INTERNATIONAL DESIGN CONFERENCE - DESIGN 2016
Dubrovnik - Croatia, May 16 - 19, 2016.

APPLICATION OF THE INTERFACE ANALYSIS TEMPLATE FOR DERIVING SYSTEM REQUIREMENTS

A. Uddin, F. Campean and M. K. Khan

Keywords: interface, interaction, use case modelling, requirements

1. Introduction to requirements engineering and interactions modelling

The requirement specification process is based on transforming the stakeholder needs of a complex system into system requirements, which are commonly structured into various categories such as functional and non-functional (performance) requirements [Burge 2007], [Buede 2009], [Salado et al. 2013] or into several checklists [Pugh 1990], [Pahl et al. 2007]. In the past decade, multidisciplinary engineers such as mechanical, electrical, and software engineers have adopted different modelling techniques to capture and define system requirements [Eisenbart 2014]. On one hand, mechanical engineers have focused on articulation of system requirements with emphasis of identifying the flows as interactions between system of interest and surrounding actors in its environment at black-box level [Otto and Wood 2001], [Burge 2007], [Buede 2009], [Kossiakoff et al. 2011]. On the other hand, software engineers have used use cases and sequence of operational events as interactions using different scenarios (failure or success) between system of interest and actors in its environment to derive the set of system requirements [Daniels and Bahill 2004]. Both these viewpoints i.e. identifying system requirements via either interaction flows or interaction events are essential and require two different mind-sets from a system engineer (or analyst). Interaction events represent stakeholder's view or language articulated by a system engineer in a way that users or other stakeholders do some actions with the system and then they expect that a use case (goal) would be realised by the system. Interaction flows represent designer's view or language articulated by a system engineer in a way that which behavioural effects as flows a system will have in order to execute its role in the interaction events so that a system would realise the desired use case.
It is observed both in literature [Otto and Wood 2001], [Buede 2009] and also from authors' experience with automotive industry, ignoring any one of these two views (i.e. interaction events and flows) may lead to derivation of incomplete system requirements. This paper aims to integrate both views in a systematic manner.

2. Review of system requirements derivation methodologies

2.1 Modelling interaction events

Use case modelling captures "who (actor) does what (interaction) with the system, for what purpose (goal), without dealing with system internals" [Malan and Bredemeyer 2001]. Several approaches have been suggested in systems engineering on how to derive the systems functional and non-functional requirements from use cases [Daniels and Bahill 2004], [Ericksson et al. 2008], [Nilsen and Muller 2014]. The functional requirements (FRs) describe what the system must/should do [Otto and Wood

2001] while non-functional requirements (NFRs) describe constraints on the system such as cost, weight or other performance requirements associated with the FRs [Otto and Wood 2001], [Burge 2007].
Daniels and Bahill [2004] presented a hybrid requirements process based on use case modelling and traditional shall-statement requirements for complex, hierarchical systems. They advocated that system-level requirements both functional and non-functional can be derived for a specific use case by using (sequence of) interaction events within various scenarios (i.e. main success, or alternate flow) as summarised in Figure 1. A scenario can represent a path of success or failure in which sequence of interactions are described. However, Daniels and Bahill [2004] discuss little on the linkage between the two types of requirements, FRs and NFRs.
Nilsen and Muller [2014] developed an approach for FRs derivation through use cases with NFRs combined/linked and represented in a single use case diagram. However, this approach does not show the role of use case scenarios.
Eriksson et al. [2008] used tabular template and language notation of RUP-SE [Rational Software 2003], illustrated in Figure 2. This tabular template aims to derive subsystems' requirements by using black-box and white-box use case scenarios, but with no distinction between FRs and NFRs. The tabular template focuses on the actor-system interactions thereby characterising them into actor actions (i.e. input from actor to system) and system responses (output from system to actor). The RUP tabular template forces the analysts to always think about system-actor interface in a structured manner, which is a key success factor for maintaining focus in the modelling of complex systems [Eriksson et al. 2008]. The term interface is generally used to denote the shared boundary between two systems facing each other [Miller and Elgard 1998], [NASA 2007]. The RUP table also provides budgeted requirement field to capture targets for the system performance related aspects, which is not discussed in similar way in the other use case modelling approaches. However, it has been concluded [Bijan et al. 2013] that use case modelling related approaches can lead to ambiguity and lack of completeness as not all types of requirements are captured.

