COMBINING FUNCTION MODELLING AND REQUIREMENTS MODELLING WITH THE IFM FRAMEWORK

Krüger, Merlin Frederik;
Zorn, Stefan;
Gericke, Kilian

University of Rostock

ABSTRACT

The transmission of information between requirements modelling and function modelling in the product development process often appears challenging because of multiple used models and different terminology of specific disciplines. The integrated function modelling (IFM) framework is used for functional analysis of technical moderate complex systems and supports cross-disciplinary modelling and communication in the design team. To improve the applicability of this method and its supporting purpose in the modelling process, the authors combined requirements as an additional entity with the existing entities of this method. Furthermore, the extended framework has been used to visualise the procedure with this approach as an example. The outlook provides the potential for further development of the method.

Keywords: Functional modelling, Requirements, Integrated system modelling, Design methods

Contact:
Krüger, Merlin Frederik
University of Rostock
Germany
merlin.krueger@uni-rostock.de

Cite this article: Krüger, M. F., Zorn, S., Gericke, K. (2023) 'Combining Function Modelling and Requirements Modelling with the IFM Framework', in *Proceedings of the International Conference on Engineering Design (ICED23),* Bordeaux, France, 24-28 July 2023. DOI:10.1017/pds.2023.99

https://doi.org/10.1017/pds.2023.99 Published online by Cambridge University Press

1 INTRODUCTION

For product modelling in the product development process (PDP), handling requirements is significant and has a crucial impact on the success of a project. So far, it has been challenging to consistently track a project's requirements through different stages of the PDP and multiple used models, whether the requirements provide the basis for a new design development or an adaptive design. The single stages of the PDP are individually interconnected with the requirements of the underlying project (Pahl et al., 2007).

Product modelling gets more complex with different disciplines being involved in the PDP. Therefore Eisenbart (2011) provides an overview of addressed design states in methodologies across disciplines, such as need, requirements specification, product functionality or conceptualisation. For these design states, many discipline-specific design models exist based on various methods and approaches. For instance, Jarratt et. al. (2011) provide an overview of common methods and approaches, such as the Failure Mode and Effect Analysis (FMEA), Quality Function Deployment (QFD), or Design For Manufacture and Assembly (DFMA). Eisenbart (2016) developed the Integrated Function Modelling (IFM) framework to improve cross-discipline modelling.

There also exist a wide variety of methods and approaches for requirements modelling, leading to various models. A common method is the V-model which serves as a phase model to handle the requirements management in larger projects (Bender, 2005). Boldt (2000) provides the "what-if" technique for treating requirements as objects. Goldin and Finkelstein (2006) chase the idea of abstracting requirements. Therefore they provide the "AbstRM" that relates to the use of "basic functions" (Roth, 1994; Pahl *et al.*, 2007). Finally, Weber (2005) provides the "Characteristics Properties Modelling (CPM)" and the "Property-Driven Development (PDD)" which basically aims to map a given set of functional requirements (FR) into design parameters (DP).

The problem is that the specific disciplines, methods, and approaches use different taxonomies, leading to inconsistent translations and transmissions of information between these disciplines and the particular models. Additionally, there exists inconsistency in terminology, which intensifies these aspects. Hence, the patency of handling requirements during the whole PDP is inadequate. Furthermore, this conflict often appears between different models that either focus on requirements modelling or function modelling because of the various design perspectives. Thus, an approach that aims to bridge the design states could improve patency in product modelling by reducing translation and transmission activities and reducing different terminology and taxonomy. Further, it prevents focussing on either requirements modelling or function modelling from different design perspectives. This will automatically result in an improved failure rate and economy of time. There already exist methods and approaches in system modelling providing solutions for solving these issues, e.g. the RFLP approach (Kleiner and Kramer, 2013) as a means to bridge requirements specifications and product functionality, or the Adaptive Redesign Method (ARM) and the Detailed Design Model (DDM) by Wong and Wynn (2022) to support the adaptive redesign of product variants by connecting product functionality and conceptualisation based on the connection to the Theory of Inventive Problem Solving (TRIZ) to resolve performance conflicts. But often approaches and their specific models are restricted by other tools, software or methods, provide unsatisfying visualizational aspects, therefore they also demand high educational effort, or do not extend through the whole PDP and further do not support method-crossing information transfer.

