INTERNATIONAL DESIGN CONFERENCE - DESIGN 2014
Dubrovnik - Croatia, May 19 - 22, 2014.

DESIGN 2014

APPLICATION OF THE IFM FRAMEWORK FOR MODELLING AND ANALYSING SYSTEM FUNCTIONALITY

B. Eisenbart, K. Gericke and L. Blessing

Keywords: Function modelling, function analysis, interdisciplinary design

1. Introduction

Conceptual design is considered to be among the most demanding design tasks, as it includes the central transition from a problem to an early solution concept, which necessitates a joint effort of all involved disciplines [Chakravarthy et al. 2001, Erden et al. 2008]. In order coordinate related design activities, a shared understanding of the requirements for the system under development and its expected functionality needs to be established among the involved designers [Frankenberger et al. 1998]. In this article, the term "technical system" encompasses technical products as well as Product-Service Systems (PSS).

The process of determining and analysing alternative solution concepts based on function considerations and with respect to function and requirements fulfilment is typically referred to as function reasoning [Umeda and Tomiyama 1997]. Function reasoning is essentially characterised by iterative synthesis and analysis steps, typically including decomposition of the design problem, in order to support synthesis of potential solution elements [Far and Elamy 2005]. Across disciplines, function modelling is proposed as a means to facilitate this reasoning process [Eisenbart et al. 2011]. Because of its use in different disciplines, it is expected to support the establishment of the required shared understanding [Stone and Wood 2000, Erden et al. 2008]. However, shared modelling and analysis of system functionality is hampered due to diverse notions of function [Crilly 2010, Vermaas 2013] and divergent, often incompatible morphologies which can be found across, but also within different disciplines [Eisenbart et al. 2012]. Function models address different sets of *function modelling perspectives*. Function modelling perspectives refer to the specific information addressed in individual function models; i.e. they relate to the content explicitly modelled to represent functions and overall system functionality (see [Erden et al. 2008, Eisenbart et al. 2012, 2013a] for a comprehensive review and comparison of function models across disciplines).

Eisenbart et al. [2013b] proposed a concept for a modelling framework that addresses these issues, in order to support cross-disciplinary modelling and analysis of system functionality: the Integrated Function Modelling (IFM) framework. The IFM framework is matrix-based, in order to provide designers with a clearly structured, integrated means for representing system functionality. In this article, the application of the framework is illustrated based on an exemplary system (see Section 3). Furthermore, specific options for function analysis are explored that result from the matrix-based setup (Section 4). Finally, the implications for integrated function modelling in practice and areas of future research are discussed.

2. The IFM framework

In the IFM framework integration is facilitated through linking function modelling perspectives prominent across disciplines. The framework consists of associated views (see Figure 1). Individual views address the different function modelling perspectives and/or their interdependencies. They comprise of matrices related to the concept of design structure matrices (DSM, after [Steward 1981]).

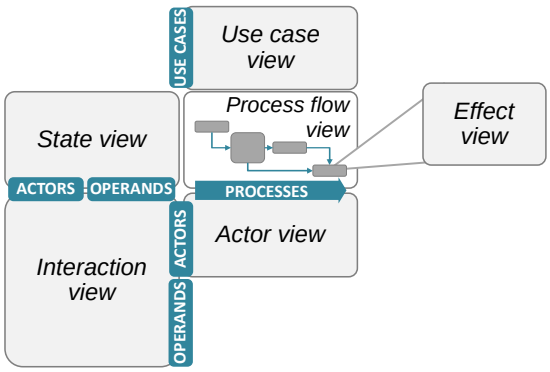

Figure 1: Integrated Function Modelling framework

2.1 Entities and their relations

Across disciplines, a central notion of function is associated to the intended, expected or perceivable behaviour of a system [Vermaas 2013, Eckert 2013]. With regard to the different function modelling perspectives, function in relation to system behaviour may be regarded through consideration of different use cases, transformation and interaction processes and/or associated state changes of involved operands or operators. These, in turn, provide required physiochemical effects for function fulfilment. Table 1 represents these entities, which are encompassed in the IFM framework. The entity relations are illustrated in Figure 2 using an UML-based domain model.

