Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 153 (2019) 285–293

Procedia Computer Science

www.elsevier.com/locate/procedia

17th Annual Conference on Systems Engineering Research (CSER)

A knowledge domain structure to enable system wide reasoning and decision making

Tjerk Bijlsma[a]*, Wouter Tabingh Suermondt[a], Richard Doornbos[a]

[a] *ESI, High Tech Campus 25, Eindhoven, 5600 HE, The Netherlands*

Abstract

To architect and design a system, the stakeholder needs have to be satisfied by technical solutions, for which decisions on trade-offs have to be made. A trend is that the number of functions, components, and interfaces in systems increase, often by an order of magnitude or more, such that reasoning about the impact of a decision becomes increasingly hard and tracing its impact throughout the system is crucial. Therefore, we decompose a system in areas of knowledge and information, which we call knowledge domains. Architecting means taking decisions, for which the impact on knowledge domains and their explicit relations are required. Existing approaches that reason across systems either do not make explicit relations between knowledge domains, or perform a quantitative computation instead of reasoning, where for both it is difficult to trace the impact of decisions.

In this paper, we present an architecture reasoning structure with which knowledge domains from different disciplines can be explicitly related, which enables system wide reasoning and decision making. With just eight language elements, a system can be described in an information structure. By applying a knowledge domain pattern, the essential information of knowledge domains is captured. Via relations, both qualitative and quantitative reasoning can be performed to trace the impact of decisions. An example is used to illustrate the approach, for which the tension for a decision is shown by tracing its impact via quantitative and qualitative relations. The approach was investigated and validated in the industrial context of Océ professional printing systems.

Keywords: knowledge domain structure; system wide reasoning; quantitative and qualitative reasoning; decision making

* Corresponding author. Tel.: +31-88-866-5420.
E-mail address: tjerk.bijlsma@tno.nl

1877-0509
10.1016/j.procs.2019.05.081

1. Introduction

To architect and design a system, the stakeholder needs, demands, and expectations have to be satisfied by technical solutions [1]. This often involves solutions from various disciplines [2]. To realize the system, decisions on trade-offs have to be made, which is tightly coupled with architecting and designing. An observed trend for technical solutions is their increasing complexity [3], due to the increasing number of system functions, components, and their interfaces. This complexity increases the time required to make decisions and trade-offs. To handle this growing complexity, support for reasoning about the impact of a decision and tracing this impact throughout the system is desirable.

To make decisions and trace them through a system, we decompose the system in areas of knowledge and information that are needed to create its architecture and design. There are teams with knowledge and information on a specific area, we call this a Knowledge Domain (KD). A KD has a representative, which we will call its owner. Because KDs can be oriented at different disciplines, they can use different languages and definitions. For system wide reasoning and decision tracing, relations between KDs should be known. Therefore, KD owners must understand each other and agree on these interfaces.

Figure 1(a) introduces an electric bicycle case, which will be used to illustrate the challenge in system wide reasoning and decision making. Note that due to its purpose this example is a simplification, where the presented approach has been successfully applied to larger and more complex cases. The challenge is to decide on the cost and the intended usage of the bicycle, where cost and usage are related, and a decision should be made that is possible and supported by all related KDs. In the figure, relevant KDs are sketched by giving their names. Important parameters are near the names to indicate their relation. Next to the KD name, an image of the owner is sketched. To make decisions on cost and usage, and reason about them, the cost and usage relation between the “Frame”, “Electric drivetrain” and “Marketing” KDs should be understood. Such a relation has to be made explicit by relating parameters of the KDs.