The author's work focuses on developing a method that solves these issues in system modelling, and specifically, to support Engineering Change activities. Herefore, various steps are necessary. This paper aims to provide an approach for combining requirements and function modelling based on an Integrated Function (IF) model as one of these steps. In the following, Eisenbart's IFM framework is presented as the primary method for this approach. But this method does not offer interfaces between requirements and function modelling yet. Hence, different types of requirements and taxonomies are discussed and a functional differentiation of types of requirements is defined for integrating requirements into the IFM framework. Furthermore, the approach of integrating requirements into the IFM framework is explained, and results are exemplified. Finally, the paper closes with an outlook of future work after discussing the opportunities and limitations.

https://doi.org/10.1017/pds.2023.99 Published online by Cambridge University Press

2 THE INTEGRATED FUNCTION MODELLING FRAMEWORK

The IFM framework is used for function modelling and function analysis. It is based on the concept of Design Structure Matrices (DSM) and Multi-Domain Matrices (MDM) (see Eichinger, Maurer and Lindemann, 2006; Kreimeyer and Lindemann, 2011) and relates the different function modelling perspectives of various design states (Eisenbart and Gericke, 2011). It facilitates cross-disciplinary modelling and analysis of the functionality of a system (Eisenbart et al., 2016). Therefore the IFM framework incorporates different entities (Fig. 1), including use cases, processes, effects, states, operands, and actors. The system may support one or more use cases. Each use case consists of multiple processes, which can be transformation or interaction processes. Processes include actors and operands. Each of them has various states, and these states may change, resulting in the process flow of each use case (Eisenbart et al., 2016).

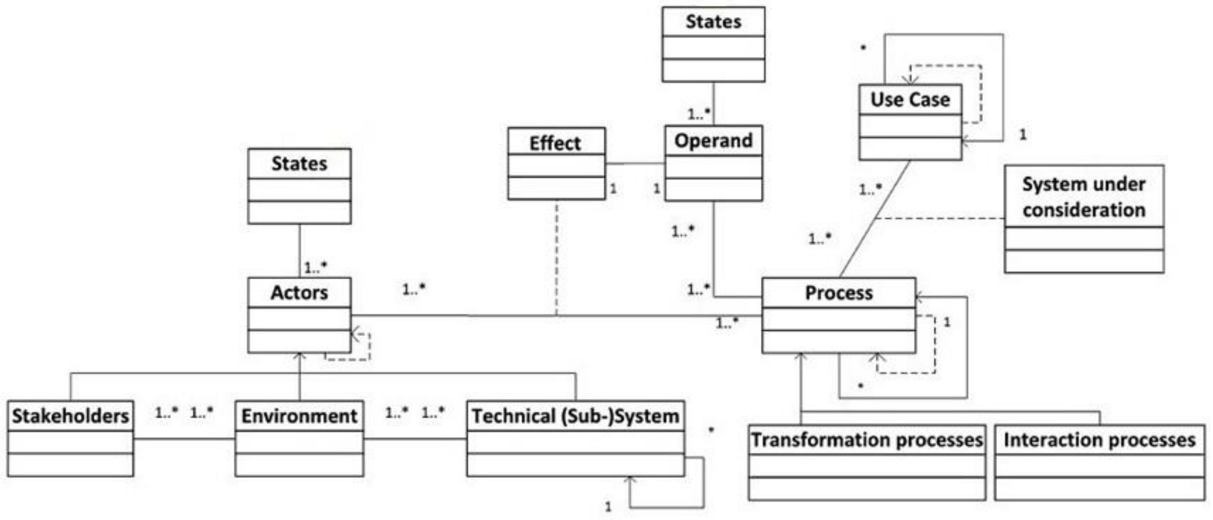

Fig. 1 Class diagram of the IFM framework (Eisenbart et al., 2016)

The entities are modelled individually in specific matrices and combined into a modular framework of multiple views (use case-, process flow-, effect-, state-, actor-, and interaction-view) as illustrated in Fig. 2a. The Interaction-, the actor-, and use case-view are most likely DSMs and show dependencies between actors and operands (Interaction view), Actors and processes (actor view), and processes and use cases (use case view) by using simple marks in the matrices, such as an "X" or "O" (also see Fig. 5). But the speciality of the framework is the centralised process flow view and the dependent state view (Fig. 2b) because of an integrated time parameter, which is indicated by a vertical arrow between the state view and the process flow view in Fig. 2a. The single processes are visualised sequentially, starting at the upper left and continuing to the lower right of the view. Dependent on a time parameter, every process and its state changes can be considered specifically in the state view. The consideration of processes is different to the consideration of functions, because it includes an input, output, and a defined instant of time. The state view visualises these inputs and outputs of actors and operands of the transformation and interaction processes. The state changes of actors depend on interaction processes and are reversible. The state changes of operands depend on the transformation processes and are irreversible. The centralised modelling of multiple design entities makes the design information more accessible and also provides a helpful self-check for the design engineer who is able to visually cross reference the entities with each other. This supports the modelling process and improves the traceability of system relations among the design entities and the system understanding (Eisenbart et al., 2016). The number of included views and their arrangement in a framework is up to the designer's discretion. For further explanations of the IFM framework, see (Eisenbart, 2013, 2015, 2017).
The IFM framework focuses on function modelling and does not yet include entities or relations to the requirements modelling. So, the first step was to identify entities which are related to requirements.

https://doi.org/10.1017/pds.2023.99 Published online by Cambridge University Press

