TOWARDS NON-HIERARCHICAL SYSTEM DESCRIPTIONS FOR AUTOMATING FUNCTIONAL ANALYSIS

Amrin, Andas (1); Spitas, Christos (1,2)
1: Nazarbayev University, Republic of Kazakhstan; 2: Delft University of Technology, The Netherlands

Abstract

The case is made that current functional representation methods for engineering systems and in particular Function-Behaviour-State models rely on defining hierarchies of states, behaviours and functions that in turn rely for a large part on subjective designer input. With the goal of automating the functional description and analysis of such systems, basic concepts of an Idea Algebra are introduced, which allow the peer-to-peer non-hierarchical functional coupling of Ideas such as components and interfaces into so-called Synaptic Networks and the objective definition of system functions and subfunctions in terms of the states of the basic Ideas forming said networks. By offering a robust mathematical definition for system topology and functionality and dispensing with several subjective steps required by current methods, the presented framework opens significant possibilities for automating functional analysis.

Keywords: Design methodology, Product modelling / models, Functional modelling

Contact:
Prof. Christos Spitas
Delft University of Technology
Design Engineering
The Netherlands
c.spitas@tudelft.nl

Please cite this paper as:
Surnames, Initials: *Title of paper*. In: Proceedings of the 21st International Conference on Engineering Design (ICED17), Vol. 4: Design Methods and Tools, Vancouver, Canada, 21.-25.08.2017.

1 INTRODUCTION

1.1 Functional modelling

Functional modelling is a set of processes that helps engineers to identify and analyse the ways in which engineering systems manifest various behaviours and produce their intended functions. Most frequently, the point of view is that of quantified function, often seen from the point of view of reliability, which concerns itself with the failure of desired functions, their causes of failure and the propagation of failure in engineering systems.

The function of a system is generally decomposed into sub-functions, which are causally connected. With regard to reliability in particular, the main goal of the analysis is to prevent or to make a prediction of the fault conditions so that the correct actions can be taken (at either the level of remedial actions for a given system that is observed to have reliability problems, or preventively, early in the design stage) to ensure the preservation of the functionality of the system. There are many reliability analysis methods that are used in engineering, including Failure Mode and Effect Analysis (FMEA), Fault Tree Analysis (FTA) and Bayesian Networks (BNs), however other methods used in the context of knowledge-based design, such as e.g. Function-Behaviour-State (FBS), have also been applied to the problem.

In this paper we shall principally consider the question of automating functional analysis and thus the focus will be on how to move from knowledge representations to mathematical models of system function with as little designer intervention/ supervision as possible. FMEA, FTA and BNs will therefore not be discussed further, in favour of FBS and an Idea Algebra (IAlg) that will be introduced specifically for this purpose.

1.2 Engineering design knowledge representation methodology

Function modelling is a class of techniques that are used to analyse development model of products, processes and objects from the point of view of functionality. The framework is also used to increase the communications between the engineers in the design process by providing a general description of a system. There are several different reported functional modelling methods, which have been developed by many researchers (Erden, 2008). Among these, function modelling techniques are a class of techniques that are used to create, modify and analyse the development model of products or processes in terms of their functionality. Function-Behaviour-Structure and Function-Behaviour-State are among the most well-known implementations of this paradigm.

The Function-Behaviour-Structure model (Gero, 2007) describes a design process as a process of transforming the function of a system to a structure through the behavioural information. The Function-Behaviour-State (FBS) model (Umeda, 1990, Van Houten, 1998) has conceptual similarities to the former, but does not explicitly represent structure, but instead more explicitly considers the physical states and chemical properties of the system. As a result, the advantage of current implementations of Function-Behaviour-State over Function-Behaviour-Structure is that they more readily admit computational models to be generated, which is very relevant to the scope of the present study. There is no known automated implementation of other variants of the Function-Behaviour-Structure method to reliability analysis, or other quantified functional analysis. Henceforth, therefore, we shall only be concerned with Function-Behaviour-State (FBS).

