TRACING PATHS AND CONNECTING MULTIPLE DESIGN DOMAINS: AN INFORMATION VISUALISATION APPROACH TO PRODUCT ARCHITECTURE MODELLING

Idrissov, Agzam; Parraguez, Pedro; Maier, Anja M.

Technical University of Denmark

ABSTRACT

Visual representation of product architecture models is crucial in complex engineering systems design. However, when the number of entities in a model is large and when multiple levels of hierarchies are included, visual representations currently in use need to be more intuitive. As such, improved visual representations that enable better system overview and better communication of essential product-related information among design participants are needed. This paper uses interactive information visualisation techniques – collapsible hierarchical tree, edge bundling and alluvial diagram – and provides the foundations of a computerised tool that improves the traceability of connections between design domains, including stakeholders, requirements, functions, behaviours and structure. The case of a cleaning robot is used as an illustrative example. The approach supports designers by providing an enhanced overview during the development of complex product architecture models, in particular in the communication with external stakeholders, in the identification of change propagation paths across several design domains, and in capturing the design rationale of previous design decisions.

Keywords: Visualisation, Product architecture, Product modelling / models, Large-scale engineering systems, Requirements

Contact:
Idrissov, Agzam
DTU - Technical University of Denmark
Engineering Systems Group
Denmark
agzid@dtu.dk

Cite this article: Idrissov, A., Parraguez, P., Maier, A.M. (2019) 'Tracing Paths and Connecting Multiple Design Domains: An Information Visualisation Approach to Product Architecture Modelling', in *Proceedings of the 22nd International Conference on Engineering Design (ICED19),* Delft, The Netherlands, 5-8 August 2019. DOI:10.1017/dsi.2019.309

1 INTRODUCTION

Designed systems are often complex, with numerous elements interacting in a "*non-simple way*" (Simon, 1965). When such systems become sufficiently large, designers often struggle to keep an overview and to make sense of all critical relations between the constituent elements (Cross, 2008). To improve overview, enable sense making, and enhance communication with participants in the design process, visual product architecture models are widely used (Bruun and Mortensen, 2012). Such models are abstract representations of systems that capture different product-related aspects of design, e.g. stakeholder requirements, the functions that realise those requirements, the structural components and their physical characteristics - in this paper referred to as *design domains*.

Typically, design domains are hierarchically broken down into smaller subsystems, whereas interactions between them are modelled as links. Traditional visual representations of product architectures include entity-relationship diagrams and matrix representations. While entity-relationship diagrams are straightforward to construct, their readability suffers drastically due to visual clutter introduced by numerous edge crossings (Maier *et al*., 2014), when the number of elements and connections between them becomes large. In contrast, matrix representations address this problem by providing a concise and structured view on connections between the elements (Eppinger and Browning, 2012; Keller *et al*., 2009). However, compared to the entity-relationship diagrams, they are less suitable for communication with external stakeholders, especially when multiple design domains are involved, where each domain consists of several levels of hierarchy. In sum, current visual representations of complex product architecture models are compromised.

This paper conceptualises a novel visualisation approach to support modelling and analysis of complex product architectures in engineering systems design. As original contribution, this paper introduces and applies techniques from the field of Information Visualisation to Engineering Design, i.e. collapsible trees, edge bundling and alluvial diagrams to improve overview, sense making, and communication between design participants. The visualisation approach is envisaged to support the following design tasks: individual and shared building of product architecture models, navigation between different design domains and levels of hierarchy within the model, identification of change propagations paths, and tracking of rationale of previous design decisions.

The remainder of the paper is structured as follows. Section 2 discusses the literature background on visual product architecture modelling. Section 3 proposes techniques from Information Visualisation to link design domains in product architecture modelling and applies the approach using a cleaning robot as an illustrative example. Section 4 discusses the contribution of linking design domains through Information Visualisation techniques. Section 5 concludes and discusses further research directions.

