A Framework for Complex Product Architecture Analysis using an Integrated Approach

Abstract

Contemporary design decomposition and synthesis analytical tasks at the conceptual design stage reply on functional and structural modelling approaches. There is a wide diversity of elements used by various modelling approaches for information and representation of product architecture, which incurs difficulties for multidisciplinary engineers working across different phases of design in capturing, visualising, sharing and tracing consistent yet common knowledge and elements across the function and structure domains. This prompts for fixation of detail and common modelling knowledge across both functional and structural analytical approaches which is also critical from automatized software perspective. A limitation of existing approaches is that they tend to focus more on 'what' and less on 'how' (and vice versa). This paper proposes an integrated conceptual product architecting approach that combines and expands the functional and structural modelling approaches, enabling capturing and tracing knowledge coherently through a common binding domain. This is underpinned by the view that most interaction requirements amongst the physical components during structural modelling can be derived from functional modelling. The proposed integrated approach is underpinned by the critical analysis and synthesis of existing approaches in literature dealing with functional and structural architecture analysis, integrated within a Multiple Domain Matrix (MDM) to fuse the knowledge of both solution independent (functional) and dependent (structural) analyses. The proposed framework is illustrated with a case study of solar robot toy, followed by discussion and suggestions for future work.

Keywords

Product design, product architecture, functional modelling, structural modelling, and multiple domain matrix

1. Introduction

In literature, many product architecting frameworks have been proposed and developed based on multi-layers ranging from integration of new technology with existing product architecture (Ravn et

al, 2015) to optimising the product architecture via algebraic analysis based on structural composition i.e. engineering components (Ko, 2013). The complexity in product development is inherent and emerges from many domains of the design environment, particularly in the scheme of product architecture. The definition of product architecture in literature varies and involves either a single or multiple domains involvement. According to Ko (2013), product architecture is the scheme by which decomposed components of a product are arranged in modules whereas Ulrich (1995) defines it as the scheme by which "the function of a product is allocated to physical components". The product is usually complex in its requirements, functions, and components domains. Researchers often consider many inter-related domains which are necessary and useful for the product design and architecture analysis (Deubzer & Lindemann, 2009; Uddin et al., 2013). In practice, design engineers tend to deal with product complexity by breaking down the complex problem into smaller problems (Chmara et al., 2008; Pimmler & Eppinger, 1994). There are two major modelling approaches in engineering design: *functional modelling*, also referred to as solution *independent* analysis, and *structural modelling*, i.e. solution *dependent* analysis by decomposing the product into its sub-components (Jarratt et. al., 2004). However, the product architecting approaches are still not well understood due to a lack of simple visual representation, consistent, and detailed knowledge integration within and across functional-structural domains. The present study focuses on this aspect.

In the functional modelling approach, the function of the product is decomposed into various sub-functions so that design engineers can search concepts for each of the decomposed sub-function (Pimmler & Eppinger, 1994; Pahl et al., 2007). From structural basis approach, Design Structure Matrix (DSM) is readily available in literature and deals with the design of a product through decomposing and integrating it on structural basis and handling interactions or dependencies between the decomposed components, mainly via qualitative and quantitative schemes.

The information elements among available various functional and structural modelling domains are diverse for analysing a product architecture. This causes difficulties with integrating the information between functional and structural modelling tools due to which multidisciplinary engineers working across different domains of design with different sets of tools suffer from capturing, visualising, sharing, and tracing consistent set of knowledge. The design modelling knowledge is also critical

from automatized software perspective that often deliver innovative solutions based on algorithms but first it requires fixation of detail among various problem solving tools on one abstraction level (Deubzer & Lindemann, 2009). The present study also supports this aspect.

This paper proposes an integrated framework for product architecture definition and analysis that aims to capture the information of both solution independent and dependent analyses in a structured manner using a combination of existing tools by synthesising, expanding, and making them compatible through a common linkage domain. The structure of the remainder of this paper is organised as follows. Section 2 reviews significant contributions to the research area of product models for representing complex product architectures from functional to physical domains. Then, the problem is formulated based on critical analysis of existing models in Section 3. Section 4 presents the proposed integrated approach steps and discuses information flow and knowledge representation schemes in it. Section 5 illustrates implementation of approach via a multidisciplinary ~~simple desktop~~ case study followed by discussion, and conclusions in the Sections 6 and 7.

2. Review of Product Modelling Frameworks

2.1 Definition of Product Architecture

Ulrich (1995) defines product architecture as ‘‘a scheme in which the function of a product is allocated to physical components.’’ He elaborated it further as: “the arrangement of functional elements; the mapping from functional elements to physical components; and the specification of the interfaces among interacting physical components”. Ulrich presented a trailer example (shown in Figure 1a) to discuss different architecture elements (i.e. functional element and structural element) and types (i.e. modular and integral) of product architecture. In the modular architecture type, a product’s components / modules are functionally self-contained (i.e. one-to-one mapping) whereas in an integral architecture type components are functionally tightly coupled (i.e. one-to-many mapping) (see Figure 1b). Referring to Ulrich’s architecture definition, contemporary products designs meet all three requirements and fall into two architecture types. Many other elements (or building blocks) have also been introduced between functions and components by researchers to define and analyse product architecture. Following sections discuss such modelling approaches and their architecting elements

for product architecture in *functional, functional to structural,* and *structural modelling* domains.

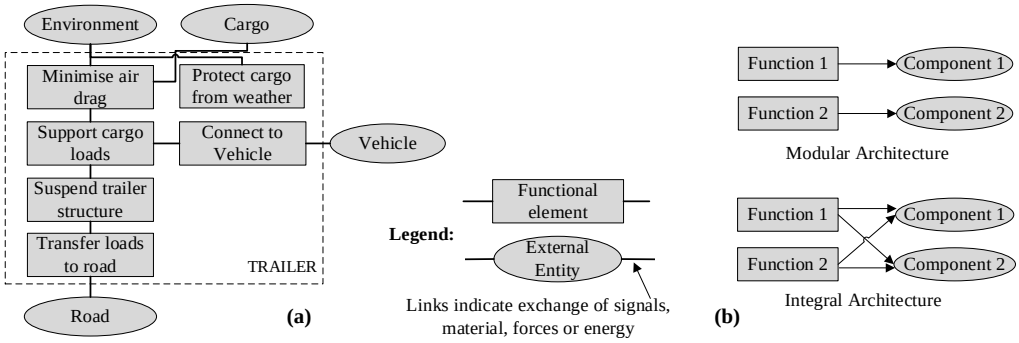


Figure 1. Product architecture elements and types (adapted from Ulrich, 1995)

2.2 Functional Modelling

A distinct step in function modelling approaches is the establishment of solution neutral *function structure*, which is used as basis for the subsequent design tasks to define the physical components and structure of the product in the early stage of design (Pahl et al., 2007; Ulrich & Eppinger, 1994). The basic purpose of a function structure is to organize the *functions* of a product in a coherent manner. The functional models have a variety of such function structures ranging from hierarchical trees to flow oriented approaches (Pahl et al., 2007; Campean et al., 2013) as illustrated in Figures 2. A hierarchical function structure is known as function *tree* which arranges functions from a product's top level overall function down to low level sub-functions which may be decomposed further or to 'leaf' functions that cannot be decomposed further as shown in Figure 2a. This function structure does not arrange functions in a time sequence and lacks such visual representation. Furthermore, it provides no information on mapping of flows on functions which is also critical. This limitation is overcome by flow-oriented function structure, introduced by Pahl et al., (2007) shown in Figure 2b, where the functions are arranged both in terms of *flows* between them: material (M), energy (E) and information (I) and also in a time sequence. These flows are often referred as operands (Hubka and Eder, 1996). However, the states and properties of the operands at input and output are not discussed and specified in such function structure and lacks such visual representation. It is also essential to take into account and specify the measurable attributes associated with operands as discussed by Hubka and Eder (1996). To overcome this limitation, another type of function structure, system state flow

diagram (SSFD), is recently introduced by Campean et al., (2013) and Yildirim & Campean (2014), shown in Figure 2c. This type of representation identifies and arranges the functions associated with the transition of main and secondary flows from their input states to output states by a set of measurable attributes. Thus, the SSFD tool is a comprehensive approach for functional modelling that provides both visual clarity and detailed information.

