# Chapter 4

# The Functional Basis for Failure Mode Avoidance in Automotive Systems Engineering Design

I.F. Campean and E.J. Henshall

## 4.1 Introduction

The engineering challenge of automotive systems engineering design has been increasing rapidly over the past couple of decades with the accelerated pace of introduction of new technologies to address environmental concerns and the drive to enhance customer satisfaction. In spite of the development and use of enhanced CAE and virtual engineering tools, the effectiveness of automotive product development process has not increased as expected; this is clearly illustrated by the pattern and cost of engineering changes (Cash, 2003; Wasmer *et al*., 2011). Related research (Webb, 2002) has also shown that the overwhelming majority of failures in the field are due to system interactions not being adequately managed during the design, which leads to failures due to lack of robustness to operational noise factors.

The failure mode avoidance (FMA) paradigm (Davis, 2006; 2007) has been embraced by the automotive industry as a strategy for enhancing the effectiveness of the product development (PD) process. Underpinned by Clausing's (2004) pragmatic definition "reliability is failure mode avoidance", FMA promotes a strategic focus on early identification of potential failure modes and development of robust countermeasures. The cornerstones of FMA are Davis' (2007) definition of a failure mode as "any condition (technical, planning, procedural) that will require a change to the plan", and the principles (i) that any failure mode should be identified in the same development phase in which it is created, and (ii) that failure modes should only be found and fixed once. The practical challenge with the FMA implementation is that early discovery of failure modes is technically difficult; this is not only due to the lack of hardware for testing early in the programme (which has been quite effectively addressed by CAE and virtual engineering developments), but also to the complexity of the automotive systems which require an integrated multi-disciplinary (mechanical, electrical, controls, software) approach to engineering design analysis and synthesis. It is therefore essential that the engineering tools employed early in the design process adequately support the complexity of the systems engineering design analysis, in particular to identify and cascade all functional requirements - including both main functions and interface functions required for system integration. On this basis the integrity of the design synthesis can be verified and validated early in the design process against critical function failure modes.

The engineering tools commonly employed within the automotive industry to support failure mode avoidance revolve around design FMEA (failure modes and effects analysis) and robust engineering design verification (Webb, 2002; Zhou, 2005). While both these tools have a functional basis, their practical deployment is often divorced from the systems engineering deployment achieved through functional requirements specification and cascade from system level down to subsystems and components. This gap between systems engineering design (SED) analysis and FMA analysis needs to be bridged in order to enable a step change in the effectiveness of product design and development.

The aim of this paper is to present an integrated framework for systems engineering design based on a Failure Mode Avoidance framework underpinned by a structured approach to function analysis of complex multi-disciplinary systems. The framework supports early deployment of function failure avoidance design strategies, within a coherent horizontal and vertical integration with the systems engineering framework. A case study on the development of an electric vehicle powertrain will be used to illustrate the framework, followed by a discussion on the authors' experience with process implementation within the automotive industry.

## 4.2 Failure Mode Avoidance Framework for Automotive Systems Engineering Design

An FMA framework has been developed by the Engineering Quality Improvement Centre at the University of Bradford, based on collaborative work with the global automotive industry over the last 15 years. The main considerations behind this development were:

- To set up an FMA *process* which is based on integrating existing practices and formal tools (such as FMEA, boundary diagram, function trees, interface matrix, P-diagram, design verification matrix, *etc.*) into a coherent information flow. It is important to base the process on existing tools in order to facilitate the take-up of the FMA process by engineers on a broad basis. The coherent information flow is necessary both to address disconnects between tools which are often used independently and on an ad-hoc basis, and to simplify the process by removing duplication;
- To strengthen the rigour of the analysis, by introducing new tools and enhancing existing tools to facilitate a more structured approach, in particular to support function decomposition, and to reduce the reliance on less structured tools based on brainstorming;

To set up a framework that facilitates the alignment and integration of the FMA analysis with (i) the systems engineering design approach used in the automotive industry, and (ii) the PD process, by enhancing the information content for design decisions at gateways and milestones.

The proposed FMA process, illustrated in Figure 4.1, is based on the following process steps:

1. *Function Analysis*: provides tools to support a structured and comprehensive analysis of functional requirements for the system engineering design, including function decomposition, mapping of functions on design solutions and evaluation of system interface functional requirements to support the flawless system integration.
2. *Function Failure Analysis*: management of early design risk assessment on the basis of function failure modes. The functional basis of failure modes analysis ensures a consistent focus on the customer required functionality (*e.g.* by supporting identification of failure modes due to lack of robustness – *i.e.* unacceptable variability in functional performance) as well as the functional safety considerations for the system.
3. *Robust Countermeasure Development*: provides a framework for robust design optimisation underpinned by a systematic consideration of noise factors (*i.e.* significant factors for functional variability), and based on engineering design strategies for managing the effect of noise factors and analytical methods for functional modelling under uncertainty.
4. *Robust Design Verification*: the aim of design verification is to demonstrate that functions are achieved robustly and reliably under real world customer usage conditions. Robust design verification tools support the development of efficient test methods and procedures to validate the functional robustness in the presence of noise factors at all levels of the system engineering design.