2 BACKGROUND

Maier and Rechtin define systems as "*collections of different things that together produce results unachievable by the elements alone*" (Maier and Rechtin, 2000). Thus, the primary goal of a system architect - the term is here used interchangeably with designer - is to establish that a system, with all interrelationships between its elements, works as a whole as intended. System architects ensure that the final combination of subsystems and components have compatible interfaces and satisfy all the necessary stakeholder requirements.

Moreover, system architects are equally responsible for critical and sometimes very fine details that may undermine the overall performance of a system (Maier and Rechtin, 2000). Failure to consider such critical details may lead to grave consequences for the whole system. For instance, in September 2009, Toyota had to recall 3,8 million vehicles due to incidents of gas pedal sticking to the floor mat and causing unintended dangerous acceleration (Gu, 2010). This case is illustrative from a systems architecting perspective, as the problem was caused by a mismatch between two seemingly unrelated subsystems - the function of a floor mat is hygienic, while the gas pedal serves the purpose of acceleration of the whole vehicle. This illustrates the need for system architects not only to have an overview about the system but also to be supported in their ability to concentrate on critical and concrete details at the lower levels of system abstraction.

2.1 Product architecture modelling

Shannon's definition of a model states "*a model is a representation of an object, system or idea in some other form than itself*" (Shannon and Johannes, 1976). The creation of product architecture models improves problem-solving in design by making knowledge surrounding the product explicit, enabling shared understanding of the system between multiple design participants and reducing complexity by providing abstractions for sophisticated concepts (Hehenberger, 2014; Maier and Rechtin, 2000).

One of the modelling approaches in product development is *Function-Behaviour-Structure (or State)* (FBS) modelling (Gero and Kannengiesser, 2004; Qian and Gero, 1996; Umeda *et al*., 1990). In essence, FBS modelling aims to identify the relations between functional, behavioural and structural domains of design. While the functional domain subjectively describes the purpose of the design object, structural domain descriptions include representations of concrete components. Functional and structural domains are connected through the behavioural domain, which expresses physical phenomena behind the actual behaviour of entities in the structural domain (Van Beek and Tomiyama, 2008; Hamraz *et al*., 2012; Koh, 2017).

Taking FBS modelling as a base, several works have focused on its extension to include more aspects of the design process and to allow for a more holistic overview of the product architecture. Ahmad *et al*. (2010) propose to organise information in requirements, functional, component and detailed design domains by using *Design Structure Matrix* (*DSM)* representations (Eppinger and Browning, 2012) to manage engineering changes. Van Beek and Tomiyama (2012) propose to include design domains that capture stakeholders and their requirements and provide a rationale on how to connect these domains to FBS modelling. In Hamraz *et al*. (2012, 2014) a change prediction method (Clarkson *et al*., 2004) is combined with FBS and DSMs to identify change propagations.

As developed systems can be rather complex, to reduce the complexity of models, FBS modelling schemes have adopted hierarchical decompositions of design domains (Hehenberger, 2014). For instance, in the functional domain of a cleaning robot, "to *clean floors*" is the highest level function, which is decomposed into three sub functions: "to *release clean air*", "to *collect dust*" and "to *move itself*" (example adapted from Habib and Komoto, 2014). Each of these sub functions can then be decomposed into further lower level functions. Similarly, behavioural and structural domains can be broken down into lower levels and represented as a hierarchy. An example of hierarchical decomposition in the structural domain would be the part structure of a cleaning robot with moving and cleaning subsystems, where the latter is further decomposed into dust sensor, brush, fan and so on. Van Beek and Tomiyama (2012) connect customer requirements with hierarchical decomposition of FBS to support design of magnetic resonance imaging system. Habib and Komoto (2014) perform hierarchical FBS decomposition to find similarities and differences among existing products to design next generation product families.