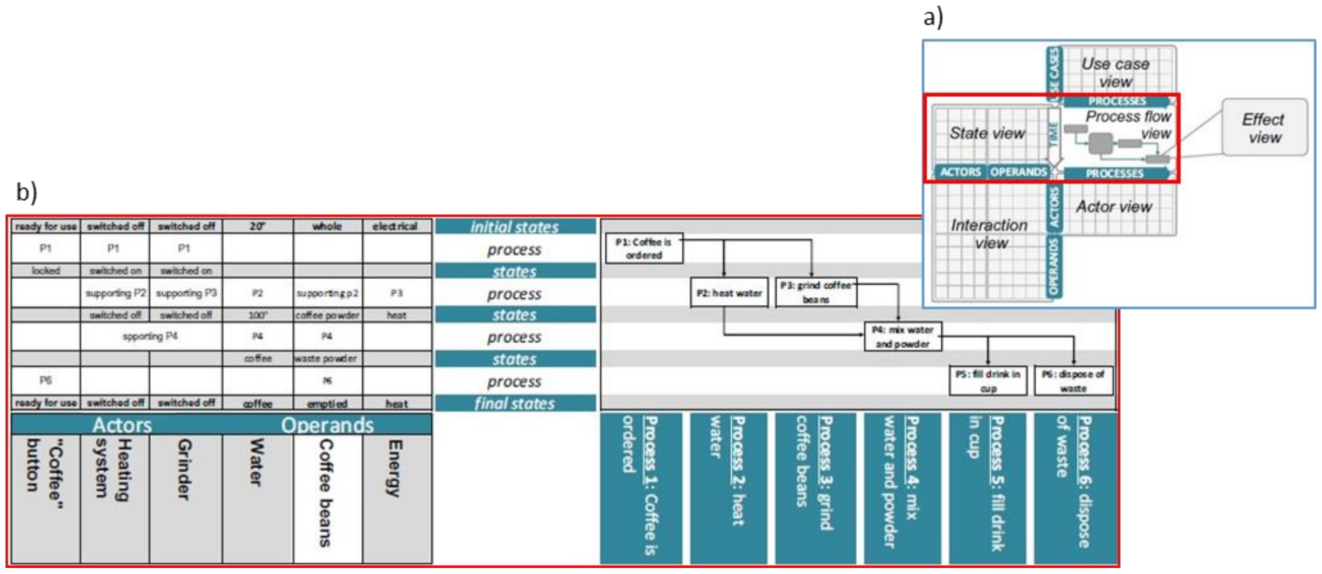

Fig. 2 a) Overview of the IFM framework; b) Process flow view and state view of a coffee machine (cutout); (Eisenbart et al., 2016)

3 TYPES OF REQUIREMENTS

Requirements modelling has a wide range over multiple disciplines and different design states and relates to specific taxonomy for individual methods, approaches or applications. In the following step, different taxonomies are indicated. Providing the notion of embedding a new entity into the IFM framework, the different taxonomies are further tailored. To identify such entities, it is necessary to clarify different types of requirements to define which types can be integrated into the IFM framework and which cannot.

The design process starts with finding and documenting all types of requirements, for example, customer needs. From the very beginning, it is helpful to formulate requirements as precisely as possible. Further, the requirements can be sorted for individual purposes. The sorting can depend on different perspectives, disciplines, design states, and relations to specific design teams or members. Often requirements are handled similarly to the product architecture and can be decomposed into different "levels/domains". The decomposition follows the principle of the problem progression by Jackson (2000), that is further improved by Li at. al. (2013), and results in a careful structuring of a problem. The problem progression additionally demands a differentiation of types of requirements that can be the same way decomposed. For instance, if the costs were one type of requirement, its decomposition would comply with the problem progression and even the product architecture. Costs are often considered as one type of requirement and are further used to define project conditions and boundaries from a manager's point of view. For the manager, also manufacturability has to be taken into account. Meanwhile, the designer focuses on functional requirements and technical feasibility (Jiao and Zhang, 2005).

Hooks and Jackson (1990) differentiate between functional, system, and detail requirements. Hood and Wiebel (2005) differentiate between user, system, and design requirements. Pahl and Beitz (2007) consider implicit and explicit requirements and mention the difference between demands and wishes. Anyway, these differentiations may provide guidance but cannot be adopted one-to-one. With the IFM framework's purpose of function modelling, the most evident and straightforward distinction would result in "functional requirements (FR)" and "non-functional requirements (NFR)", whereby the definition of FR and NFR has to be clarified.

Therefore Weber (2005) provides suitable guidance. He differentiates between characteristics and properties. Characteristics can be directly influenced or determined by the designer (e.g. material, shape, dimensions, etc.). Whereby the properties describe the product's behaviour (e.g. weight, safety, manufacturability, costs, etc.) and are driven indirectly by the characteristics. For Instance, weight (as a property) cannot be changed directly; it depends on the material, dimensions, etc. (characteristics) and can only be changed by changing the driving characteristics.

This distinction is further used to define functional requirements as "direct characteristics (driving)" and non-functional requirements as "indirect characteristics (driven)". This distinction is also similar to Pahl and Beitz's consideration of implicit and explicit requirements.