Five sources of misunderstanding for decision making have been identified, as illustrated in Figure 1(b). The sources of misunderstanding are illustrated by a black circle with a number. Source *one* shows that there can be a misunderstanding between KDs on the used terminology and definitions, which hampers decision making. For example, when the "Marketing”, “Usability”, “Frame”, and “Electric drivetrain” discuss the usage of the bicycle, they refer to the “usage”, “stiffness”, and “efficiency”, respectively. These parameters probably have different units and meanings, because the KDs are from different disciplines. A *second* source of misunderstanding can be an unclear relation between KDs. For the relation between “Marketing” and “Frame”, the intended “usage” of the bicycle should be translated in the desired “stiffness”. A *third* source of misunderstanding can be the absence of a clear owner of a KD, which results in difficulties to agree on a relation. A KD that has an unclear scope with non-related information is the *fourth* source of misunderstanding, because the encompassed KDs may each have different interests, which can obfuscate the discussion and thereby the realization of relations. As a *fifth* source of misunderstanding, there can be a difference in used abstraction level. One KD can have an abstract high-level description, whereas the description of another KD can be focused and detailed. These five sources of misunderstanding can make it difficult to explicitly relate KDs and thereby reason about the impact of a decision.

Approaches are available for system wide reasoning or decision making. The A3 architecture overviews [4] and DoDAF 2.0 [5] provide multiple views on a system, but do not explicitly relate the parts. The Geeglee approach [6, 7] prescribes a process with six filled score cards to compute design risks and make decisions. To make decisions and to reason about their system wide impact, explicit relations with the impacted parts of a system are desired.

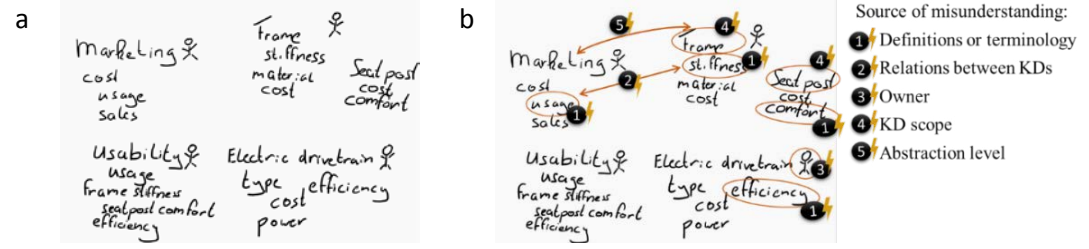

Figure 1: Electric bicycle case, to illustrate the system wide reasoning challenge for making architecture and design decisions (a), and an annotation with possible sources of misunderstanding for decision making (b).

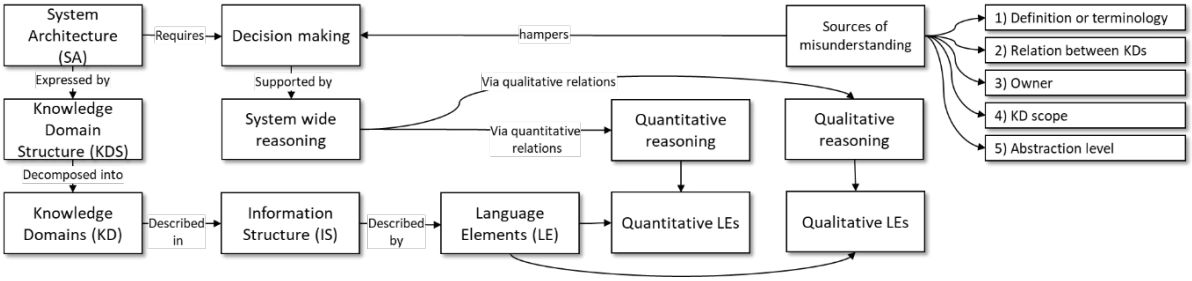

Figure 2: Ontology for system wide architecture reasoning structure.

In this paper, we present an architecture reasoning structure with which KDs from different disciplines can be explicitly related, which enables system wide reasoning and decision making. A fixed set of eight Language Elements (LEs) is given that is sufficient to describe the KDs of a system in a Knowledge Domain Structure (KDS). With these LEs the information of KDs can be captured, where quantitative and qualitative relations inside and between KDs can be described. Via these relations, both qualitative and quantitative reasoning can be performed, such that the impact of design decisions can be traced. This approach was validated in the industrial context of Océ professional printing systems, where it has been applied for system wide quantitative and qualitative reasoning and decision making.

