Development of a structured approach for decomposition of complex systems on a functional basis

Unal Yildirim[1] and I. Felician Campean[1]

[1]Bradford Engineering Quality Improvement Centre (BEQIC),
University of Bradford, United Kingdom, BD7 1DP

E-mail: u.yildirim@student.bradford.ac.uk

Abstract. The purpose of this paper is to present the System State Flow Diagram (SSFD) as a structured and coherent methodology to decompose a complex system on a solution-independent functional basis. The paper starts by reviewing common function modelling frameworks in literature and discusses practical requirements of the SSFD in the context of the current literature and current approaches in industry. The proposed methodology is illustrated through the analysis of a case study: design analysis of a generic Bread Toasting System (BTS).

1. Introduction

Product development (PD) organizations are facing the increasing challenge of engineering of complex systems driven by the pressure for delivery of customer requirements and the need to address environmental and safety concerns (e.g. emission standards for an automotive manufacturer) within a highly competitive market.

Within a customer focused engineering approach, systems engineering design must focus on robust and reliable delivery of customer required functions. Therefore, function analysis plays a central role in the systems engineering design process. It has been recognized that the increasing complexity of systems places a significant challenge for function analysis [1,2], in particular due to the complicated nature of systems interdependencies. Robust systems engineering design must ensure that all systems interfaces are identified, characterized and managed through functional requirements both at the system level and subsequent levels of systems decomposition [3].

Structured approaches to design, based on the axiomatic design framework [4], make the clear argument that a separation between functional domain and the design domain must be maintained, and design development should take place through zig-zagging between the functional domain and the design domain. This ensures that functions are articulated in a solution-neutral manner, and that decomposition is pursued in the functional domain first. However, most engineering practice focused design approaches deviate from this fundamental principle, in that the decomposition of systems is pursued on structural basis, commonly carried out by clustering the elements into architectural and team chunks on a design structure matrix (DSM) [5,6]. The authors' observation of current systems engineering design practice in the automotive industry has highlighted the prevalence of this approach in the decomposition of systems; a clear element for this is that design responsibilities are typically allocated to engineering teams on either a disciplinary basis (mechanical, electrical, software) or structural basis relating to design units or chunks (e.g. taking an automotive example - body, powertrain, chassis), and not functions, which does not guarantee that a flawless systems integration is achieved. There is overwhelming evidence from industry [7] that most field problems occur at systems interfaces which have not been appropriately managed at the design stage.

Another weakness of current practice in function analysis of complex systems stems from the common reliance on brainstorming as the basis for identification of functional requirements [3]. While brainstorming can be a powerful tool, it cannot guarantee completeness of functional requirements. The introduction of structured approaches for systems decomposition based on either the DSM [5] or reverse engineering [8] improve the effectiveness of brainstorming, however, this deviates from the axiomatic design principle that analysis in the function domain should take place first, and often compromises the solution neutral articulation of the function. The effect of this is that opportunities for driving innovation in the design are missed.

The System State Flow Diagram (SSFD) has been recently introduced [9] as a framework for functional representation and decomposition of a complex system on a solution-independent functional basis. This paper presents research to enhance structure of the SSFD framework, focusing in particular on the integrity of the analysis of flows through the system, as well as functional representation and articulation. The organization of the paper is as follows: next section reviews current function modelling frameworks in literature; section 3 introduces the proposed SSFD framework, discussed along with a case study based on the design analysis of a generic Bread Toasting System (BTS); section 4 reflects on the merits of the proposed approach, and outlines further research work.

2. An Overview of Function Modelling Frameworks

An engineered system is generally represented in terms of an input-output relationship and the function of the system is described with respect to this relationship [10]. The system decomposition is developed on the basis of this input and output. The identification of the inputs and the outputs in terms of the flows of energy (E), material (M) and information (I) is common in literature [8,11-12]. The system decomposition is managed based on mapping the flows of energy, material and information through the system; sub-functions are defined as successive operations on flows, with output from one sub-function providing the input to the next [11].

The Functional Basis (FB) model of Otto and Wood [8] uses the flows of energy, material and information for design decomposition of a system. The FB model represents the overall function of a system in terms of interconnected sub-functions which are defined as operations on the flows of E, M and I. The model provides a consistent way for the description of the flows and the functions by introducing a taxonomy for both. The function is described in verb-object form [8]. Design development is carried out through successive decompositions of functions and sub-functions into lower level sub-functions which can be referred to an iterative decomposition in the functional and the design solution domains in the context of axiomatic design [4]. While the FB model is limited to the decomposition of a system based on the flows at successive levels, system integration can be supported at any level by utilizing a DSM based approach [5], which can be used in the identification of the relationships between design solutions [13] or grouping design solutions into structure-function units [6].