1.3 Automation of functional and reliability analysis

Van Houten et. al. (1998) have implemented a method for fault and deterioration diagnosis of products using the Function-Behaviour-State model as a basis for data representation. Echavarria et. al. (2007) have introduced a fault diagnosis system for wind turbine using an FBS based model-based reasoner and functional redundancy designer models (Umeda, 1994). In these cases, the starting point is a functional representation of the system. Explicitly, the hypothesis is made (Van Houten, 1998) that there is a 1-1 mapping between structure (elements) and function.

Lapp and Powers (1977) developed an algorithm that transforms a directed graph into a fault tree, Taylor (1982) introduced a method that transforms a component model based on the mini fault trees into a fault tree. Bossche (1991a, 1991b, 1991c) developed a method that constructs a fault tree from models of components that were developed based on the system description. Latif-Shabgahi and Tajarrod (2009) implemented a method that constructs a fault tree from a Simulink file. Moreover, a fault tree can be

constructed in the Simulink environment. Wang et. al. developed (2002) a tool that automatically maps a system block diagram into a fault tree. Khan et. al. (2002) developed an algorithm for generating of probabilistic fault trees.
In all cases it can be seen that the starting point for functional and reliability analysis is a functional representation of the system: Therefore, it assumes that the mapping of structure and topology to function has already been done. This is a particularly problematic assumption, especially in the case of complex systems and multiply coupled system functions, where a single physical component may be contributing to more functions and each function may require more components; therefore, the requisite mapping of topology to function is non-obvious, and certainly not 1-1.

1.4 Idea Algebra

To overcome these perceived deficiencies, the authors have previously proposed the introduction of a generalised class of Ideas, which can encompass all the potential elements of a system model at any level of abstraction (Spitas 2011, 2012, 2013). Components (hardware, software), states, behaviours, functions, are all Ideas. A main characteristic of such a class is that it facilitates and encourages the non-hierarchical –and potentially ad-hoc- definition of Ideas. This departs from all current paradigms, that must by definition define hierarchies of functions.
In this paper, we shall demonstrate how this class of Ideas may be equipped member functions and operators suitable for automatic functional analysis, forming an Algebra. An obvious advantage of this approach is that it allows the elimination of several steps involving the identification and hierarchical organisation of functions and function-component relationships; steps that otherwise require substantial and potentially subjective designer intervention.

2 FUNCTION-BEHAVIOUR-STATE AND ITS USE IN FUNCTIONAL ANALYSIS

The Function-Behaviour-State model, as introduced by Umeda et.al. (1990) and elaborated in subsequent publications discussed previously, is represented via a hierarchical tree, where the design information is classified into three types: function, behaviour and state. The framework considers systems only from the physics point of view.

State
(to highlight that the state and structure is different things by itself, it is the main difference between the FBS model and the IA model)
According to the model, the structural and state information is presented as a set of States, which have a set of parameters: a set of entities, a set of attributes and a set of relations, where an entity is represented as a real world entity identifier, an attribute is described as scientifically observable properties of the entity, like chemical, physical, geometrical, etc. and a relation is presented as “what relates attributes, entities, or relations”.
There is no reported method for rigorously representing/ naming states, so in this paper we shall use the descriptor Si, where i is any suitable integer counter.

Behaviour
The authors of the model identify behaviour as “sequential one and more changes of states”. The behaviour is considered as a State which is captured for a moment. As the states can be produced infinitely, the change of States are produced by some rules, like physical laws, which are defined as “a rule which determines behaviours of an entity under a specific condition of states”. The relationship between the Behaviours and States are called B-S relationship.
There is no reported method for rigorously representing/ naming behaviours, so in this paper we shall use the descriptor Bi, where i is any suitable integer counter. From the B-S relationship definition, we can define such a relationship as a mapping:

$$S_i(t) \rightarrow B_i \tag{1}$$

Views

It is a type of representation for behaviours and states which describes the analysis of a system depending on the domain. The same behaviour can be interpreted using different engineering knowledge, also it relates to the state. The States and Behaviours can be described only after the view is chosen.