Table 1: Entities and relations in the IFM framework

Entities	Description
Use Case	Different cases of applying the system. This is typically associated to the interaction of actors with the system under development, which may require subsequent transformation processes to take place. The associated set of processes lead to an observable result, in order to provide some kind of value to users.
Transformation processes	Processes executed by actors, which (from the designers' perspective) are part of the system under consideration and may lead to a change of state of actors or of operands. *Technical processes* are transformation processes related to technical sub-systems; *human processes* are related to stakeholders (thus, including service activities).
Interaction processes	Representation of interaction processes of actors, which (from the designers' perspective) are *not* part of a system, with actors, which *are* part of the system under consideration.
Effects	Representation of the required physiochemical effects, which have to be provided, in order to enable or support transformation and/or interaction process(es).
States	Representation of the states of actors or of operands before (input) and after (output) a transformation process.
Operands	Operands are typically specifications of energy, material, and information.
Technical sub-system	Technical sub-systems encompass technical systems (i.e. technical products, potentially combining mechanical, electrical and software systems with associates services), which are part of the system under consideration. They can be composed of more technical sub-systems.
Stakeholder	Stakeholder comprises (groups of) animate beings affected by or affecting the system under consideration (including any related services).
Environment	Environment includes all active and passive parts of nature in general surrounding the system under development.

A technical system may support one or more use cases. Each use case may be decomposed into sub-use cases. Use cases may have dependencies among each other (mutually exclusive, mutually inclusive etc.). A use case may have one or more transformation processes associated to it, which may be composed of sub-processes and may be mutually dependent. A transformation process may result in the transformation of operand and/or actor states. Such state transformations are enabled, respectively supported by effects, which are provided by actors. Actors thus serve as operators in transformation processes. Actors comprise stakeholders, technical sub-systems, and environment. Actors also may have

dependencies among each other. Operands may also interact and have dependencies among each other. Operands may temporarily assume the role of an actor during a use case by supporting the state transition of actors or other operands in relation to the execution of specific processes (as already discussed by [Nevala 2005]).

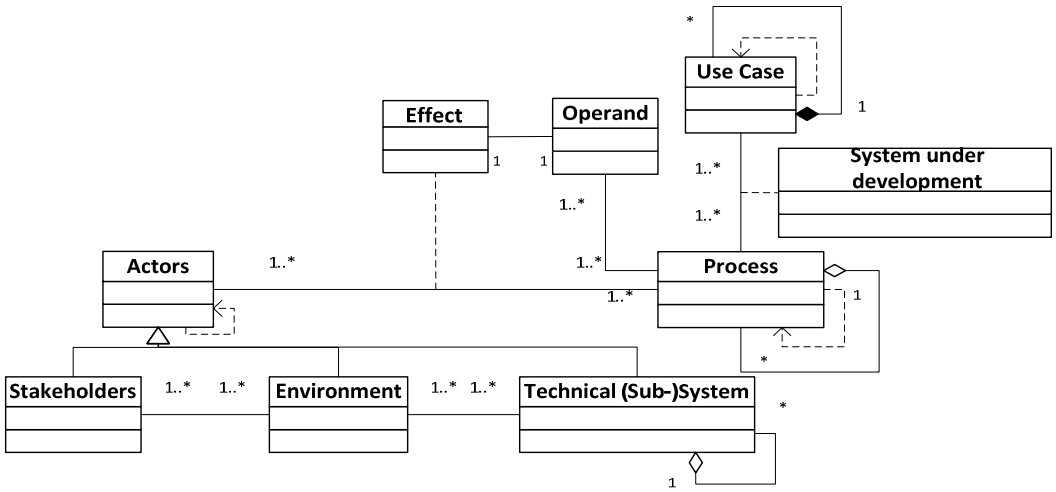

Figure 2: Domain diagram of the IFM framework

2.2 Adjacent views onto a system's functionality

The views in the IFM framework are modular, as individual views may be added or omitted, in order to allow for demand-actuated adaptation (augmenting or tailoring) of the framework related to different design contexts (see [Eisenbart et al. 2013c]). Also, the application of (alternative) function taxonomies is enabled, if the designers prefer to utilise one of them. The different views are strongly interlinked through the adjacent placement and the respectively shared header rows and header columns in the specific matrices forming the individual views (see Figure 1). Examples are presented in Section 3.