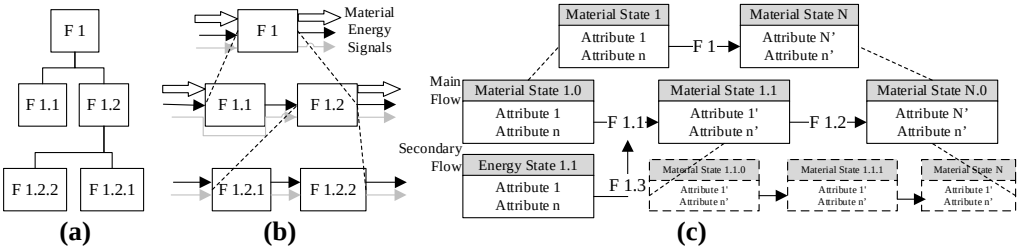
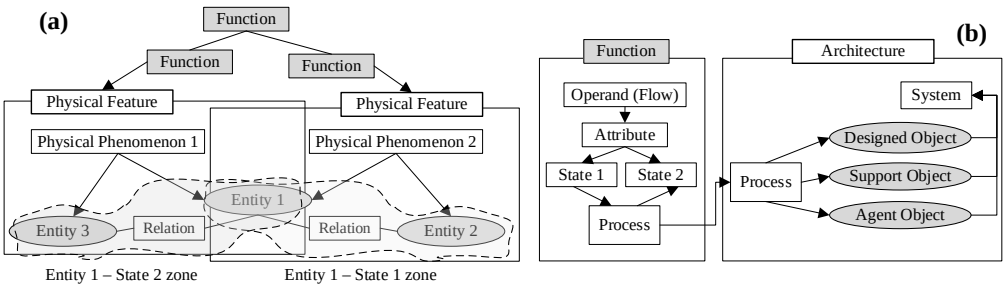


Figure 2. Representations of elements of functional models

2.3 Function to Structure Modelling

The *function* domain is often considered as the intermediary between requirement and *structural* domains. In the Axiomatic Design (AD) methodology, a product is modelled hierarchically via a zigzag procedure between functional requirements and design parameters of functional and physical domains respectively (Suh, 1998). According to Suh (1998), design parameters may be *physical parts*, parameters or assemblies. However, concrete decomposition operations on the product description have not been explained (Chmarra et al., 2008; Komoto & Tomiyama, 2011). Several researchers (Gero, 1990; Umeda et al., 1996), argued that function to structure domains mapping require interpretation of physical *behavioural elements* in between functions and entities for conceptualising the product architecture which are perceived in various ways by several researchers. In FBS (Function-Behaviour State) model, behaviour is perceived as the 'physical phenomenon' that causes the change of the 'states' of entities (components) of the system (Umeda et al, 1996), shown in Figure 3a. However, no information on operand is stated or visualised. In OPM (Object-Process Methodology) model, behaviours describe the processes that cause the transition of 'states' of an operand (i.e. flow) and also describe the operation performed by the entities (or objects) of the system (Soderborg et al, 2002), as shown in Figure 3b. The graphical representation of both OPM and FBS approaches is relatively complex and require learning of many building blocks (or notions) for

product modelling. A similar concept of working principle (i.e. physical principle) has also been proposed by Pahl et al (2007) which supports for searching concepts via Morphological~~y~~ Matrix/Chart that could satisfy the developed function structure. Concept generation is an important modelling activity that often provides various working solutions for a same function structure of a product. The conventional morphological schemes are often based on abstract knowledge of functions, and operands. Woldemichael & Hashim (2011) refer to working principles as *concepts* in their morphological scheme. In their concept generation method, the input is the set of sub-functions along with inputs and outputs operands' descriptive information and on the basis of which the output is the set of alternative concepts displayed on morphological~~y~~ chart that can satisfy those sub-functions. However, their concept generation scheme does not discuss information related to flowing operand's *attributes* and constraints. *Constraints* are often regarded as attributes on operands variables such as parameter values, position, and orientation etc. (Cao & Fu, 2011) that help in minimising the solution space. Constraints help in determining the accuracy, sufficiency, and completeness of design decisions (Lin & Chen, 2002).



(c)

			Domain 1	Domain 2	Domain 2
Function	Operations (Op)	Domain 1	Op	St_{out}	Ph_{Op}
	States (St)	Domain 2	St_{in}	St	Ph_{in}
	Physical Parts (Ph)	Domain 3	Ph_{Op}	Ph_{out}	Ph

Figures 3. Representations of function to structure modelling approaches

These function to structure modelling approaches involve many complex, overlapping and diverse notions that engineers often find hard to grasp, implement, and integrate them. Also it is hard to differentiate and visualise *interactions* within and across various domains separately via such graphical approaches. To overcome such limitations, many matrix-based approaches, such as Design

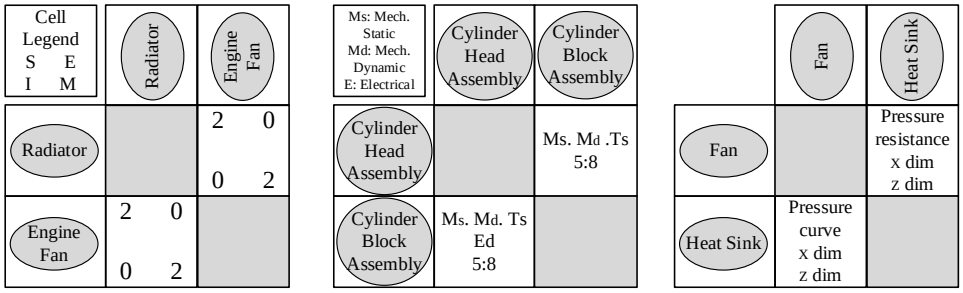
Structure Matrix (DSM), Domain Mapping Matrices (DMM) and Multiple Domain Matrix (MDM) (Deubzer & Lindemann, 2009; Lindemann et al., 2009) have also been developed for representing product architecture due to their simple visual appeal and for managing interactions both within a single and across multi-domains. Matrix based approaches have the ability to be generated through functional models to maintain consistent design strategy (Duebzer & Lindemann, 2009).