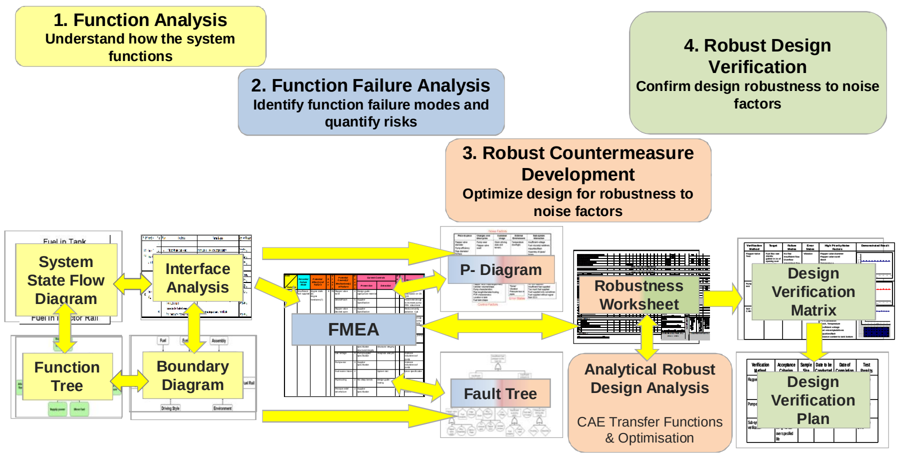

1. Function Analysis
Understand how the system functions
2. Function Failure Analysis
Identify function failure modes and quantify risks
3. Robust Countermeasure Development
Optimize design for robustness to noise factors
4. Robust Design Verification
Confirm design robustness to noise factors
System State Flow Diagram
Interface Analysis
Function Tree
Boundary Diagram
FMEA
P- Diagram
Fault Tree
Robustness Worksheet
Analytical Robust Design Analysis
CAE Transfer Functions & Optimisation
Design Verification Matrix
Design Verification Plan

**Figure 4.1.** FMA framework, showing process steps and support tools

The FMA process as a whole, including the integration of the support tools illustrated in Figure 4.1, has been described elsewhere (Henshall and Campean, 2009; Campean *et al*., 2013); the main focus of this paper is the function analysis step which has been identified as a main weakness in the current automotive engineering design practice. The following sections of the paper provide an outline of the functional analysis framework and tools, illustrated with examples based on a case study of an electric vehicle powertrain (EVP) development. The alignment of the FMA process on the basis of the functional requirements information flow and the integration with the systems engineering design framework and the product development process will be subsequently discussed.

## 4.3 Function Analysis Framework

Within a consumer focused engineering approach, systems engineering design must focus on robust and reliable delivery of customer required functions. It is therefore essential that the functional focus is maintained throughout the systems engineering design process - from requirements analysis through to design verification and validation.

### 4.3.1 Function Analysis Challenge

The common theoretical basis for function decomposition analysis consists of the iterative mapping of functions and their solutions (sub-function structures) at increasing level of detail until a solution concept is reached (Chakrabarti and Blight, 2001). The axiomatic design paradigm (Suh, 1995) provides a useful conceptual framework in which engineering design is presented as an iterative (zigzagging) mapping between the functional domain and the design domain. This is illustrated in Figure 4.1, which shows the customer needs and requirements ($C_i$) in the customer domain which are mapped onto functions ($F_i$) in the functional domain; functions are mapped onto design solutions ($S_i$) in the design domain, which in turn are mapped onto part and process characteristics ($P_i$) in the process/hardware domain. Also illustrated in Figure 4.2 are the design verification loops; this shows that the design verification must be carried out against function at all levels of the design decomposition.

The separation between the functional and physical domains is essential, as it encourages engineers to focus on the functional requirements that need to be delivered by the system, rather that honing in on the design solution at hand. The systems engineering design cascade can be described as zigzagging iterations between the functional and design domains through the levels of the systems hierarchy, until a level of resolution is achieved where engineering design can be carried out (*i.e.* component level). It is essential that *all* functions are identified and mapped/cascaded, and not just the main functions; it is often the case that design engineers focus on main functions, and pay less attention to the functions that support system integration. Functional decomposition has further practical importance in that it helps to define the scope for responsibility of a design team (Eppinger, 1991).

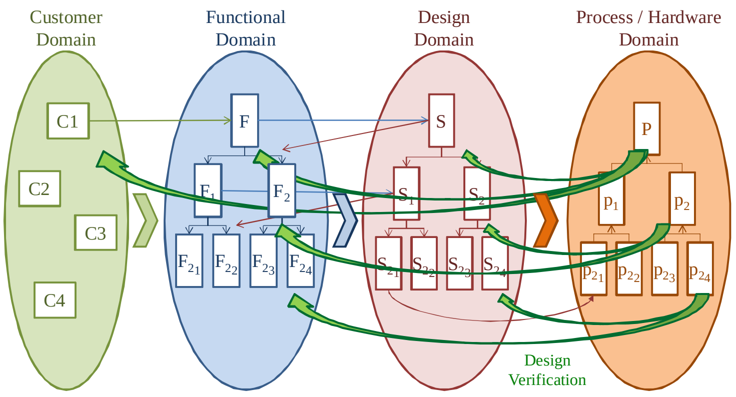


**Figure 4.2.** Functional decomposition by mapping and zigzagging between customer, function, design and process domains