The KDS is developed and validated in the context of the Model-Based System Architecting (MBSA) methodology. In this methodology, the clarity and transparency of models is used in the conceptualization, quantitative reasoning, and decision making. The LEs that are used to describe KDs include validations, transformations, models, and parameters. In KDs, transformations can be used that have a model which determines the values of its output parameters based on input parameter values. Between KDs, validations that have a model can be used. The model of a validations shows if similar values are used for the related parameter, which indicates the level of agreement between KDs on the parameter values. The validations and transformations, with their models, make relations between parameters explicitly and can be used to calculate parameters values, which enables quantitative reasoning. Qualitative reasoning is made possible by using relations inside and between KDs.

An ontology of our system wide reasoning structure for architectures is given in Figure 2. This figure shows that a system architecture can be expressed by a KDS that is described by quantitative and qualitative LEs. Decision making is hampered by sources of misunderstanding, whereas it can be supported by system level reasoning via the LEs.

In the remainder of the paper, we use a bottom-up presentation order, starting from LEs towards a KDS, to enable quantitative and qualitative reasoning. First, Section 2 presents related work. Section 3 introducers the LEs. Using these LEs, KDs are described in Section 4. A KDS suitable for system wide reasoning and making design decisions is presented in Section 5. Finally, Section 6 presents the conclusions.

2. Related work

Related work describes methods to support decision making for architectures or designs. First, methods to architect systems will be discussed, followed by approaches that visualize and describe architectures, to support decision making. Finally, approaches to make decision based on quantitative system models are discussed.

Methods to architect a system are given by the CAFCR and BoDERC methods. The CAFCR method [1] uses customer objectives, application, functional, conceptual, and realization viewpoints to analyze the system and help the architect in deciding on a consistent and balanced system. However, the structure of the information for a viewpoint is not described and relations are not made explicit via parameters. The BoDERC design methodology [8] describes a quantitative model-based architecting process. A design is prepared by obtaining key drivers and core domain knowledge, followed by the selection of critical design aspects for which tensions and conflicts are identified, for which options are evaluated via models and measurements. However, this methodology does not explicitly relate the knowledge in a way that is needed for quantitative reasoning.

Approaches to visualize and describe architectures are given by the 4+1 view model, the A3 architecture overviews, DoDAF, and SysML. The 4+1 view model [9] for software architectures uses a development, logical, physical, and process view, and a set of scenarios to describe these views. Each view has an architecture, and these architectures are

related. However, this approach focusses on the software discipline and does not relate views explicitly via parameters. The A3 architecture overviews [4] enable an informal process to make architecture knowledge explicit, with an A3 per important system aspect for which qualitative reasoning is allowed. Compared to traditional documents, the information in A3s is easier to understand and read for stake holders. However, the relation between the information in A3s is not explicitly checked and managed. DoDAF 2.0 [5] describes eight views with corresponding models. In each view, the system is described with a certain focus, e.g. data and information. Together, the views and models describe the architecture, such that stakeholders can more effectively make key decisions. The views are prescribed, and models suggested, but the exact models and their relations are free to choose. SysML [10] is a software oriented visual modeling language to support amongst others the specification, analysis, and design of systems. Multiple diagrams with each a modelling language are described. The software-oriented languages focus on models for realizations, whereas for a system and its KDs also system aspects and their relations are important.

The Geeglee and Contact and Channel approaches can be used to make decisions, based on computation of a quantitative system model. The Geeglee approach [6, 7] prescribes a process in which six score cards are filled and used to identify design challenges and make design decisions. The score cards rely on quantitative reasoning, for which engineers have to determine and provide all values for relations in a predefined format, to perform a computation on which decisions can be based. Instead of providing all values in a fixed format, architects will desire the support of incrementally building up the system information, where they can see the KDs, such that they get insight in design risks and can reason about decisions. The Contact and Channel approach [2] can be used to model the functional behavior of a system, with linked simulation models and visualizations of the design in SysML. A visual description of the system and the mapping of functions in the system is provided. However, this approach focuses on the functional realization and does not consider for example system aspects like cost.