4 INTEGRATING REQUIREMENTS INTO THE IFM FRAMEWORK

As mentioned in chapter 2, the speciality of the IFM framework is the process flow view and the dependent state view. The state view shows the state changes of processes in detail. The visualisation of state changes is based on Hubka's (1988) theory of technical systems, which shows the stepwise transformation of an operand by the "Transformation Process" as shown in Fig. 3. Because the interaction processes are therefore less essential, the following focuses on the transformation processes. An operand can reach multiple intermediate states, based on the number of operations in the transformation process. It is essential that Hubka explains the operand evolves through multiple intermediate states until it reaches a state that is desired by the designer. The desired state is then considered the output that has to be achieved by the transformation process. Furthermore, the combination of the outputs of multiple transformation processes, i.e. multiple desired states for specific operands, is the aim of the technical (sub-)system.

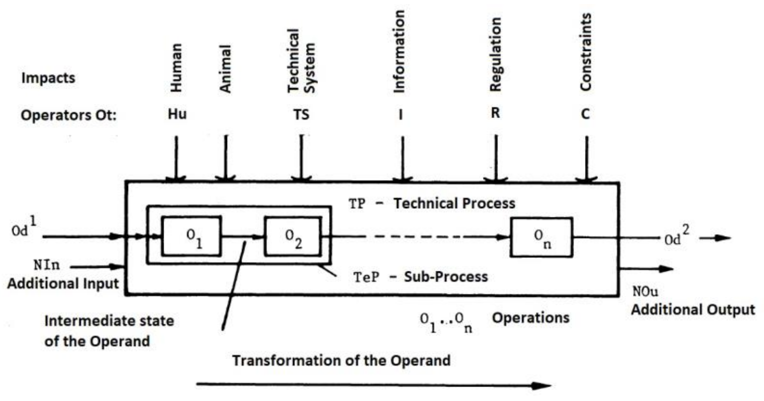

Fig. 3 Transformation Process (Hubka and Eder, 1988)

Hence, considering desired states as the aim of a technical system, the desired states can further be connected to the requirements which also represent a technical system's aim.
The IFM framework provides the same type of detail for considering processes. Based on combining states and requirements, and the distinction between NFR and FR, the following extended class diagram (Fig. 4) shows an approach to connect the FRs and NFRs with the already existing entities of the IFM framework. Guided by Hubka's desired states, the relation between FRs and states is considered the basic step of integrating requirements into the IFM framework. Hence, the relations are explained in the following, and then the paper continues by focusing on the relation between FRs and states. The focus is also highlighted via different colouring. All relations between FRs and other entities can be explained "bottom-up", therefore also recall the explanations of the class diagram in chapter 2: The FRs are related to the states of operands and actors. The states and state changes specifically describe the processes. I.e., the FRs related to the states describe what has to be achieved by a process. Furthermore, multiple processes describe one use case. So, the FRs can also be related to the use cases to describe what has to be achieved by the use case (at the end of a use case). One or more use cases fulfil the technical (sub-) system, i.e. the FRs can be related to the technical (sub-) system. Hence, it is possible to decompose the requirements to the appropriate level of detail that the designer needs, somewhere in between the system, its use cases, or its states. This type of decomposition recalls the relation to Jackson's problem progression that was already mentioned in chapter 3.
Because the NFRs can only be changed indirectly, they are not connected to the system's actions, i.e the functionality that is captured in use cases or processes, but to the system overall and specifically to its actors and operands. For Instance, the designer could change an actor to reduce the weight of that specific actor, e.g. changing a metal one to a synthetic one. If the weight of a specific actor is reduced, the weight of the system is automatically reduced too.

https://doi.org/10.1017/pds.2023.99 Published online by Cambridge University Press