Function
The function is "a description of behaviour abstracted by human through recognition of the behaviour in order to utilise it". It's an abstracted behaviour which is described as "to do something". Γab corresponds to the human process of recognition and abstraction of behaviour to the function, which is called F-B relationship.
There is no reported method for rigorously representing/ naming functions, so in this paper we shall use the descriptor Fi, where i is any suitable counter. From the F-B relationship definition, we can define such a relationship as a mapping between the entire sets comprising all Bi and Fi:

$$\{B_{\{\dots\}}\} \rightarrow \{F_{\{\dots\}}\} \tag{2}$$

The framework comprises two types of parts which are presented as relationships: the subjective and objective parts. The F-B relationships and the views are identified subjectively, whereas the B-S relationships are identified objectively by using the physical laws. The main function is decomposed into sub-functions and it has a hierarchical structure.

The question then remains how to identify the functions, which is, as discussed, a largely subjective process. The Functional Evolution Process proposed for this purpose by Shimomura et. al. (Shimomura, 1998) includes three steps: the first step is a functional realisation which represents a process that retrieves physical descriptions from functions, the second step is a functional evaluation which describes a process of function reliability assessment and the last step is a functional operation which represents a process of design improvement by operating the functions.

FMS Modifier Diagrams (FBS/m) can thus be drawn, utilising "decomposed into" and "conditioned by" relations.

3 IDEA ALGEBRA: FOUNDATION AND APPLICATION TO FUNCTIONAL ANALYSIS

3.1 Definitions and concepts

For the purpose of representing engineering systems, we define as an Idea anything that is currently or potentially present to consciousness.
Consequently, all elements of these systems (having physical/ material manifestations) are Ideas; (non-physical/ non-material) relationships are Ideas; physical states are Ideas; behaviours and functions are Ideas; design objectives are Ideas; etc.
Thus we define a generalised class of Ideas, which can be used to identify and reference any aspect of design. Equipping said class with operators results in the definition of an Idea Algebra (IAlg).

3.2 Data representations

3.2.1 Ideas as class elements

An Idea, in the form of a class element, is a mathematical representation of a cognitive construct. In a formal computer programming context this may be, at its most rudimentary definition, simply a name pointing to a block of data (or null), denoted as Ii, or Ii(), where i is any suitable counter. Such an idea does not externally reference other ideas as its arguments. Within the namespace/ context of Ii any number of ideas may be defined locally, having appropriate meanings for the context of Ii only. i.e., given the definition of Idea, the probability p of Ii being currently present (True) (as opposed to not being currently present (False)) would be Ii.p. In the context of formal computer programming, any Idea such as p would be a member of Ii.

3.2.2 Values of Ideas

An Idea can be subjected to evaluation, returning one or more member values that may be constant or variable, if the idea is equipped with methods that allow the determination of said values. The value can be any simple or complex representation (and by definition is itself an Idea). By way of being subject to evaluation, Ideas are in a generalised way identical to functions and in a formal computer programming context can be represented as such.

3.2.3 States of Ideas

The values of Ideas may be used to indicate States of said Ideas. A State can refer to a physical state (e.g. in the Lagrangian or Hamiltonian sense), or a logical state (True or False), or in fact anything potentially variable that may be used to characterise the idea in a particular instant.

Here we see the first possibility for mapping to FBS, as this definition of State is compatible in both FBS and IAlg. The difference is that in IAlg, State is just a member of an Idea object, represented as Ii.Sj, whereas in FBS it would be an original entity, represented as Si. The possibility in IAlg to assign various different States to the same Idea becomes obvious, whereas in FBS there is no such explicit possibility.