The *Process flow view* qualitatively visualizes the flow of sequential or parallel (interaction and/or transformation) processes related to a specific use case. For each use case an associated set of views is created. Individual transformation processes are represented as chronologically numbered blocks. In the vertical direction, the process flow is visualized related to time. This matches to the flow of states from initial to final of operands and actors in the associated *state view*. In horizontal direction, the process blocks are spread from left to right, to enable a direct link to the *actor view*.

The *Effect view* represents the effects, which enable individual transformation processes and are provided by actors. For each process block in the *process flow view*, a separate *effect view* may be created, preferably using a similar representation. This allows for detailed analysis of the basic physiochemical effects that are affecting or contributing to the individual transformation processes.

The *Use case view* lists the different use cases associated with the system under consideration and indicates the involvement of individual processes within them. Dependencies between processes, which hinder their parallel or sequential execution, could affect the operability of use cases in which these processes are involved. The *use case view* is intended to support analysis of such dependencies. Use cases are listed in the header column. The flow of processes builds up the header row, which interlinks *use case view* and *process flow view*.

The *Actor view* indicates the involvement of one or more actors in the realization of individual transformation processes related to a use case. Involvement may be differentiated between e.g. as either "affecting/ supporting" or "being affected" by a process. The view allows differentiating actors according to whether they – from the designers' point of view – are part of the system (e.g. service

operators as part of a PSS) or not (e.g. the targeted users or external service providers). This differentiation also separates transformation processes from interaction processes (see Table 1).

The *States view* represents the states from initial to final of operands and actors as well as their changes associated to the flow of individual processes. The *states view* consists of the *actor state matrix* and the *operand state matrix*. Individual operands and/or actors may also merely support a transformation process without changing their states.

The *interaction view* depicts the specific bilateral interactions and impacts between actors and operands as well as their complementary contributions in the realization of use case, associated processes etc., i.e. their complementary contribution to fulfilment of expected system functionality. Additionally, information about the embodiment of specific bilateral impacts may be included. Hence, this view essentially results in an initial system structure or interface matrix of the system, respectively. However, in principle any kind of bilateral dependency between actors can be represented in this view, such as e.g. business relations.

Apart from these views, it may prove beneficial to individual designers to add entirely new views, in case specific information is particularly important or in order to ease use of the framework in a specific design context (see [Eisenbart et al. 2013c]). For instance, in electrical engineering petri nets or finite state machines are of particular importance. Thus, electrical engineers may benefit from including an additional *state/state DSM*, indicating all possible transitions between actor and/or operand states. Such a DSM can be conveniently embedded into the structure of the IFM framework.

3. Application

The framework – through its modular and interlinked character – explicitly allows for application in alternative ways. That includes different starting points and alternative sequences of individual modelling activities. This is exemplified in [Eisenbart et al. 2013c] for a variety of design approaches proposed in literature across disciplines. In the following, one potential way of applying the IFM framework is described for an original design project. That includes specific modelling steps and the application of the framework onto an exemplary system.

3.1 Approach

Across disciplines, the conceptual design stage typically encompasses requirements specifications, function modelling and results in an (initial) system model [Eisenbart et al. 2011, Gericke et al. 2013]. In Table 2 central modelling activities for the application of the IFM framework are described, which can be iteratively performed.

For application of the IFM framework the requirements list is analysed in an initial step and central expected functions as well as associated sub-functions are derived. These may change over a system's life cycle, which is of particular relevance for life-cycle oriented systems, such as PSS. In the next step, the central use cases are determined and specified. Changes in the use of a system over its life-cycle may be regarded as variant use cases and specified as such. In this step, use cases can simply be listed and roughly outlined (e.g. in terms of central goals and involved actors) in a few sentences.

Modelling with the IFM framework may then start by sorting expected functions into the outlined use cases, i.e. as flow of associated transformation processes. Subsequently, respective process flows are gradually detailed and complemented by required sub- and supporting transformation processes. Considering the respective associated state changes of operands and/ or involved actors and their states may support this. On the next level of detail, individual process blocks may then themselves be decomposed into sub-processes. Processes are enabled by actors, which may again be comprised of general function carriers (see e.g. [Andreasen et al. 2014]) including any related sub-system or service operator. Function carriers may be subsequently concretised. Thus, the framework allows modelling the functions and actors of a system under development from very abstract to detailed and specific. Detailing of individual views may be achieved through decomposition of individual actors, process blocks etc.

