DEVELOPMENT OF AN INTERFACE ANALYSIS TEMPLATE FOR SYSTEM DESIGN ANALYSIS

Uddin, Amad; Campean, Felician; Khan, Mohammed Khurshid
University of Bradford, United Kingdom

Abstract

Interface definition is an essential and integral part of systems engineering. In current practice, interface requirements or control documents are generally used to define systems or subsystems interfaces. One of the challenges with the use of such documents in product development process is the diversity in their types, methodology, contents coverage, and structure across various design levels and across multidisciplinary teams, which often impedes the design process. It is important that interface information is described with appropriate detail and minimal or no ambiguity at each design level. The purpose of this paper is to present an interface analysis template (IAT) as a structured tool and coherent methodology, built upon a critical review of existing literature concepts, with the aim of using and implementing the same template for capturing interface requirements at various levels of design starting from stakeholders' level down to component level analysis. The proposed IAT is illustrated through a desktop case study of an electric pencil sharpener, and two examples of application to automotive systems.

Keywords: Design methodology, Interface, Interaction, Interface requirement document, System design levels

Contact:
Amad Uddin
University of Bradford
School of Engineering Design and Technology
United Kingdom
a.ud-din1@student.bradford.ac.uk

Please cite this paper as:
Surnames, Initials: *Title of paper*. In: Proceedings of the 20th International Conference on Engineering Design (ICED15), Vol. nn: Title of Volume, Milan, Italy, 27.-30.07.2015

1 INTRODUCTION

Interface definition and analysis are core activities that describe the performance of the system and require sensibly chosen and effective *tools* deployed as early as possible in the development process to evaluate and model the system interface requirements. It is important that the selected tools capture and communicate system's interface information with minimal or no ambiguity (Lalli et al., 1997). Incomplete definition of interfaces lead to un-necessary design iterations in systems engineering design resulting in growing costs and delays in delivery time. Therefore, an efficient tool for capturing rich interface information is fundamental to systems engineering design.

In the engineering design literature there are numerous tools presented to define and analyse system interfaces. Tools based on the design structure matrix (DSM) are extensively used in academic research for analysing interactions among subsystems in a system architecture (Browning, 2001) as well as for change propagation analysis (Hamraz et al., 2013). A limitation of many DSM approaches is that the analysis is exclusively focused on subsystems within the system boundary and don't often extend to higher level external system interfaces (such as stakeholders, user, designer, supporting systems and natural environment etc.).

Systems engineering disciplines have evolved to promote the system-of-systems thinking with a sharp focus on understanding and characterising systems interfaces as the basis for requirements specification. The common practice to systems engineering is underpinned by a number of graphical tools such as context diagram, use case diagram, activity, sequence diagrams, are commonly deployed through UML / SysML software tools (OMG SysML, 2012) to model and document the system's interactions with its high level interfaces such as user, designer, maintainer, and supporting systems (Kossiakoff et al, 2011; Buede, 2009; Weilkiens, 2006). These tools aim to provide a high level abstract view of the system. Individually, each tool promote a different representation and thus reflect a different viewpoint of the analyst, which is both a strength and weakness: strength because analysing a system from several viewpoints allows a more comprehensive elicitation of interfaces and interactions; however, this is time consuming, and sometimes can lead to lack of integration between the analysis conducted from different viewpoints, in particular in relation with coupling the downstream analysis of physical systems - which is a very important consideration for the development of complex multidisciplinary systems.

Tabular templates for interface requirements and control have also been introduced, (Lalli et al, 1997; Wasson, 2006; Buede, 2009; Campean et al, 2013; Grady, 2014;) to capture the detail of the interface exchange and convert into functional requirement for system. The key principle is that functional requirements need to be introduced in order to manage interface exchanges, and these functional requirements need to be both cascaded through the systems levels and appropriate design verification methods need to be put in place to validate the system integration (Campean et al, 2013). Many software packages have been developed for capturing customer and system technical requirements such as CORE, DOORS, CaseComplete, underpinned by common system modelling graphical tools and features; none of these agreed to design a standard common template for documenting, representing and communicating the interfaces information that could be applied in different design levels with similar methodology. Even though there are standards and outlines for the form of interface requirements documents in certain domains, such as medicare and medicaid services (CMS) (CMS, 2013), aircraft/stores (Schlatt, 2004) national aeronautics and space administration (NASA) (NASA, 2007), there is no universal standard for the content and form of interface requirements documents, and this has been highlighted as an area for improvement (Rahmani and Thomson, 2012).