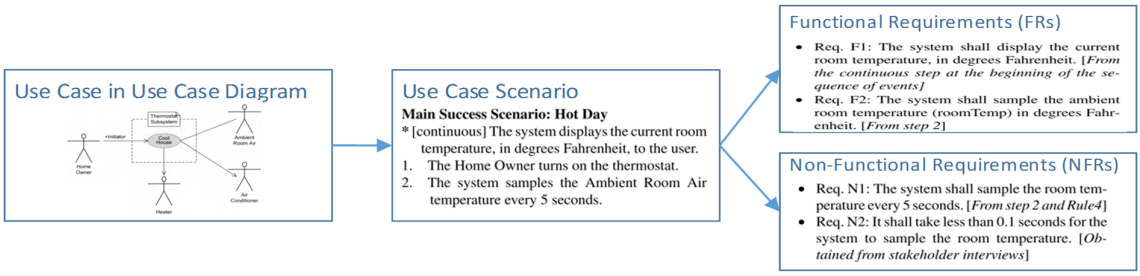

Figure 1. Daniels and Bahill [2004] system requirements methodology

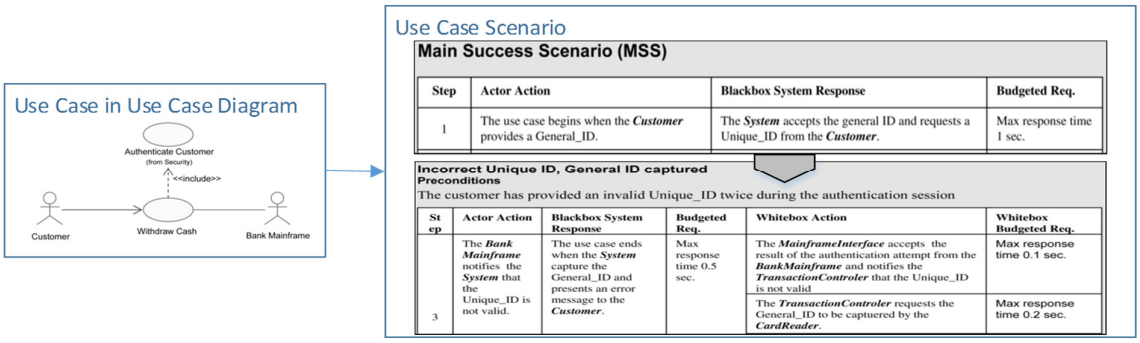

Figure 2. Eriksson et al. [2008] system requirements methodology

2.2 Modelling interaction flows

In the engineering design field, the interactions between two actors are commonly characterised in terms of flows and form based exchanges such as energy (E), material (M), information (I), and physical (P) or spatial (S), respectively [Pimmler and Eppinger 1994], [Otto and Wood 2001], [Pahl et al. 2007],

[Kossiackoff et al. 2011]. System interfaces with several interactions characterisation have been described at both high level (functional) black-box model using context diagram or/and at white-box (component) level using design structure matrix (DSM) shown in Figure 3.

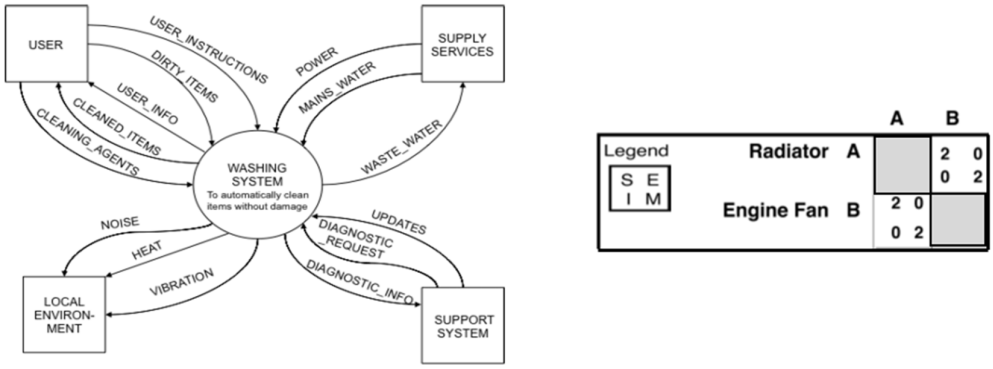

Figure 3. Black-box analysis via context diagram (left side) (adapted from Burge [2011]) and white-box analysis via DSM (right side) (adapted from Pimmler and Epppinger [1994])

The characterisation of interactions in terms of the flows is very powerful in terms of deriving functional requirements for the system in order to manage interface exchanges, as discussed by Campean et al. [2011] and further illustrated by Henshall et al. [2015] with a complex automotive case study. However, this approach has been introduced at a subsystem level, and hence not fully discussed within a complex systems requirements framework.