We introduce a KDS in which explicit relations can be used between KDs from different disciplines. The explicit relations enable a combination of qualitative and quantitative reasoning. LEs to describe the KDs of a system are provided to capture and visualize information about the whole system, making the information easily accessible for reasoning and decision making.

3. Language elements and information structure

In this section, the LEs needed to describe the structure and definitions of a system in an IS are introduced. Eight LEs are sufficient to describe a system in an IS.

The eight LEs to describe an IS, are given in Table 1. The LEs can be subdivided in three categories; structural elements, quantification elements, and relational elements. The usage of the LEs in an IS will be illustrated for the bicycle case in Figure 3.

Table 1: Eight LEs, with which an IS can be created.

Language element	Reasoning	Category	Usage
Block	Qualitative	Structural	Creates structure, or qualifies an entity, with its name.
Image			Clarifies or illustrates other LEs.
Text			Annotates or explains a LE.
Parameter	Quantitative	Quantification	Quantifies an aspect from a block.
Model			Makes the relation of a validation or transformation quantitative and explicit. Examples are a mathematical expression, a simulation, executable code, or a loop-up table.
Relation	Qualitative	Relational	A qualitative relation between parameters, blocks, images, or text.
Validation	Quantitative		A comparison of parameter values to satisfy a given condition.
Transformation			A transformation of input parameter values into output parameter values.

The block, image, and text are structural elements, used to create structure in an IS. They can be used to create a kind of hierarchy, with a parent child structure between structural LEs.

The parameter and model are quantification elements that explicitly quantify parts of a system. A parameter quantifies structural elements, e.g. weight, or cost. Typically, a parameter has a unit attached, like “m/s” or “$”. When hard quantification is not possible, a parameter can contain qualitative discrete values like “+”, “heavy”, or “acceptable”, which may not have a unit.

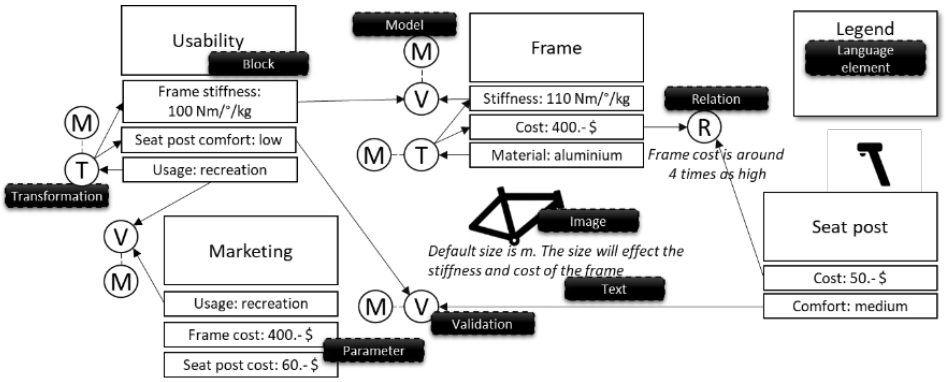

Figure 3: IS for the electric bicycle case from Figure 1.

Relational elements can be used between blocks, images, texts, and parameters. Both a transformation and a validation are defined by a quantitative model. A transformation uses its model to compute the values of its output parameters, given the values of its input parameters. A validation validates values of input parameters by using its model to compare these values. The result of the comparison can be true, false, maybe, or undetermined.

The presented LEs can be used to describe entities, relations, and quantifications of a system in an IS. In an IS, blocks are used to represent entities. Parameters are related to a block, because they quantify an aspect of its entity. Blocks can be related, either directly via qualitative relations, or indirectly via quantitative transformations or validations between their parameters. The relations between entities in an IS make it possible to perform qualitative and quantitative reasoning about the impact of decisions, as will be presented in Section 5.