The research presented in this paper aims to address this lack of common standard interface analysis template with a common methodology that could be used across different technical disciplines and system levels, starting from stakeholders' level down to component level for capturing the rich interface information between two interacting systems or entities. An interface analysis template (IAT) is introduced, as a tool that could be applied across system design levels and on different nature interfaces with a common structure and methodology. The proposed framework is of a tabular type as capturing the details of the exchanges at the interfaces and the associated functional requirements is essential for real world complex systems design. This development is based on an in-depth analysis of current frameworks for interface analysis, presented in the next section of the paper. A desktop case study of the design of an electric pencil sharpener is used to explain key features of the framework and

to demonstrate its deployment. The application of the tool is further illustrated in relation to two real world case studies.

2 LITERATURE REVIEW ON INTERFACE DEFINITION AND ANALYSIS

In engineering design and modelling literature, the terms 'interface' and 'interaction' are often used interchangeably while very few texts describe the clear difference between the interface and interaction such as in (Fosse and Delp, 2013; Gedell et al., 2011; Miller and Elgard, 1998).

The term *interface* is generally used to denote the shared boundary between two systems facing each other (Miller and Elgard, 1998; NASA, 2007; Otto and Wood, 2001). In design literature, researchers define and conceptualise an interface in numerous ways such as a plane or place (Grady, 2014), a logical or physical relationship (Rahmani and Thomson, 2012), an internal feature (Gedell et al., 2011), an intended interaction location (Liang and Paredis, 2004), a spatial region (Wie et al., 2001), a linkage (Mikkola, 2001), a mating face (Blackenfelt, 2000) and also in other ways in (Fosse and Delp, 2013; Sellgren and Anderson, 2005; Sellgren, 1998; Ullman, 2010). In interface modelling, the first step is often to define the boundary of the system which requires the identification of the interacting entities in its environment that can be human, supporting systems, and natural environment (Kossiakoff et al., 2011; Lalli et al., 1997). An interface can involve a geometric connection as well as non-contact *interactions* between two systems (Ulrich, 1995).

In design literature, the term *interaction* or interaction-*exchange* is commonly defined in relation to the input-output *flow* and *form* aspects between interacting systems (Miller and Elgard, 1998; Pimmler and Eppinger, 1994). Pimmler and Eppinger's four-interaction exchange taxonomy is built on the fundamental functional models (Pahl et al., 2007) in relation to flows of energy (E), material (M), Information (I) with the consideration of form aspect related to adjacency and orientation i.e. spatial (S). This taxonomy is widely used in current automotive practice for interface analysis (Campean et al., 2013), often referring to physical (P) interactions with same abstract meaning as 'S'. Many researchers have adopted (Hamraz et al., 2013; Otto and Wood, 2001; Rahmani and Thomson, 2012) and adapted (Blackenfelt, 2001; Helmer et al., 2010; Sosa et al., 2003) this four-interaction taxonomy. For example, Blackenfelt (2001) looking at product variety and modularity in embodiment design, replaces the 'S' relation by inter-domain relation of 'FP' i.e. two entities contributing to the same function or parameter (FP). Sosa et al. (2003) extended the four-exchange taxonomy with an introduction of fifth type as 'structural' that indicates the requirements related to transferring loads or containment between two interfacing entities. Since there can be geometric connections between interacting components (Ulrich, 1995), Cansler et al., (2014), looking at mapping the excess relationships, recently introduced geometric exchange category that indicates the requirements related to length, volume, and area. Thus, there can be form (S, P) and flows (E, M, I) related exchanges in an interface. Pimmler and Eppinger (1994) also suggest that the definition of interaction types can vary depending upon the context of the given problem. Two other distinct evidences to this fact is found in Jarratt (2004) engineering change management method and in Bruun et al. (2013) interface diagram tool that supports developing modularity of complex systems.

The *interaction information* between interfacing systems needs to be sufficiently detailed to communicate meaning with minimal or no ambiguity (Kossiakoff et al., 2011; Lalli et al., 1997). *Interaction requirement* (i.e. statement) is normally made in relation to the functional or logical relationships (including physical characteristics) that are required to exist at a system boundary in its operating environment (Wasson, 2006). There can occur many operational scenarios or instances in a system-entity interface. The operational *scenarios* aim to consider all possible system-entity interactions, activities, conditions, and assumptions that might occur under certain or worst case conditions (Wasson, 2006). In essence, scenarios include consideration of the fact how two entities should interact. Once the scenarios are identified, they can provide a basis for translation into specification requirements suitable for system design (Wasson, 2006). Therefore, interaction scenarios can be stated in terms of interaction descriptions.

