Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 50 (2016) 210 – 215

www.elsevier.com/locate/procedia

26th CIRP Design Conference

Function Allocation Theory for Creative Design

Tetsuo Tomiyama[a]*

[a]*Cranfield University, Cranfield MK43 0AL, UK*

* Corresponding author. Tel.: +44-1234-750111 x5610; fax: +44-1234-754605. *E-mail address:* t.tomiyama@cranfield.ac.uk

Abstract

Function structure influences on systems architecture (or product architecture). This paper discusses a design method for creative design solutions that focuses on the allocation of functions. It first proposes a theory called "Function Allocation Theory" to allocate a function to an appropriate subsystem or component during the systems decomposition phase. By doing so, the complexity of design solutions can be reduced. The theory is applied to some examples including collaborative robots and robotics maintenance. Finally, the paper illustrates a case study of designing a reaction-free fastening system using this theory.

Peer-review under responsibility of the organizing committee of the 26th CIRP Design Conference

Keywords: Function; modelling; architecting

1. Introduction

Modern products are increasingly becoming more complex due to their size, advanced technologies, multi-disciplinarity, and social and economic circumstances. To develop such products, architecture plays a critical role and complex systems architecting is a useful method [1, 2]. However, systems engineering hasn't established a strong link with, for example, more technically oriented design methodologies [3].

Komoto, et al. [4, 5] described a series of work to develop a systematic method for system decomposition within the V-model from functional requirements (see Fig. 1). The V-model [6] is a well-accepted model to develop complex systems, although its drawbacks and deficiencies are recognised.

Fig. 1 illustrates the V-model in the context of complex systems architecture [1]. In the systems decomposition (architecting) phase, the focus of the design moves from the high-level abstract top level to the systems level and then down to the detailed component level. At the bottom level, components are designed and implemented. This is followed by the integration phase in which components and subsystems will be integrated while being verified and validated.

All through this architecting process, systems requirements will develop into systems decomposition using functions, which defines "functional architecture". Function allocation theory facilitates this architecting process in which global functional structure will be defined, designed, integrated, verified and validated.

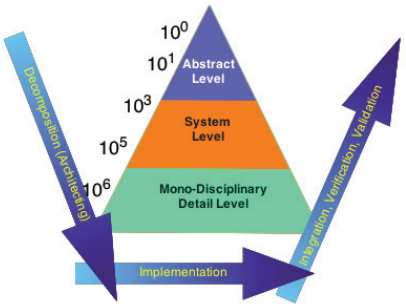

Fig. 1. Systems architecture and the V model [1].

This paper is organised as follows. The next section illustrates a systematic method and a computer-based tool called SA-CAD that can systematically generate architectural variations (solutions) [4, 5]. It uses a function modelling method called the FBS (Function-Behaviour-State) modelling [7]. Section 3 proposes a new functional design method called

Peer-review under responsibility of the organizing committee of the 26th CIRP Design Conference
doi:10.1016/j.procir.2016.04.060

"function allocation theory" that can be used for systems architecting. To explain the concept of function allocation, two cases of robot design will be depicted as examples. Section 4 presents a case study of designing a reaction-free fastening system to demonstrate the power of the function allocation theory. Section 5 concludes the paper.

2. Function Modelling and Systems Architecting

2.1. Function-Behaviour-State modelling

The starting point of this research is the FBS (Function-Behaviour-State) modelling and the FBS modeller [7]. From functional requirements, a metamodel [8] will be built with the FBS modeller. This metamodel describes a product with a network of concepts including function, behaviour, physical phenomenon, state, attribute, and entity. The FBS modeller is connected to a qualitative reasoning system based on Qualitative Process Theory [9] to reason about physical behaviour of the object.

2.2. Systems architecting

Fig. 2 depicts the systems architecting process using SA-CAD proceeds as follows [4, 5]. This process can be carried out with two methods. First, a function level metamodel is built from functional requirements. The architect will then decompose the top level function into subfunctions until each subfunction can be associated with "building blocks" such as physical features as embodiment. A building block might be a physical feature that contains physical phenomena and a set of entities, parameters, and relationship among these concepts. The collection of these building blocks represent a design solution or an architectural variation.

The second method of systems architecting kicks in when these building blocks are not available. In this case, from functional decomposition, appropriate behaviours as state changes will be instantiated. These state changes are triggered by physical phenomena. Therefore, a design solution is a network of parameters (attributes) that can induce these state changes. The parameter network will be then chunked into clusters which signify embodiment. This clustering of parameters is carried out in such a way that for each of clusters a physical entity is assigned from the component database.