The applications of MDM approach have continuously increased over the last few years in product design (Lindemann et al., 2009). Among the existing MDM approaches, an inspiring framework is introduced by Deubzar & Lindeman (2009) that supports product architecture analysis by using operations, states, and physical components' domains, shown in Figure 3c. They divorce function description into operation (via verb) and operand state (via noun) domains and map interactions between them. This framework supports in deriving solutions based on given requirements in operations and state domains. However, the detail knowledge in terms of operands' states and their properties is not considered. The framework is visually simple and useful to accumulate the knowledge of both solution independent and dependent analysis and can be integrated with functional models. Due to such reasons, this framework is used in this paper.

2.4 Structural Modelling

A DSM approach is widely recognized for modelling the product in a structural domain. Looking at the structural domain of a product, a DSM approach decomposes the product into its components/parts/entities and represents its architecture by capturing *interactions/relationships* between the components (Browning, 2001). Multiple types of interactions can be specified between interacting entities as realized by researchers (Pimmler & Eppinger, 1994; Martin & Ishii, 2002; Jarratt et al., 2004). Pimmler & Eppinger (1994) used static DSMs to identify and examine alternative product architectures and described four types of interactions referred as spatial (S), material (M), energy (E), and information (I) between the decomposed entities along with the quantification scheme that facilitated weighing interactions amongst them, as shown in Figure 4a. Many researchers (Otto & Wood, 2001; Rahmani & Thomson, 2012; Hamraz et al., 2013; Uddin et al, 2015) have adopted and (Sosa et al., 2003; Jarratt et al, 2004) extended this four-interaction taxonomy. For example, Sosa et al. (2003) extended the four-exchange taxonomy with an introduction of fifth type as

'structural' (P) that indicated the requirements related to transferring loads or containment between two interfacing entities. Jarratt et al (2004), looking from an engineering change management perspective, also used the concept of S/E/M/I interactions between two interacting components but with an addition to multiple-type *linkages* definitions with steady and dynamic states, shown in Figure 4b. Martin & Ishii (2002) used the concept of *specification flows* in the product platform architecture. In their component-based DSM, an interface is characterized by specification flows that seem to represent a combination of flowing operands and *parameteric* constraints, shown in Figure 4c. As a conclusion, a product is modelled via DSM on structural basis thereby representing three operands based interactions (Energy, Material, Information) and also one to two physical based interactions (Spatial, Structural).



Figures 4. Representations of structural models and their information elements

3. Contemporary Analysis and Critique

A number of product modelling approaches have been discussed in previous section in the context of functional, function to structure mapping, and structural domains. There are a number of commonalities in these modelling approaches. For instance, the concept of multiple flows or (or interactions) i.e. E/M/I between functional and structural elements is quite distinct as discussed in Section 2; however, following differences and issues are observed.

From design effort perspective, it is observed, for conceptualising the product architecture, theories like AD, and OPM, on the one hand, support the product evolvement between different levels, whilst knowledge-based (i.e. rule-based) approaches do not. On the other hand, cross domain matrix

approaches such as MDM support the product analysis but do not discuss design solutions search activities based on functional domain.

From visualisation, domains separation, and detail knowledge perspectives; on one hand methodologies like, OPM, and FBS analyse product architecture via dense graphical representations with several behavioural elements and diverse notions between function and components but lack to provide compact visual representation without showing the interactions or shared attributes both within as well as across domains. On the other hand, matrix based approaches analyse product architecture with compact and simple visual representation and identify interactions both with and across function and component domains, lack to cover both detail and shared attributes knowledge in between them. Both views are essential and have their pros and cons. Thus an integrated approach that could separate and show binding elements common across both functional and structural domains among different sets of tools will support product architecture analysis in a more appropriate and consistent way.

The other central issue in existing approaches is the lack of clear discussion and representation on common and detailed information link across both functional and structural domains with the emphasis that how mostly interaction requirements between components can be derived from functional elements. This is also critical from tools' information integration and software automation perspective.

This fact is formulated and elaborated below in typical knowledge rule format, and illustrated within Figure 5;

IF an operand object E has input state (Esti) with measurable attributes (Ai) and constraints (Ci)

AND is an output of Function F2 but input to Function F3,

AND IF Function F2 is mapped/allocated to Component C1,

AND Function F3 is allocated to Component C3,

THEN there is an understood requirement that the interaction between Components C1 and C3 will possess operand E with same state and measureable attributes at input-output.

The operand E can be a material, or an energy or information related.

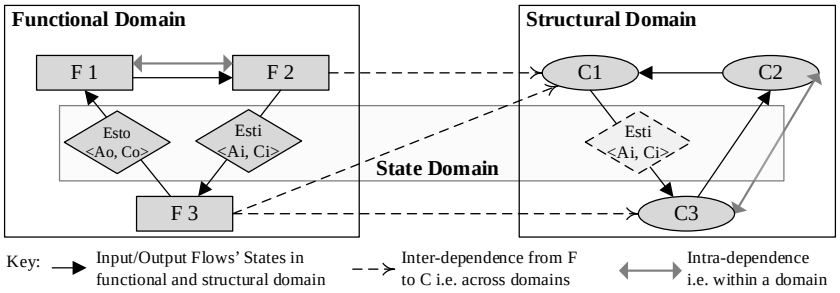


Figure 5: Proposed structure and elements for functional to structural domains mapping

Therefore, looking at Figure 5, a product architecture definition is revised from Ulrich's (1994) definition and the following set of requirements to be met by any product architecting approach, are extracted based on the existing literature.

- *Req. 1:* the arrangement of functional elements (as discussed in Section 2.1) but with measurable attributes and specifications (or constraints) of input-output flows (as discussed in Section 2.2)
- *Req. 2:* the mapping from functional elements to chosen physical components (as illustrated in Section 2.1) but with clear intermediary domain (as discussed in Section 2.3) and,
- *Req. 3:* the specification of the interfaces among interacting physical components (discussed briefly in Sections 2.1 and 2.4) but both *form* (P, S) and *operands* (E, M, I) related interactions (as summarised in Section 2.4)

Furthermore, there is a need for an integrated approach that takes into account aforementioned central issue, re-visited requirements, and consideration of the *following issues* for product architecture analysis:

- *Req. 4:* To deliver a *balance*d architecting process for the design effort, both across solution independent (what) and dependent (how) analysis.
- *Req. 5:* To support the consistent knowledge transferability between multidisciplinary engineers dealing with different sets of tools

- *Req. 6:* To support both types of product architecture analysis, i.e. modular and integral architectures.

4. Framing the Proposed Integrated Approach

The current authors have selected three key tools: SSFD, Morphologically Chart (MC) and MDM for following reasons. For solution neutral analysis and function structure development, SSFD is the comprehensive tool (as discussed in section 2) and thus meets Req. 1 (articulated in section 3). It involves representation of functions and operands states knowledge except constraints elements and is expanded in this aspect. The conventional MC helps in searching and guides in choosing potential concepts for identified functions and operands and thus meets Req. 2. The conventional MC lacks discussion and representation of constraints and is thus expanded in this aspect. The MDM fulfils Req. 2 and Req.3 as it helps identifying, mapping and tracing interactions both within and across domains. The analytical knowledge from SSFD, and MC based on constraints on operands' states can be transformed into MDM. The knowledge is defined and captured through graphical tools such as SSFD and MC whilst dependencies and interactions are traced through matrix based tool MDM.