Several methods have also addressed visualization of functional structure of a system on the basis of flows, but in different manners, e.g. Function Flow Block Diagram and Integrated Definition for Function Modeling (IDEF0) [14].

The Contact and Channel Approach (C&C^2-A) [15] describe the input and the output of a system as a Working Surface Pair (WSP) and specifies a Channel and Support Structure (CSS) to connect two WPSs. The C&C^2-A suggests that the description of a function should include at least two WSPs, the connecting CSS and at least two Connectors which embed the model into the environment. A WSP is described as a state characterized by measurable attributes and the function of a system is described as a transfer of one state into another [16]. Design decomposition is carried out through describing the basic elements WSP, CSS and Connectors at different levels of the system.

The Object-Attribute-Function (OAF) framework [17] and TRIZ [18] propose similar approaches. The Object-Attribute-Function (OAF) framework defines the input and the output as generic objects which are described by their measurable attributes (e.g. mass), with a clear taxonomy developed to describe both. TRIZ describes a function with respect to interaction between two components of a system and identifies an element which provides the interaction.

Object-Process Methodology (OPM) [19] represents a system both graphically (Object-Process Diagram (OPD)) and textually (Object-Process Language (OPL)). The method describes a function as a function sentence which begins with the list of processes followed by objects and represents the function as a box named function box that encloses at least one object and one process. The method can represent a system at any level through its complexity management mechanisms (e.g. in-zooming/out-zooming).

Description of the functional structure of a system with respect to its intended behavior and structure at different phases of the design process is proposed by [20-22]. The FBS framework [20] suggests that functions are defined in relation to states of the structures (design objects), which are represented by entities, their attributes, and relations between entities. Within FBS the functions are defined by the combination of verb-object which relate to the designer intentions, and behaviors expressed through adjectives, which can instantiate the function, thus embedding the time dimension into the function definition.

Several methods possess similar characteristics, e.g. description of a function in verb-object format. Implementation of some methods in engineering practice lends itself to useful software implementation and automation. The FBS framework is an example of this. Decomposition of a complex system in a solution independent manner is a generic issue which is not addressed by any framework. System decomposition tends to be based on brainstorming at any level, i.e. by asking the question “how is this function achieved?”

Based on a review of current methods and practices in industry, which emphasized the need for a structured and coherent methodology for the decomposition of a complex system on a solution-independent functional basis, the requirements for a functional framework have been summarized as follows [23]:

- To be integrated with other tools commonly used in industry to encourage broad take-up of the framework;
- To have a graphical representation to provide shared understanding of the analysis of a system;
- To promote the primacy of function and solution-neutral thinking in systems engineering design analysis;
- To be based on tools and methodology which can be applicable across disciplines (i.e. electro-mechanical, control and software).

3. System State Flow Diagram

3.1. The Basis of the System State Flow Diagram

An engineered system is commonly represented as a system block diagram, which is a black-box representation of the system [11]. The overall function of the system is shown in the box, with the inputs and outputs to the system also indicated on the box. Figure 1 illustrates a system block diagram for a Bread Toasting System (BTS), which will be used as a reference case study.

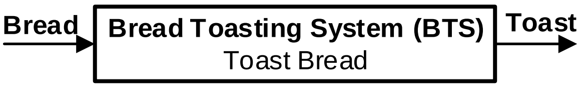

Figure 1. Representation of a Bread Toasting System as a system block diagram

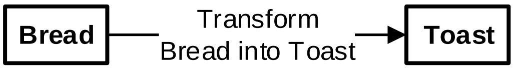

Figure 2. A system state diagram of a BTS

Most function modelling frameworks define the inputs and the outputs of the system in terms of the flows of energy, material, or information [10]. The function of the system is commonly articulated in verb-noun format in relation to flows.

The SSFD, illustrated in Figure 2, follows the general principles of state diagrams [24]: a box denotes a state and an arrow denotes the function required to achieve the state transition. Coherent with [17,19-20], a state can be thought of as a generic object described by a set of measurable attributes. The function can be described in verb-object structure which is common practice in literature (see [10] for a review). Coherent with [17], this characterization is related by the rule that the verbs correspond to the operations on the attributes and the nouns to the objects. The function is articulated in respect of relevant object attribute change required to transition between states.