The common practical approach to system decomposition is to use the “hardware” basis, *i.e.* decompose the system into elements. This is sometimes justified by the fact that automotive design is largely evolutionary, so the hardware structure is stable, in particular at high level (*e.g.* all car models will have a body structure, an engine or powertrain, transmission, driver interface, *etc*.). The function mapping is often done by attributing customer required functions (defined in the *customer* domain) to “hardware” clusters, or by setting up functional or “attribute” teams (*e.g.* driveability) with responsibility for the mapping and integration of a function across hardware groups. However, the increased complexity and multi-disciplinarity of the automotive systems, with a clear shift of focus towards mechatronic and control systems, have made this approach less effective and often impractical. This defines the need for a more structured process for function decomposition which satisfies the following criteria:

- Supports the upfront analysis of the system on a functional basis, leading to a decomposition based on functions, which are then mapped onto design solutions and hardware;
- Facilitates the analysis of interfaces between components and subsystems to identify all functions required for system integration to ensure a robust and reliable delivery of customer required functions;
- Is based on tools and methodology which can be applied across the engineering disciplines (mechanical, electrical, electronic, control and software systems);
- Is integrated with, or based upon, tools currently in use by automotive engineers, to encourage the take up of the process on a broad basis.

The following sections describe the function analysis framework and tools developed on the basis of the above considerations.

## 4.3.2 Function Decomposition Based on System State Flow Diagrams

The hierarchical functional decomposition is often difficult in practice (Ariyo *et al*., 2008) and can result in different function tree structures depending upon the team conducting the analysis. This can have severe consequences if support functions required for systems integration remain un-mapped.

Several function based analysis and decomposition frameworks have been discussed in literature (van Eck *et al*., 2007). The "functional basis" approach by Stone and Wood (2000) provides a consistent framework including a taxonomy for functions and a coherent representation of the overall function in terms of interconnected sub-functions, defined as operations on flows of *energy*, *material* and *signals* between identified inputs and outputs to the system. The Contact and Channel (Albers *et al*., 2009) framework provides a strong structure of support for functional decomposition. It is based on identifying working surface pairs (WSPs) at the system input and output, and the channel that connects the WSPs within the engineered system. A working surface is described in terms of a state characterised by measurable attributes, and the system function defined as "transfer between the states" (Albers *et al*., 2011). The functional decomposition is carried out by defining surface pairs with the channel, which correspond to design subsystems. While this framework is highly structured, it uses a taxonomy which is not conducive to the analysis of multi-disciplinary systems.

The system state flow diagram (SSFD) has been introduced (Campean and Henshall, 2008; Campean *et al*., 2011) as a diagrammatic approach to facilitate a more disciplined functional decomposition of the system. The fundamental idea behind the SSFD is the identification of discrete (stationary or pseudo-stationary) observable states of the flow of energy, material or information through a system, and then the identification of the functions that the system needs to provide in order to achieve the transition between successive states. The SSFD diagram convention is that the *states* are represented by boxes, which are joined by arrows which denote the *functions* that need to be achieved by the system to transition between states.

Figure 4.3 illustrates graphically a SSFD for an electric vehicle powertrain (EVP), based on a case study presented in Campean *et al* (2011). In an EVP there are three main flows, associated with the three main functions of the system, *i.e.*:

1) charge and store energy;
2) deliver controlled torque to the rear axle;
3) provide power for vehicle consumer units.

The SSFD analysis normally starts with the identification of the inputs (mains energy and driver demand) and outputs (controlled torque at rear axle and electric power to the fuse box) of the system. The SSFD in Figure 4.3 maps the flows through the system based on identification of states and functions that need to be provided to achieve the transitions between the states. For example, following the flow of electrical energy from the input (mains energy, alternative current AC), a next state of the energy flow is "electric energy/direct current (EE/DC)"; the function needed is to "*convert mains AC into DC*". On a diagrammatic representation such as the SSFD it is convenient to illustrate the mapping of functional requirements onto design solutions, established on the basis of design analysis and synthesis. At high level system analysis, as considered in the EVP example, design solutions are usually thought of in generic terms; *e.g.* a "*Charger*" is a generic design solution for the function to "*convert mains AC into DC*". This is illustrated in the SSFD in Figure 4.3, which includes the generic design elements in boxes alongside the functions they achieve.

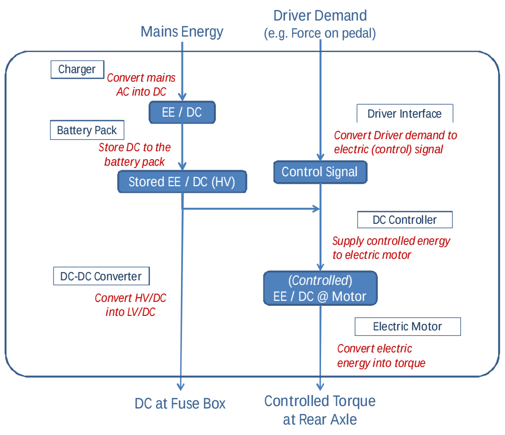


**Figure 4.3.** System state flow diagram for an electric vehicle powertrain (EVP) system

Thus, the SSFD is a composite graphical representation which combines an analysis in the function domain (mapping the main flows through the system in terms of states and functions), and also mapping the design elements in the design domain as systems that will deliver the function. From the SSFD we can extract a conventional function tree, illustrated in Figure 4.4. In common engineering practice the function tree would normally be derived through brainstorming. It is clear that mapping the states of the flow through the system provides a more objective way of deriving the function tree, addressing difficulties of multiple tree shapes for the same system discussed by Ariyo *et al.* (2008).