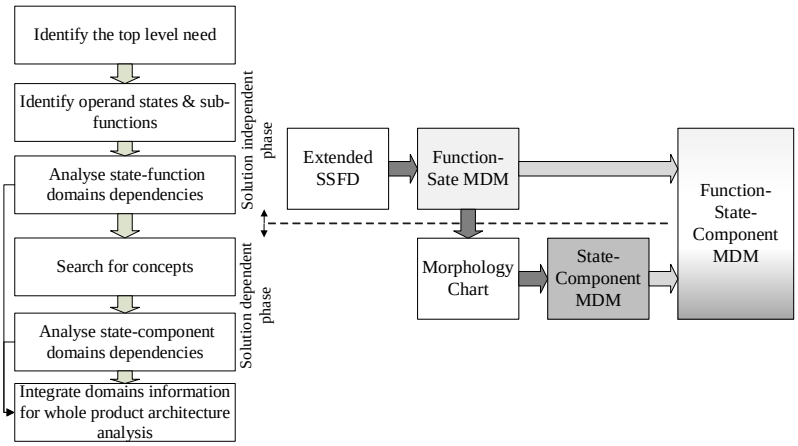


Figure 6. Proposed integrated approach - *process steps* and information flow in *tools*

The integrated approach process steps are presented, on the left side of Figure 6. It involves six key steps: from the customer's top level need to integrated product architecture analysis. The tools' sequence and information flow for the integration of both phase analysis is shown, on the right side of Figure 6.

4.1 Solution Independent Analysis Phase and Representation Elements

Referring to Figure 6, the solution independent phase is based on two key tools: System State Flow Diagram (SSFD) and Function-State (FS) Multiple Domain Matrix (MDM).

4.1.1 System State Flow Diagram (SSFD)

A distinct feature of SSFD is its ability to show transitions in measurable attributes with main and secondary operands' input and output states based on graphical representation. The generic structure of SSFD is shown in Figure 7 and follows the basic principles of state diagrams (Harel, 1987): a box represents a flowing object's (or operand's) states and an arrow denotes the function required to achieve the state transition. A flowing object or an operand can be material (e.g. bread, etc.), energy (e.g. solar power, etc.) or information (digital or analog signal or data) related and can be represented with attributes such as operand temperature, operand velocity etc. Authors extend box representation by attaching constraints underneath attribute box as shown in Figure 8a. Originally, in SSFD, an operand object is composed of attributes <An> shown in Figure 7. Now with extension of constraints <Cn>, an operand object is composed of set <An, Cn>, as shown in Figure 8a. Three types of constraints are adapted from literature (Cao and Fu, 2011); a *constancy* shows an operand stability, *track* shows an operand movement direction and *range (R)* shows associated measuring minimum and maximum values. For example, shown in Figure 8a, an operand 'rotational energy' at its input state is composed of single attribute <Ai: angular velocity> and constraints <Ci: Constant, R, Clockwise>.

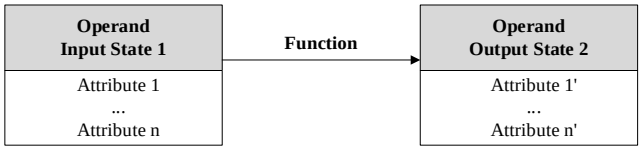


Figure 7: SSFD representation (adapted from Yildirim & Campean, 2014)

Figure 8. is provided at the last page.

Formatted: Font color: Text 1

4.1.2 Function – State Multiple Domain Matrix (FS-MDM)

The knowledge is then transferred and traced by FS-MDM tool (shown in Figure 8b) comprising of function domain DSM, operands' state domain DSM and two function vs states' FS-DMMs, driven

by SSFD. The purpose of FS-MDM is to *divorce* the functions from operands' states in order to visualise each domain separately along with analysing the interactions within and across the two domains (which are usually represented in a unifying manner in SSFD and can be hard to visualise the dependencies between and across domains). The FS-MDM groups states input-output pair knowledge within a state domain as well as maps interactions across a functional domain.

4.2 Solution Dependent Analysis Phase and Representation Elements

In this phase, again two key tools are available to engineers: the morphological~~y~~ chart (MC) and the State-Component (SC)-MDM.

4.2.1 The Morphological~~y~~ Chart (MC)

In this paper, the MC is incorporated with constraints elements, as shown in tabular format in Figure 8c. Their importance and consideration in MC are now discussed from concepts generation and design decisions accuracy perspectives.

In conventional morphological ~~y~~ chart, a number of concepts (see column C5 in Figure 8c) are searched and shortlisted based on functions description (column C1) and then input-output operands description and attributes knowledge (column C2-C3) is used for decision making as shown in Figure 8c. However, at times such knowledge for decision making becomes insufficient, inaccurate, and leads to same concepts. For example, a function description 'change rotational energy' (in Figure 8c) can be used to search three concepts 'pulley belt mechanism' (row R2), 'gear pair box' (row R1), and 'crank rocker' (R3). Then input-output operands knowledge i.e. 'rotational energy' in and 'rotational energy' out along with attributes 'angular velocity' again would shortlist same three concepts. But if constraints (Column C4) on input and output operands' attributes are specified then a desired concept can be further shortlisted. For example, if a designer specifies constraints on angular velocity attribute as <constant magnitude, clockwise direction> at input and also <variable magnitude, to-and-fro direction> at output, then a single potential concept 'crank rocker' would be available, shown in Figure 8c. It should be noted that this paper focuses on what kind of information can be used and automated to search for a product solution within developed morphological~~y~~ scheme and not on the development of grammar rules for automation search mechanism. Designers can use functions and

states' knowledge for obtaining solutions by providing relevant depth of information in the database search engine to derive possible concepts.

4.2.2 State-Component Multiple Domain Matrix (SC-MDM)

Since the input-output flows' pair knowledge (along with the function in the morphological ~~y~~ chart~~table~~) is used to generate a product design solution, therefore the concepts that fulfil those operands' constraint information are then placed into State-Component MDM as shown in Figure 8d. The state DSM knowledge remains the same, both across function and structure domains. The SC-DSM helps in identifying and visualising that output of one solution (e.g. angular velocity from C1: crank rocker) is the input to the other (e.g. angular velocity to C2) shown in Figure 8d. Similarly other interaction operands related to material (M) and information (I) can be traced between C1 and C2. The next step for design team in the same DSM is to identify and specify physical interactions (P). Therefore, SC-MDM first divorces the components from operands' states and then maps out the relevant relationships between and across them. Consequently, the selected concepts in the component DSM are represented via four types of interactions collectively i.e. P, M, E, I with their specifications.

4.3 Integrating Knowledge Across Analysis Phases

In the final step, the knowledge of both phases is integrated and visualised into a single framework named as Function-State-Component (FSC) MDM and represents the whole product architecture in a structured manner, shown in Figure 8e. This matrix also helps in mapping and tracing the relationships between function, state and component domains. Although MDM analysis is conducted on the modelling concepts and framework given by Deubzer and Lindemann (2009), however, the representation of knowledge of functions, states, and components within FSC-MDM of proposed integrated approach is different. For example, the interactions in Deubzer's MDM are represented via qualitative and quantitative schemes between the operation and states' domains whereas the proposed approach in FSC-MDM covers the input-output operands states' *specifications* within state and component domains. The state domain matrix also groups input-output states pair knowledge which is not represented in similar fashion in the state domain of Deubzer and Lindemann's (2009) MDM framework.