3.2. Functional Decomposition of a System Based on the System State Flow Diagram

The SSFD starts by generation of a high level system state flow diagram which represents input and output states of the system. This follows the identification of the main flow through the system. This requires the identification of intermediate states between the input state and the output state described in the high level system state flow diagram. An intermediate state is thought of as an object and its measurable attributes like for the input state and the output state. The flow of the states is mapped through the system and then functions required to achieve these state transitions are articulated. Next, the auxiliary flows of the system are identified based on the main flow. The structure of the auxiliary flows follows the same logic and structure as the one used for the main flow. Once all the flows of the system are identified they are aggregated into a single model. Lastly, design elements are identified for the fulfilment of the functions. In this paper, all design decisions will be taken based on the structure of a common slotted toaster as an example.

Step 1: Generate a high level system state flow diagram

The SSFD provides a high level representation of a system as a state diagram, showing the input state and the output state to the system. From an engineering point of view, the analysis should start by defining the fundamental working principle of the system to describe its input and output states. In fundamental terms bread toasting is achieved by heating the bread surface to 310°F [25], when chemical transformations (known as the Maillard reaction) in the bread are triggered generating the characteristic flavours of toast. Further analysis of physics shows that increasing the temperature of a slice of bread changes its size (i.e. length and thickness), temperature, weight, composition (i.e. sugars and starches start to caramelize and turn brown) and moisture capacity (i.e. vaporization of the moisture of the bread which directly reduces the weight of the bread). Location of the bread/toast (i.e. transport, orientation, position, etc.) is also considered during transition between states. The system should address these attributes for the generation of a slice of toast. Using this information the high level SSFD for the BTS can be updated, as shown in Figure 3.

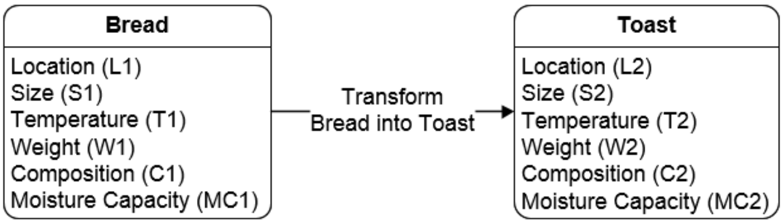

Figure 3. BTS high level system state flow diagram

Step 2: Create the main flow through the system for the input-output state

The main flow through the system represents the system`s required functionality by identifying required state transitions with reference to the input and the output states of the system shown in

Figure 3. As discussed, the fundamental working principle of the system is associated with the increase of the bread temperature. A design decision should be made to heat the bread. There are three basic ways of heat transfer: conduction, convection, and radiation and there are a variety of ways of increasing the temperature of the bread on the basis of these modes, e,g. over an open fire, focused sunlight and radiant heat. A common slotted toaster uses radiant heat to heat the bread. In this case, the system must apply radiant heat directly to the bread slice [25]. The usage of radiant heat in the system points out the requirement for the orientation of the bread relative to the heat source. According to this requirement, the flow should start by positioning the bread relative to the heat source (i.e. load bread). Once the bread is toasted (i.e. heat bread) it comes out of the system (i.e. eject toast). Figure 4 shows the main flow through the system. The box around the system defines the "system boundary" which shows the limits of the scope for responsibility for the design team.

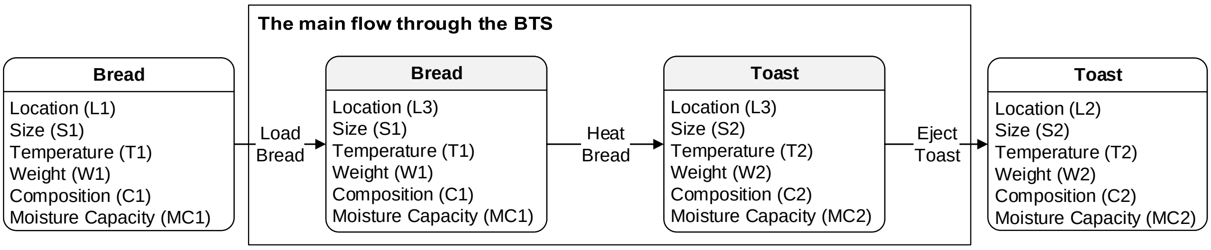

Figure 4. The main flow through the BTS

Step 3: Identify the auxiliary flows based on the main flow of the system

As mentioned earlier, the BTS engineering principle is associated with the increase of the bread temperature through applying radiant heat directly to the bread slice. This step requires a way of delivering the function `heat bread` through generating radiant heat from a given energy source. A variety of energy sources (e.g. mains electricity, sunlight) can be considered to achieve the function `heat bread`. A common slotted toaster uses mains electricity as energy source, which should be converted into radiant heat. Figure 5 represents the high level SSFD for the auxiliary flow.