To describe *qualitative relations* from a system in an IS, blocks and relations are required, where images or texts are used to clarify the relations or blocks. Blocks represent entities of a system, like the "Frame" or the "Usability". Between blocks there can be a qualitative relation, where a text or an image explains the relation.

To describe *quantitative relations* from a system in an IS, blocks, parameters, transformations, validations, and models are required. The quantitative transformation or validation requires the involved blocks to have parameters. The transformation writes values in its output parameters, and thereby keeps the parameter values consistent among the blocks, according to its model. The validation compares parameter values and shows the outcome, according to its model.

To describe a system with both qualitative and quantitative relations in an IS, the eight LEs from Table 1 are required. Qualitative relations are easier to capture and derive, compared to quantitative relations for which values, units, and models are needed. Therefore, typically initially an IS will contain only qualitative relations. Only relations that are unclear or critical should be made quantitative, because quantification requires a significant amount of time and effort. Note that as a step towards quantitative relations, blocks can be quantified with parameters where a qualitative relation is used between these parameters.

We use the term IS, rather than the more commonly used term knowledge graph [11]. For our IS, we use eight distinctive and different LEs, whereas a graph typically consists of a single type of nodes.

In Figure 3, an example of an IS for the electric bicycle case is shown, annotated with black boxes that contain the name of the LEs. The "Marketing", "Usability, "Frame", and "Seat post" KDs are represented by blocks. The qualitative relation is made explicit by the relation between the "Cost" parameters of the "Frame" and "Seat post" KDs. To make quantitative relations explicit, parameters are added to blocks. This is visualized by placing these parameters near their blocks. Values and units are added to the parameters. In the "Usability" KD, the parameter "Usage" has the value "recreation". This value is compared via a validation with the value of "Usage" in the "Marketing" KD and it is used as input for the transformation with which the "Frame stiffness" and "Seat post comfort" values are determined. Models are added to transformations and validation to perform the computation. In a visualization, the colour of a validation shows the result of its comparison, where green is used for true, red for false, orange for maybe, and white for no output.

The example in Figure 3 shows KDs for the bicycle case from Figure 1 for which the information is made explicit by capturing it with LEs. Explicit relations between and inside the "Marketing", "Usability", "Frame", and "Seat post"

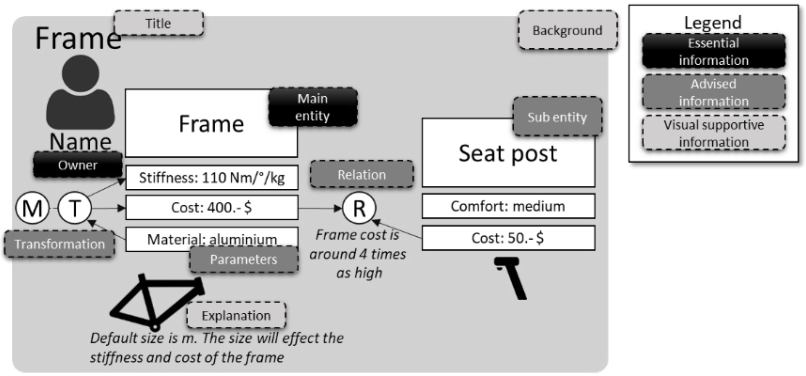

Figure 4: Application of the KD pattern with essential, advised, and visual supportive information, for the "Frame" KD from Figure 3.

KDs are shown. This IS of the bicycle shows its architecture and the parameters on which the owners of the KDs have to agree, for this example related to usage and cost.

The presented IS reduces, or avoids, misunderstanding on the used terminology, definitions, and relations in a system. LEs explicitly describe the information of and relations in a system.

4. Knowledge domain pattern

In this Section, a KD pattern is described that helps to define the scope and to describe essential information of a KD. This pattern can be applied for KDs in an IS. The pattern describes essential, optional and visual supportive information.