With respect to computational design support for product architecture modelling, several requirements for design support tools were outlined in Van Beek *et al*. (2010), Van Beek and Tomiyama (2008) and Alvarez Cabrera *et al*. (2011). They include:

- to provide a "birds eye" overview of product architecture capturing relevant design domains,
- to allow effortless change between levels of abstraction – from high level system descriptions to low level concrete details,
- to perform traceability between stakeholder requirements and designed functions and embodying structures, and
- to allow navigation according to how architects imagine product architecture in their minds for shared understanding of constructed models.

However, even with the proliferation of computerised design support tools, it is not trivial to represent complex product architectures. The combination of multiple domains, each decomposed into multiple hierarchical layers introduces additional challenges of appropriate representation in design support tools. While seeing the link between the function and associated behaviour is trivial, as these domains are close to each other, tracking relations between stakeholders and related structural components across multiple design domains may result in unnecessary cognitive load for the user. In addition, there may be multiple levels of hierarchy even between the neighbouring domains (e.g. between higher level functions and lower level structural domain entities) which makes it hard to track these linkages.

2.2 Current representations of product architecture models

Traditionally, a multi-domain product architecture has been represented as a vertical hierarchical entity-relationship diagram (Alvarez Cabrera *et al*., 2011; Mortensen *et al*., 2008; Umeda *et al*., 1990, 1996). While being intuitively clear for simpler products, these types of diagrams become visually cluttered and hard to perceive as the number of entities and interconnections between them grows. Moreover, tracking connections across several domains becomes challenging when each domain is decomposed into multiple levels of hierarchy.

Traditionally, UML diagrams (Booch *et al*., 1999) and later, SysML diagrams (Peak *et al*., 2007) were adopted in product development to visualise connections between system entities using software tools, such as UNICOM System Architect or MagicDraw. However, tracing relationships within diagrams and across different domains becomes challenging for complex products (Chandrasegaran *et al*., 2013), as users have to constantly switch between different detailed views, while overall system views do not explicitly highlight connections across several domains.

The *Design Structure Matrix* (DSM) (Steward, 1981) is a squared matrix, where columns and rows contain two types of connected entities and their intersection represents the nature of that connection. Used for tracking component-component connections (Pimmler and Eppinger, 1994), this visualisation was later extended to represent dependencies between people, organisations and processes (Eppinger and Browning, 2012). By combining several domains into one DSM, a multi-domain matrix (MDM) (Maurer and Lindemann, 2008) can be built to represent dependencies between multiple design domains. Nonetheless, studies have shown that it might be challenging to understand and construct DSMs and especially MDMs for complex product architectures. In a study by Keller *et al*. (2005), fish-eye network visualisations (Furnas, 1986) are used to display change propagations between components of a helicopter, as designers "*seem to prefer a network representation, rather than a DSM*" or normally "*create a network first and then transformed it into a DSM*". Although this study concentrated on the structural domain, it is safe to assume that visualising connections across other domains will lead to difficulties in understanding a designed system. Additionally, according to Ghoniem *et al*. (1997), for path-finding tasks, network representations perform better than matrix representations. Another study by Novick and Hurley (2001) argues that network diagrams are inherently better suited to illustrate hierarchical structure, which is hard to capture by a matrix representation, although there are adaptations of DSMs for hierarchical information (Eppinger and Browning, 2012).

Therefore, a unifying view of product architecture that would allow tracking of connections between entities throughout several domains and levels of hierarchy is required. Ideally, such a visual representation guides users during product architecture model construction and provides a history of design decisions for future reference. To address these issues, we propose to apply techniques from the field of *Information Visualisation (InfoVis)*, which is a branch of Human-Computer Interaction that focuses on "*computer-supported, interactive, visual representations of data to amplify cognition*" (Card *et al*., 1999). In the next section, we combine three InfoVis techniques – alluvial diagram, collapsible hierarchical tree, and edge bundling – to visualise product architectures. It is then discussed, how such an approach addresses the abovementioned representation challenges and sketches potential applications in engineering systems design.