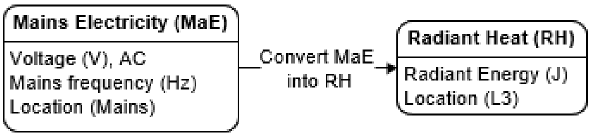

Figure 5. The high level system state flow diagram for the auxiliary flow

Reflecting the energy source choice made (i.e. mains electricity supply), the auxiliary flow should provide "controlled supply of mains electricity" (i.e. conversion of AC into DC) and then its conversion into Radiant Heat (i.e. convert EE into RH) which is used (i.e. channel RH) to achieve the function `heat bread`. Figure 6 shows the auxiliary flow through the BTS.

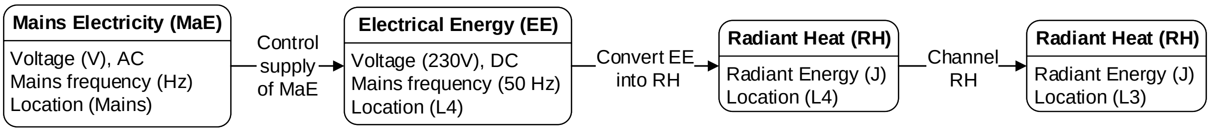

Figure 6. The auxiliary flow through the BTS

Step 4: Aggregate function chains into a functional model

In this step, all of the function chains from Step 2 and Step 3 are aggregated into a single model. Figure 7 shows the SSFD for the BTS. The dashed state is used to connect the distinct chains together.

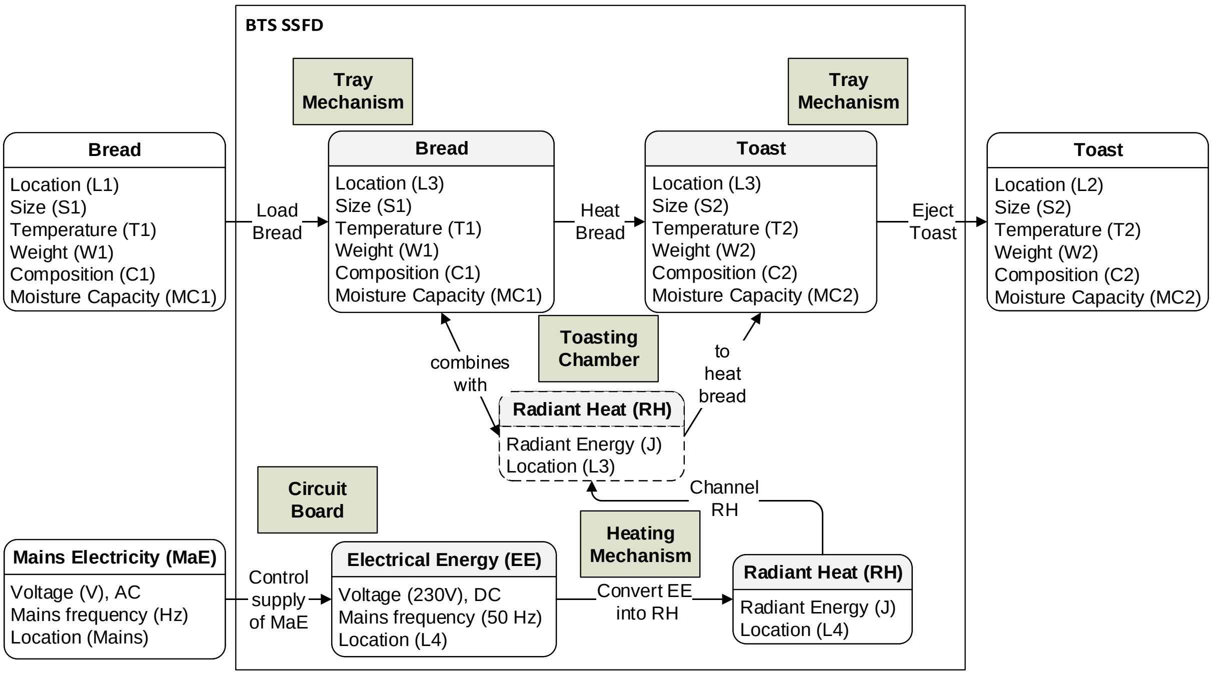

Figure 7. BTS SSFD

An engineered function can completely be defined in terms of the triad of an input state, an output state and a design element required to transition between states coherent with [16-18]. Figure 7 includes design elements for the fulfillment of the functions.