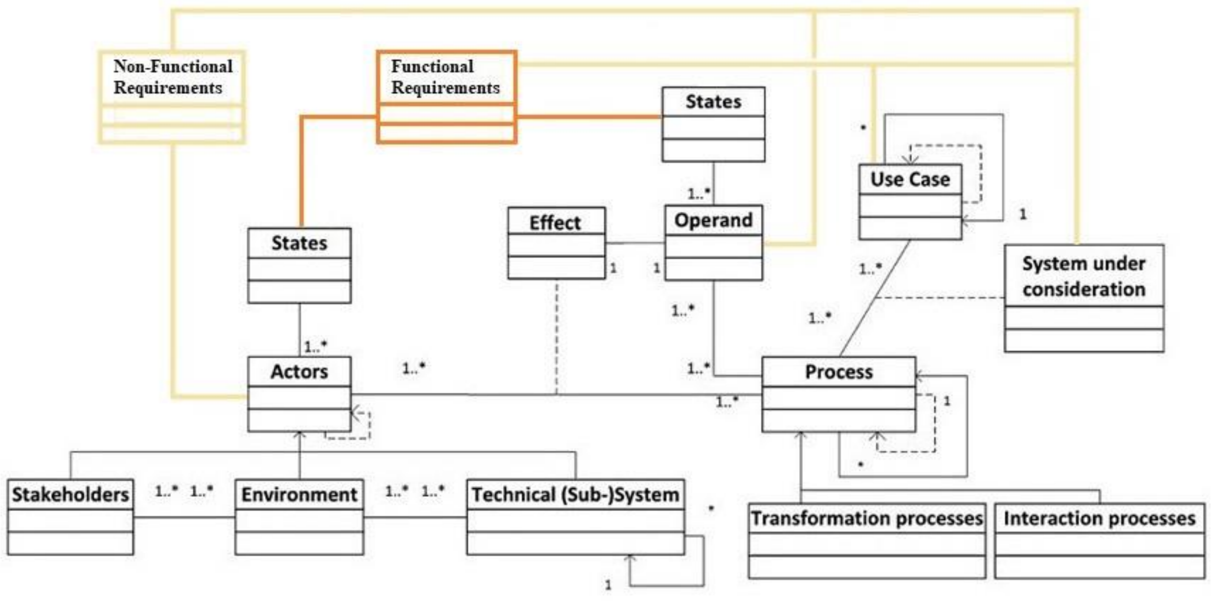

Fig. 4 Extended class diagram of the IFM framework

The FRs are connected to the states and can be changed by controlling the specific transformation process or the use case. For instance, as a transformation process "heat water" to 40°C (FR → 40°C), heat water to 50°C (FR → 50°C), or heat water to 60°C (FR → 60°C). But with controlling this exemplified transformation process the NFRs do not change, i.e. no matter how the system is regulated, the weight (for example) remains the same.The following section shows how the FRs can be visualised in the IFM framework.

5 APPLICATION OF THE EXTENDED IFM FRAMEWORK

The previous section showed how the FRs and NFRs can be related in the framework theoretically. As described, the NFRs are driven by the FRs. Hence the following focuses on how to visualise FRs in the framework. Because of the relation between FRs and states, it seems suggested to integrate requirements in the state view, next to the defined states. The implementation of requirements is shown on the example of the IFM framework of the coffee machine (Fig. 2) modelled by (Eisenbart et al., 2016).

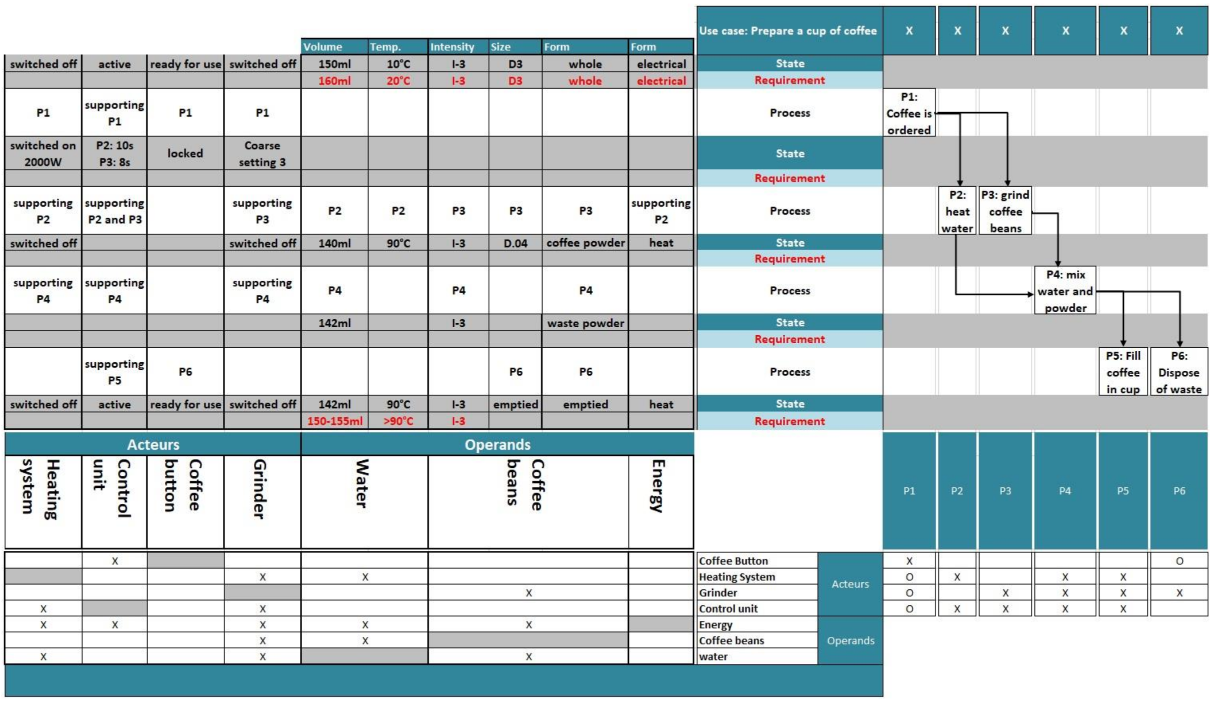

Fig. 5 IFM framework with requirements

https://doi.org/10.1017/pds.2023.99 Published online by Cambridge University Press