Alternatively, associated partial views may be generated, which focus on one specific process block (i.e. “zooming in” on individual process blocks), see e.g. Figure 3.

Table 2: Modelling activities

Modelling activity	Description
Use Case definition	…includes the consolidation of the different use cases (and their sub-use cases, if applicable) the system under development is expected to support in the different phases of its life-cycle. The use cases are then listed in the respective column in the *use case view*.
Process flow modelling	…involves modelling separate flows of required processes related to each (sub-) use case. A multitude of alternative process flows may fulfil a use case. While modelling the process flows, the involvement of individual processes in multiple use cases needs to be considered. Modelling and selecting an alternative process flow may be facilitated through considering the required state changes of operands and actors in parallel (combining *process flow view* and *state view*).
Operand state modelling	…includes modelling the state changes of involved operands in the operand state matrix (as part of the *state view*) related to the chosen process flows.
Effect modelling	…involves modelling the required effects related to the specific process blocks or entire flows, respectively.
Actor allocation	…includes allocation of the actors, which are involved in the individual processes, either as affecting or being affected through the delivered effects.
Actor state modelling	…includes modelling the state changes of allocated actors in the actor state matrix (as part of the *state view*) related to the chosen process flows.
Interaction specification	…involves analysing and detailing the specific bilateral impacts among actors, among operands, and between actors and operands.

3.2 Example: coffee vending machine

The chosen example system is a customary coffee vending machine, which provides customers with a variety of warm drinks. In the following, the application of the individual views focuses the use case of “preparing a cup of coffee”. Therein, a set of associated sequential and parallel transformation processes are required for use case fulfilment, as illustrated in Figure 3. Further use cases associated to the vending machine may include

- preparing a cup of cappuccino,
- preparing hot water (e.g. for tea),
- automated cleaning, etc.

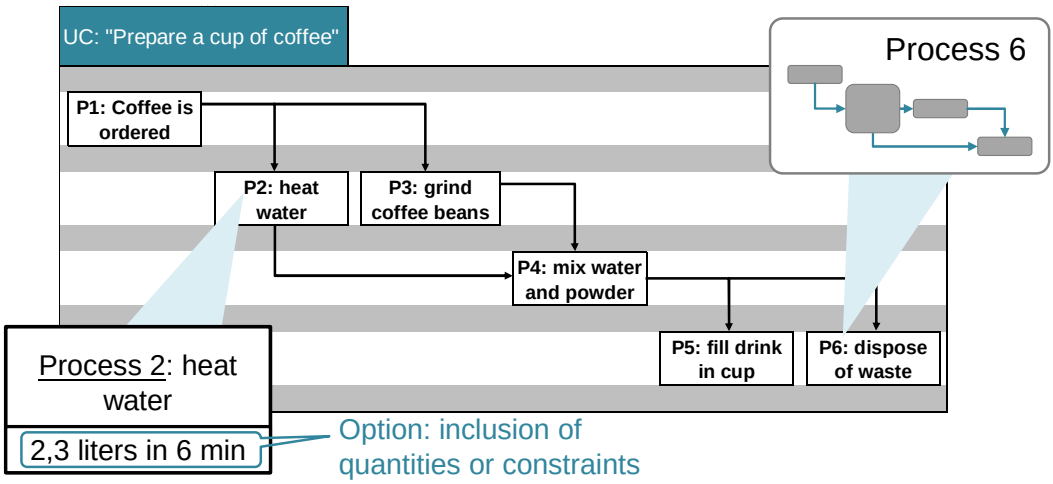

Figure 3: Process flow view for preparing a cup of coffee

In this simple example, the use case is initially modelled using high-level transformation processes. As described in the previous section, these may subsequently be decomposed and detailed. For instance, process 6 “dispose of waste” may involve a separate flow of sub-processes, possibly associated to a service offering. These may be modelled through detailing the existing process flow (as illustrated in Figure 3). Alternatively, an existing service blueprint may simply be attached.