Given that the SSFD includes the design elements that deliver the functions, we can easily convert from the SSFD to a conventional system boundary diagram (SBD), illustrated in Figure 4.5, which is a representation of the system in the *design* domain, showing the system components as boxes, placed within the boundary of the system. The SBD also includes the mapping of the main energy flows through the system, represented as arrows connecting the boxes.

It is common practice to include in an SBD the external elements and systems with which the system interfaces (shown in Figure 4.5 outside the box which defines the system boundary). The double-headed arrows between the system boundary and the external interfacing system indicate that exchanges take place in both directions between the system and the external interfacing systems.

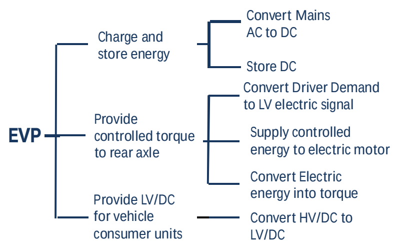

EVP
Charge and store energy
Convert Mains AC to DC
Store DC
Provide controlled torque to rear axle
Convert Driver Demand to LV electric signal
Supply controlled energy to electric motor
Convert Electric energy into torque
Provide LV/DC for vehicle consumer units
Convert HV/DC to LV/DC

**Figure 4.4.** Function tree for the EVP system

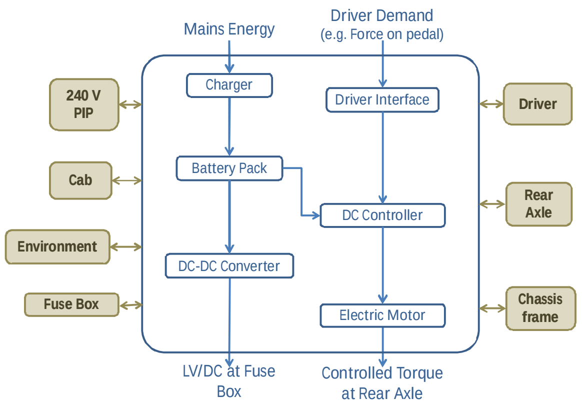


**Figure 4.5.** System boundary diagram for the EVP system

### 4.3.3 Interface Analysis

The SBD provides a concise graphical representation of the system, indicating the existence of interfaces between components. There can be multiple and complex exchanges at interfaces both within the system boundary and at external interfaces. The SSFD and the SBD are focused on the main energy flows through the system (associated with the main functions, and represented by arrows in the SBD) and, as such, do not provide a meaningful way of documenting multiple exchanges at interfaces. A matrix based tool, referred to as an *interface matrix* (IM), is commonly used in the automotive industry (Webb, 2002) to systematically analyse the interface exchanges. This type of analysis has been introduced in an automotive context by Pimmler and Eppinger (1994), referred to as *interaction matrix*. In broader literature this is commonly referred to as Design Structure Matrix (Browning, 2001).

Figure 4.6 illustrates an IM analysis for the EVP. The IM analysis includes both *internal* (*i.e.* within the system boundary) and *external* (*i.e.* between the system and external systems) interfaces. The analysis of the exchanges is carried out on a flow basis (Pimmler and Eppinger, 1994), *i.e.* identifying whether at any given interface there is a flow of *energy* (E), *material* (M) or *information* (I). It is also common practice to evaluate whether an interface involves *physical* (P) touch or contact; this information is primarily useful for capturing any geometrical compatibility requirements at the interface.

| | | INTERNAL INTERFACES | | | | | | EXTERNAL INTERFACES | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | A | B | C | D | E | F | E1 | E2 | E3 | E4 | E5 | E6 | E7 |
| | **P** Physical / **E** Energy / **I** Information / **M** Material | Charger | Battery Pack | Driver Interface | DC Controller | Motor | DC-DC Converter | *Driver* | *240V PiP* | *Cab* | *Environment* | *Chassis Frame* | *Rear Axle* | *Fuse Box* |
| 1 | Charger | ▨ | E | | | | | P E | P E | | E | P E | | |
| | | ▨ | I | I | | | | I | | | M | | | |
| 2 | Battery Pack | E | ▨ | | E | | E | P | | | P E | P E | | |
| | | I | ▨ | | | | | I | | | M | | | |
| 3 | Driver Interface | | | ▨ | | | | P E | | P E | P E | | | |
| | | I | | ▨ | I | | I | I | | I | M | | | |
| 4 | DC Controller | | E | | ▨ | E | | | | | P E | P E | | |
| | | | | I | ▨ | I | | | | | M | | | |
| 5 | Motor | | | | E | ▨ | | | | | P E | P E | P E | |
| | | | | | I | ▨ | | | | | M | | M | |
| 6 | DC-DC Converter | | E | | | | ▨ | | | | P E | | | P E |
| | | | | I | | | ▨ | | | | | | | |