In Fig. 5 we see additional rows for requirements (marked in red letters) to visualise states and related requirements in parallel in the state view. The requirements can be visualised as set-based, point-based or ranged. For instance, in the use case "prepare a cup of coffee", the coffee is related to a volume between 150ml and 155ml (set-based), a temperature higher than 90°C (ranged), and an intensity of I-3 (point-based) as an exemplarily scale. Hence, the considered characteristics are "volume", "temperature", and "intensity", and they extend over multiple operands (water, coffee beans, and energy). To make the state-requirement-relation more accessible, it is important to define "quantified" requirements as proposed by Pahl and Beitz (2007). Because then the "values" of the characteristics, e.g. "90°C" for the "temperature" of the water, that are intermediate states, can be related to the requirements. It is up to the designer, which intermediate states can be considered "initial" (input) or "final" (output), and most requirements will probably relate to the initial and final states.
FRs are defined as driving characteristics, therefore these characteristics have to be individually visualised for each actor or operand. As a change to Eisenbart’s framework, the single column for each actor or operand to describe its state and state changes can be expanded to multiple columns for a single operand or actor to describe multiple specific characteristics and changes that are important to the design. Hence, states can further be considered in greater detail. As mentioned, requirements can be related specifically to each intermediate state, although most requirements only relate to “initial-” or “final”-states, i.e. inputs and outputs (of the system). But it is possible to decompose a single requirement to an intermediate state. This enables the opportunity for a very high-level detailed consideration of requirements in a specific environment if necessary. It is up to the designer to choose an appropriate level of detail.
The designer can further compare states and requirements directly to check the fulfilment of the requirements. It enables the designer to detect more effects, for instance, a non-fulfilled, a tolerated or a fulfilled requirement. These could be visualised via different colour coding. The different colouring also helps the designer to promptly realise which states the FRs are related to, for this example there only exist relations to the initial and final states of specific operands to define the coffee (final/output) that has to be prepared by the use case and the defined initial states (inputs).
Considering states and requirements individually for operands and actors during the process flow gets more abstract when/if single operands get combined as in the example of the coffee machine. In such a combined state it would be more coherent for the designer to visualise states, state changes and requirements related to a combined operand. The combination of the operands is shown in the following example (Fig. 6).