In existing literature, the interaction *requirement's description* is usually articulated in three ways that reflect somehow slightly different meanings such as using a natural or verbal language for describing generic interface information in terms of *interface exchanges* (Campean et al, 2013), verb-noun and/or standard shalls format for describing *interface functions* (or *interface requirements*) (Grady, 2014; Liang and Paredis, 2004; Wasson, 2006), and the noun-format for describing the system *input-output*

flows or exchanges from/to external entities e.g. in context diagram (Kossiakoff et al., 2011). Interface requirements are often described as statements and constraints that define the reception of inputs and transmission of outputs between the system and the system's environment (Buede, 2009). In some texts, a combination of aforementioned formats is also used e.g. a specific interface exchange description begins with noun-format and then thereafter interface functional requirement is specified with verb-noun to manage that interaction exchange (Campean et al, 2013).
The interface requirement *specification* should also address the qualitative or quantitative controlled attribute or parameter with constraint or bounding relations (<, >, =), target value, and unit (Grady, 2014). The interface specification is derived from the interface requirements that describes the desired mechanical properties and logical connection between interfacing systems thereby including the format and structure of the exchange data (Kossiakoff et al., 2011; Wasson, 2006).
One of the crucial decisions in the interface definition and analysis is to assess or prioritise the *interactions criticality* that are normally associated with interaction exchanges among two systems. It is an important consideration as some interactions are more important than others. According to Pimmler and Eppinger (1994) interaction exchanges can have desirable effects (i.e. necessary for functionality) or detrimental (i.e. causing negative effects thus affecting system functionality). They use five-point scale scheme from -2 to +2 to highlight the interactions criticality to the system function. This scheme was further improved by Helmer et al (2010) in many aspects with a particular focus on clustering technique and that which interaction to prioritise over another in an interface having different ratings. Helmer et al (2010) summarise direction of decreasing importance in following order (+2) > (-2) > (+1/-1) > (+0.5/-0.5) > 0. The other schemes have also been developed to assess the subsystems interfaces in different study contexts such as evaluating risks (Hamraz et al., 2013) and evaluating interface standards (Cabigiosu et al, 2013).
As a conclusion, an interface definition and analysis activities require identification of system interfaces, categorisation of interactions in an interface, their requirement's description, specification, criticality or prioritization and the documentation (Lalli et al, 1997). There is one more category i.e. to analyse compatibility from control perspective which is not a main theme of this research.

3 INTERFACE DEFINITION AND ANALYSIS APPROACH

3.1 The IAT Framework

Based on the critical analysis of existing literature, the interface analysis can be summarised as a six steps process discussed below; an Interface Analysis Template (IAT), illustrated in Figure 1, has been introduced to document the analysis.

C1	C2	C3	C4	C5			C6	C7	C8	C9	C10
Interface	Interaction Code	Generic Interaction Scenario Description	Interaction Exchange Type	Interaction Function and Exchanges			From	To	Requirement Specifications (Attribute + Target Value + Relation + Unit)	Interaction Criticality	High Level Function
				Verb	Object/Noun						
					Input	Output					

Figure 1. Structure of interface analysis template (IAT)

(1) Identify System interfaces [IAT C1&2]: An interface describes a system's boundary with respect to external entities that can impact or influence each other in unidirectional or bidirectional manner. This step requires identification of interacting entities around the system; interfaces are documented in columns C1 and C2 of the IAT.
(2) Define Interaction scenario [IAT C3]: Interaction scenarios describe how a system - entity interface can or should interact. These are the operational scenarios that are defined in narrative or verbal manner thereby describing the all possible interactions, activities and assumptions that can occur at the system-entity interface. The descriptive statement, documented in IAT column C3, can start either with the system (subject) or the entities (modifiers) interacting with it.
(3) Interaction exchanges & classification [IAT C4-5]: The four interactions taxonomy (i.e. energy-E, material-M, information-I, and physical-P) was adopted and extended to distinguish between Physical touch and Spatial relationships at the interface. E-M-I exchanges are related to flow and S/P are related to form aspects. The S describes the adjacency and orientation relation among two entities when they are not in physical contact whereas P describes the adjacency and orientation relationship when they

are in physical contact. Interaction exchanges can be stated in noun-format describing either a flow aspect as an input/ output or a form aspect. The IAT column C4 is used for documenting interaction exchange *type* and column C5 for the interaction *exchange* definition as "object/noun".
(4): Interaction function [IAT C5-7]: Interaction function describes an operation of a system with its interacting entities in a *verb* format and in distinct manner which may not be possible in interaction scenario. Interaction function should be articulated from the perspective of the system of interest based on the functional basis taxonomy (Stone and Wood, 2000) and the five interface functions taxonomy from (Scalice et al., 2008) i.e. verb-object pair. The verb describes an operation at system's interface whereas object defines either flow or form related aspect in terms of inputs-outputs. Column C5 of IAT (Figure 1) is used for this purpose. Columns C6 and C7 help in determining the interaction directionality (i.e. system's interaction functions and exchanges to/from external entities). Therefore, a *system interaction function* is described via verb-object pair. For example, a sharpener system can have two interaction functions with the user interface in terms of inputs and outputs: "import (verb) human energy (object exchange)" as an input to the sharpener from user and the "transmit mechanical shock" as an output from sharpener to the user.
(5) Interface requirement specification [IAT C8]: An interaction function can be expressed in terms of its requirement specifications. A specification should specify the controlled attribute or parameter with constraint or bounding relations (<, >, = or minimum and maximum), target value, and unit.
(6) Interaction criticality [IAT C9 & C10]: Interaction criticality specifies how much a certain requirement *specification* is critical among interacting entities which in turn critical for the overall functionality accomplishment. The five point rating scheme (i.e. -2, -1, 0, 1, 2) of Pimmler & Eppinger (1994) was adopted. The evaluation of criticality is in relation to the *high level function* affected or context / use case scenario. A high level objective function (an operational requirement) of a system is represented in column C10 of IAT which describes the overall system's context that has to be accomplished via other interacting entities. One can define the context of the problem particularly related to the life-cycle phases such as 'design the sharpener', 'transport the sharpener', operate the sharpener', 'dispose the sharpener', and 'manufacture the sharpener' etc.