2.3 Requirements grouping and engineering attributes

In automotive engineering practice, requirements are often organised according to various engineering / vehicle level attributes framework (such as NVH, vehicle dynamics, cost, weight, etc., as shown in Figure 4-c). The role of such engineering attributes from requirements' grouping perspective is often not explicitly discussed in the requirements analysis methodologies. The vehicle requirements in automotive industry are directly organised and grouped along with these engineering attributes. Each attribute is further broken down and reveals several performance aspects of the system. The proposed approach in this paper also aids in grouping both FRs and NFRs according to such engineering attribute types.
Based on aforementioned approaches from literature review, it is quite evident that both viewpoints of modelling the interactions as either 'flows as well as form related exchanges' or 'events' (i.e. actions/responses) between two interacting actors or between a system and its surrounding actors are essential; however none of the established literature shows a distinct relationship between these two viewpoints at interfaces.

Table 1. Characteristics and integration of interactions modelling methods

Methods Characteristics & Criterion	Interactions Based on		Hybrid
	Use Cases	Flows	IAT
Interactions modelled as operational events	√		√
Sequence based interactions	√		√
Main or/and exceptional (alternate) flow based scenarios	√		√
Interactions modelled as exchanges / flows		√	√
Non-sequence based interactions		√	√
Desireable & undesireable input/ouput flows based interactions		√	√
Requirements grouping against engineering attributes			√

The aim of the research presented in this paper is to develop a structured approach for systems requirements analysis that integrates characteristics of use case modelling with a coherent flows based approach for describing interface exchanges, summarised in Table 1. The approach is based on the

Interface Analysis Template (IAT) discussed in [Uddin et al. 2015], which is extended with the integration of the engineering attributes framework and expanded to incorporate use case modelling and reasoning in the interface analysis; summarised briefly in Section 3. The paper outlines the proposed approach in Section 4 and illustrates its application for the design analysis of an automotive feature in Section 5, followed by discussion and conclusions in Sections 6 and 7 respectively.

3. Research contribution

It is observed that the use case modelling based approaches share common steps for deriving system requirements: (1) specifying use cases; (2) analysing them via main success or alternate flow scenarios and interactions in those scenarios; and (3) then deriving system requirements from interaction descriptions (shown in Figure 4-a). In this paper, the authors aim to show the integration of the IAT with the generic use case modelling technique and to extend its scope to integrate with the engineering attributes (as shown in Figure 4-c). The proposed methodology is based on six-steps, summarised in Figure 4-b, for deriving system requirements: (S1) use cases are specified; (S2) interfaces are identified; (S3) interactions (events) are defined; (S4) the single or multiple (flows) exchanges are identified that can exist at each interaction of actor and a system or between two internal actors of a system; (S5) assess and associate the impact of the exchanges with relevant engineering attributes; and (S6) then derive system requirements (both FRs and NFRs) based on steps S(4)-(5). Characterisation of an interaction in terms of the multiple exchanges within actions and responses events at the interface, in the context of a specific use case as well as across multiple use cases facilitates a more systematic and robust approach to requirements elicitation.