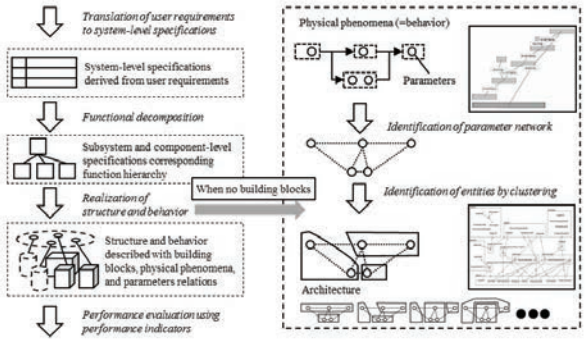

Fig. 2. Conceptual design process [5].

Either way, these building blocks (in the former method) or embodied entities after clustering parameters (in the latter) collectively perform the functional requirements. Each architectural variation performs the same total function using the same set of physical phenomena and has the same subfunctions. However, how each architectural variation is structured in terms of entities can be completely different.

3. Function Allocation Theory

The systems architecting method implemented in SA-CAD assumes that it generates all possible architectural variations and then best ones will be selected. Fig. 3 shows a screen hardcopy of SA-CAD [4]. It is suitable for exhaustive search of variations at conceptual design stages, while SA-CAD doesn't allow designation of a particular function to a particular entity.

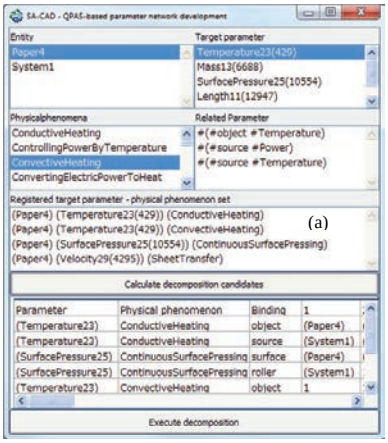

Fig. 3. SA-CAD [4].

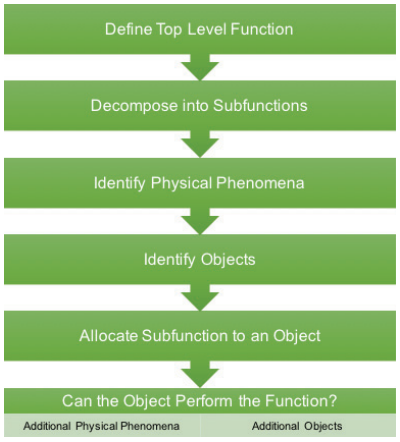

Fig. 4. Function allocation process.

Function allocation does this designation. Fig. 4 shows the process of function allocation. Given functional requirements, a top level function is first decomposed into subfunctions and then these subfunctions are intentionally "allocated" to

available entities that are identified in the decomposition process. As a design method, function allocation can generate creative architectural solutions, because this designation can be “arbitrary” and “out-of-the-box”.

This is the value of function allocation. However, because of this, allocating a function to an entity does not necessarily guarantee that this function can be realised by the allocated entity. Therefore, this function allocation generates a lower level design problem, which aims at a detailed architecture that may come with new additional physical phenomena and objects.

4. Case Studies: Toward Robotic Maintenance

The function allocation theory is now explained in the context of maintenance robot design as well as an innovative fastening system suitable for robotic maintenance.

4.1. Robotic maintenance

Among various life cycle phases, maintenance is one of the most labour intensive, hence expensive life cycle phase. To this end, approaches of robots and autonomous systems are considered promising. However, the use of robots in maintenance is yet limited compared with production and other tasks for a variety of reasons [10]. At this moment, the majority of maintenance robots are limited to inspection, monitoring, or servicing robots. Many of them are remotely controlled and often dragging power cables [11].

As an activity, maintenance is difficult to automate for four reasons [12, 13]. First, maintenance is irregular in that failures happen quite stochastically. Second, maintenance is not uniform in that every failure is different. Third, maintenance is non-deterministic in that a maintenance operation may change the system’s state. This means that the consequence of of one operation determines subsequent maintenance operations. Forth, maintenance is not standardised and requests often specialised tools and methods.

Due to these, intelligence for maintenance must deal with a variety of situations flexibly. This means a general purpose maintenance robot requires huge general maintenance intelligence. Therefore, although expectations for truly versatile intelligence are high, it is extremely difficult to develop such a maintenance robot that can deal with almost any failures and situations.