3.2 UML Illustration of Relationships within the IAT

The relations among the interface contents of IAT are now discussed using a formal modelling language UML class diagram as shown in Figure 2.

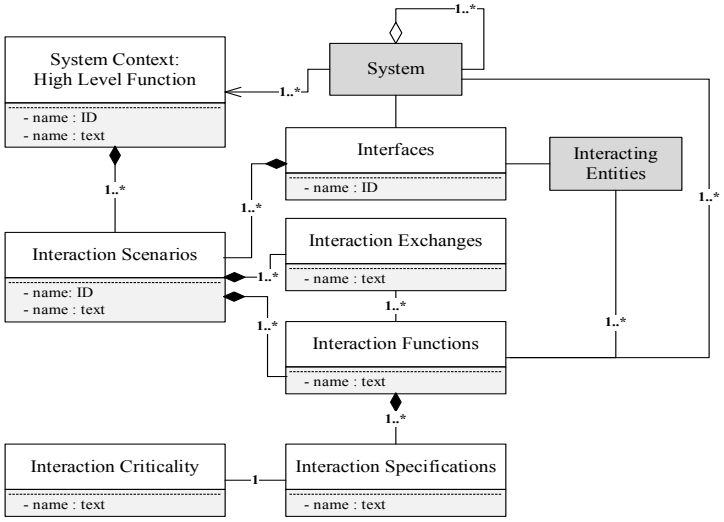

Figure 2. IAT approach key contents and their relations via UML model

Figure 2 shows that a system requires external entities at its interfaces to achieve the high level objective function (use case). The external entities can be a user, designer, supporting enabling system, and/or natural environment etc. To achieve this high level objective function, many interaction scenarios are possible among system and its external entities interfaces. Each interaction scenario represents how a high level function can be achieved, applied, used, or misused. The interaction scenarios can possess multiple interaction exchanges (both flows and form related) that are in turn

constrained and specified by interaction requirements i.e. interaction (or interface) functions. These interaction functions identify what the system has to do in terms of inputs and outputs in association with its external interacting entities as well as for the fulfilment of high level function. The interaction functions are quantified by at least one or more (performance) requirements specifications. An interface is assumed to be failed if it does not deliver the desired interaction function and performance requirement when required as part of an overall (high level) function. Each performance requirement criticality is quantified or ranked using a five scale scheme.

To summarise, following structured hierarchical information flow scheme is established (see Figure 3) to generate interface information from each of the steps explained earlier in a systematic manner;

- There can be one or many interfaces to and within the system of interest (i.e. system),
- There can be one or many interaction scenarios in an interface,
- There can be one or many interaction exchange types in an interaction scenario,
- There can be one or many interaction functions associated with an interaction exchange type,
- There can be one or many interaction requirement specifications with an interaction function, and
- There is only one interaction criticality associated with a requirement specification.

This one-to-one and one-to-many relationships is already depicted in UML model.

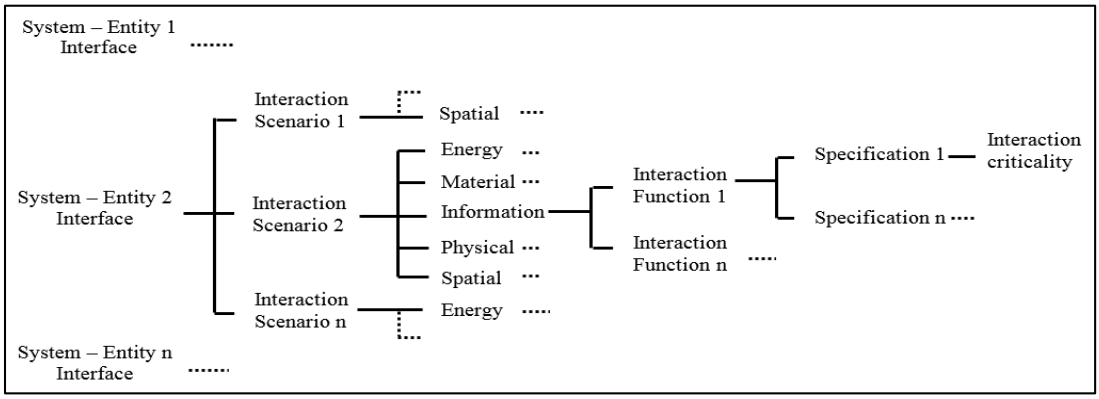

Figure 3. Hierarchical representation of left to right information flow in IAT [C1 to C9]