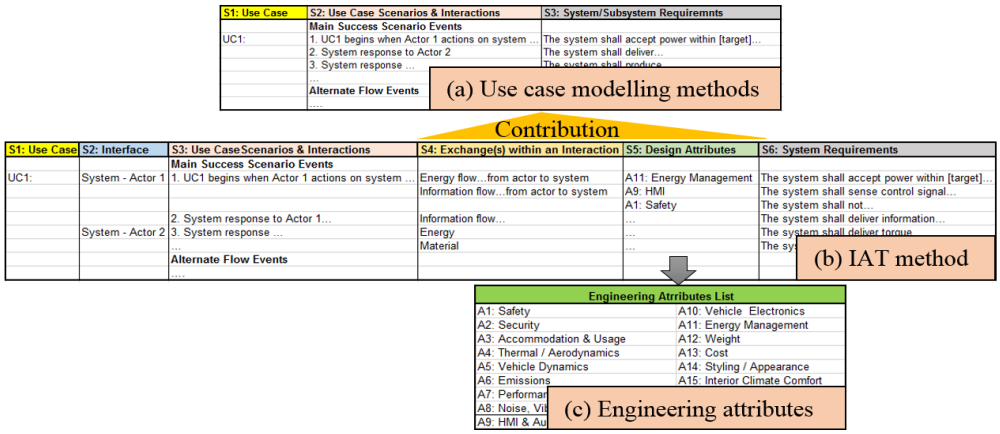

Figure 4. IAT application and alignment with interactions modelling viewpoints

4. Extended IAT framework

Based on the critical analysis of requirements derivation approaches via use case modelling, the IAT tabular template model has been introduced for documenting system requirements [Uddin et al. 2015] based on a six steps approach, illustrated in Figure 5.

4.1 IAT modelling viewpoints and descriptions

With regard to different viewpoints of use case modelling, the intended behaviour of a system or feature of the system can be articulated through consideration of its use cases, interfaces with surrounding actors, interactions, exchanges, attributes. Table 2 defines the key viewpoints of the IAT framework.

S2: Interface	S3: Specification of Interaction		S4:Specification of Exchange					S5: Exchange Effect / Impact		S6: Specification of System/Feature Requirement				S1: System Context
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11			C12	C13
Interface	Interaction Step	Interaction Description	Exchange Type	Exchange Description	Exchange Properties	From	To	Attribute Type or Purpose	Criticality	Functional Requirement			Nonfunctional Requirement (Performance Related)	Use Case Description
										Verb	Object			
											Input	Output		

Figure 5. Interface Analysis Template (IAT) framework

Table 2. Modelling viewpoints in the IAT framework

IAT Framework Viewpoints		Column	Description of contents of IAT Framework
S2: Interface	External interfaces	C1	Denotes the shared boundary (interfaces) between the system and external interacting actors, at black-box level.
	Internal interfaces		Interfaces between the components of a system at white-box.
	External to internal		Interfaces between the system external and internal actors.
S3: Interaction	Interaction Step	C2	Description of number of interaction steps in an interface.
	Interaction Description	C3	All possible system-actors interaction events, and conditions occurring under necessary/worst case conditions either in sequential or in non-sequential manner.
S4: Interaction Exchanges	Exchange Type	C4	Characterises the type of an exchange via E/M/I/P.
	Exchange Description	C5	Specification of energy, material, information and physical related exchanges.
	Exchange Property	C6	Specifies the multiple properties of an exchange.
	From	C7	Specify the directionality of an exchange from an actor to system (and vice versa) or from one actor to another actor.
	To	C8	
S5: Interaction Exchange Effect	Attribute Type	C9	Specifies the high level attributes affected by an exchange.
	Criticality	C10	Evaluates the criticality of the effect of an exchange on an attribute using five point scale (-2,-1,0,+1,+2); negative denotes detrimental effect, positive denotes desired effect.
S6: Interaction Requirement	Interaction Functional Requirement (FR)	C11	Specification of both input/output and non-input/output functional requirements in 'verb-noun' format or 'subject shall + verb + object' format in relation to exchanges.
	Interaction Non-Functional Requirement (NFR)	C12	Specifies the performance aspects with constraint or bounding relations (<, >, = or min / max), target value, and unit associated with functional requirement.
S1: Interaction Lifecycle Context (Use Case)		C13	Specifies system lifecycle use cases where the relevant exchange is relevant.

5. IAT application: analysis of a Surround View feature

To illustrate the deployment of the proposed IAT framework, a case study is considered based on a Surround View (SV) driver assist feature [Sainz and Stich 2015]. Such features have been introduced for safety (in particular in relation to avoiding collisions with pedestrians and other road users in conditions where vision is restricted) and to support the driver with parking manoeuvres. From a systems engineering point of view, this feature is introduced into the vehicle context, which in turn contains many actors. We will start by considering a Black Box level analysis for the feature. While several physical architectures can be considered for the SV feature; (e.g. e.g. solution 1: LCD screen on dashboard + Camera +other actors OR solution 2: Hologram + Camera + other actors), at a black-box level, the general expectations of the external actors from the feature would be similar and therefore the intended behavioural requirements need to be captured first.