**Figure 4.6.** Interface matrix for electric vehicle powertrain

While the IM provides a compact analysis of exchanges at interfaces, both internal and external, it does not capture the detail of the actual exchange, normally discussed by the engineering team while analysing a particular interface. This is particularly important as there could be multiple exchanges of the same type at an interface. More significantly, if an exchange (*i.e.* a flow of E, M or I) is identified at an interface, then a functional requirement must be specified to manage this flow. Any exchange can be either detrimental to the main function of the system (potentially leading to a robustness failure), or beneficial, if not essential, for the system function. In both cases a function is required to manage the exchange. An interface analysis table (IAT) has been suggested (Campean *et al.*, 2011) as an enhancement to the IM. The IAT extract, illustrated in Figure 4.7 as an example shows two interfaces - one internal (*Charger - Battery Pack*) and one external (*Motor - Chassis*). The table includes a description of the exchange, a statement of the engineering function required to manage the exchange and an evaluation of the effect of the interface exchange on the main ("high level") function to which it relates with this main function also being documented in the table. Following Pimmler and Eppinger (1994), the rating of the effect on the main function uses a numeric scale from -2 to +2, the "-" sign indicating that the effect is detrimental to the main function and therefore the interface exchange must be minimised, whereas the "+" sign indicates a beneficial exchange which must be provided to support a main function of the system.

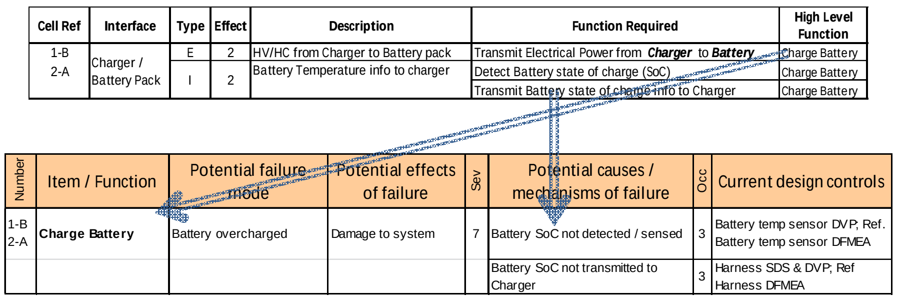
| Cell Ref | Interface | Type | Effect | Description | Function Required | High Level Function |
|---|---|---|---|---|---|---|
| 1-B 2-A | Charger / Battery Pack | E | 2 | HV/HC from Charger to Battery pack | Transmit Electrical Power from ***Charger*** to ***Battery*** | Charge Battery |
| | | I | 2 | Battery Temperature info to charger | Detect Battery state of charge (SoC) | Charge Battery |
| | | | | | Transmit Battery state of charge info to Charger | Charge Battery |
| 5-E6 | Motor / Chassis | P | 2 | Motor mounted on chassis | Mount motor securely | Propel Vehicle |
| | | E | 2 | Electric exchange motor - chassis | Isolate motor electrically from chassis | Propel Vehicle |
| | | E | 2 | Ground motor electrically | Maintain electrical contact to ground through chassis | Propel Vehicle |
| | | E | 2 | Heat exchange from motor to chassis | Dissipate heat from MCU through chassis | Propel Vehicle |

**Figure 4.7.** Interface analysis table (IAT) for EVP

It is good practice for the IAT to include the unit or method of measurement of the interface exchange and the associated function, as well as a range or target for the function. It is important that the interface functions are part of the functional requirements specification for the system, and thus part of the functional requirements cascade. For example, we can extract from the IAT all functions associated with the "*Charger*" which will form the basis for the functional requirement specification. While it is tempting to attribute an interface function requirement to one or other of the interfacing systems, this is not always beneficial early in the design process as it might unduly narrow the engineering design options. The recommendation is that interface functions should be cascaded to both interfacing elements with the decision as to which element(s) provides this function left until later in the design process.

## 4.4 Information Flow within the FMA Process

The IAT is a very comprehensive, information rich document which not only provides a sound basis for the functional requirements specification and cascade, but also feeds into the other tools in the FMA process. The flow of information from the IAT to the other tools in the FMA process is shown in Figure 4.1, with Figure 4.8 illustrating the flow of information from the IAT to the FMEA, which is the main tool for the FMA process step 2 function failure analysis. At each level of analysis (*i.e.* system, subsystem, component) the FMEA focuses on the main functions for the system under investigation. In taking a function failure mode approach to FMEA (Stamatis, 2003; McDermott *et al.*, 2008), the potential failure modes are (i) no function, (ii) partial function, (iii) intermittent function or (iv) function when not required (command failure). Figure 4.8 illustrates an example of partial function failure of the function "charge the battery", showing that the interface functions documented in the IAT provide the potential root causes of this function failure mode, recognising that failure to manage an interface function is likely to cause system failure. Thus completing the FMEA based on the IAT is a much more straightforward process compared to the conventional approach to FMEA (which starts with "brainstorm functions" and continues with the identification of causes in manner which often tends to be based more on the previous experience of failure than on a more fundamental engineering approach). The experience with the FMA process depicted in this paper is that the SSFD based functional decomposition facilitates better (more concise and precise) FMEAs than those developed in a more conventional manner.