5. A Multidisciplinary Case Study: Solar Robot Toy

A multi-disciplinary case study of a domestic scale Solar Robot Toy (SRT) was employed to illustrate the proposed framework. A specific design problem in the SRT i.e. movement of arm or leg has been considered to demonstrate the working of the proposed integrated approach. Solar energy powers the solar board of the SRT and is absorbed via thermal collectors, and this energy is then used to drive mechanical components. The SRT receives solar energy as an input and generates rotational energy to rotate its arm/leg as an output. In order to keep the case study scope 'brief' for the purpose of illustrating the approach, only the *energy* related operand is considered here.

5.1 Identifying the Top Level Need

At the desired abstract level, the top function of SRT is to 'convert solar energy into rotational energy' in order to rotate the robot arm a full 360^0. It is represented by using an extended-SSFD tool, shown in Figure 9. The input operand is the 'sunlight or solar energy' (i.e. flowing object/operand) whose measurable attribute is 'solar power' that has got multiple constraints and needs to be converted into an output state of 'rotational energy' possessing attribute of 'angular velocity' with desired constraint values. The constraints on the operands' attributes help in establishing the desired target values to be achieved by the designed product.

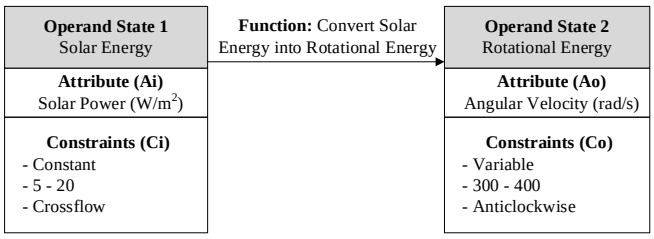


Figure 9. Solar Robot Toy (SRT) high level representation via extended SSFD

5.2 SSFD Tool: High Level Function Decomposition

The design team can decompose the high level function into sub-functions with the identification of intermediate states via SSFD, as shown in Figure 10. The engineers at this stage may perform many iterations for function structure development, thereby identifying requirements associated with attributes and constraints on intermediate states of operands and corresponding sub-functions. For

example, the identified intermediate state ‘electrical energy’ may have two types of constraint e.g. constancy constraint can be either of <constant magnitude> or of <variable magnitude>. In this case, electrical energy state is defined with a constraint of <constant magnitude>.

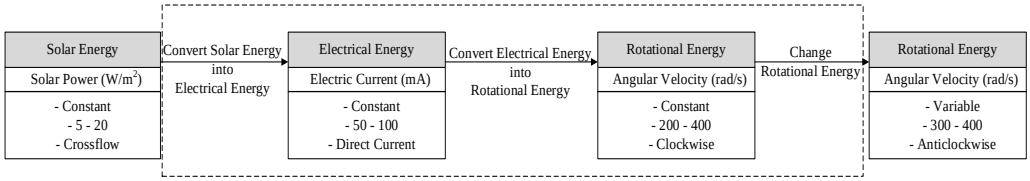


Figure 10. The main operand representation with sub-functions via SSFD

5.3 Knowledge Transformation into FS-MDM

The function structure obtained via the SSFD is then divided into function and state domains’ knowledge and relevant information is organised and placed into FS-MDM as shown in Figure 11. Once the identified sub-functions and states are placed into FS-MDM, the dependency within each domain and across domains is then mapped. For example, across domains (i.e. from function to state), sub-function ‘convert S.E (solar energy) to E.E (electrical energy)’ has input state ‘solar energy’ and output state ‘electrical energy’ so the relevant cells in state domains are shown by ‘I’ and ‘O’ symbols respectively. Similarly, the same procedure is repeated for other sub-functions and states. Also within a single domain, relevant mappings are performed e.g. in the function domain, ‘convert E.E (electrical energy) to R.E (rotational energy)’ is dependent on ‘convert S.E to E.E’ whereas ‘change R.E (rotational energy)’ is dependent on ‘convert E.E to R.E’ hence relevant cells are marked by ‘X’. The same applies to the state domain, relevant input-output states pair knowledge is aggregated which reveals a states’ transition information without function.

5.4 Morphologicaly Chart (MC)

The possible concepts and working solutions for the developed SRT function –structure are then explored via MC. For example, designers can obtain two solutions of ‘DC-motor’ and ‘AC-motor’ using an information of function: ‘convert electrical energy into rotational energy’, and also specifying ‘current’ as an attribute at input state and ‘angular velocity’ as an attribute at output state (rows R5 & R6 shown in Figure 8c). After this, further information of constraints is specified on the attributes then two different concepts filter down to one concept. For example, entering the constraints

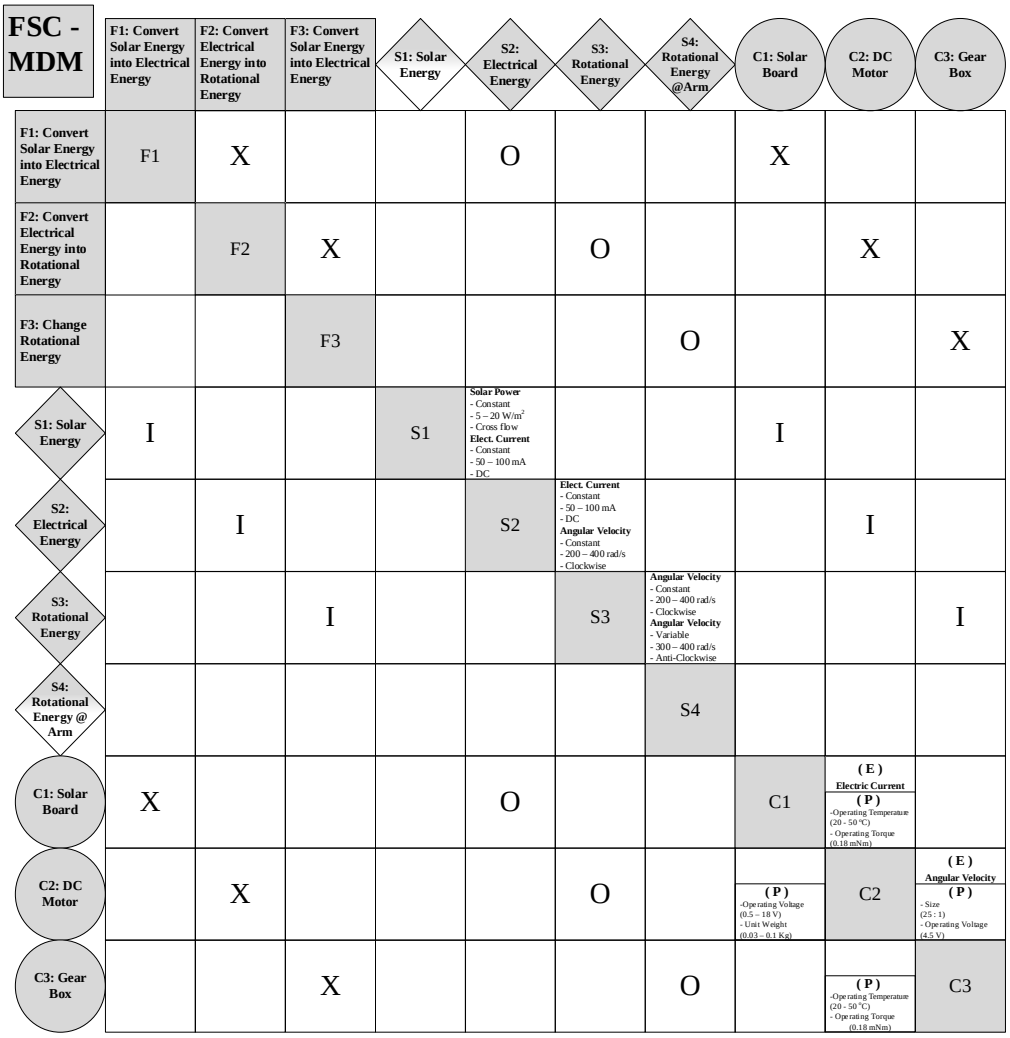
information on input current attribute (constant, DC, and range values) and on output angular velocity attribute (constant, clockwise, and range values) associated with function ‘convert electrical energy into rotational energy’ would filter ‘DC motor’ concept only. Similarly, other concepts ‘Gear Box’ and ‘Solar Board’ are explored for SRT’s sub-functions ‘change rotational energy’ and ‘convert solar energy into electrical energy’ respectively and can then be grouped into a working solution i.e. ‘Solar Board + DC Motor + Gear Box’.