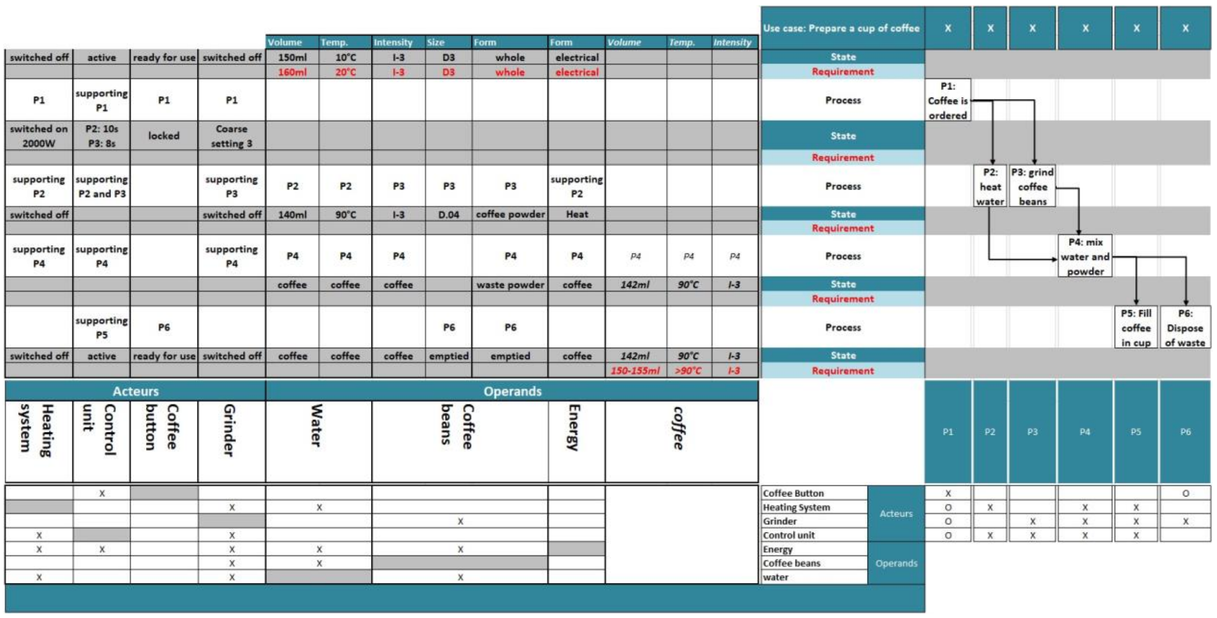

Fig. 6 IFM framework with combined operand

For this example, a combined operand would be "coffee" for the use case "prepare a cup of coffee". Other combined operands could be cappuccino, espresso, etc., related to the specific use cases. To support the designer, the combined operand can be listed in a separate column (Fig. 6) with all characteristics that are important to the design, e.g. "volume", "temperature", and "intensity". The state changes of the characteristics get added to the combined operand coffee sequentially or in parallel during the process flow. Thus, it is not astonishing that the combined operand coffee has no initial states of characteristics at the beginning of the process flow. In the case of the coffee machine, the state changes of the characteristics get added in parallel. Still, it is possible that a combined operand evolves during the process flow with a sequential summing up of state changes of the characteristics. The designer reads the framework as follows: Every state change to a characteristic of an operand is visualised in an individual column of the specific operand until it reaches a combined state. With reaching the combined state, the visualisation of every following state change of the considered characteristic switches to the column of the combined operand.

Considering the three basic flows (material, energy, and information) (Pahl *et al.*, 2007), only the basic flow of material is combined by combining operands. For instance, water (material) and beans (material) are combined to "coffee". However "energy" is listed as an operand, it still has to be handled differently, e.g. for heating the water (see Fig. 6, state view; water; temp.; P2: → 90°C) the energy is not combined with the water, because there already exists energy in the water, the energy in the water is somehow changed (increased or reduced). Hence, this affection is still visualised in the specific column of the operand water. Instead, by combining the materials, the combined operand "coffee" evolves, which is why all the dependent characteristics switch to the column of coffee.

As mentioned in chapter 4, there do not only exist relations between FRs and states, but also between FRs and use cases, or FRs and the system overall. Based on the highlighting in the given class diagram (Fig. 4), this paper focuses on the relation between FRs and states. The visualisation of the other relations still has to be explored.

6 DISCUSSION

The presented work differentiates into two types of requirements, the FRs and the NFRs, and further provides an approach to visualise FRs in the state view of the IFM framework. In the first step, the visualisation is based on considering transformation processes, and relates between FRs and states. In the state view, requirements can be visualised at the process level, i.e. the FRs related to states of the transformation process. As already mentioned before, requirements can be decomposed, similar to the problem progression. Meanwhile, the relation between FRs and states is on a very specific and detailed level, the requirements can also be considered on a coarse level. Therefore, the extended class diagram (Fig. 4) provides the relation between FRs and states, FRs and use cases, and FRs and the system under consideration. For future work, it has to be examined, how the FRs can be visualised in the IFM framework on a coarse level, e.g. in relation to the use cases or the system.

The NFRs are also related to the existing entities of the IFM framework but are not part of the visualisation yet. Therefore, the most appropriate way of considering the NFRs in working with the IFM framework has to be discovered.

The state view now visualises states and FRs but given the provided design purpose a “state” can be considered differently. If there already exists a system, the states can be described as intended states or realised states, i.e. the way the system is actually working (Rötzer et al., 2022). Hence, the designer first has to decide, what is defined by "state". Alternatively, the designer can visualise both types of states in addition to the FRs and tailor the framework for specific considerations. Resulting to the design purpose the designer then has to face either state and requirement for new design development, also state and requirement for adaptive design development in case of functional extension, or state and realised in case of a functional failure of performance (see Krüger et al., 2020).

Considering operands individually for the whole use case can lead to abstractive perspectives and result in deficient design. Therefore, multiple operands can be combined, meanwhile, the combining refers to the flow of material. It has to be clarified, if it is possible and/or necessary to visualise the combining of single energies or signals without combining material. Hence, it has to be analysed, if it is reasonable to list energy as an operand.

An outlook for engineering change

With modelling FRs in the IFM framework, the function modelling perspective is extended by the missing part for also visualising the engineering change. The following explains how to identify the impacts of changes in the framework manually.

For instance, if the initial characteristic “size” of the coffee bean (see Fig. 6) in the state view is changed from D3 to D4 (D3 and D4 exemplify the size by a diameter of 3mm or 4mm), then the column of “size” will be tracked row by row until a state change has been realised by a process (see Fig. 6 - Process P3). Secondly, the provided process will be tracked column by column through the row for the whole state view. Every single actor or operand that supports or is supported by the same process (P3) is perhaps affected by the state change. The extent of the impact cannot be defined, but it is somehow subjective to a considered actor or operand and all its interactions, processes and states and further possible impacts. The basic principle is to explore where a change has an impact. The other way round, e.g. an actor has been changed, it is possible to capture all the processes and dependent states this actor is involved in or related to and further the change may have an impact on. The tracking can be coupled with colouring to highlight the impacts. Because manually tracking the changes can be too extensive, the approach demands tool support. A tool could improve multiple handlings of the framework, such as highlighting or tailoring. Also, the comparison of FRs and states as mentioned in chapter 5, i.e. the fulfilment of a requirement, could be automatically highlighted by a tool.