3.2.4 Ideas as arguments to ideas. Synapses

Furthermore, an Idea Ij may reference other Ideas defined independently from it and external to it, i.e. Ii, in which case it can be denoted as Ij(Ii). We term such an assignment of an argument to an Idea a 'synapsis'. For all intents and purposes an Idea with synapses behaves like a function with arguments, whereas an Idea without synapses behaves like a function without arguments. However, whereas traditionally the arguments of a function must be decided/ declared upon its definition, the arguments for any Idea are situational (and thus volatile); i.e. any Idea may be declared with no arguments and later subjected to modification, or an Idea may be stripped of its arguments. This is a unique property of the class of Ideas: In all other existing engineering data representations (e.g. Gero 2007, Umeda 1990), the equivalents to Ideas are always defined within a class hierarchy (i.e. component, subsystem, system etc), whereas obviously any hierarchies built among ideas are situational and volatile.
Thus defined, synapses are used to link Ideas into interacting clusters/ systems. In formal computer programming terminology, Ideas are in principle agnostic to anything that occurs outside their own scope, so synapses allow Ideas to expand their scope to the ideas that are linked to them as arguments. At the same time, Ideas that are linked arguments to other Ideas also share this awareness.
Synapses thus allow the formation of hierarchies (systems), and of peer-to-peer interfaces between two or more Ideas. If Ii, Ij, Ik, … are ideas, then Ii(Ij,Ik,…) can be used to define a hierarchy (i.e. an assembly) from Ij, Ik, … or to refer to a physical interaction (i.e. an interface) between Ij, Ik (or possibly more than two ideas. In both cases the synapsis is the following mapping:

$$\{I_{\{\dots\}-\{i\}}\} \rightarrow I_i \tag{3}$$

Typically, Ii as a hierarchy may not have any special purpose other than to express the existence of the hierarchy. However, Ii as an interface will typically be equipped with member functions (i.e. laws of physics) that couple its State and the States of its arguments. In the context of IAlg, this allows States to become connected and interact, not requiring the explicit definition of Behaviours, or Functions.

3.2.5 (Non-)hierarchies of ideas

Looking at the previous definitions, probability p, seen as an Idea, need not be defined as a member of any given Ii, but may be defined independently from it and externally to it, thus it may also be appropriate to write p(Ii). The subtle but important difference is that Ii.p implies a hierarchical relationship (p is a member of Ii and defined internally to it), whereas p(Ii) indicates a situational relationship, in that p is defined independently from and externally to Ii.
So in general for any two related Ideas, Sj(Si) can be interchangeable with Si.Sj, thus formal and informal hierarchies are interchangeable ad hoc. This lack of imposed hierarchy in the class of Ideas (although it can still be emulated where desired), is proposed as an essential characteristic that makes Idea class objects similar to the human ideas, here denoted as used in design: able to transform at will, allowing generalisation, specification and ultimately design freedom, exploration and creativity.

3.3 Behavioural coupling- Idea state propagation (causally driven)

3.3.1 Hierarchical causalities and functionality

Normally the hierarchy for causalities is determined explicitly. Just as human cognition must be used to map the structure of an engineering system to a hierarchically organised topology, it must also be used to determine a hierarchy of events and functions describing the state changes of the engineering system: Functions of different levels must typically be organised in hierarchies of their own, building up to the most important system level functions. This is the underlying concept of FTA, FMEA, DSM, FBS and all state of the art methods for systems analysis in general.

3.3.2 Behavioural coupling: Ad-hoc causalities

The concept of behavioural coupling fundamentally re-addresses this approach: If the topology of an engineering system is known (without any knowledge of hierarchy) and the basic functional interactions between ideas and their arguments are known (again without any knowledge of hierarchy), can the system function states be calculated automatically? If so, this would eliminate the need to define/ understand functional -and even structural- hierarchies and along with this much of the present demand on human cognition.
Behavioural coupling essentially enables the automatic and ad-hoc propagation and interaction of state information through the synapses that link a network of ideas, as follows:

Argument assignments to Ideas can create causal relationships between the States of all involved Ideas, if they are equipped with suitable member functions. We consider in principle that not only is the State of an Idea affected by the States of its arguments, but in fact that a bi-directionally coupling exists, such that if an Idea is an argument to other Ideas, its State can be affected by the states of said Ideas.
Furthermore, one State of an Idea can be affected by other States of the same Idea; this becomes important when it is necessary to couple e.g. structural states to functional states.

$$\left\{I_i.S_{\{\dots\}-\{j\}}, I_{\{\dots\}-\{i\}}.S_{\{\dots\}}\right\} \rightarrow I_i.S_j \tag{4}$$

Considering that State information propagates at finite speed through the system, also in accordance with causality, we consider that the State of an Idea will be affected only by precedent States of other Ideas and itself, leading to the following mapping in the time domain:

$$\left\{I_{\{\dots\}}.S_{\{\dots\}}(t_n)\right\} \rightarrow I_i.S_j(t_{n+1}) \tag{5}$$

3.3.3 The concept of non-hierarchical functionality

In the context of functional analysis, the described causal framework for behaviour coupling affords the significant advantage that component states can be affected by system states, as opposed to the one-way state propagation used in all FTA and FMEA models, thus providing a highly realistic approach in the case of (multi)physical systems.