5.1 Step 1 - (S1): identify use cases of the system

A use case diagram, illustrated in Figure 6 (on left side), is used to describe an abstract information of a SV feature by describing its use cases occuring due to its interactions with external actors such as the driver, the vehicle (and enabling systems), and the environment. As the system lifecycle has many phases, there is a need to consider requirements for the SV feature in all lifecycle related use cases, such as operational, design, manufacturing, and transportation. The use cases of the SV feature in each lifecycle can either be represented on a single use case diagram or in separate diagrams. For example, Figure 6 shows the use case diagram of a SV feature from only operational perspective. A base use case

"operate SV feature" in turn can have many sub-goal use cases, e.g. 'show front view' and 'show rear view' that are represented via inclusion relationships. If the SV feature functionality is extended from manual to automatic then other sub-goal use cases such as 'show rear view automatic' can be linked to base use case via extension relationship as shown in Figure 6 (on left side). The identified use cases are listed down in C13 of IAT, shown in Figure 7.

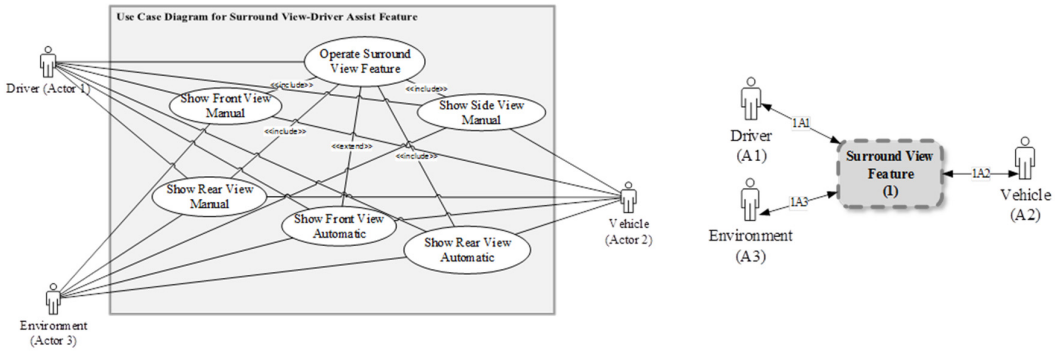

Figure 6. Use case and context diagrams for Surround View feature

5.2 Step 2 - (S2): identify the system interfaces

Figure 6 (on right side) shows an adapted version of a system context diagram for a SV feature. The system context diagram reflects the system context from a physical world perspective, derived from use case diagram, in the sense that it identifies external actors as sources for inputs into the system and destinations of outputs from the system. The system context diagram describes the black-box functional view of the system and contain less information than the elaborated context diagrams suggested by Burge [2011] and Kossiakoff et al. [2011] (e.g. see Figure 3 on left side). There can be multiple interactions between external actors and the system which are usually represented via a combination of arrows and specific labels with abstract meaning and information. Here, the abstract information is omitted. The reason behind this is that, as discussed earlier, the interaction is perceived in different ways, such that it can represent many viewpoints, e.g. a set of flows (in/out to a system) [Burge 2011], [Kossiakoff et al. 2011] and/or activities (i.e. sequence of interactions events) between an external actor and the system [Daniels and Bahill 2004], [Ericksson et al. 2008]. Labelling each such interaction in the system context diagram between system-actors would lead to an unnecessarily complex diagram, and time consuming process. Therefore, in order to improve the readability of the diagram, it is 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 6 (on right side), an interface '1A1' between the SV feature and the driver is represented through a bidirectional relationship. A single arrowhead would indicate whether an actor affects or is affected by the system.