Consequently, an approach that combines “autonomous maintenance” and “robotic maintenance” could be promising, because of the possibility of reducing complexity associated with the development of maintenance intelligence [10]. Autonomous maintenance is a way of maintenance performed by a system that is equipped with sensors, controller, and actuators and responding to the internal and environmental changes without human interventions.

This means the machines to be maintained will be equipped with maintenance intelligence specialised for the machine rather than the maintenance robot is equipped with versatile maintenance intelligence. In this way the necessary knowledge can easily be limited to only the one about the machine to be maintained. This local intelligence controls and conducts maintenance by collecting data through embedded sensors and by using robots as a way to deliver maintenance capabilities (such as inspection, exchanging components, adjustments, etc.).

4.2. Collaborative autonomous maintenance robots

Consider a robot arm whose function is “to transport something”. The arm is connected to a hand that can grab an object (something) and transports it from position P_1 to P_2. Here, subfunctions are “to grab”, “to lift”, and “to move”. Physical phenomenon, “moving fingers”, will close fingers for grabbing, whereas changing values of various joint parameters of the arm will realise behaviour of lifting the object and moving the arm.

Now imagine a situation in which two or more autonomous robots (each equipped with one arm) collaboratively grab and transport an object. In Fig. 5, the object (i.e., the black door) needs to be lifted and opened for a maintenance purpose. Since the door is too heavy, this task requires at least two robots. The door has two handles which is equipped with a sensor to detect if a robot hand grabs the door at the handle. First, the door issues a help to open the door. To do so, it calls for a robot to come to the position of a handle and waits until two robots come to these two positions. Once the presence of two robots is confirmed, the door tells the robots to lift the door to open it. Once the door is lifted, then actual maintenance tasks can begin.

Fig. 5. Collaborative two robots.

Apparently, this requires not only grabbing and transporting an object with visual assistance with a camera, but also synchronising the timing to grab the object and making a decision to start moving the arms simultaneously. Therefore, the collaboration function requires other subfunctions such as “to synchronise timing”, (two arms) “to communicate”, and “to make a decision about timing”.

At this level of abstraction, there are three objects that can be identified; the surrounding environment, two arms, and the object to be transported (the door). Obviously the two arms provide the transportation (including grabbing and lifting) function. The environment sets up the support and usually provides no other function. In a traditional design case (Design A), the function of communication (between the arms) and synchronising the timing as well as decision making that are necessary for collaboration can be allocated to

the arms. Therefore, the arms need to be equipped with, e.g., a CPU, sensors, and communication devices. The object plays a passive role of just being transported.

This traditional "allocation" of function can be re-investigated (Table 1). A new design approach would be to "re-allocate" these subfunctions to the environment or the object. For example, Design B would assume that the arms perform the transportation subfunction, while the object can perform the collaboration subfunction (including the synchronisation and decision making subfunctions). The communication subfunction needs to be distributed among the arms and the object. As is Design A, in Design B the environment plays no role.

Table 1. Different function allocation possibilities.

Design Variation	Environment	Arms	Object
A	N/A	To transport To communicate To synchronise To make a decision	N/A
B	N/A	To transport To communicate	To synchronise To make a decision To communicate
C	To communicate To synchronise To make a decision	To transport To synchronise To communicate	N/A

Another design variation (Design C) would be that the arms perform the transportation subfunction, whereas the other functions would be distributed to the object and the environment. For example, the decision making as well as sensing for synchronisation (e.g., camera) can be carried out by the environment. This design is a more drastic design in which the environment makes the decision and the synchronisation can be done jointly by the environment and the arms.

These three design variations can be compared in the followings. Design A embodies the traditional concept the robot arms should be intelligent. This way, the arms (or the robot) need to make the decision and synchronise the timing, which may require advanced sensors (e.g., cameras) and communication and synchronisation capabilities between the two arms. This means the control algorithm becomes complicated. If multiple arms are involved, this becomes even more complicated.

Compared with this, Design B can lead to a simpler design ("collaboration without communication"), because the most complicated control task which is synchronisation is done by one entity (i.e., the object) and no collaboration between the two arms is necessary. Complexity reduction through the introduction of the concept of the leader and the use of the environment for communication is often experimented in collaborative robotics (e.g., [14]). In the past research, this architectural principle could be found in Holonic manufacturing [15], cellular machines (Fig. 6) [16], and more recently in Industry 4.0 [17].