3.3 IAT Example 1: System Design Analysis for an Electric Pencil Sharpener

To illustrate the deployment of the proposed IAT framework, a desktop case study of the design analysis of an electric sharpener was considered. For brevity, only one specific context was considered, for which high level objective function is defined as "Operate the sharpener to sharpen the lead pencil".

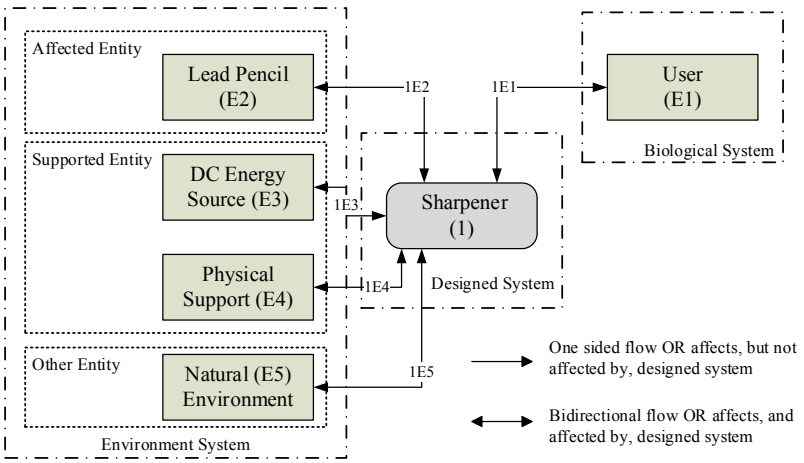

Figure 4. System block diagram for electric sharpener

3.3.1 Step 1: Identify the system interfaces

Figure 4 shows an adapted version of a context diagram (or system block diagram-SBD) at system level (i.e. black box level) for the electric sharpener system. This diagram scopes down the context for the system design team and identifies external entities as sources for inputs into the system and destinations of outputs from the system for the achievement of high level function "operate the sharpener to sharpen a pencil". There can be multiple interactions between external entities and the system which are usually represented via a combination of arrows and specific labels with abstract meaning and information, as observed in context diagram (Kossiakoff et al., 2011). Here, we omit the abstract information. The reason behind this is that interaction can represent anything such as a set of flows and/or activities between an external entity and the system. Labelling each such interaction in the SBD between two entities will lead to unnecessarily complex diagram, and timely process. Therefore, it would be better if initially only the distinction between influencing and affecting entities upon each other are made, which is, consistent with the systems theory (Hitchins, 2007). This can be visualised via bidirectional or one directional arrows. For example, in Figure 4, an interface '1E1' between an electric sharpener and user represent bidirectional relationship. A single arrowhead indicates whether an entity affects or affected by the system. These interfaces are listed down in column C1 of IAT e.g. for user-sharpener interface, shown in Table 1 along with high level function in C10.

3.3.2 Step 2: Define the interaction scenarios

Having established the SBD and the high level objective function, the next step is to identify the possible interaction scenarios between the external entities and the system. This can be done in two ways: (i) analyse external entities one at a time; or (ii) all entities at once. We recommend the former one as it allows the design team to think in converging manner. For example, as shown in Table 1, six interaction scenarios labelled as 1E11 to 1E16 are identified in column C3 of IAT for user-electric sharpener interface. If possible, listing interaction scenarios in sequential manner would be ideal.

Table 1. User-hardware interface - Interface analysis template (IAT) applied at system level

C1	C2	C3	C4	C5			C6	C7	C8	C9	C10
Interface	Interaction Code	Generic Interaction Scenarios Description	Interaction Exchange Type	Interaction Functions and Exchanges			From	To	Requirement Specifications (Attribute + Target Value + Relation + Unit)	Interaction Criticality	High Level Function
				Verb	Object/Noun						
					Input	Output					
Electric Sharpener (ES) – User (1-E1)	1E11 A	User grips the sharpener during sharpening operation	P	Support	Human Hand		User	ES	Contact Surface Area ($X\ mm^2 < A < Y mm^2$)	1	Operate the sharpener to sharpen a lead pencil
									Contact Surface Static Friction ($X<u<Y$)	1	
	1E11 B		E	Import	Human Energy		User	ES	Human Gripping Force ($XN< F <YN$)	1	
	1E12	User grips the sharpener but finds pencil and graphite debris material	M	Seal		Graphite debris	ES	User	Debris (X to Y mg)	-1	
	1E13	Sharpener exerts mechanical shock on user's grip during operation	E	Transmit		Mech. Energy	ES	User	Mechanical Vibrations (Vibrations <YJ)	-1	
	1E14	User inserts pencil into sharpener during operation	M	Import	Lead Pencil		User	ES	Lead Pencil Flow (Pencil weight < X mg)	2	
	1E15	Sharpener virtually alerts the user when sharpening operation is finished	I	Display		Visual Signal	ES	User	Luminous Flux (X lumens ≤ lumens)	1	
									Visible Spectrum (X nm < wavelength <Y nm)	2	
	1E16	Noise exchange between user and sharpener during operation	E	Transmit		Acoustic Energy	ES	User	Noise (X db < z < Ydb)	-1	
	...	...	...	...	...	...	...	...	...	..	