7 CONCLUSION

This paper shows an approach to combining function and requirements modelling with the IFM framework. Therefore a differentiation into different types of requirements was necessary and has been defined as functional requirements and non-functional requirements based on literature. The existing entities of the IFM framework have been extended by the different types of requirements. Furthermore, the framework has been extended by visualising the functional requirements in the state view. It has been noticed, that the non-functional requirements are not part of the IFM framework so far.
Further improvements of the IFM framework have been discussed, such as the decomposition of requirements in relation to use cases or the (sub-)system, or the consideration of visualising different types of states in the framework, i.e. realised states and intended states.
Also, the framework provides a basis for visualising the engineering change and demands of an automation tool.

REFERENCES

Bender, K. (2005) Embedded Systems - qualitätsorientierte Entwicklung. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg. https://dx.doi.org/10.1007/b138984.

Boldt, L. (2000) ‘Managing requirements at the object level’, SAE Technical Papers, (724). https://dx.doi.org/10.4271/2000-01-2570.

Eichinger, M., Maurer, M. and Lindemann, U. (2006) ‘Using multiple design structure matrices’. 9th International Design Conference, Design 2006, pp. 229–236.

Eisenbart, B. (2013) ‘Der Integrated Function Modelling (IFM) Framework’, (January).

Eisenbart, B. et al. (2015) ‘Integrated function modelling: Comparing the IFM framework with SYSML’, Proceedings of the International Conference on Engineering Design, ICED, 5(DS 80-05).

Eisenbart, B. et al. (2016) ‘A DSM-based framework for integrated function modelling: concept, application and evaluation’, Research in Engineering Design, 28(1), pp. 25–51. https://dx.doi.org/10.1007/s00163-016-0228-1.

Eisenbart, B. and Gericke, K. (2011) ‘A Framework for Comparing Design modelling approaches across disciplines’, International Conference of Engineering Design, (January), pp. 1–12.

Gericke, K. and Eisenbart, B. (2017) ‘The integrated function modeling framework and its relation to function structures’, Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM, 31(4), pp. 436–457. https://dx.doi.org/10.1017/S089006041700049X.

Goldin, L. and Finkelstein, A. (2006) ‘Abstraction-Based Requirements Management’, pp. 3–9.

Hood, C. and Wiebel, R. (2005) Optimieren von Requirements Management & Engineering. Springer Berlin Heidelberg.

Hooks, I. and Jackson, B. G. (1990) ‘Why Johnny can’t write requirements’, AIAA Space Programs and Technologies Conference, 1990. https://dx.doi.org/10.2514/6.1990-3561.

Hubka, V. and Eder, W. E. (1988) Theory of technical systems : a total concept theory for engineering design. Berlin: Springer. Available at: https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=3089886.

Jackson, M. (2000) ‘Problem Frames: Analysing & Structuring Software Development Problems: Analysing & Structuring Software Development’, Addison-Wesley.

Jarratt, T. A. W. et al. (2011) ‘Engineering change: An overview and perspective on the literature’, Research in Engineering Design, 22(2), pp. 103–124. https://dx.doi.org/10.1007/s00163-010-0097-y.

Jiao, J. and Zhang, Y. (2005) ‘Product portfolio identification based on association rule mining’, CAD Computer Aided Design, 37(2), pp. 149–172. https://dx.doi.org/10.1016/j.cad.2004.05.006.

Kleiner, S. and Kramer, C. (2013) ‘Model Based Design with Systems Engineering Based on RFLP Using V6’, Proceedings of the 23rd CIRP Design Conference, pp. 93–102. https://dx.doi.org/10.1007/978-3-642-30817-8_10.

Kreimeyer, M. and Lindemann, U. (2011) ‘Complexity Metrics in Engineering Design’. Springer Berlin / Heidelberg.

Krüger, M. F. et al. (2020) ‘The Application of the IFM Framework and FIDD Method on an Industrial Cigarette Filter Maker’, (September), pp. 10–10. https://dx.doi.org/10.35199/dsm2020.19.

Li, Z., Hall, J. G. and Rapanotti, L. (2013) ‘On the systematic transformation of requirements to specifications’, Requirements Engineering, 19(4), pp. 397–419. https://dx.doi.org/10.1007/s00766-013-0173-8.

Pahl, G. et al. (2007) Engineering Design : A Systematic Approach. 3. Edition. Edited by K. Wallace and L. Blessing. Springer Vieweg.

Roth, K. (1994) ‘Konstruieren mit Konstruktionskatalogen’. Berlin [u.a.]: Springer.

Rötzer, S. et al. (2022) ‘Attribute dependency graphs : modelling cause and effect in systems design’, Design Science, 8. https://dx.doi.org/10.1017/dsj.2022.20.

Weber, C. (2005) ‘CPM/PDD – an Extended Theoretical Approach To Modelling Products and Product Development Processes’, Proceedings of the 2nd German-Israeli Symposium on Advances in Methods and Systems for Development of Products and Processes, pp. 159–179.

Wong, F. S. and Wynn, D. C. (2022) ‘A systematic approach for product modelling and function integration to support adaptive redesign of product variants’, Research in Engineering Design, (0123456789). https://dx.doi.org/10.1007/s00163-022-00401-3.