3 USING INFORMATION VISUALISATION TO CONNECT AND TRACK CONNECTIONS BETWEEN DESIGN DOMAINS

3.1 Connecting design domains

In this paper, we consider a product architecture model that focuses on five design domains presented in Figure 1: stakeholders, stakeholder requirements, functional, behavioural and structural domains. This representation schema is an extended version of an FBS product architecture model and is motivated by works of Komoto and Tomiyama (2010), Ahmad, Wynn and Clarkson (2010), Van Beek and Tomiyama (2012), Habib and Komoto (2014). These studies, in addition to functional, behavioural and structural domains, discuss another two domains: stakeholders and stakeholder requirements. An explanation for the inclusion of these domains is provided below. Then, the approach proposed here explicitly links these five design domains, integrating their hierarchical decompositions and design decision history in the form of design rationale to improve traceability and holistic understanding of a product architecture.

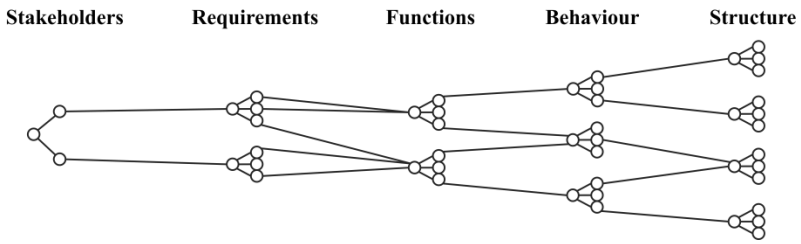

Figure 1. Linking design domains of a product architecture model

3.1.1 Stakeholder requirements

Interdisciplinary teams of designers often jointly design systems with other stakeholders (e.g. customers, financial or legal teams). Thus, the stakeholder requirements domain acts as an overall map that guides the design process by constantly reminding the designers about the purpose of the system. Besides the requirement itself, requirement nodes may contain links to images, documents, video/audio interviews with additional explanations for the requirement in detail. Such modelling of stakeholder requirements "*facilitates design team meetings and discussions by acting as an explicit and understandable medium for all stakeholders*" (Van Beek and Tomiyama, 2012).

3.1.2 Stakeholders

Seeing not only stakeholder requirements but also the people and organisations that initiated said and often conflicting requirements provides designers with better insights. Adapting an example from Crawley *et al*. (2015), in the design of a hybrid car, marketing departments might require a car of low cost, customers need power to carry cargo, and the government may impose environmental regulations on the amount of produced emissions. Thus, by tracing a specific requirement back to its stakeholder, system architects can avoid incompatible design decisions and produce better overall system designs (Crawley *et al*., 2015).

3.1.3 Hierarchical decomposition and linkage between design domains

Similar to FBS, each domain of the model is hierarchically decomposed into the lowest adequate level suitable for building and redesigning the product. For instance, for larger systems it would be impractical to decompose standard components into individual parts. Stakeholder decomposition is performed similar to an organisational hierarchy; for example organisations are divided into departments, departments into divisions, divisions consist of people. Stakeholder requirements are decomposed into sub requirements; for example, the "*easy to use*" customer requirement for a cleaning robot design can be further decomposed into sub requirements, such as long battery lifetime, small form-factor, or intuitive controls.

Mapping the stakeholder domain to the requirements domain is trivial: requirements are linked to the stakeholders that initiated them. Then, stakeholder requirements are mapped to the product architecture using Quality Functional Deployment (QFD) (Akao *et al*., 1990) or Axiomatic Design (AD) (Suh, 1990). Finally, mapping between functions, behaviour and structure domains is addressed by FBS Linkage (Hamraz *et al*., 2014). What is lacking, however, is a visual representation of such linkages for complex systems that would address the requirements outlined in Section 2.