State propagation testing to determine behaviourally significant subsystems and associated behaviours and functions
We propose that (sub)systems of any behavioural significance manifest a strong coupling of the states of the ideas that comprise them. Conversely, a (sub)system definition that is not characterised by such a coupling will likely be of little behavioural significance.
Following from this proposition and considering that behaviour is the sequential-/ time-function of state, we hereafter propose a 'state propagation testing' method to automatically identify behaviourally significant subsystems within in a given system. The method is implemented as follows:

- We assign initial State values (the initial conditions) across all Ideas of the system.
- We disturb this steady state by imposing to any single Idea in the system a different State value and thereafter allow the Sates of all Ideas in the system to re-evaluate, according to their member functions (e.g. the laws of physics). The Ideas whose states changed as a result can be understood to form a functional subsystem whose behaviour is dependent on the reference Idea.
- The same process can be repeated for several or all Ideas in the system. Typically, only a limited number of distinct subsystems will emerge from this process.

This method can be used to automatically identify behaviourally significant subsystems and their associated behaviours and functions. Instead of considering the subjective mapping Γab that is required by the FBS model, in an IAlg it is possible to map Behaviours to States and also Functions directly to (clusters of) States.
We accomplish this by considering the following:
State change can be expressed as a function of time derivatives of said State. However, given the definition of State in section 3.2.3 said derivatives can also be considered as States. Given the definition of Behaviours as changes of State, as per section 2, it follows therefore that is possible to consider so-called Behaviour as State. Given then that Functions, as per section 2, are functions of behaviours, it follows that they are functions of States.
It is possible then that a function of any complexity can be mapped to the State of an appropriate Idea. E.g. a system-level function could be mapped to a State of the Idea representing the system itself, the State representing a functionality of said Idea. For an automotive drivetrain the "propulsion" function would involve the entire automobile system, together with the ground. Conventional FBS thinking would dictate that to test if this function is achieved, the functionality (States/ Behaviours/ Functions) of the entire automobile system (or perhaps only the drivetrain subsystem) would need to be modelled and tested. Representing the system with an IAlg, where the interfaces of the automobile system to the ground coincide with the interfaces of the drivetrain system to the ground, and in fact the interfaces of the four wheels to the ground, it becomes possible and rather obvious to define the "propulsion" function instead as the State of these interfaces, where torque is being transmitted. Thus IAlg allows a much simpler definition of what is normally considered a high-level function, without ever having to construct a hierarchy of Behaviours and Functions for the automobile system. This is only possible because IAlg contains intrinsically the functional interrelations of the system topology, whereas FBS dispenses with topology, dealing primarily in States.

It follows that a main difference of IAlg from FBS and other existing implementations is that the identification of functional dependency does not rely on subjective definition, but emerges from the basic dependencies of States intrinsic to the system topology. It is a peer-to-peer, non-hierarchical functionality.

4 GRAPHICAL (NETWORK) REPRESENTATION OF AN IDEA ALGEBRA

A set of ideas {Ii} can be mapped visually to any set of points or in general shapes in 1-, 2- or 3-dimensional space, without a priori attributing any particular significance to the location of the particular points in said spaces. Synapses (argument references) may be mapped to vectors (directional lines). The depiction of such synaptic vectors creates the visual appearance of a network and is shown in Fig. 4. We term such a network of Ideas connected via synapses a Synaptic Network (SN).

If the chosen shapes for the synapses are curves and they are made to coincide/ intersect locally with the points/ shapes that correspond to their arguments, then the corresponding reference vectors will vanish, having their endpoints coincide, leading to an alternative representation of the SN that can be used in the manner of intuitive shorthand (Spitas, 2013). Recognising that synapses may be in turn referenced by other synapses, or that they may visually appear to intersect inadvertently, this shorthand representation does not lend itself well to representing complex SNs and can be error-prone.