3.3.3 Step 3: Identify interaction exchanges by analysing use case scenarios

In this step, each interaction scenario is examined to find the flows and form related exchange types. For example, the interaction operational scenario 1E11 - 'user grips the sharpener during sharpening operation' encompasses two types of interaction exchanges, associated with 'human hand' contact as a physical (P) exchange and 'human energy' as energy (E) exchange, shown in columns C4 and C5 (Table 1) for the user-electric sharpener interface.

3.3.4 Step 4: Specify interaction requirements

There may be one or many interaction functions associated in each scenario as well as in each interaction type. For example, see Table 1 - column C5, in user-electric sharpener interface, a scenario with label '1E11' has two interaction functions i.e. 'support human hand' and 'import human energy' associated with different exchange types (i.e. P and E), that can be labelled further as '1E11A' and '1E11B'. It should be noted here that the interaction functions and exchanges need to be written from sharpener's perspective (which is a system of interest) but in inputs-outputs matrices form although previously scenarios in C3 were written either from sharpener or external entity perspectives and the reason for that is to give freedom to the designer to think from both sides of interface perspectives.
Also note that to be more specific, one can even specify the 'interfaces' in much detail in 'from/to' columns (i.e. C6-C7) such as 'user_hand' from rows 1E11 to 1E14 and 'user_eye' in row 1E15 instead of only specifying 'user' and similarly for electric sharpener (ES) as 'sharpener_surface'.

3.3.5 Step 5: Quantify the requirement specifications

Using the interaction scenarios as the basis for identifying the interface requirements, we then identify the key operational and physical characteristics in each interaction function for the interface. The requirement specification should address the quantitative controlled attribute or parameter with compatibility relations, values and units. There can be many requirement specifications associated with the single interaction function. For example, in Table 1, an interaction function 'support human hand' between sharpener and user has got two key shared performance attributes: (i) the contact functional surface area and (ii) the co-efficient of dry static friction.

3.3.6 Step 6: Evaluate interaction criticality

Having identified the requirement specification, the next task is to analyse its criticality in an interface. Interactions criticality can be quantified by using a five-point scale as discussed in IAT content section. It is envisioned that the interaction scenarios' thinking should cover both detrimental and desirable interactions when analysing an interface. Both positive (1E11, 1E14, and 1E15) and negative (1E12, 1E13, and 1E16) interactions are shown in Table 1 for user – electric sharpener interface. In a single scenario there can be multiple functions that in turn can have multiple specifications. Assuming that one specification can be more critical than the other, therefore assigning a criticality scale to each requirement specification is beneficial. For example, interaction scenario 1E15 has got two specifications with different criticalities scale: luminous flux (+1) and visibility spectrum (+2). In this case, it is important that a user should detect the luminous flux of a light source embedded in a sharpener that would indicate the sharpening operation completion and if it is not visible or in visible range (i.e. visible spectrum) then user would never know when a job is completed.
It should be noted that in electric sharpener SBD (Figure 4), only a single biological entity is considered i.e. user. There can be other entities such as supplier, maintainer etc. and thus their possible interactions from design perspectives can also be analysed in similar manner in the IAT depending upon the specific context of the problem defined at high level.

3.4 IAT Example 2: Automotive Subsystem Design Analysis

We have shown so far how IAT tool can be applied at high level of electric sharpener to capture stakeholder's requirements when its internal solutions are not known. In many practical applications, such as automotive systems design, the focus of the interface analysis is at subsystem and component levels as design solutions are carried over from one design to the next. The applicability of the IAT framework will be illustrated by reconsidering two industrial case studies previously reported by the present authors. Figure 5 illustrates an extract of the IAT based analysis for the interface between the Battery Pack and the Charger within a full electric vehicle powertrain, initially discussed in (Campean et al., 2011).