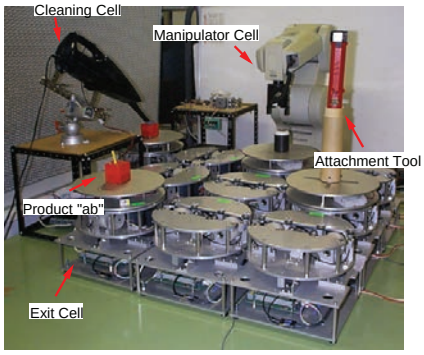

Fig. 6. The concept of cellular re-manufacturing system [16].

Design C is more drastic with a distributed intelligence scenario. This is more like IoT (Internet of Things), because an observer (environment) can control the arms and the object by collecting information from distributed sensors. While it has disadvantages like complexity of collaboration of multiple arms but has a strong advantage that it can obtain a systems level overview. As opposed to distributed intelligence in Design B which collects local information based on local intelligence, Design C can collect global information such as global performance, optimality of the entire system, and congestion. This feature can improve the system performance drastically.

4.3. Reaction-free fastening system

Maintenance activities are non-uniform. Even a simple task like disassembling can be difficult for a robot for this reason. First, unfastening rusty or corroded fasteners (bolt and nut) requires rust removing chemicals such as phosphoric acid and powerful tools such as an impact screwdriver or pneumatic screwdriver. Second, the torque to be applied to a bolt/nut is difficult to predict even without corrosion. However, excessive torque may damage the fasteners. Therefore, a brute-force approach isn't always a solution.

Applying torque leads to reaction force. If a robot uses a powerful handheld screwdriver and a bolt is stuck due to corrosion which is a common situation in maintenance, it is likely that the screwdriver moves the robot or damages the arm rather than unfastens the bolt. For this purpose, various types of reaction-free fastening systems are useful.

One new design of such a system can be derived using the function allocation theory. First, Fig. 7 depicts a (simplified) FBS model of function "to fasten" an object using a bolt. (In this figure, a rounded square signifies an entity and a diamond a subfunction or physical phenomenon.) Torque is applied to the screwdriver which rotates the bolt. The bolt then moves forward relative to the object and this evokes the "fastening" phenomena. To apply torque, the screwdriver needs to be supported by a support, which is usually the operator's body (weight). In this setting, as a response to the torque application, reactive torque is generated and comes back to the support.

A new design can be then created using the function allocation theory. In Fig. 7, the support and the object are

separate objects. However, by instantiating the object as the support, allocating “to apply torque” partly to the object, the torque and reactive torque cancel out. This means “reaction-free”, which is illustrated in Fig. 8. A schematic drawing is depicted in Fig. 9.

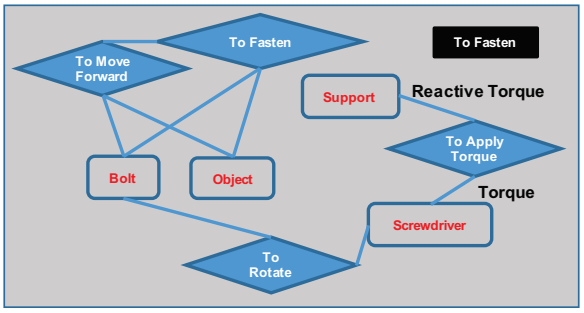

Fig. 7. Function “to fasten”.

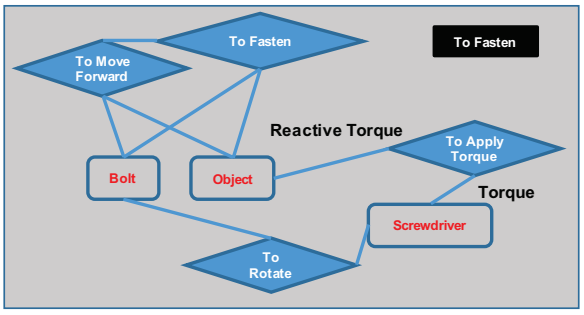

Fig. 8. New fastening system design.

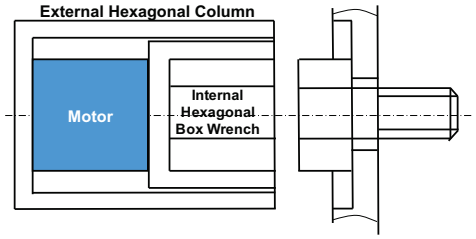

Fig. 9. Reaction-free fastening system

This design in Fig. 9 uses a conventional bolt. The object to be fastened has a pocket with a shape of a bigger hexagon than the external shape of the bolt head. This can be replaced by a washer that has the same function. The (un)fastening screwdriver has an internal hexagonal box wrench that is driven by the motor inside the external hexagonal column. This external hexagonal column fits in the pocket of the object.