The **essential** information for a KD, annotated with black boxes in Figure 4, are a main entity and an owner. A block is used to represent the main entity of a KD, which should have a name that clearly describes the area of a KD. Typically, this name is a noun. The name should clearly reflect the scope of the KD area. With an image and text, the picture and name of the owner for a KD should be included. Explicitly defining ownership helps other users to recognize the responsible person for a KD.

For a KD, the **advised** information, annotated with dark grey boxes in Figure 4, are sub entities, parameters, relations, and transformations. Besides the main entity, additional information and structure for KD can be given by including blocks for sub entities, like the "Seat post" in the given example. With sub entities, a decomposition of the important parts of the main entity can be made, where it is possible to create a nesting of sub entities. The names of sub entities are typically a noun. Parameters can be added to entities to indicate the presence of quantitative or qualitative information about an aspect of the entity, the name of the parameter gives information about the aspect. A value and a corresponding unit can be added to refine the information of the parameter. Entities in a KD should be related, this can be via a relation between entities or via relations or transformation between parameters of entities.

Visual supportive information, annotated with light grey boxes in Figure 4, make a KD easier to understand and thereby supports the reasoning for decisions. An image can be used as a background, to contain all LEs belonging to the KD. This shows the boundary and scope of the KD. Text can be used to show the title of the KD. An explanation of entities in the KD by using text or images, shows the knowledge area and eases clear understanding.

Qualitative reasoning can be performed via relations. For parameters or blocks that have a relation, the owner should decide if their values should be changed due to a decision, via qualitative reasoning. The owner can decide if the impact of a change is acceptable, by tracing the impact of the change via the qualitative relations, with support from annotations in text or images.

Quantitative reasoning about decisions in a KD is enabled by transformations and their models. Quantitative reasoning can be performed by deciding if a change to a parameter is acceptable given its computed output parameter values. To reason on a desired change to an output parameter value of a transformation, a search for acceptable input parameter values is needed.

Figure 4 shows the application of the KD pattern for the "Frame", from Figure 3, which enables reasoning and decision making. By applying the pattern, the "Frame" and "Seat post" entities are included in the KD, such that the

covered knowledge area is explicit. The relations and transformations in this KD can show if there is a tension between parameter values. As an example, assume that the owner has to minimize the cost of the "Frame" whilst maintaining a stiffness above the 100 Nm/°/kg. For the qualitative relation between the "Cost" of the "Frame" and "Seat post", the owner has to reason about the changes to the values. The owner is supported by a quantitative transformation for the less obvious transformation of "Material" into "Cost" and "Stiffness". If the owner would change the value of the "Material" parameter into "steel", the outcome of the transformation may result in an acceptable "Cost" for the "Frame", but a too low "Stiffness" and an unrealistic low "Cost" for the "Seat post". By observing the changes and tracing them, the owner can make decisions for his KD.

The KD pattern sets an explicit scope for the knowledge area, via the name of its entities and the background. If information is unclear, there is an owner that can be contacted. By applying the KD pattern, essential information is described which avoids misunderstanding.

5. Knowledge domain structure

System wide reasoning to make architecture or design decisions, becomes possible by relating KDs explicitly, as discussed in this Section. By applying the KD pattern on all information in an IS, it becomes a KDS. Quantitative and qualitative relations between KDs make it possible to reason about the impact of a change in one KD by tracing its relation to the other KDs, such that a balanced decision can be made that takes trade-offs into account. At the end of this Section, system wide reasoning is explained for the bicycle case.

In a KDS, KDs are related via quantitative validations or qualitative relations. Each KD should have at least a validation or relation with another KD. An isolated KD indicates that something is wrong and that a discussion with its owner is needed. KDs in a system should be related. Besides relating, a validation or relation also decouples the KDs. If an owner of a KD makes a change to a parameter value, it will not change the information in other KDs that are managed by other owners.