| Cell Ref | Interface | Type | Effect | Description | Function Required | High Level Function |
|---|---|---|---|---|---|---|
| 1-B 2-A | Charger / Battery Pack | E | 2 | HV/HC from Charger to Battery pack | Transmit Electrical Power from ***Charger*** to ***Battery*** | Charge Battery |
| | | I | 2 | Battery Temperature info to charger | Detect Battery state of charge (SoC) | Charge Battery |
| | | | | | Transmit Battery state of charge info to Charger | Charge Battery |

| Number | Item / Function | Potential failure mode | Potential effects of failure | Sev | Potential causes / mechanisms of failure | Occ | Current design controls |
|---|---|---|---|---|---|---|---|
| 1-B 2-A | **Charge Battery** | Battery overcharged | Damage to system | 7 | Battery SoC not detected / sensed | 3 | Battery temp sensor DVP; Ref. Battery temp sensor DFMEA |
| | | | | | Battery SoC not transmitted to Charger | 3 | Harness SDS & DVP; Ref Harness DFMEA |

**Figure 4.8**. Illustration of flow of information from IAT to FMEA

There is a similarly straightforward flow of information from the IAT to other FMA tools (such as function fault tree analysis and P-diagram), supporting both FMA step 3 "countermeasure development" and the FMA step 4 "robust design verification". The fundamental conjecture is that a noise factor can only affect a system through an interface and so if the interface analysis is complete (*i.e.* all exchanges have been identified and characterised, and engineering functions specified), then all noise factors that can affect the functional performance of the system should have been captured (so there is no need to "brainstorm" noise factors in developing a P-diagram). Consequently, countermeasure development is in fact design optimisation based on all functional requirements and constraints (expressed in relation to the interface management functions) documented in the IAT. Similarly, the robust design verification process should demonstrate that the system achieves its function given the effect of the interface exchanges identified in the IAT, *i.e.* these interface exchanges need to be included in the design verification matrix.

## 4.5 Integration with the Automotive Systems Engineering Design

Systems engineering design in automotive industry is carried out at successive levels from high level system-of-systems (*e.g.* vehicle level) down to subsystems (*e.g.* powertrain), sub-subsystems (*e.g.* charger unit) and components (*e.g.* sensor), as illustrated by the Systems Engineering Vee model in Figure 4.9.

The FMA process should be applied at each level, starting with the highest level. Function analysis should always start at the highest level possible, where it can be directly linked to customer requirements, followed by iterative decomposition, setting the scope and resolution for analysis at each level. Within an FMA based SED framework, the functional requirement cascade should be underpinned by the IAT, which documents all functions needed, both for the main flow and the interface exchange management. Function failure modes must also be identified and prioritised early in the process, *i.e.* starting with the highest level system analysis, and cascaded down through to subsystems. As discussed earlier, ultimately, robust countermeasure development is based on robust design optimisation and as such can only take place at component level. Therefore countermeasure development cannot be completed at the higher levels system analysis (*e.g.* vehicle or powertrain), but functional requirements (including interface requirements) and critical function failure modes are cascaded down through the Systems Engineering Vee, to support subsequent robust countermeasure development. Design verification can be planned at each level of the system analysis, incorporating the effect of noise factors identified through the interface analysis. Verification tests can be planned at each system level to ensure that the reliability and robustness of the system is confirmed given the design countermeasures are in place.

The integration of the FMA process described in this paper with the SED Vee framework is illustrated in Figure 4.10. This shows the holistic two-dimensional integration of the FMA framework with the SED framework, based on:

- *Horizontal integration*: based on the iterative deployment of the FMA process at each system engineering level – from vehicle level down to powertrain, charger unit and sensor component;
- *Vertical integration*: based on the cascade of functional requirements, critical failure modes and design verification plans through the system levels, with iterative upward validation on the basis of the design verification results.

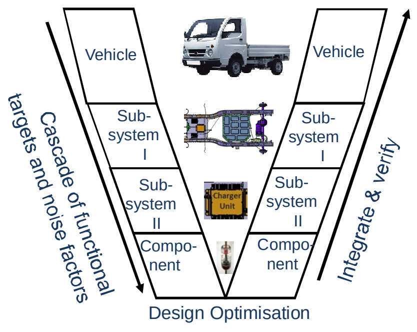


**Figure 4.9.** Systems Engineering Vee

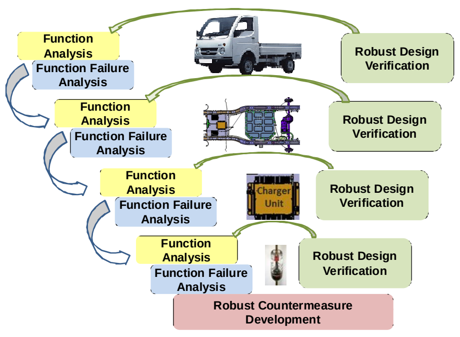

Function Analysis
Function Failure Analysis
Robust Design Verification
Function Analysis
Function Failure Analysis
Robust Design Verification
Function Analysis
Function Failure Analysis
Charger Unit
Robust Design Verification
Function Analysis
Function Failure Analysis
Robust Design Verification
Robust Countermeasure Development