With this design, a maintenance robot does not have to resist against reaction force. The robot only has to support the weight of the fastening system, which means its design might become much simpler and lighter.

5. Conclusions

This paper discussed a new functional design method named “function allocation theory” to arrive at creative design solutions. Our previous work on systems architecting proposed a function decomposition method that exhaustively generate different architectural variations. In contrast, the function allocation theory allows arbitrary allocation of functions to objects through which new design concepts can be generated.

The power of this function allocation theory was demonstrated through a couple of design cases, including collaborative autonomous robots and a reaction-free fastening system in the context of maintenance robots. This also demonstrated the usefulness of functional modelling and reasoning at systems architecting and conceptual design in arriving at creative design solutions [18].

The future work may include the development and implementation of computer-based tools of function allocation theory.

Acknowledgements

The author would like to thank Dr. Michael Farnsworth (Cranfield University, UK) and Dr. Hitoshi Komoto (AIST, Japan) for their help and contributions in conducting research which formed the basis of this paper. This work was partially funded by the EPSRC, grant number EP/1033246/1, through the kind support of the EPSRC Centre for Innovative Manufacturing in Through-life Engineering Services.

References

[1] Muller G. Systems Architecting: A Business Perspective. Boca Raton, FL: CRC Press; 2011.
[2] INCOSE. INCOSE Systems Engineering Handbook, Fourth Editon, INCOSE-TP-2003-002-04. Hoboken, NJ: John Wiley & Sons; 2015.
[3] Tomiyama T, Gu P, Jin Y, Lutters D, Kind Ch, Kimura F. Design methodologies: Industrial and educational applications. CIRP Annals Manu Tech 2009; 58:2, pp. 543-565.
[4] Komoto H, Tomiyama T. Multi-disciplinary system decomposition of complex mechatronics systems,” CIRP Annals Manu Tech 2011; 60:1, pp. 191-194.
[5] Komoto H, Tomiyama T. A framework for computer-aided conceptual design system and its application to system architecting of mechatronics products. Comp-aided des 2012; 44:10, pp. 931-946.
[6] Forsberg K, Mooz H. The relationship of system engineering to the project cycle. Proc Joint Conf Nat Council Sys Eng and American Soc for Eng Mngmt. Chattanooga, TN. 21–23 Oct 1991.
[7] Umeda Y, Ishii M, Yoshioka M, Shimomura Y, Tomiyama T. Supporting conceptual design based on the function-behavior-state modeler. Art intell eng des analy manu 1996; 10:4, pp. 275-288.
[8] Tomiyama T, Kiriyama T, Takeda H, Xue D, Yoshikawa H. Metamodel: A key to intelligent CAD systems. Res Eng Des 1989; 1:1, pp. 19-34.
[9] Forbus KD. Qualitative process theory. Art intell 1984; 24:1-3, pp. 85-168.
[10] Farnsworth MJ, Bell C, Khan S, Tomiyama T. Autonomous maintenance for through-life engineering. In: Redding L, Roy R, editors. Through-life engineering services. Switzerland: Springer International Publishing; 2015, pp.395-419.
[11] http://www.darpa.mil/program/darpa-robotics-challenge, accessed on 18 February 2016.

[12] Takata S, Kimura F, van Houten FJAM, Westkamper E, Shpitalni M, Ceglarek D, Lee J. Maintenance: changing role in life cycle management. CIRP Annals Manu Tech 2004; 53:2, pp. 643-655.
[13] Farnsworth M., Tomiyama T. Capturing, classification and concept generation for automated maintenance tasks. CIRP Annals Manu Tech 2014; 63:1, pp. 149–152.
[14] Miyata N, Ota J, Arai T, Asama H. Cooperative transport by multiple mobile robots in unknown static environments associated with real-time task assignment. IEEE Trans robotics automation 2002; 18:5, pp. 769-780.
[15] Van Brussel H, Wyns J, Valckenaers P, Bongaerts L, Peeters P. Reference architecture for holonic manufacturing systems: PROSA. Comp ind 1998; 37:3, pp. 255-274.
[16] Kondoh S, Sato RK, Tomiyama T, Umeda Y. Self-organization of the cellular manufacturing system. CIRP J Manu Sys 2000; 30:6, pp. 507-513.
[17] German National Academy of Science and Engineering (Acatech). Recommendations forimplementing the strategicinitiative INDUSTRIE 4.0. Recommendations for implementing the strategicinitiative INDUSTRIE 4.0. 32013.