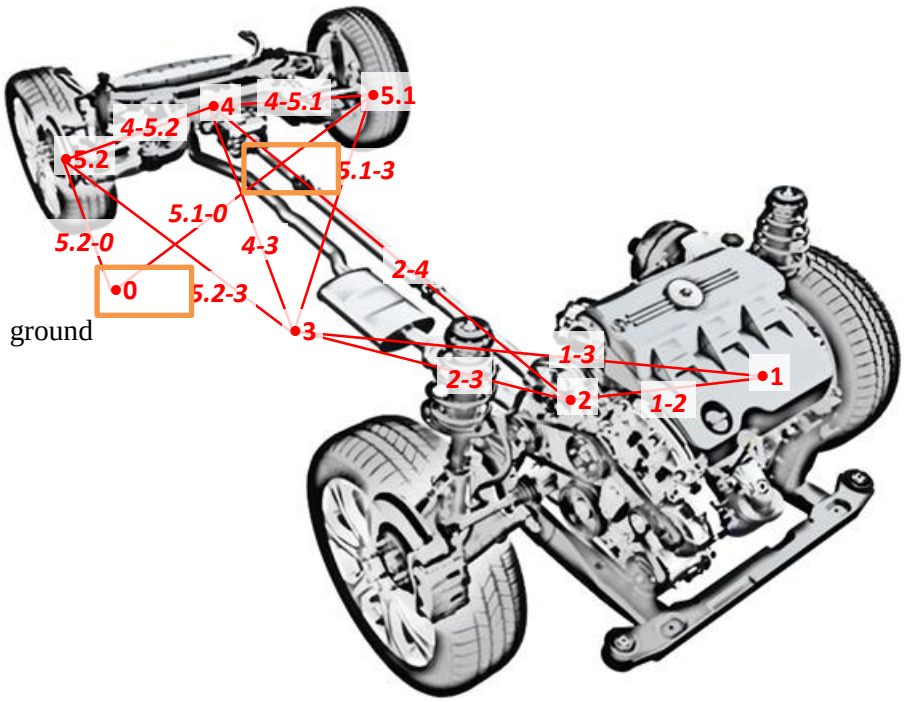

Figure 1. Graphical representation of a system of Ideas for an automotive drivetrain (synaptic vectors are omitted for clarity). The two indicated interface Ideas appropriately embody the function "propulsion", without need for any functional hierarchy within the drivetrain.

5 APPLICATION TO FUNCTIONAL ANALYSIS: AUTOMOTIVE DRIVETRAIN

To demonstrate the application of the presented model, let us consider the automotive drivetrain subsystem illustrated in Figure 1. The system identification is as follows:
0: ground, 1: engine, 2: gearbox, 3: chassis, 4: differential, 5.1: rear left wheel, 5.2: rear right wheel
All of these ideas do not reference each other and are completely independent.

The physical interfaces between these ideas are themselves ideas that use the former as arguments. They are identified as follows:
1-2: shaft and couplings, 2-4: shaft and couplings, 4-5.1: shaft and couplings, 4-5.2: shaft and couplings, 5.1-0: tyre-to-ground contact, 5.2-0: tyre-to-ground contact
1-3: bolted connection, 2-3: bolted connection, 4-3: bolted connection, 5.1-3: suspension, 5.2-3: suspension

Functional analysis is set up automatically by assigning force states (F) to each interface, i.e. (5.1-0).F, (5.2-0).F, (5.1-3).F, (5.2-3).F etc and equilibrium equations to each system component, i.e. for wheel 5.1 the force equilibrium: (5.1-0).F+(5.1-3).F=0, the torque equilibrium (4-5.1).T+(5.1).R ×(5.1-0).F=0 and so on for other system components. These definitions and equations are generated automatically from the system topology, as part of the respective Idea Algebra. Note that not only mechanical states and constitutive equations, but any domain of multiphysics as well as structural integrity states can be considered as per the functional requirements of the design.
Back to our example, the propulsion function requires that at least one of (5.1-3).F or (5.2-3).F is non-zero. Considering (5.1-3).F, as per the previously generated state definitions and equations, loss of said function can only happen if (5.1-0).F=0, which in turn can only happen if the interface 5.1-0 loses its ability to transmit force, e.g. through sliding or complete loss of contact; or if (4-5.1).T=0, which can happen if power is lost throughout the powertrain. A similar consideration applies with regard to (5.2-3).F. Note that the ideas 1, 1-2, 2, 2-4, 4, 4-5.1, 5.1, 5.1-0, 4-5.2, 5.2, 5.2-0, 0 form paths in the SN that correspond to the power transmission and any loss in structural integrity will result in loss of torque.