5.5 Knowledge Transformation into SC-MDM

The concepts obtained in previous step are then placed in SC-MDM shown in Figure 11. The SRT’s SC-MDM shows engineers which concepts are responsible for which operands states’ transitions. Furthermore, the SC-MDM shows that what sort of interaction operand exists between two components and also that it is an output of one component but input to another. For example, ‘Solar Board’ will deal with input (I) operand ‘solar power’ and will deliver ‘electric current’ as an output attribute (O) shown in Figure 11. This output of ‘Solar Board’ becomes the input for ‘DC-Motor’. Similarly, the design team can trace and visualise the interactions between other components rather than brainstorming. After this, the physical specifications can be defined between components’ interfaces.

5.6 Unifying Knowledge in a Single Framework

The FS-MDM and SC-MDM analysis are then combined in the FSC-MDM which also helps in tracing the relationships between functions and components domains as shown in Figure 11. For example, ‘convert S.E into E.E’ is achieved via ‘Solar Board’ and the corresponding cell is mapped as ‘X’. It should be noted that the analysed problem in SRT is of modular architecture nature (referred to Figure 1b) in which a single function is allocated to a single component which is observable in Figure 11.



The semi filled diamond of S4 in row shows the final state must provide by design and also that output rotational energy at arm is not used further as an input. The semi filled diamond of S1 in column shows the initial state acquire by design and also that input solar energy is not generated as an output

Figure 11. SRT architecture analysis via FSC-MDM for SRT

In most cases, a product can possess integral architecture nature i.e. a single component can perform more than single function or inversely speaking a single function can be achieved by more than a single component (recalled Figure 1b). According to Ulrich (1995), “an integral architecture includes a complex (one-to-many) mapping from functional elements to physical components and/or coupled interfaces between components”. If a modification to same existing design of the SRT (shown in Figure 11) is required; for example, a robot arm of SRT should work when there is no sunlight, the new functionalities such as ‘Store Energy’ are analysed via SSFD and solution is searched via MC

and then the updated information can be visualised via MDM shown in Figure 12 (with only relevant information). This would reflect and provide the overview of whether the SRT product architecture is of modular nature or integral design as can be seen from Figure 12 that more than one function is allocated to a single component 'Solar Board'.

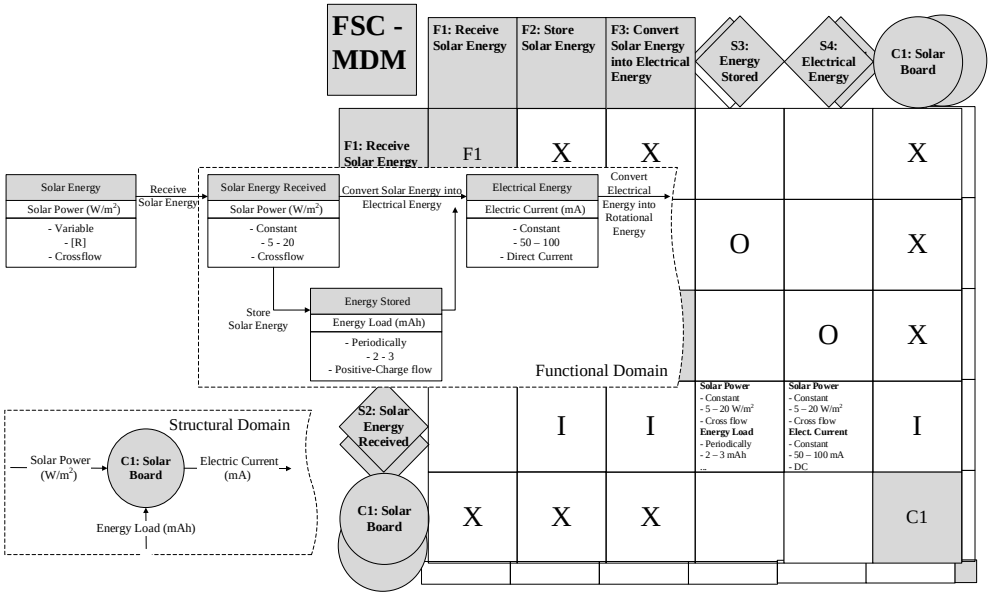


Figure 12. SRT architecture analysis with an inclusion of different design problem

In a similar way, the approach can be applied depending upon the context of the defined problem for the whole SRT and at any level of abstraction. Thus, the FSC-MDM framework has the ability to not only manage multiple relationships with detailed information within and across *functions*, *operands' states*, and *components* domains in consistent and structured manner but also support in analysing both modular and integral architecture types of products and hence fulfil requirements Req. 4 to 6 (articulated in section 3).

6. Discussion: Benefits of Proposed Integrated Approach

The proposed approach shows consistent information flow in a way that functional and structural elements share the same type of input-output states' specifications. It also supports in deriving physical components' E/M/I interaction requirements which as a whole is visualised in the FSC-MDM. The state domain is the key linkage, binding the function and component domains in a

coherent and structured manner with detailed information. An application of the approach to the Solar Robot Toy (SRT) is also discussed in following paragraphs from the perspectives of engineering change management and generation of different architectures.

Design engineers working across different phases of design can trace the changes from top-down and bottom-up in FSC-MDM. Firstly, from bottom-up perspective, the interactions between components are based on functions and states information rather than brainstorming. For example, in the case study, the DC-Motor and Solar Board share electric current (energy operand) in which the current is an input to DC-Motor and output from Solar Board (see Figures 11 and 13). Secondly, the same ~~components based~~ DSM shows that the DC Motor *provides* specifications information to Solar Board and Gear Box or, inversely speaking, the Solar Board and Gear Box *require* the specific information from the DC-Motor such as operating voltage, motor weight, and motor size (see Figure 13). This managed information helps the designers to visualise that if they change or replace a DC-motor then the specifications need to be carefully handled for other components. Any change in the DC-motor will cause the Solar Board specifications to change to meet the DC-motor specifications; similarly this will also impact on the corresponding functions and states all the way up. This distinct feature is observable in Figure 11. Hence these two interactions specifications (i.e. operands and form related) in ~~components~~ DSM not only help the final conceptual design but also the assembly design and/or from change management perspective.

