + 1*
**End while**
*y = number of modules*
*Write the selected modules as final modules.*
**End** *Procedure.*

Matlab® software is used for the implementation of the preflow-push max flow algorithm. Using this algorithm and Matlab Code for preflow-push max flow algorithm with a four-gear speed reducer with 17 components, the solution is as follows:

- Module 1: Gear 1, Shaft 1, Bearing 1, Bearing 2, Key 1, and Gear 2.
- Module 2: Gear 2, Gear 3, Shaft 2, Bearing 3, Bearing 4, Key 2, and Key 3.
- Module 3: Gear 4, Shaft 3, Bearing 5, Bearing 6, and Key 4.

## 10.5 The Benefits of Product Modularity

One of the most important benefits of promoting modularity is the need to allow a large variety of products to be built from a smaller set of different modules and components. The result is that any combination of modules and components, as well as the assembly equipment, can be standardized. The benefits of product modularity also include the following:

1. *Reduction in Product Development Time*

   Modularity relies on dividing a product into components with clear definition of the interfaces. These interfaces permit the design tasks to be decoupled. This decoupling results in a reduction in the design complexity and enables design tasks to be performed concurrently, which will eventually reduce the product development time.
2. *Customization and Upgrades*

   Modular products accomplish customer requirements by integrating several functional components interacting in a specific manner. This integration allows products to be improved and upgraded by using more efficient components that can perform the required functions effectively. In addition, components can be replaced by custom-made ones to fulfill different functions.
3. *Cost Efficiencies Due to Amortization*

   Modular components are used in several product lines, which infer that their production volumes are higher. This will allow the amortization of the development expenses over a large number of products.
4. *Quality*

   Modularity allows production tasks to be performed simultaneously. Thus, independent components can be produced and tested separately before they are integrated into a modular product. This will help build quality into the product.
5. *Design Standardization*

   Modular design facilitates design standardization by identifying the component functions clearly and minimizing the incidental interactions between a component and the rest of the product.
6. *Reduction in Order Lead Time*

   Modular products can be made by combining standardized and customized components. This allows standard components to be inventoried, and then

customization can be focused on the differentiating components. Also, modular products can be a combination of standard components, that is, the same standard components (usually kept in inventory) are integrated in different ways to form a variety of products that can respond to customer requirements.

## References

1. Amirouche, F., Computer Aided Design and Manufacturing, Prentice Hall: Englewood Cliffs, New Jersey, 1993.
2. Erhorn, C., and Stark, J., Competing by Design: Creating Value and Market Advantage in New Product Development, Essex Junction, VT, 1994.
3. Finger, S., and J. R. Dixon, A Review of Research in Mechanical Engineering Design, Part I: Descriptive, Prescriptive, and Computer-Based Models of Design Processes, *Research in Engineering Design*, Vol. 1:51–67, 1989.
4. Johnson, R. C., and R. C. Benson, A Basic Two-Stage Decomposition Strategy for Design Optimization, *Transactions of the ASME*, Vol. 106, September 1984.
5. Kamrani, A., A Methodology for Manufacturing Cell Design in a Computer Integrated Manufacturing Environment, Published Ph.D. Dissertation, University of Louisville, 1991.
6. Kamrani, A., and S. Salhieh, Product Design for Modularity, 2nd Edition, Kluwer Academic Publishers, 2002.
7. Kamrani, A. K., Modular Design Methodology for Complex Parts, Industrial Engineering Research Conference, Miami Beach, Florida, May 1997.
8. Kusiak, A., and N. Larson, Decomposition and Representation Methods in Mechanical Design, *Transactions of the ASME*, Vol. 117, June 1995.
9. Pahl, G., and W. Beitz, Engineering Design: A Systematic Approach, 3rd Edition, Springer-Verlag: New York, 2007.
10. Pimmler, T. U., and S. D. Eppinger, Integration Analysis of Product Decompositions, Design Theory and Methodology—DTM'94, DE-Vol. 68, ASME, 1994.
11. Rogers, G. G., and L. Bottaci, Modular Production Systems: A New Manufacturing Paradigm, *Journal of Intelligent Manufacturing*, Vol. 8, No. 2, pp. 147–156, April 1997.
12. Roozenburg, N. F. M., Product Design: Fundamentals and Methods, John Wiley & Sons: Chichester, New York, 1995.
13. Salhieh, S., and A. Kamrani, Macro Level Product Development Using Design for Modularity, *Robotics and Computer Integrated Manufacturing Journal*, No. 15, pp. 319–329, 1999.
14. Singh, N., and D. Rajamani, Cellular Manufacturing Systems: Design, Planning, and Control, Chapman & Hall, Norwell, Massachusetts, USA 1996.
15. Stone, R., and K. Wood, A heuristic method for identifying modules for product architectures, *Design Studies*, Vol. 21, pp. 5–31, 2002.
16. Ulrich, K., and K. Tung, Fundamentals of product modularity, In *Issues in Design/Manufacture Integration 1991*, pp. 73–79. A. Sharon Ed., ASME: New York, NY, 1991.