5.3 Step 3 - (S3): define the scenarios & interactions in it

Having established the use cases and context diagram, the next step is to define the possible scenarios (i.e. 'desired or main success' and 'undesired or exceptional') between external actors and the system, shown in Figure 7 in IAT. The interactions analysis in a scenario can be done in following two ways: (i) analyse external actors one at a time (with or) without sequence of events; or (ii) all actors at once (with or) without sequence of events in terms of actions and responses. Herein, the former approach is adopted, i.e. considering one actor at a time and without considerations of the sequences of events. This allows the engineering design team to think both in convergent and divergent manner; convergent due to the fact that only one actor with system is analysed first, and divergent because all possible actions (both desired and undesired interactions) of an actor on the system, and the systems responses on the actor, are exhaustively searched. For example, in Figure 7 there are three interactions labelled as '1A11' to '1A13' in C2, describing the drivers' possible interactions (i.e. actions) specified in column C3 of IAT. Similarly, the SV feature responses (or actions) on the driver are listed in '1A14 - 1A16'.

This procedure can be repeated for other interfaces. If possible, listing interaction in sequential manner would be ideal and this is often recommended by systems engineering community; however, this is only possible for a single use case, whereas the IAT aims to analyse and map the captured interactions across the multiple use cases concurrently once FRs and NFRs are derived. It is also envisioned that the interaction scenarios reasoning should cover both desirable and detrimental interactions when analysing an interface. This is illustrated in Figure 7, where both positive i.e. [1A11 - 1A15], & [1A21] and negative [1A16], & [1A22] interactions are captured for 'driver-SV feature' and 'vehicle-SV feature' interfaces, respectively, in corresponding scenarios (i.e. main success and exceptions).

5.4 Step 4 - (S4): identify interaction exchanges

In next step, each interaction (or event) is examined by the analyst to find the flows (i.e. E/M/I) and form (P/S) related exchanges. For example, the interaction '1A11 - driver activates the front SV feature' encompasses two types of interaction exchanges, 'driver input' as an energy exchange in order to activate the feature and 'activation signal' as an information exchange for switching it ON. These exchanges are characterised and specified in columns C4 and C5 respectively. The IAT-C6 is used to specify the properties of exchanges e.g. ‘driver input' can possess properties like ‘driver effort (joules)’ where as for ‘activation signal’ it can be ‘frequency (hertz)'. To specify the directionality of an exchange from/to system with its external actors, columns C7-C8 in IAT are used. An exchange (or flowing object) can have more than one property as discussed in theory given by Hubka and Eder [1988] and is evident in Figure 7 with '1A14' as 'image' exchange has many properties. Also, to be more specific from a redesign perspective, one can even specify the interfaces in much greater detail in 'from/to' columns (i.e. C7-C8) such as 'driver_hand' from rows 1A11 to 1A12 and 'driver_eye' in row 1A14 instead of only specifying 'driver'; similarly for SV feature as 'feature_activation surface'. This sort of specification becomes really important particularly if an existing system is reverse engineered. It should also be noted that column C3 may express an interaction from exchange perspective e.g. interaction 1A16 is itself revealing an exchange description of 'vibration' so same information of exchange would be specified and repeated in C5. Alternately, C5 can be left empty which will indicate that C3 is describing an exchange and not an interaction event or activity of an actor or feature response.

5.5 Step 5 - (S5): assess interaction exchange effect

The IAT column C9 is used to list down the design attribute affected by an exchange. For example, shown in Figure 7 with interaction labelled as '1A11'; ‘driver input’ and ‘activation signal’ exchanges are desired for delivering the design attributes 'A3: Accommodation & Usage' and ‘A9: HMI & Audio Visual Performance’ related to the SV feature within the use case of 'show front view manually'. It is quite possible that a single exchange may be essential for multiple attributes which need to be specified. Interaction criticality is specified in C10 that evaluates the impact of the exchange (e.g. 'ON activation signal') in relation to attribute, i.e. ‘A9: HMI’. The interaction exchange criticality (desired or undesired) is assessed using a five point scale, presented in Table 2, adopted from Pimmler and Eppinger [1994].

5.6 Step 6 - (S6): specify system requirements

5.6.1 Specify interaction functional requirements (FRs)

There may be one or many interaction functional requirements associated with each interaction exchange at interface. For example, looking at exchange description of 'driver input' in C5, in driver-SV feature interface labelled as ‘1A11, the SV feature will require interaction FR of 'accept driver input' as an input and is specified in C11 in IAT (see Figure 7). It should be noted here that the interaction FR is articulated by the analyst from the system's perspective. This can even be articulated in traditional shall format i.e. 'SV feature shall accept driver input'. Similarly, the feature's output behavioural requirements can also be articulated in similar way looking at exchange description (see e.g. column C11 in rows '1A14' and '1A15' in Figure 7). It is possible that a designer may find an exchange hard to put in the input/out category in C11 then in that case the requirement can be articulated in generic way e.g. 'maintain operational life' as labelled '1A13'. This reflects the fact that system can also have non-input/output requirements.