Thus the proposed Idea Algebra also helps to visualise failure modes as 'paths' in the SN of the system. Importantly, all the mentioned processing and reasoning can be automatic (a straightforward parsing of the topology) and no a priori reasoning or anticipation of failure modes is required.

We observe that the actual function of propulsion involves only the ideas 0, 5.1, 5.2, 3 and their respective interfaces 5.1-0, 5.2-0, 5.1-3, 5.2-3 (which form two parallel paths), as follows:
Force is transmitted via interfaces 5.1-0 and 5.2-0; due to the equilibrium of 5.1 and 5.2, the same force is transmitted via interfaces 5.1-3 and 5.2-3; and thus the sum of said forces acts on 3, creating propulsion.
In hindsight, this is easy to understand and identify. It is, however, not so obvious a priori, particularly to a non-expert.

In the preceding discussion it is thus seen how the proposed Idea Algebra allows a direct mapping between system structure and system functionality, complete with automatic definitions of the relevant component/ sub-system states (represented by idea states) and constitutive relationships relating said states, thereby producing a model of system functionality. Functions of any level (e.g. propulsion in the discussed case) can be directly mapped to specific idea state values.

Still looking at the propulsion function, let us now briefly compare the above analysis to how it would proceed under an FTA, FMEA, DSM or FBS paradigm.
FTA would require the definition of various 'basic events' for failures, then connect them via a logic tree to form 'intermediate events', up until the 'top event' corresponding to propulsion failure. All of these steps would require independent cognitive identification by a human, thereby making this a subjective process heavily dependent on expert input. Due to e.g. cognitive bias, it would be easy to miss a less frequent failure mode, such as the structural failure of interfaces 5.1-3 or 5.2-3, as most persons, including experts, would tend to get fixated on the loss of torque in the powertrain, or loss of contact at the interfaces 5.1-0 or 5.2-0.
FMEA likewise is heavily dependent on the cognitive identification of the possible failure modes (for which part of the process there is no automation available) and is just as dependent as FTA on expert input and similarly prone to omissions and cognitive bias.
A DSM would be helpful in revealing the topological links (interfaces) between system elements and visualising them in matrix form, but would offer no more insights. It would be dependent on expert inference (for which there is no automation available) to determine functional couplings from the topological couplings. The SN already offers the insight of the topological couplings by representing the interfaces as ideas having other ideas as arguments, but in addition the supporting Idea Algebra explicitly and automatically maps the topology to function.
Finally, FBS would require the hierarchical functional decomposition of the propulsion function. Again, this would have to be dependent on the subjective cognitive processing of the system by the designer and would be subject to the same limitations as discussed under FTA, FMEA and DSM. Lacking automation and subject to cognitive bias, e.g. again it would be easy to miss the significance of the interfaces 5.1-3 and 5.2-3 in light of the more obvious findings. Also, the hierarchy itself would not provide any particular insight: The Idea Algebra-based analysis showed that the propulsion function is actually directly dependent on the force carried by a single pair of interfaces (5.1-3 and 5.2-3) and is actually very 'low level' and can be influenced ad-hoc by various other states in the entire system.

6 CONCLUSION

This paper presented a number of concepts relevant to automating the functional analysis of engineering systems. While the FBS model is a perfectly adequate for the functional representation of engineering systems, it requires the subjective definition of functions and uses States as its starting point of system description, assuming a 1-1 mapping of system topology to functionality that is in many cases oversimplifying and untrue. It was shown that a direct mapping of system topology (components, interfaces) to Ideas can be used to define Idea Algebras (IAlgs) that possess a number of desirable characteristics:

- They make possible the ad-hoc connection of Ideas (and their States) via so-called synapses in a peer-to-peer network, called a Synaptic Network (SN).