A validation between KDs relates offer or target parameters. Validations relate KDs via parameters, where the parameter in one KD sets a target and the parameter in the other KD represents the offer. An example of this is the "Cost" target set by "Marketing" and the corresponding offer from "Frame", in Figure 3. The validation has a model for the comparison of both "Cost" values to show if the offer is sufficient for the target. The model used for a validation should be agreed upon by the involved KDs, therefore it is a contract between the KDs on the comparison. In this case, a green validation indicates agreement between the KDs for the offer and target parameter values.

Different types of KDs can be identified for a system, being component, aspect, and stakeholder KD types. These different KD types contain different information about the system. Note that all can be captured with the KD pattern. A component KD contains information, captured with blocks and parameters, regarding the realization of a part of the system. Typically, this KD offers parameter values towards other KDs. An aspect KD describes an aspect that should be realized by the system, for which typically targets are given towards component KDs, e.g. on cost or usability. A stakeholder KD contains information regarding the stakeholder perspective on the system. From a stakeholder KD there will be targets towards aspect KDs, because stakeholders desire a system where the aspects are such that their goal is achieved as good as possible.

In a KDS, the validations show the level of reached agreement and where there are tensions. The absence of agreement requires either changes in one of the KDs or a new agreement on the model for the comparison. If its owner considers changes for a KD, the impact can be observed for its validations that indicate the agreements with other KDs. If a KD has relations and changes are made, the owner has to agree on the change with the involved KDs, found via the relations. The validations and relations allow tracing the impact of a change over KDs, which makes it possible to decide on a change whilst considering the system wide impact.

The intention of using a KDS is to perform *qualitative* and *quantitative reasoning* on changes at system level, such that agreement between related KDs can be achieved. To realize an architecture, the system architect can use the KDS to determine for which relations discussions with the KD owners are needed to come to an agreement. KD owners can observe and trace the impact of a change towards and also for other KDs. They can trace these changes via quantitative validations or qualitative relations, which enables quantitative or qualitative reasoning, respectively. Because system wide important information is captured and related in a KDS, it can also be used as a form of documentation.

Figure 5 gives a KDS for the electric bicycle case, based on which decisions can be made. The green color for most of the validations between the KDs shows that most KDs reached agreement on the relation of parameter values.

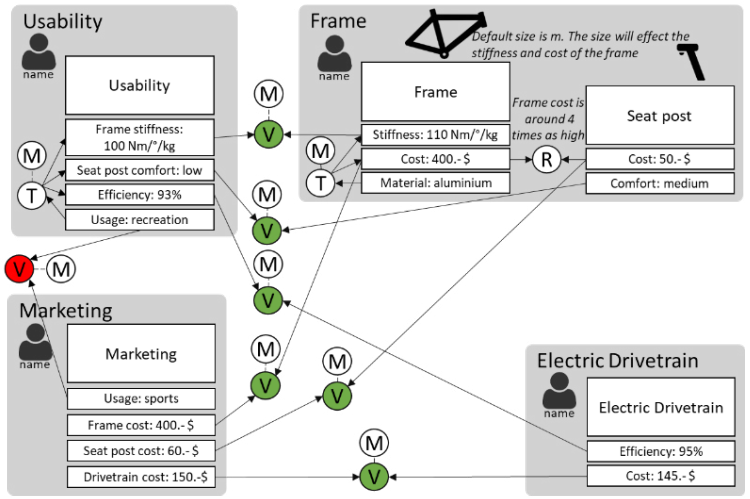

Figure 5: KDS that enables system wide quantitative reasoning and decision making.