Interface	Interaction Code	Generic Interaction Scenarios Description	Interaction Exchange Type	Interaction Functions and Exchanges			From	To	Requirement Specifications (Attribute + Target Value + Relation + Unit)	Interaction Criticality	High Level Function
				Verb	Object/Noun						
					Input	Output					
Battery Pack (A) – Charger (B)	AB1	Battery pack requires electric energy from charger	E	Import	Electric Energy		Charger	Battery	Electric Power (X<W<Y)	2	Charge Battery
									Electric Voltage (72 +/- 4 Vdc)	2	
	AB2	Signal exchange between Battery Pack and Charger	I	Transmit		Thermal Signal	Battery Pack	Charger	Temperature signal (2+/- 0.5 mV)	2	
	AB3			Transmit		State of Charge	Battery Pack	Charger	State of charge (2 +/- 0.5 mV)	2	
	---	---	---	---	---	---	---	---	---		

Figure 5. Hardware - hardware interface: Battery pack - charger interface

Figure 6 describes the interface analysis between the exhaust gases and a Diesel Particulate Filter (DPF), within an exhaust aftertreatment system, based on the case study presented in (Campean et al, 2013). For both case studies the IAT delivers a superior completeness and accuracy of the interface exchange characterisation.

Interface	Interaction Code	Generic Interaction Scenarios Description	Interaction Exchange Type	Interaction Functions and Exchanges			From	To	Requirement Specifications (Attribute + Target Value + Relation + Unit)	Interaction Criticality	High Level Function
				Verb	Object/Noun						
					Input	Output					
DPF (M) – Exhaust Gas (N)	MN1	Thermal Energy exchange between Exhaust Gas and DPF	E	Transmit		Thermal Energy	Exhaust Gas	DPF	Heat (X +/- y J/s)	-1	Filter and Store Soot
	MN2	Obstruction of EG flow due to high soot level	E	Increase		Pneumatic Energy	Exhaust Gas	DPF	Back Pressure (X +/- y kPa)	-2	
	MN3	Gathering of soot from EG	M	Store	Soot		Exhaust Gas	DPF	Soot mass (X<mg<Y)	2	
	MN4	DPF accommodates the Exhaust Gas	P	Secure	Exhaust Gas			DPF	Gamma (X< y < Y)	2	
	MN5	Gathering of soot from EG	P	Filter	Soot		Exhaust Gas	DPF	Soot particle size (X< nm < Y)	2	

Figure 6. Hardware-supplied service interface: Diesel particulate filter - exhaust gas

4 CONCLUSIONS

The overall aim of this paper was to introduce an IAT tool as a structured approach for interface analysis, in the context of using and implementing it across multidisciplinary systems and different system design levels. The proposed IAT framework is based on a tabular template and is derived from the review of current methods and tools used in the academic and industrial practice. The IAT tool supports the identification of the system design requirements at early stages of design on the basis of interfaces. It provides richer interface information in a systematic manner compared with the existing visual (graphical) and textual (tabular) tools that often work either in isolation for providing a specific view of a system or require comprehensive linking integration among several tools for representing collective information. An Electric Sharpener case study was used to illustrate the development and working of the proposed IAT tool, with further validation on two automotive examples.

Since, IAT has the ability to capture both desired and detrimental interaction scenarios and requirements; it can be described a useful approach to support redesigning or reverse engineering an existing system. In engineering design, for the development of a new system, design engineers mostly consider desired interactions and not often detrimental interactions. The reason being that detrimental interactions are discovered late once the decisions on design solutions are made. They basically appear when design solutions are synthesized and tested. Therefore, keeping the information of those interactions via IAT can be a good practice for future systems design.

A potential weakness of the IAT is that it can generate long documents for relatively complex systems, which requires a relational database to organise this information. Although IAT has been tested with real world case examples, it would be further tested with a group of independent engineers working in the development of large complex systems. This test would highlight the strengths and weaknesses of the suggested IAT methodology and template.

REFERENCES

Blackenfelt, M., 2000. Design of robust interfaces in modular products, in: ASME Design Engineering Technical Conferences. Baltimore, Maryland, pp. 1–9.