- They allow the objective identification of functional subsystems via "state propagation testing", as opposed to the subjective definition/ hypothesis of the same.
- They allow the definition of all functions directly from States, dispensing with the need to subjectively define/ hypothesise functional hierarchies.
- They allow a simple visualisation of both the system topology and functionality, which is intuitive and consistent with network representation conventions -as opposed to the multilevel graphs typical of FBS-type models.

As a result, by offering a robust mathematical definition for system topology and functionality and dispensing with several subjective steps required by current methods such as FBS, the presented framework opens significant possibilities for automating functional analysis. These will be explored in depth in future research.

REFERENCES

Bossche A., (1991a), *Computer-aided fault tree synthesis I (system modeling and causal trees)*, RESS, Volume 32(3), pp. 217-241

Bossche A., (1991b), *Computer-aided fault tree synthesis II. Fault tree construction*, RESS, Volume 33(1), pp. 1-21

Bossche A., (1991c), *Computer-aided fault tree synthesis III: Real-time fault location, RESS*, Volume 33(2), pp. 161-176

Echavarria, E., Tomiyama, T. and van Bussel, G.J., (2007), "Fault Diagnosis approach based on a model-based reasoner and a functional designer for a wind turbine. An approach towards self-maintenance". In *Journal of Physics: Conference Series* (Vol. 75, No. 1, p. 012078). IOP Publishing.

Erden, M.S., Komoto, H., van Beek, T.J., D'Amelio, V., Echavarria, E. and Tomiyama, T., (2008), "A review of function modeling: approaches and applications". *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 22(02), pp.147-169.

Gero, J.S. and Kannengiesser, U., (2007), "A function–behavior–structure ontology of processes". *AI EDAM: Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, 21(04), pp.379-391.

Khan, F.I., Husain, T. and Abbasi, S.A., (2002), "Design and evaluation of safety measures using a newly proposed methodology 'SCAP'". *Journal of Loss Prevention in the Process Industries*, 15(2), pp.129-146.

Lapp, S.A. and Powers, G.J., (1977), "Computer-aided synthesis of fault-trees". *IEEE Transactions on Reliability*, 1, pp.2-13.

Latif-Shabgahi, G. and Tajarrod, F., (2009), "A New Approach for the Construction of Fault Trees from System Simulink". In *ARES* (pp. 712-717).

Shimomura Y., Yoshioka M., Takeda H., Umeda Y., Tomiyama T., (1998), "Representation of design object based on the functional evolution process model", *J. Mech. Des* 120(2), 221-229

Spitas C., (2011), "Analysis of systematic engineering design paradigms in industrial practice: Scaled experiments", *Journal of Engineering Design*, 22(7), pp 447-465

Spitas C., (2012), "Definition of a functional class of ideas for Integrated Product Development and supporting theory", *9th International Workshop on Integrated Product Development (IPD Workshop)*, 5-7 September, Magdeburg

Spitas C., (2013), "Beyond frames: A formal human-compatible representation of ideas in design using non-genetic ad-hoc and volatile class memberships and corresponding architecture for idea operators", *13th International Conference on Engineering Design (ICED13)*, 19-23 August, Sungkyunkwan Univerity, Seoul, Korea, paper 554

Taylor, J.R., (1982), "An algorithm for fault-tree construction". *IEEE Transactions on Reliability*, 2, pp.137-146.

Umeda, Y., Tomiyama, T., Yoshikawa, H. and Shimomura, Y., (1994), "Using functional maintenance to improve fault tolerance". *IEEE Expert*, 9(3), pp.25-31.

Van Houten, F.J.A.M., Tomiyama, T. and Salomons, O.W., (1998), "Product modelling for model-based maintenance". *CIRP Annals-Manufacturing Technology*, 47(1), pp.123-128.

Wang, Y., Teague, T., West, H. and Mannan, S., (2002), "A new algorithm for computer-aided fault tree synthesis". *Journal of Loss Prevention in the Process Industries*, 15(4), pp.265-277.