3.1.4 Design rationale

A common criticism of hierarchical functional modelling approaches is that during functional decomposition, designers tend to choose temporary design solutions and subsequently improve them which leads to information overload (Fiorineschi *et al*., 2016). This issue is addressed by DRed tool (Aurisicchio and Bracewell, 2013; Bracewell *et al*., 2009). The design rationale approach allows designers to deconstruct their decision making process by keeping a diagram that shows why they have chosen a particular design solution and what the alternatives were.

While we have described an example of a model that aims to provide a holistic understanding of a product architecture, in this paper, our focus is on the visual representation of complex product architecture models that contain decomposable and interlinking hierarchies. Although hierarchical decomposition and linkages between design domains may be performed differently than in FBS (e.g.

via function-means tree (Hansen and Andreasen, 2002), the aim of this paper is to demonstrate how to represent these models in a visual way.

3.2 Applying information visualisation techniques

To address the needs for a design support tool outlined above, we propose to employ interactive information visualisations to represent product architecture for complex systems. For each design domain, collapsible tree visualisations are constructed, where nodes represent entities of that design domain and edges represent connections between entities within one and simultaneously also between several design domains. The purpose of collapsible tree visualisations is to exhibit hierarchies across all design domains and it allows switching between different levels of hierarchy within each domain. To avoid edge crossings in complex models, edge bundling algorithms cluster similar edges together and improve readability of the visualisation. Then, the alluvial diagram highlights the path from the node of interest to all the connected nodes and edges across all design domains. In this way, design participants can observe connections between entities across all domains, even the ones that are distant.

3.2.1 Collapsible tree diagrams

While representing an ontology in an entity-relationship diagram is a popular technique, when the number of entities and links between them is sufficiently large, the display becomes cluttered with numerous intersecting links which makes it hard to read (Maier *et al*., 2014). The study by Plaisant *et al*. (2002) suggests that the *collapsible tree visualisation*, where the user can choose which entities to expand, while shrinking the rest, is well suited for both tree navigation and topology overview tasks. An example of a collapsible tree for the requirements domain is presented in Figure 2. When clicking on the "*non-functional requirements*" node, all the related sub requirements will be hidden (e.g. "*lightweight*", "*safe to use*", etc.). This lets designers choose an adequate level of abstraction and can support a large number of entities (Heer and Card, 2004).

3.2.2 Edge bundling

One of the ways to overcome visual edge clutter is *edge bundling* - a visualisation technique that clusters similar edges together, while preserving all connections between entities (Holten and Van Wijk, 2009; Zhou *et al*., 2013). Instead of drawing a direct edge between entities, edge bundling algorithm groups edges so that edge crossings are minimised. Figure 2 illustrates an example of such and edge bundling technique for representation of connections between stakeholders and their requirements. Similarly, edge bundling helps to minimise edge crossings when displaying connections between other domains.