From this SSFD representation a high level BTS Function Tree and a BTS boundary diagram can be directly extracted as shown in Figure 8 and in Figure 9 respectively.

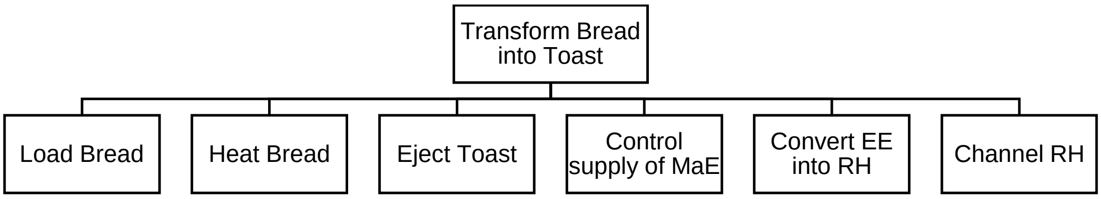

Figure 8. BTS Function Tree

A System Boundary Diagram (SBD) is a graphical representation of the system and it includes the design elements identified on the SSFD along with the flows of energy, material and information required to transform system inputs into system outputs [11].

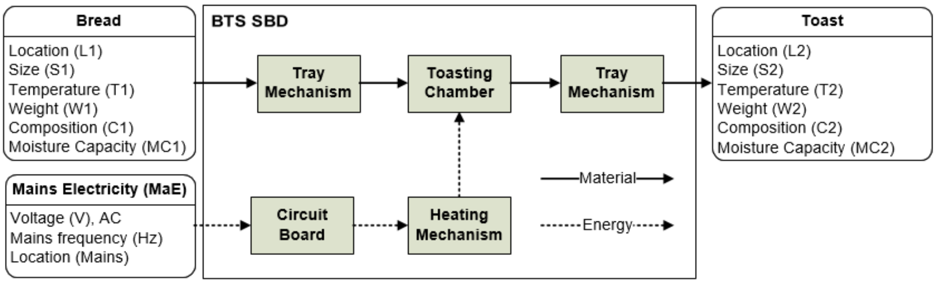

Figure 9. BTS SBD

The diagram is extracted from the BTS SSFD by deleting all information other than Input/Output states and Design elements boxes. Then, the design elements boxes are joined with arrows denoting the flows of Energy, Material and Information between them.

4. Discussion and Conclusion

The aim of this paper was to present the System State Flow Diagram (SSFD) as a structured framework for the decomposition of a complex system on a solution-independent functional basis. The SSFD draws from the methodology of Paul and Beitz [11] and Otto and Wood [8] for system decomposition, and from the C&C^2-A [16], the OAF framework [17] and TRIZ [18] for function modelling. The SSFD offers a more structured way for system decomposition based on its function representation model, addressing the weaknesses of the current practice, discussed in the first section of the paper.

Through the implementation of the SSFD on the Bread Toasting System case study it is demonstrated that the requirements outlined in section 2 are met, as discussed below:

- The SSFD can be integrated with other tools currently used in the systems engineering design. A function tree can be extracted from a SSFD in a solution neutral way rather than conduct brainstorming (How-Why) which is the common approach in practice [8]. A SSFD also facilitates the development of a system boundary diagram which enables interface functions to be identified through interface analysis [7] (see [23] for an example).
- It improves communication within the engineering team in the sense that its system representation is easy to understand and portable across multiple engineering disciplines. Thus it supports the achievement within an engineering team of a common understanding of the functional decomposition of the system in a solution neutral way.
- Thinking of the input and the output in terms of an object and its measurable attributes is an important feature of the SSFD which divorces the consideration of function from the consideration of the design solution, that is, it promotes solution independent thinking in the design of a system, for example, a slice of bread can be toasted in a variety of ways (e.g. Electromagnetic waves, over an open fire) as long as the bread's surface temperature reaches about 310°F.