Several transformation processes may be involved in multiple use cases, as illustrated in Figure 4. For instance, process 2 “heat water” is also involved in the use cases of “preparing a cup of cappuccino”, “automated cleaning”, etc. This view supports analysis of dependencies between use cases, involved actors and processes.

Use Cases		Process 1	Process 2	Process 3	Process 4	Process 5	Process 6
	Prepare a cup of capuccino		X	X	X	X	X
	Prepare a cup of hot tea water		X			X	X
	Automated cleaning		X				X
	...						

Figure 4: Use case view (excerpt)

Figure 5 illustrates the allocated actors for the given transformation process flow. These include technical sub-systems such as heating system, grinder, etc. as well as stakeholders, such as service provider, user etc. The specific role of individual actors in the realization of the different processes is indicated with “X” for *affecting* or “O” for *being affected* by a transformation process, respectively.

Actors	Process 1	Process 2	Process 3	Process 4	Process 5	Process 6
Heating system	O	X		X	X	
Grinder		O	X	X	X	X
Cup					O	
Control unit	O	X	X	X	X	
Service provider						X
...						
System Border						
User	X					
Environment		O				O

Figure 5: Associated actor view

Based on the given process flow and allocation of actors, the corresponding *state view* may be completed (see Figure 6). Individual states of actors and operands are transformed associated to the individual processes in the transformation flow. For instance, the water’s (operand) state will change from 20 degrees to 95 degrees Celsius through process 2 (P2) “heat water” and is finally transformed in its state from water to coffee through process 4 (P4) “mix water and powder”.

	Actors								*Operands*		
	Heating system	**Grinder**	**Cup**	**Control unit**	**Service Provider**	**...**	**User**	**Environ-ment**	**Water**	**Coffee beans**	**Energy**
initial states	**switched off**	**swichted off**	**empty**	**active**	**inactive**		**desires coffee**	**N.A.**	**20°**	**whole**	**electrical**
process	P1	P1		P1			supporting P1				
states	switched on	switched on									
process	supporting P2	supporting P3		supporting P2 and P3					P2	supporting p2	P3
states	switch off	switched off							100°	coffee powder	heat
process	spporting P4			Supporting P4					P4	P4	
states									coffee	waste powder	
process			P5	supporting P5	supporting P6		P5			P6	
final states	**switched off**	**switched off**	**filled**	**active**	**inactive**		**has coffee**		**coffee**	**emtptied**	**heat**

Figure 6: Associated state view

Effect views may be generated for one or more transformation process blocks, using a similar representation as the *process flow view* and associated *state view*. For the given example, process 2 “heat water” may thus be modelled, as illustrated in Figure 7: it requires physiochemical effects, in order to

transform electrical energy into heat, which needs to be channelled towards the involved water, in order to heat it.

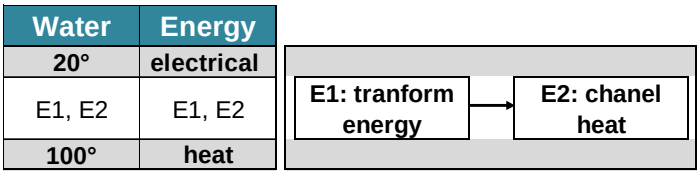

Figure 7: Effect view with associated partial state view for process 2 "heat water"

Finally, the bilateral impacts between individual actors, between individual operands and between operands and actors are modelled in the *interaction view*. For illustration purposes, occurrence of bilateral impacts is simply indicated with an "X" in Figure 8. Examples of bilateral impacts include,

- the service provider (actor) impacts on the grinder (actor) e.g. during waste disposal, embodied through physical contact transmitting forces and waste material;
- the hot water (operand) may impact on the grinder or cup (actors) through transmitting heat, embodied through physical contact and radiation;
- finally, operands may impact each other, as e.g. hot water the coffee powder.