**Figure 4.10.** SED framework based on FMA

It is important to note that the IAT provides the basis of a strong information flow throughout the systems engineering cascade. For example, from the EVP IAT we can extract all the functional requirements associated with the *charger unit*, (including interface functions) and cascade these down from the powertrain level to the charger as a subsystem. The subsystem level analysis will be carried out by a different team, often at a different company (*e.g.* a supplier) if the component is outsourced. It is therefore very important that all interface functions are identified and cascaded, as well as the significant potential failure modes, because otherwise the supplier team are unlikely to have a vision and understanding of the exchanges between their system and other systems (both internal and external). The implication is that the subsystem FMA analysis should be carried out within the context of the system level analysis, rather than as a stand-alone separate analysis, which is often the engineering practice. For example in analysing the charger unit, we are in fact zooming in with the analysis to one of the subsystems within the EVP. Looking at this cascade in terms of the SBD, it is clear that the charger unit will have the same external interfaces as the EVP, shown in Figure 4.5 (although the charger will not interface with all EVP external elements - *e.g.* there should be no interface with the electric motor), and some of the EVP internal interfaces will become external interfaces for the charger (*e.g.* the battery pack is an external interface for the charger unit). Figure 4.11 illustrates the cascade of interfaces to the charger unit in graphical format, the detail of the interface exchanges being already documented in the EVP IAT.

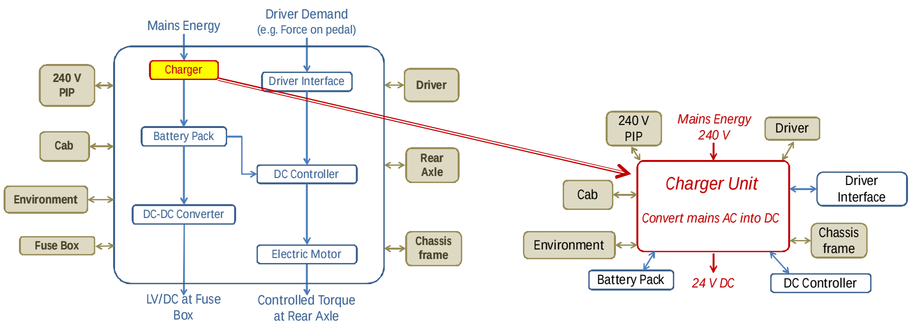


**Figure 4.11**. Illustration of the cascade of interfaces to subsystem level

Being integrated with the SED framework, the FMA process is also integrated with the Product Development framework. In this context the FMA process fulfils an important role in that it provides the information content for effective gateway review process, on the basis of functional requirements, critical functional failure modes and robust design verification outcomes (Campean *et al*., 2013).

## 4.6 Discussion and Conclusions

The aim of this paper was to present an approach to systems engineering design which embeds the failure mode avoidance paradigm, framework and support tools. The FMA framework developed by the Engineering Quality Improvement Centre at the University of Bradford is based on a four step process, illustrated in Figure 4.1, which is iteratively applied at all levels within the systems engineering cascade, as illustrated in Figure 4.9. The FMA framework has a very strong functional basis, and it promotes the discipline of maintaining the domain separation throughout the systems engineering design process. This is seen as an effective way of avoiding the common pitfall of honing in on a known design or "hardware" solution, without considering the functional requirements in a holistic way (*i.e.* including the system integration requirements).

A main focus of the paper has been the discussion of a structured approach to function decomposition. This has great practical importance, not just in order to produce a representation of the team's understanding of how the system achieves its functions, but it also helps to define the scope for responsibility for the design teams on a functional basis. This is useful both in terms of aligning the PD organisation with the functional decomposition of the system, and also in communicating functional requirements for outsourced subsystems.

The function analysis and decomposition is based on three main tools: system state flow diagram, boundary diagram, interface matrix and interface analysis table. The function tree, which has long been regarded in industry as the main tool for function analysis, can be derived as a by-product from the more structured and fundamental SSFD tool. The iterative use of these tools, as discussed and illustrated in Section 4.2 of this chapter, provides a highly structured framework which maintains the separation of the domains throughout the analysis, leading to a complete and comprehensive functional decomposition and mapping, covering both the main functions and those required to manage interfaces. Information gained from this analysis is compactly documented in the IAT. Of the methods discussed in the literature, the contact and channel method (CCM) (Albers *et al.*, 2009) offers a similarly structured and comprehensive approach, which offers both a rigorous functional decomposition and potential for identifying interface exchanges as functional requirements. However, the CCM appears to be less portable across engineering disciplines in particular on modelling information flows, and it is less integrated with other tools commonly used in the automotive industry, which will likely inhibit a large scale take up by the engineering teams.