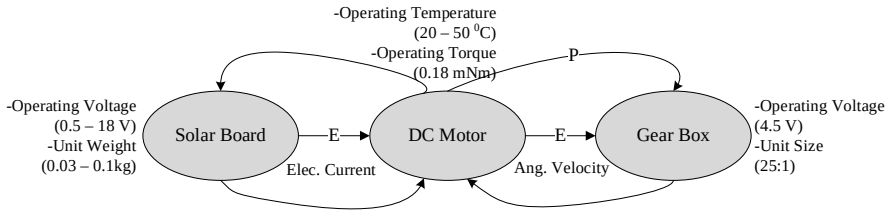


Figure 13: Energy operand (E) and physical form (P) related specifications between DC Motor and adjacent components of SRT

Secondly, from a top-down perspective, if any functional element is changed or updated at the top level then it will also impact all the way down to component levels and the interactions information

may vary between them. For example, the product architecture solution based on *Solar Board + DC-Motor + Gear Box* was selected for a defined problem. However, if the problem is changed, e.g. assuming that specification for arm rotation is changed from '0-360^0 rotation to 0-180$^{0'}$ <i.e. Range> and the arm should return back from 180-0^0 <i.e. Track> then the design team will need to look for another solution; e.g. 'Crank Rocker' from the morphologicaly chart (see Figure 8c with row R3) meets the same sub-functional and states requirement fitting with the new constraint conditions. In that case, the product architecture solution would be *Solar Board + DC-Motor + Crank Rocker* and will require the designers to manage the specifications and interactions between components accordingly. This illustrates that the proposed integrated approach supports both engineering change management and different product architectures generation activities.

7. Summary, Conclusions and Future Work

The main aim of this paper was to present an integrated and structured design approach for complex product architecture in the context of solution independent and solution dependent analysis based on tools that are currently in practice. The review of current methods' and tools' representational elements and information for conceptualising the product architecture across functional and structural modelling domains pointed out the gaps and the need for a novel approach. The novel approach is built-upon the adaptations of principles of SSFD, MC and MDM tools available in literature. The SSFD tool helps for functional decomposition, MC for searching solutions and MDM for design architecture synthesis. The framework provides a systematic and balance procedure to perform both solution independent (i.e. *what*) and solution dependent (i.e. *how*) analysis. A core binding linkage between the function and structure domains is the state domain with measurable set of attribute and constraints that helps in finding solutions, extracting the interaction requirements in structural modelling from functional modelling and keeps the information in a structured way. A multidisciplinary case study, i.e. Solar Robot Toy (SRT) was used to illustrate the proposed integrated approach.

Although the proposed approach serves its primary objectives articulated in section 3, however it has some limitations which are discussed and considered worthy for future work.

The proposed integrated approach has a systematic and consistent structure in terms of information sharing between different domains and tools but is conceptually document based in current shape which takes time and thus requires a software support to populate, automate and handle the information among various tools. For example, the FS-MDM, SC-MDM, and FSC-MDM matrices would be automatically generated based on information feed into SSFD and MC by design engineers. In the future work, a set of software requirements will be formulated for embedding the integrated approach into a software package.

Since, the approach involves a matrix based tool, the FSC-MDM results in establishing and completion of large matrices which is also a common practice in engineering design. For example, the matrix based approaches such as QFD (Quality-Function Deployment) are often applied in engineering practice and frequently generate large matrices. It is no surprise that same applies to FSC-MDM and hence, relatively large documents are expected while implementing the approach on many industrial and real world complex products such as automotive engines, and aeroplanes etc. At the moment framework mainly focuses on technical domains and internal structure of the product. There are other matrix based approaches such as PRD model (Product Requirement Development) of Bonev et al, (2013) that provide compact visual overview and insights on the internal relations of the product as well as integrate business aspects along with the requirements coming from different stakeholders thereby assessing the development tasks of a physical product from redesign or changes perspective. In future, to allow for more comprehensive support, it would be explored how other domains related to business and different stakeholders can be integrated into the proposed integrated approach.

Though the approach has filled the gaps in using and extending the existing tools and integrating them in a structured manner, however at the same time few compromises were made in certain tools. For example, throughout the approach, the input-output operands' states have been defined and considered with only *single* attribute and its *multiple* constraints. According to SSFD (Yildirim & Campean, 2014), a state is a generic object which can be described by a set of multiple measurable attributes, rather than a single attribute (see Figure 7). For example, sunlight (solar energy) can possess multiple attributes i.e. solar power, solar intensity or solar temperature etc. In the future, it

will be researched on how *multiple attributes* on both single and *multiple operands* along with *multiple constraints* can be managed within the same approach and it will be assessed how this can impact the integrity of the approach. This may change the structure of morphological chart since it currently supports in identifying the concepts based on a function having one input-output states pair information underpinning a single attribute but multiple constraints.

References

Bonev, M, Wörösch, M, Hauksdóttir, D and Hvam, L. (2013) Extending Product Modeling Methods for Integrated Product Development." Proceedings of the 19th International Conference on Engineering Design (ICED13)-: Design for Harmonies. Seoul, Korea, Republic of: The Design Society,
219–28.

Browning, T. R. (2001) Applying the design structure matrix to system decomposition and integration problems a review and new directions. *IEEE Transactions on Engg. Management*, 48(3): 292-306.

Blackenfelt, M. (2001) *Managing complexity by product modularisation*. PhD Thesis, Royal Institute of Technology, Stockholm, Sweden.

Campean, F., Henshall, E., & Rutter, B. (2013) Systems engineering excellence through design an integrated approach based on failure mode avoidance *SAE Int. J. Mater. Manf.*, 6(3):389-401.

Campean, F., Henshall, E., Yildirim, U., Uddin, A., & Williams, H. (2013) A structured approach for function based decomposition of complex multi-discipinary systems *Proceedings of the 23th CIRP design conference*. 113-123.

Cao, D.X., and Fu, M.W., 2011. A knowledge-based prototype system to support product conceptual design. *Computer Aided Design & Applications*, 8(1), 129-147.

Chmarra, M.K, Cabrera, A.A.A, van Beek, T., D'Amelio, V., Erden, M.S., &Tomiyama, T. (2008) Revisiting the Divide and Conquer Strategy to Deal with Complexity in Product Design. *Mechatronic and Embedded Systems and Applications, IEEE/ASME International Conference*: 393 – 398.

Deubzer, F. & Lindemann, U. (2009) Product architecture definition and analysis using matrix-based multiple-domain approach. *Proceedings of ASME IDETC/CIE*. 1197-1205.

Gero, J. (1990) Design prototypes a knowledge representation scheme for design, *AI Magazine*, 11(4): 26-36.

Hamraz, B., Hisarciklilar, O., Rahmani, K., Wynn, D.C., Thomson, V., & Clarkson, P.J., (2013) Change prediction using interface data. *Concurrent Engineering: Research and Applications* 21: 141–154.

Harel, D. (1987) Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8: 231-274.

Hubka V and Eder WE (1996) *Design science: introduction to the needs, scope and organisation of engineering design knowledge*, London, Springer-Verlag.

Jarratt, T. A. W., Eckert, C., & Clarkson, P.J. (2004) Development of product model to support engineering change management. *Proceedings of TMCE* April Switzerland

Ko, Y.T. (2013) Optimizing product architecture for complex design. *Concurrent Engineering: Research and Applications* 21(2) 87–102.