However, the red validation, between the "Marketing" and "Usability" KDs, shows the system architect that the KD owners disagree, such that this solution is not possible. As a solution, consider that the owner of the "Usability" KD agrees with the "Marketing" KD owner and changes its "Usage" to "sports". The owner of the "Usability" KD can trace the relations for "Usage" in his or her KD and the validations to other KDs. Inside the KD a transformation is used, which enables quantitative reasoning. The "Usability" owner can change "Usage" and observes the output values of the transformation. As a continuation of the example, the output value for "Frame stiffness" will become 115 Nm/°/kg. Because this is larger than the "Stiffness" parameter value of the "Frame" KD, the corresponding validation will become false, i.e. red. If the "Frame" and "Usability" owners would decide on increasing the "Stiffness", the "Cost" of the "Frame" KD will increase, for example to 500 $. This results in a false validation on "Cost" between the "Marketing" and "Frame" KD. With this information, the architect and owner of the "Marketing" KD can now decide on the trade-off between the "Cost" and "Usage" values. The example shows a tension between the "Cost" and the "Usage" values, where qualitative and quantitative relations make it possible to perform system wide reasoning on the impact of changes. The result of this reasoning can be used to come to a balanced decision, with which the involved KD owners can agree.

6. Conclusions

We presented a knowledge domain structure that enables a combination of qualitative and quantitative system wide reasoning, such that the tensions of changes become visible and decisions can be made. Eight language elements are used to structure the system information in a knowledge domain structure, which we illustrated with an electric bicycle system. In this knowledge domain structure, the knowledge domains are decoupled by validations, where the result of the validation indicates the level of agreement between the knowledge domains.

With the language elements and a knowledge domain pattern, the information of a system can be described in a knowledge domain structure. The knowledge domain pattern describes essential information, such that the scope and owner of the knowledge domain are captured. Relations can be used in a knowledge domain for qualitative reasoning on the impact of a change. Transformations enable quantitative reasoning, because changes to input parameter values of a transformation are propagated to its output parameter values. Knowledge domains can be related via relations and validations. The models that define the validations can be seen as a contract between knowledge domains on the comparison of the parameter values. Via the relations inside and between the knowledge domains, a combination of qualitative and quantitative reasoning can be performed. This approach has been investigated and validated in the Océ professional printing system context, where it was used for system wide reasoning and decision making. An interesting extension is the support for reasoning in large industrial systems with hundreds of knowledge domains, which requires a pattern to further structure information and scale the approach.

Acknowledgements

The research is carried out as part of the Octo+ program under the responsibility of the Embedded Systems Innovation (ESI) group with Océ Technologies B.V. as carrying industrial partner. The Octo+ research is supported by the Netherlands Organisation for Applied Scientific Research TNO.

References

[1] G. J. Muller, CAFCR: A Multi-view Method for Embedded Systems Architecting; Balancing Genericity and Specificity, Delft: Technical University Delft, 2004.

[2] A. Albers and C. Zingel, "Interdisciplinary systems modeling using the contact & channel model for SysML," in *Internation conference on engineering design(ICED)*, 2011.

[3] INCOSE, "A world in motion - Systems engineering vision 2025," International Council on Systems Engineering, 2014.

[4] P. Juzgado, A3 Architecture Overviews: A tool for effective communication in product evolution, Enschede: University of Twente, 2010.

[5] DoD Chief Information Officer, "DoDAF Architrecture Framework 2.02," 12 June 2018. [Online]. Available: https://dodcio.defense.gov/Library/DoD-Architecture-Framework/dodaf20_background/.

[6] M. Jankovic, V. Holley and B. Yannou, "Multiple-Domain Design Scorecards: A method for architecture generation and evaluation through interface characterisation," *Journal of Engineering Design, Taylor and Francis,* vol. 23, no. 10-11, pp. 746-766, 2012.

[7] Geeglee, "Geeglee," 26 March 2018. [Online]. Available: http://www.geeglee.net/.

[8] W. Heemels, v. d. E. Waal and G. Muller, "A multi-disciplinary and model-based design methodology for high-tech systems," in *proceedings of CSER*, 2006.

[9] P. Kruchten, "Architectural Blueprints — The “4+1” View Model of Software Architecture," *IEEE Software,* vol. 12, no. 6, pp. 42-50, 1995.

[10] M. Hause, "The SysML modelling language," in *proceedings of European Systems Engineering Conference*, 2006.

[11] L. Ehrlinger and W. Wöß, "Towards a Definition of Knowledge Graphs," in *proceedings of SEMANTiCS*, Leipzig, Germany, 2016.