The IM tool widely used in the automotive industry is similar to the design structure matrix (DSM) (Browning, 2001; Clarkson *et al.*, 2004), except that it places a strong emphasis on the external interfaces (with external elements and systems) - which play an important role within the automotive industry. As argued by Davis (2007), the noise space that a vehicle is subject to even under "normal" driving conditions is much more complex than the noise space in industries such as nuclear or even aerospace. The framework used by the IM tool to identify interface exchanges is based on the generic classification of flow as energy/material/information and physical touch. An alternative framework discussed in literature is the so-called “linkage modelling method” (LMM) (Jarratt, 2004), which suggests a characterisation and classification of interface exchanges in more detail - which has some advantage from a mechanical engineering analysis point of view, as it provides a more accurate description of the linkage compared to the PEIM framework. However, the IM approach is much easier to apply to a multi-domain context - which is an important practical consideration.

The authors’ extensive experience of facilitating the implementation of this process in a real world automotive systems engineering design context has been very positive. Feedback from engineering teams working across different systems (representative of the multi-domain context of automotive systems engineering design) has highlighted (i) the structured approach to function decomposition which removes the reliance on brainstorming, delivering a more objective and comprehensive analysis; (ii) the portability of the approach across multiple domains - the same tools and process can be used to analyse predominantly mechanical components as well as software features; and (iii) the strong integration of the whole process through the information flow between the tools. While completing the function analysis tools, in particular the IAT, still require a significant effort/resource, this is seen as an integral part of the systems engineering design process (as the basis for functional requirement specification) and it greatly simplifies the completion of the FMA downstream tools - such as the FMEA. Most importantly, this analysis is carried out early in the product development process, providing strong facilitation for failure mode identification.

## 4.7 References

Albers A, Braun A, Clarkson PJ, Enkler H-J, Wynn DC (2009) Contact and channel modelling to support early design of technical systems. International conference on engineering design. In: Proceedings of the 17th International Conference on Engineering Design (ICED’09), Stanford, CA, US

Albers A, Oerding J, Alink T (2011) Abstract objectives can become more tangible with the contact and channel model (C&CM). In: Proceedings of the 20th CIRP Design Conference, Nantes, France

Ariyo OO, Eckert CM, Clarkson PJ (2008) Hierarchical decomposition for complex product representation. In: Proceedings of the 10th International Design Conference (DESIGN 2008), Dubrovnik, Croatia

Browning TR (2001) Applying the design structure matrix to system decomposition and integration problems: A review and new directions. IEEE Transactions on Engineering management, 47(3): 292-306

Campean F, Henshall EJ (2008) A function failure approach to fault tree analysis for automotive systems. SAE Technical Papers Series 2008-01-0846

Campean F, Henshall EJ, Brunson D, Day A, McLellan R, Hartley J (2011) A structured approach for function analysis of complex automotive systems. SAE International Journal of Materials and Manufacturing, 4(1): 1255-1267

Campean F, Henshall EJ, Rutter, B (2013) Systems engineering excellence through design: An integrated approach based on failure mode avoidance. SAE Technical Paper Series 2013-01-0595

Cash PA (2003) Right first time production releases. MSc Thesis, University of Bradford, Bradford, UK

Chakrabarti A, Bligh TP (2001) A scheme for functional reasoning in conceptual design. Design Studies, 22: 493-517

Clarkson PJ, Simons CS, Eckert CM (2004) Predicting change propagation in complex design. ASME Journal of Mechanical Design, 126(5): 788-797

Clausing DP (2004) Operating window: An engineering measure for robustness. Technometrics, 46(1): 25-29

Davis TP (2006) Science, engineering, and statistics. Applied Stochastic Models in Business and Industry, 22(5-6): 401-430

Davis TP (2007) Failure mode avoidance. Short Course at the University of Bradford, Bradford, UK

Eppinger SD (1991) Model-based approaches to managing concurrent engineering. Journal of Engineering Design, 2(4): 283-290

Henshall EJ, Campean F (2009) Implementing failure mode avoidance. SAE Technical Paper Series 2009-01-0990

Jarratt TAW (2004) A model-based approach to support the management of engineering change. PhD Thesis, Cambridge Engineering Design Centre, University of Cambridge, Cambridge, UK

McDermott RE, Mikulak RJ, Beauregard MR (2008) The basics of FMEA, 2nd edn. Productivity Press, New York, NY, US

Pimmler TU, Eppinger SD (1994) Integration analysis of product decompositions. In: Proceedings of the ASME 6th International Conference on Design Theory and Methodology, Minneapolis, MN, US

Stamatis DH (2003) Failure mode and effect analysis: FMEA from theory to execution, 2nd edn. American Society for Quality Control, Milwaukee, WI, US

Stone RB, Wood KL (2000) Development of a functional basis for design. Journal of Mechanical Design, 122(4): 359-370

Suh NP (1995) Designing-in of quality through axiomatic design. IEEE Transactions on Reliability, 44(2): 256-264

van Eck D, McAdams DA, Vermaas PE (2007) Functional decomposition in engineering: A survey. In: Proceedings of the ASME International Design Engineering Technical Conference (IDETC/CIE2007), Las Vegas, NV, US

Wasmer A, Staub G, Vroom RW (2011) An industry approach to shared, cross-organisational engineering change handling - The road towards standards for product data processing. Computer-Aided Design, 43(5): 533-545

Webb RD (2002) Investigation into the application of robustness and reliability tools to the design process. MSc Thesis, University of Bradford, Bradford, UK

Zhou J (2005) Reliability and robustness mindset in automotive product development for global markets. SAE Technical Papers Series 2005-01-1212