Figure 7. IAT framework example for the Surround View feature

5.6.2 Specify non-functional requirements (NFRs)

Using the interactions (as actions/responses) and their exchanges as the basis for identifying the interaction functional requirements, the IAT framework then aids in specifying the non-functional (performance) requirements. There can be many NFRs associated with the single interaction FR. For example, in Figure 7, an interaction requirement labelled as 1A11 'accept driver input' between SV feature - driver interface, has got two key shared performance attributes: (i) the number of activation point (X < No. of Interface Point < Y) and (ii) the minimum and maximum amount of driver's effort (X < N <X). Once the FRs and NFRs in interaction 1A11 are identified within a single use case 'show front view manual' then same requirements are checked against other multiple use cases, listed in IAT (earlier

in Step 1). The requirements in 1A11 do not belong or contribute to other use cases hence not mapped across other use cases whereas requirements in 1A14 and 1A21 do contribute and common across multiple use cases and hence mapped.

6. Discussion and future work

It should be noted that IAT can be approached in two ways: either horizontally (left-right) or vertically (top-down) from C3 column to C12 column in Figure 7. For example, on one hand in horizontal direction, having identified an interaction e.g. '1A11' in C3, then at the same time all other columns can be filled step by step (left-to-right), before listing down all the interactions in C3. On the other hand, on the vertical deployment, all interactions in C3 are listed first (in top-down manner) and then other columns are filled one after another. The latter approach is suggested by most of the existing use case modelling approaches i.e. identifying scenarios and listing down the interactions first and then extracting requirements. But in this case, first use case interactions are brainstormed whereas listing multiple use cases and then a single interaction event and extracting exchanges and FRs and NFRs from one use case and thereafter checking those against across multiple use cases provides a more robust and iterative view. A single FR and its NFR can be common and essential across multiple use cases as evident in '1A13' and '1A21, 1A22' interactions and hence these can be mapped upfront. These requirements that are mostly common across more than one use cases can then be prioritised. Thus, in this sense, the IAT framework provides more benefits, and is more flexible from usage perspective, and is also compatible with other existing approaches due to use of common viewpoints of interactions modelling techniques. Moreover, the IAT helps in grouping the different types of requirements according to design attributes e.g. as can be seen from Figure 7 that all the FRs and NFRs related to 'A9: HMI' such as FRs: 'sense ON signal (from 1A11)', 'sense OFF signal (from 1A12)', 'display front view symmetrically (from 1A14)' can be grouped and organised. Similarly, requirements also for 'A3: Accommodation & Usage' can be grouped and organised as can be seen from Figure 7, which is often not discussed in existing use case modelling and requirements derivation approaches. 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 often miss out detrimental interactions. The reason being that often detrimental interactions are discovered late once the decisions on design solutions are made. These are typically identified when design solutions are synthesized and tested. Therefore, keeping the information of those interactions via IAT framework is good practice for future systems design. A potential strength of the IAT is that it provides structured guidelines for capturing system requirements by analysing and modelling the necessary viewpoints at system interfaces. The IAT can be updated based on system's actual behaviors during its integration and validation tests which requires a relational database support to organise the learned information and to aid data change management process. The IAT framework is being validated with a number of industrial case studies in an automotive industry and is being tested with a group of independent engineers working in the development of large complex systems such as automobile, at mainly black-box level. The validation process so far has highlighted the strengths and also weaknesses of IAT framework. The key strength is that mechanical and software engineers used the IAT framework in slightly different manner but managed to obtain a robust set of requirements thereby identifying requirements that were not discovered with previous requirements documentation processes. The weakness being that it happened at the cost of timely and complex process of IAT framework due to many columns. In future, it will be assessed how this framework can be simplified and transformed into a software or package such as by utilising the commercial requirements management tool Telelogic DOORS to manage our IAT model.