Further work will concentrate on the development of the SSFD to promote identification of further functional requirements through its integration with the interface analysis method [7] to provide a comprehensive interface management to ensure all system interfaces are identified, characterized and managed through function requirements both at the system level and the subsystem level.

References

[1] Lu SC-Y and Suh N-P 2009 Complexity in design of technical systems *CIRP Annals* **58** pp 157-160

[2] Tomiyama T, D`Amelio V, Urbanic J and ElMaraghy W 2007 Complexity of multi-disciplinary design *CIRP Annals* **56**(1) pp 185-188

[3] Campean I F and Henshall E J 2012 Failure mode avoidance in automotive systems engineering design *Proceedings of the 2nd International Workshop on Modelling and Management of Engineering Processes (MMEP)* P Heisig, P J Clarkson eds. University of Cambridge Cambridge UK pp 15-29

[4] Suh N P 2001 *Axiomatic design: advances and applications* (New York: Oxford University Press)

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

[6] Pimmler T U and Eppinger S D 1994 Integration analysis of product decompositions *Proceedings of the ASME 6th International Conference on Design Theory and Methodology* Minneapolis

[7] Webb R D 2002 *Investigation into the application of robustness and reliability tools to the design process* MSc thesis University of Bradford

[8] Otto K and Wood K 2001 *Product design: techniques in reverse engineering and new product development* (New Jersey: Prentice Hall)

[9] Campean I F, Henshall E, Yildirim U, Uddin A and Williams H 2013 A Structured Approach for Function Based Decomposition of Complex Multi-disciplinary Systems *Proceedings of the 23th CIRP design conference* M Abramovici, R Stark eds. Bochum Germany pp 113-123

[10] Srinivasan V, Chakrabarti A and Lindemann U 2012 A framework for describing functions in design *Design methods: proceedings of the 12th international design conference* Dubrovnik 21-24 May the Faculty of Mechanical Engineering and Naval Architecture University of Zagreb pp 1111-1122

[11] Pahl G, Beitz W, Feldhusen J and Grote K H 2007 *Engineering design: A systematic approach* 3rd edn. (London: Springer)

[12] Ullman D G 2010 *The Mechanical design process* 4th edn. (Singapore: McGraw-Hill)

[13] Jarratt T A W 2004 *A Model-based approach to support the management of engineering change* Ph.D. thesis University of Cambridge Cambridge

[14] Department of Defence System Management College 2001 *Systems Engineering Fundamentals* (Virginia: Defense Acquisition University Press)

[15] Matthiesen S and Ruckpaul A 2012 New Insights on the Contact&Channel-Approach – Modelling of Systems with Several Logical States *Proceedings of the International Design Conference (DESIGN)* Dubrovnik Croatia pp 1019-1028

[16] Albers A, Oerding J and Alink T 2011 Abstract Objectives can Become More Tangible with the Contact and Channel Model (C&CM) *Proceedings of the 20th CIRP Design Conference* A Bernard ed. Springer Berlin pp 203-213

[17] Sickafus E N 1997 *Unified Structured Inventive Thinking: How to Invent* (Michigan: Ntelleck)

[18] Fey V and Rivin E 2005 *Innovation on Demand* (New York: Cambridge University Press)

[19] Dori D 2002 *Object-Process Methodology: a Holistic Systems Paradigm* (Germany: Springer)

[20] Umeda Y, Ishii M, Yoshioka M, Shimomura Y and Tomiyama T 1996 Supporting Conceptual Design based on the Function-Behaviour-State Modeller *AI EDAM* **10**(4) pp 275-288

[21] Gero, J S 1990 Design prototypes: A knowledge representation schema for design *AI Magazine* **11**(4) pp 26-36

[22] Goel A, Rugaber S and Vattam S 2009 Structure, behaviour, and function of complex systems: the structure, behaviour, and function modelling language *AI EDAM* **23**(1) pp 23-35

[23] Campean I F, Henshall E, Brunson D, Day A, McLellan R, and Hartley J 2011 A Structured Approach for Function Analysis of Complex Automotive Systems *SAE Int. J. Mater. Manuf.* **4**(1) pp 1255-1267

[24] Harel D 1987 Statecharts: A Visual Formalism for Complex Systems, *Science of Computer Programming* **8** pp 231-274

[25] Mital A, Desai A, Subramania A and Mital A 2008 *Product Development: a Structured Approach to Consumer Product Development, Design and Manufacture* (Oxford: Butterworth-Heinemann)