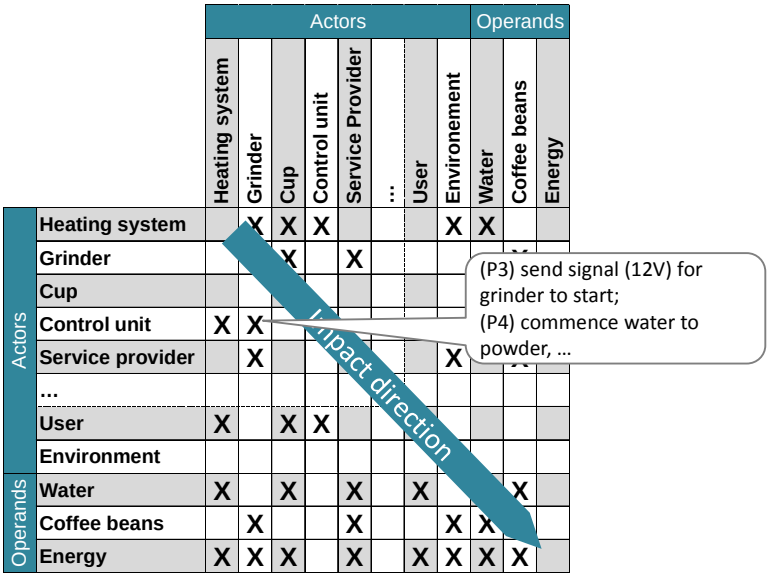

Figure 8: Interaction view

4. Potentials for analysing system functionality

DSM represent the dependencies between entities in a matrix-format. A DSM can be used to model any kind of dependency – usually focusing on one or few dependencies at a time – but different extensions to the basic DSM have been proposed in the last years. Research into analysis options resulting from a DSM representation is considerable and on-going (see e.g. [Lindemann et al. 2009, Eppinger and Browning 2012]). The specific setup of interlinked DSM-based, adjacently placed views in the IFM framework is expected to facilitate application of established analysis methods for DSM and multi-domain matrices (MDM). Analysis may be performed for a completed IFM model or during the process of model creation. Central analysis options that may be applied are described in the following.

Conflict analysis

Dependencies between inherent transformation processes and involved actors can affect the inter-operability of individual use cases. For instance, process 2 "heat water" in the provided example may be associated to alternative quantities or constraints in different use cases. Regarding the preparation of coffee, process 2 is expected to heat 150 ml of water to 100 degrees Celsius (see Figure 3), whereas in

the use case of “automated cleaning” 500 ml of 60 degrees Celsius may be required for flushing. Thus, both use cases are mutually exclusive and may not be performed in parallel, due to the deviating quantities and temperatures associated to the same operand. Similarly, actors and operands may be mutually exclusive, such as e.g. nuclear radiation and humans as well as any related use case these are involved in (unless precautions are taken). For other examples additional steps may be required before switching between use cases. The framework may support the designers in the analysis of such interdependencies and related conflicting actors and processes. In addition, changes in individual use cases over the system’s life cycle (e.g. as variant use cases discussed above) and their effects on expected functions, i.e. process and involved actors and operands, may be analysed.

Consistency analysis

Modelling functionality with the IFM framework is explicitly intended to support getting started with conceptual design on paper. However, the inherent matrices may become considerably large, thus analysing internal consistency is of particular importance, in order to prevent flaws in modelling. Individual views are strongly interlinked through the shared header rows and columns of inherent matrices in adjacent views. This facilitates verification of their mutual consistency and thus the parallel development of different views in the entire framework. For instance, considering the required changes from initial to final states of operands and actors in the *state view* facilitates the parallel development of the *process flow view* and vice versa. In the process of modelling and developing one particular view in the framework, the corresponding header rows and columns in adjacent views are simultaneously filled. Furthermore, completeness of process flows related to individual use cases may be evaluated using logical consistency analysis of associated actor and operand states, e.g. in order to identify missing steps or eventual additionally required actors etc. With regard to the provided example, it allows for consolidation whether e.g. the logical flow of states from water and coffee beans to become coffee and waste is complete, as are the related transformation processes (heating water, grinding, etc.) and enabling actors (e.g. heating system grinder, etc.). This kind of analysis is particularly important while gradually detailing respective process flows, as described in section 3.1.

Property analysis and change prediction

A DSM allows for assessment of changes of sub-systems and the propagation of effects of changes on other sub-systems. In redesign projects a completed IFM framework may already exist. Property analysis and change prediction may be applied in two ways: (a) reasoning from a changed set of requirements towards the *required* changes of use cases and inherent transformation processes, involved actors etc. or (b) reasoning backwards from a changed actor, use case etc. towards consequent changes in the offered functionality of the technical system under consideration.
The IFM framework particularly aims to support designers in reasoning from required functionality towards the structure of a potential solution concept. Information about how actors and operands may impact on each other facilitates the design of the interfaces between them accordingly in later design stages. Similarly, analysis of interferences between actors and/or operands may highlight problems with function fulfilment or even with malfunctioning in use. This is of particular benefit analysing the specific reasons for system failure.