7. Conclusions

The overall aim of this paper was to introduce a novel approach, an IAT framework for analysing and defining system's interfaces as well as for deriving its requirements robustly, in the context of using and implementing it across multidisciplinary teams and different system design levels. The IAT framework is tabular template and is derived from the review of current methods and tools used in the academic and industrial practice based on interface and use case modelling techniques. The IAT model supports

the identification of the system design requirements at early stages of design on the basis of various interaction modelling contents of interfaces. It has the ability and aids in identifying the conflicting requirements at customer level and groups the functional and non-functional requirements according to attribute types. 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 comprising of different viewpoints for representing collective information. A Surround View (SV) driver assist feature, as a case study was used to illustrate the development and working of the proposed IAT tool along with framework's strengths and weaknesses.

References

Bijan, Y., Yu, J., Stracener, J., Woods, T., "Systems Requirements Engineering - State of the Methodology", Systems Engineering, Vol.16, 2013.

Buede, D. M., "The Engineering Design of Systems: Models and Methods", 2nd ed., John Wiley & Sons, Inc., New Jersey, 2009.

Burge, S., "A Functional Approach to Quality Function Deployment", Burge Hughes Walsh, 2007.

Burge, S., "The Systems Engineering Tool Box", Available at: <http://www.burgehugheswalsh.co.uk/>, 2011.

Campean, F., Henshall, E., Brunson, D., Day, A., Mclellan, R., Hartley, J., "A structured approach for function analysis of complex automotive systems", SAE Tech. Pap., 2011.

Daniels, J., Bahill, T., "Combines Traditional Requirements and Use Cases", Systems Engineering, Vol. 7, 2004, pp. 303-319.

Eisenbart, B., "Supporting interdisciplinary system development through integrated function modelling", PhD dissertation, University of Luxembourg, 2014.

Eriksson, M., Kjell B., Jürgen, B., "Use Cases for Systems Engineering - An Approach and Empirical Evaluation", Systems Engineering, 2008, pp. 39-60.

Henshall, E., Rutter, B., Souch, D., "Extending the Role of Interface Analysis within a Systems Engineering Approach to the Design of Robust and Reliable Automotive Product", SAE Int. J. Mater. Manf., Vol.8, No.2., 2015, pp. 322-335.

Hitchins, D., "Systems Engineering: A 21st Century Systems Methodology", John Wiley & Sons Ltd., 2007.

Kossiakoff, A., Sweet, W. N., Seymour, S. J., Biemer, S. M., "Systems Engineering Principles and Practice", John Wiley & Sons, Inc., New Jersey, 2011.

Malan, R., Bredemeyer, D., "Functional Requirements and Use Cases", Bredemeyer Consulting, Available at: <http://www.bredemeyer.com>, 2001.

Miller, T. D., Elgard, P., "Defining modules, modularity and modularization evolution of the concept in a historical perspective", In: 13th IPS Research Seminar, Aalborg University, Fuglsoe, 1998.

NASA, "Systems Engineering Handbook", 2007.

Nilsen, A. F., Muller, G., "Use Cases and Non-Functional Requirements Presented in Compact System Description A3s", In: INCOSE, Las Vegas, 2014.

Otto, K., Wood, K., "Product design: techniques in reverse engineering and new product development", Prentice-Hall, New Jersey, 2001.

Pahl, G., Beitz, W., Feldusen, J., Grote, K. H., "Engineering Design: A Systematic Approach", 3rd ed., Springer, 2007.

Pimmler, T. U., Eppinger, S. D., "Integration Analysis of Product Decompositions", In: ASME Design Theory and Methodology Conference, 1994.

Pugh, S., "Total Design: Integrated methods for successful product engineering", 1990.

Rational Software, "Rational Unified Process for Systems Engineering", IBM Rational Software, 2003.

Salado, A., Roshanak, N., "A Categorization Model of Requirements Based on Max-Neef's Model of Human Needs", Systems Engineering, 2013, pp. 348-360.

Uddin, A., Campean, F., Khan, M. K., "Development of an Interface Analysis Template for System Design Analysis", Proceedings of the Internnational Conference on Engineering Design, ICED15, Milan, Italy, 2015.

Amad Uddin, Research Fellow
University of Bradford, University of Bradford
130 Heath Terrace, BD3 9PH Bradford, United Kingdom
Email: a.uddin3@bradford.ac.uk