Blackenfelt, M., 2001. Managing complexity by product modularisation. Royal Institute of Technology, Stockholm, Sweden.
Browning, T.R., 2001. Applying the design structure matrix to system decomposition and integration problems: a review and new directions. IEEE Trans. Eng. Manag. 48, 292–306. doi:10.1109/17.946528
Bruun, H.P.L., Mortensen, N.H., Harlou, U., 2013. Interface diagram: Design tool for supporting the development of modularity in complex product systems. Concurr. Eng. 22, 62–76. doi:10.1177/1063293X13516329
Buede, D.M., 2009. The Engineering Design of Systems: Models and Methods, 2nd ed. John Wiley & Sons, Inc., New Jersey.
Campean, F., Henshall, E., Brunson, D., Day, A., Mclellan, R., Hartley, J., 2011. A structured approach for function analysis of complex automotive systems. SAE Tech. Pap. 1–14.
Campean, I.F., Henshall, E., Butter, B., 2013. Systems engineering excellence through design: An integrated approach based on failure model avoidance. SAE Int. J. Mater. Manf. 6, 389–401.
Cansler, E.Z., Ferguson, S.M., Mattson, C.A., 2014. Identifying and mapping excess relationships in complex engineered systems, in: Proceedings of ASME, IDETC/CIE. New York, pp. 1–12.
Fosse, E., Delp, C.L., 2013. Systems engineering interfaces: A model based approach. 2013 IEEE Aerosp. Conf. 1–8. doi:10.1109/AERO.2013.6497322
Gedell, S., Michaelis, M.T., Johannesson, H., 2011. Integrated model for co-development of products and productionsSystems - A systems theory approach. Concurr. Eng. 19, 139–156. doi:10.1177/1063293X11406751
Grady, J.O., 2014. System Requirements Analysis, 2nd ed. Elsevier Inc.
Hamraz, B., Hisarciklilar, O., Rahmani, K., Wynn, D.C., Thomson, V., Clarkson, P.J., 2013. Change prediction using interface data. Concurr. Eng. 21, 141–154. doi:10.1177/1063293X13482473
Helmer, R., Yassine, A., Meier, C., 2010. Systematic module and interface definition using component. J. Eng. Des. 21, 647–675.
Hitchins, D., 2007. Systems Engineering: A 21st Century Systems Methodology. John Wiley & Sons Ltd., Chichester.
Jarratt, T.A.W., 2004. A Model-Based Approach To Support the Management of Engineering Change. PhD Thesis, University of Cambridge, UK.
Kossiakoff, A., Sweet, W.N., Seymour, S.J., Biemer, S.M., 2011. Systems Engineering Principles and Practice. John Wiley & Sons, Inc., New Jersey.
Lalli, V.R., Kastner, R.E., Hartt, H.N., 1997. Training Manual for Elements of Interface Definition and Control Training Manual for Elements of Interface Definition and Control.
Liang, V.-C., Paredis, C.J.J., 2004. A port ontology for conceptual design of systems. J. Comput. Inf. Sci. Eng. 4, 206. doi:10.1115/1.1778191
Mikkola, J.H., 2001. Modularity and interface management of product architectures, in: International Conference on Management of Engineering and Technology, PICMET'01. Portland, pp. 599–609.
Miller, T.D., Elgard, P., 1998. Defining modules , modularity and modularization evolution of the concept in a historical perspective, in: 13th IPS Research Seminar. Aalborg University, Fuglsoe.
NASA, 2007. Systems Engineering Handbook.
Otto, K., Wood, K., 2001. Product design: techniques in reverse engineering and new product development. Prentice-Hall, New Jersey.
Pahl, P., Beitz, W., Feldusen, J., Grote, K.H., 2007. Engineering Design: A Systematic Approach, 3rd Editio. ed. Springer.
Pimmler, T.U., Eppinger, S.D., 1994. Integration analysis of product decompositions, in: ASME Design Theory and Methodology Conference.
Rahmani, K., Thomson, V., 2012. Ontology based interface design and control methodology for collaborative product development. Comput. Des. 44, 432–444. doi:10.1016/j.cad.2011.12.002
Scalice, K.R., Andrade, L.F.S., Forcellini, F.A., 2008. A design methodology for module interfaces, in: Collaborative Product and Service Life Cycle Management for a Sustainable World. pp. 297–304.
Schlatt, H., 2004. The Aircraft / Store Common Interface. SAE Tech. Pap.
Sellgren, U., 1998. Attachment of mating faces - an interrelational feature approach, in: 8th Int. ANSYS Conference. Pittsburgh, PA, pp. 95–110.
Sellgren, U., Anderson, S., 2005. The concept of functional surfaces as carriers of interactive properties, in: Int. Conf. on Engieering Design (ICED). Melbourne, pp. 1–15.
Sosa, M.E., Eppinger, S.D., Rowles, C.M., 2003. Identifying modular and integrative systems and their impact on design team interactions. J. Mech. Des. 125, 240–252.
Stone, R.B., Wood, K.L., 2000. Development of a functional basis for design. J. Mech. Des. 122, 359–370.
Ullman, D.G., 2010. The mehanical design process, Fourth. ed. McGraW Hill.
Ulrich, K., 1995. The role of product architecture in the manufacturing firm. Res. Policy 24, 419–440. doi:10.1016/0048-7333(94)00775-3

Wasson, C.S., 2006. System Analysis, Design, and Development: Concepts , Principles , and Practices. John Wiley & Sons, Inc., New Jersey.
Wie, M.J. Van, Greer, J.L., Campbell, M.I., Stone, R.B., Wood, K.L., 2001. Interfaces and product architecture, in: ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference. Sept 9-12, Pittsburgh, Pennsylvania, pp. 1–12.