Analysis and identification of possibilities for system improvement

Typically, alternative variants of actors and transformation process flows may fulfil desired requirements and use cases for a system under development. Depending on the design context, alternative solutions may offer specific advantages over other variants. For instance, reducing complexity within a technical product or associated to stakeholder interaction processes is a desirable goal in a development project. Using the IFM framework, designers can directly analyse and evaluate the amount of actors within the system and complexity of their bilateral impacts. Thus, alternative combinations for process flows and involved actors may directly be compared and evaluated. Related methods and tools for complexity management of systems based on DSM and MDM is discussed in more detail e.g. in [Lindemann et al. 2009, Eppinger and Browning 2012].

Analysing possibilities for system modularization

In case a set of actors possesses many bilateral impacts and dependencies among each other, but only few to other surrounding actors, they may be regarded as a "cluster" [Lindemann et al. 2009]. Integrating these into one system module with defined inputs and outputs towards other actors may support modularization. Established methods for such analysis can directly be applied to the matrix representation used in the IFM framework.

Analysis of consistency between requirements and potential solution concepts

As discussed above, one particular benefit of the IFM framework is to facilitate moving from function modelling towards an initial system structure or interface matrix, respectively. It thus allows analysis of consistency between specific interfaces or functional requirements predefined in the requirements specification and the structure as well as offered functionality, established in a potential solution concept. Furthermore, the DSM-based representation allows for direct links to a variety of established system simulation tools, which offers quick an impartial means for evaluation of alternative solution concepts with respect to requirements and function fulfilment (see [Eisenbart et al. 2013d]).

5. Discussion and conclusion

The IFM framework has been proposed, with the specific aim of facilitating joint modelling and analysis of system functionality, i.e. facilitating joint function reasoning across disciplines. Different perspectives onto system functionality are included and linked. In summary, the framework

- uses an established DSM-based representation, in order to facilitate clearly structured representation and analysis of system functions;
- is modular, in the sense that it enables adaptation (augmenting and tailoring) related to a specific design context, i.e. the specific needs of designers;
- allows focusing on individual modelling perspectives, thus supporting modelling and retrieving information relevant to individual designers quickly;
- allows different starting points and alternative sequences for function modelling;
- can address the system functionality on different levels of detail or abstraction.

In this article, the specific application of the framework onto an example system has been illustrated. Furthermore specific options for function analysis have been explored which directly result from the DSM-based setup of the framework. Considering the discussed potentials for modelling and analysing system functionality across disciplines, the framework may provide designers with a valuable modelling approach capable of supporting joint function reasoning. Feedback received from designers in industry suggests that it is in particular the discussed options for analysis of system functionality that could proof very valuable in design practice, especially in redesign projects. Most notably that refers to malfunctioning analysis of existing systems, modularisation efforts and change prediction of system properties, discussed in the previous section.

The framework is thus expected to be capable of facilitating the establishment of a shared, comprehensive understanding of the system under development among involved designers. The specific setup of interlinked views may further direct designers to look beyond those function modelling perspectives, typically relevant to them. The explicit inclusion of the specific interactions between individual actors (i.e. resulting in an initial system structure or interface matrix) further provides links to models used in later design stages and established analysis methods.

Current and future research addresses the practical application of the developed framework in industry. The gained insights will be used to develop the framework further and improve its applicability in different design contexts. Furthermore, future work will include the development of a software-tool support for assisting the application of the framework in modelling system functionality. Such a software-tool is expected to provide suitable means for managing the represented information. Another potential benefit of such a software-tool could be the automated placement of information into respective matrices, as soon as it becomes available.

Acknowledgements

The authors would like to thank the Fonds Nationale de la Recherche Luxembourg for funding this research. Special thanks go to Prof. M.M. Andreasen (DTU Copenhagen), Dr. H. Moser (LuxSpace), and Prof. W.E. Eder (Royal Military College) for valuable discussions and feedback as well as Prof. Tim McAloone (DTU Copenhagen) for continuous support.