Komoto, H., & Tomiyama, T. (2011) A theory of decomposition in system architecting. *International Conference on Engineering Design*

Konig, C., Kreimeyer, M., & Braun, T. (2008) Multiple-domain matrix as a framework for systematic process analysis. 10th *Int. Design Structure Matrix Conf.* November Stockholm Sweden

Lin, L and Chen, L-C. Constraints modelling in product design. *Journal of Engineering Design*, 2002 13(3): 205-214.

Lindemann, U., Maurer, M., & Braun, T. (2009) *Structural Complexity Management*. Springer Berlin Heidelberg.

Martin, M. V. and Ishii, K. (2002) Design for variety developing standardized and modularized product platform architectures. *Research in Engineering Design*, 13: 213-235.

Otto, K., Wood, K (2001) *Product Design: Techniques in Reverse Engineering and New Product Development.* Prentice-Hall, New Jersey.

Pimmler, T. U. & Eppinger, S. D. (1994) Integration analysis of product decompositions. *ASME Design Theory and Methodology Conference*, September Minneapolis.

Pahl, G, Beitz, W., Feldhusen, J., & Grote, K.H. (2007) *Engineering design a systematic approach*, Springer.

Rahmani, K., Thomson, V., (2012) Ontology based interface design and control methodology for collaborative product development. *Computer Aided Design*, 44(5): 432–444.

Ravn, P. M., Gudlaugsson, T.V., Mortensen, N.H., (2015) A multi-layered approach to product architecture modeling: Applied to technology prototypes. *Concurrent Engineering: Research and Applications* 1–14.

Sosa, M.E., Eppinger, S.D., Rowles, C.M., (2003) Identifying modular and integrative systems and their impact on design team interactions. *Journal of Mechanical Design*, 125: 240–252.

Suh, N. P. (1998) Axiomatic Design Theory for Systems. *Research in Engineering Design*, 189-209.

Uddin, A., Campean, F. & Khan, M.K. (2015) Development of interface analysis template for system design analysis. *International Conference on Engineering Design (ICED)*, Milan, Italy.

Uddin, A., Khan, M.K., & Campean, F. (2014) Function-based conceptual design expert (CDE) systems. *Applied Mechanics and Materials* 564: 590-596.

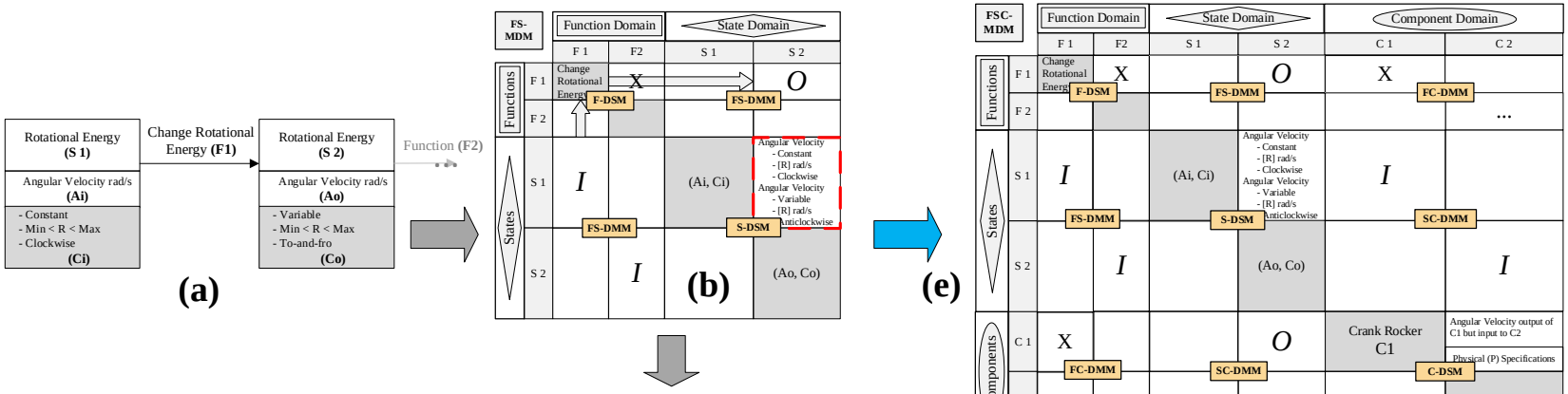
Ulrich K (1995) The role of product architecture in the manufacturing firm. *Research Policy* 24(3): 419–440.

Ulrich, K. T. & Eppinger, S. D. (1994) *Product design and development*. McGraw Hill.

Umeda, Y., Ishii, M., Yoshika, M. & Tomiyama, T. (1996) Supporting conceptual design based on the function-behaviour-state modeller. *Artificial Intelligence for Engg Design, Analysis, and Manuf.* 10(4): 275-288

Yildirim, Y. & Campean, F. (2014) Development of a structured approach for decomposition of complex systems on a functional basis. *27th International Conference on CADCAM, Robotics and Factories of the Future*, IOP Publishing.

Woldemichael, D. E., & Hashim, F. M. (2011) A framework for function-based conceptual design support system. *Journal of Engineering Design and Technology*, *9*(3), 250-272.



	C1: Functions	C2: Operand States		C3: Attributes		C4: Constraints						C5: Concepts
						Constancy		Track		Range		
		In	Out	Input	Output	Input	Output	Input	Output	In	Out	
R 1	Change Rotational Energy (R.E)	R.E	R.E	Angular Velocity	Angular Velocity	Constant	Variable	CW	ACW	[R]	[R]	Gear Pair Box
R 2	Change Rotational Energy (R.E)	R.E	R.E	Angular Velocity	Angular Velocity	Constant	Constant	CW	CW	[R]	[R]	Pulley Belt Mechanism
R 3	Change Rotational Energy (R.E)	R.E	R.E	Angular Velocity	Angular Velocity	Constant	Variable	CW	To and fro	[R]	[R]	Crank Rocker
R 4	Change Rotational Energy (R.E) to Translational Energy (T.E)	R.E	T.E	Angular Velocity	Linear Velocity	Variable	Variable	ACW	Axial	[R]	[R]	Screw Mechanism
R 5	Convert Electrical Energy (E.E) to Rotational Energy (R.E)	E.E	R.E	Current	Angular Velocity	Constant	Constant	DC	CW	[R]	[R]	DC-Motor
R 6	Convert Electrical Energy (E.E) to Rotational Energy (R.E)	E.E	R.E	Current	Angular Velocity	Constant	Variable	AC	CW	[R]	[R]	AC-Motor
R 7	Change Electrical Energy (E.E)	E.E	E.E	Current	Current	Constant	Constant	DC	AC	[R]	[R]	DC-AC Inverter
R 8	Convert Solar Energy (S.E) into Electrical Energy (E.E)	S.E	E.E	Solar Power	Current	Constant	Constant	Cross-flow	DC	[R]	[R]	Solar Board
R 9	Convert S.E to E.E	S.E	E.E	Solar Power	Current	Constant	Constant	Cross-flow	DC	[R]	[R]	Solar Board
R 10	Convert S.E to E.E	S.E	E.E	Optical Current	Current	Variable	Variable	Radiated	DC	[R]	[R]	Solar Battery

Key: CW: Clockwise, ACW: Anticlockwise, DC: Direct Current, AC: Alternating Current, [R]: Range for minimum & maximum values

(c)

Figure 8. Information flow between tools of proposed integrated approach