References

Andreasen, M.M., Howard, T. and Bruun, H., "Domain Theory, its Models and Concepts", in Chakrabarti, A. and Blessing, L.T.M. (Eds.), An Anthology of Theories and Models of Design: Philosophy, Approaches and Empirical Explorations, Springer-Verlag, Berlin (2014).

Chakravarthy, B.K., Albers, A. and Schweinberger, D, 'Collaborative Environment for Concept Generation in New Products', Proceedings of International Council of Societies of Industrial Design (ICSID 2001).

Crilly, N.: The Role that Artefacts Play. Technical, Social and Aesthetical Functions, Design Studies Vol. 31, pp. 311–344 (2010).

Eckert, C. 'That Which is not Form. The Practical Challenges in Using Functional Concepts in Design', AI EDAM, Vol. 27 (3) (2013).

Eisenbart, B., Gericke, K., and Blessing, L., 'A Framework for Comparing Design Modelling Approaches Across Disciplines', Proceedings of 19th International Conference on Engineering Design – ICED (2011).

Eisenbart, B., Blessing, L. and Gericke, K., 'Functional Modelling Perspectives Across Disciplines', Proceedings of 12th International Design Conference, Design (2012).

Eisenbart, B., Gericke, K., and Blessing, L., 'An Analysis of Functional Modelling Approaches Across Disciplines', AI EDAM, Vol. 27 (3) (2013a).

Eisenbart, B.; Qureshi, A.J.; Gericke, K. and Blessing, L., 'Integrating Different Functional Modelling Perspectives', ICoRD'13, Springer, pp. 85-97 (2013b).

Eisenbart, B., Gericke, K. and Blessing, L., 'Adapting the IFM Framework to Functional Approaches Across Disciplines', Proceedings of 19th International Conference on Engineering Design – ICED (2013c).

Eisenbart, B., Dohr, F., Gericke, K., Vielhaber, M. and Blessing, L.: Potentials for Realising a Consistent Transition Between Function Modelling with the IFM Framework and Early System Simulation', Proceedings of ICED (2013d).

Eppinger, S. D. and Browning, T.R., 'Design Structure Matrix Methods and Applications', MIT Press, Cambridge (USA), London (2012).

Erden, M., Komoto, H., van Beek, T.J., D'Amelio, V., Echavarria, E., and Tomiyama, T., 'A Review of Function Modeling, Approaches and Applications', AI EDAM, Vol. 22, p. 147–169 (2008).

Far, B.H. and Elamy, H., 'Functional Reasoning Theories, Problems and Perspectives', AI EDAM, Vol. 19, pp. 75–88 (2005).

Frankenberger, E., Birkhofer, H. and Badke-Schaub, P. (Eds.), "Designer: The Key to Successful Product Development", Springer-Verlag, London (1998).

Gericke, K., Qureshi, A.J. and Blessing, L., 'Alanyzing Transdisciplinary Design Processes in Industry – an Overview', Proceedings of the ASME 2013 IDETC/CIE (2013).

Lindemann, U., Maurer, M., and Braun, T., 'Structural Complexity Management', Springer, Berlin (2009).

Nevala, K., "Content-based Design Engineering Thinking. In the Search for Approach", Dissertation, University of Jyväskylä, Jyväskylä (2005).

Steward, D., 'The Design Structure System, A Method for Managing the Design of Complex Systems', IEEE Transactions on Engineering Management, Vol. 28 (3) (1981).

Stone, R.B. and Wood, K.L., 'Development of a Functional Basis for Design', Journal of Mechanical Design Vol. 122, pp. 359–370 (2000).

Umeda, Y. and Tomiyama, T., 'Functional Reasoning in Design', IEEE Expert, Vol. 12 (2), pp. 42–48 (1997).

Vermaas, P.E., 'On the Co-Existence of Engineering Meanings of Function, Four Responses and Their Methodological Implications', AI EDAM, Vol. 27 (3) (2013).

Contact: Boris Eisenbart
Research Unit in Engineering Sciences
University of Luxembourg
6, rue Richard Coudenhove Kalergi
L-1359 Luxembourg
Email: Boris.Eisenbart@uni.lu