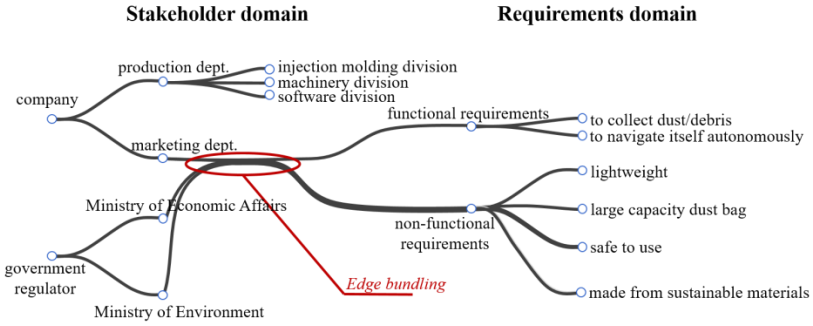

Figure 2. Edge bundling of inter-domain connections

3.2.3 Alluvial diagrams

Interactive *alluvial diagrams* are useful for highlighting and summarising structural changes while concisely providing overview in complex systems (Rosvall and Bergstrom, 2010). Taking as an example the model presented in Figure 3: Let us assume that a designer wants to know what parts of a cleaning robot are related to the "to *suck air and dust*" function in the functional domain. When the designer hovers over the entity of interest (e.g. the "to *suck air and dust*" node), all related entities in other domains are highlighted accordingly: e.g. the "*marketing dept.*" entity in the stakeholder domain, the "to *collect dust/debris*" requirement in the requirements domain, the "*air flow*" entity in the behavioural domain, and the "*suction fan*" and the "*suction motor*" entities in the structural domain.

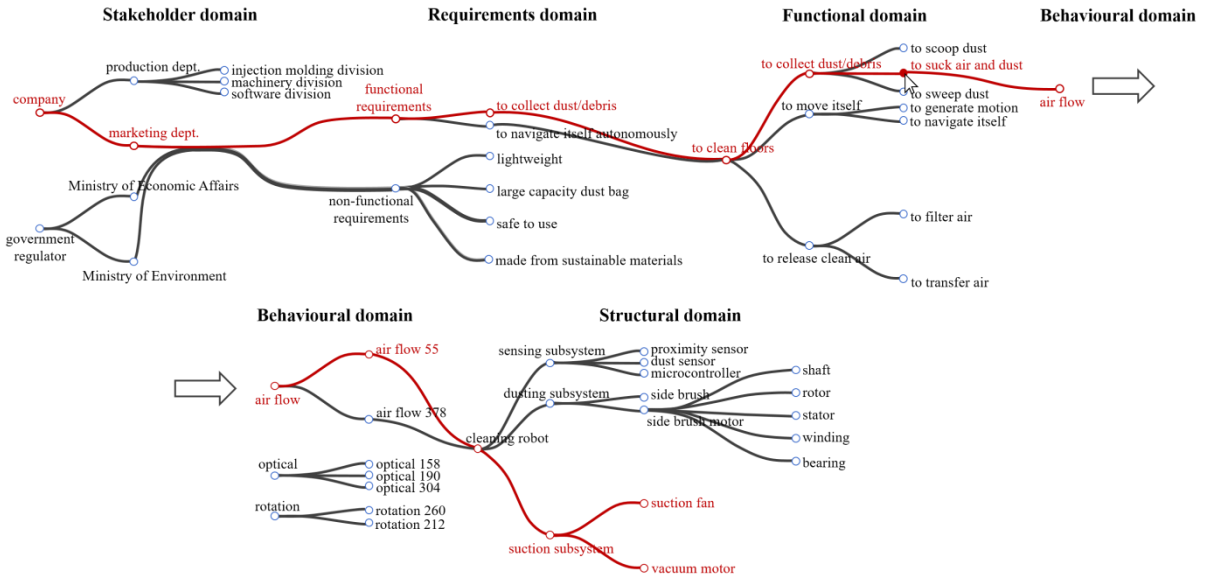

Figure 3. Alluvial diagrams to highlight path connections

In summary, by combining these three visualisation techniques, we aim to decrease the visual clutter that emerges when displaying complex product architectures. While a collapsible tree diagram represents complex hierarchies within a product architecture, interactive alluvial diagrams are designed to highlight connections between entities throughout several distant levels of hierarchies, edge bundling displays all related connections reducing the number of undesired edge crossings.

3.3 Features of the proposed visualisation approach

In this section, we discuss how the proposed visualisation approach could address the requirements for visual representations outlined in Section 2.

3.3.1 Guided model construction

First, similar to the suggestion of Van Beek and Tomiyama (2012), using market research or stakeholder interviews, system architects, together with marketing and other related departments, start by constructing stakeholder and stakeholder requirements domains. Once hierarchical decompositions for the domains of stakeholder and stakeholder requirements are constructed, interdisciplinary design teams jointly perform FBS decomposition linkages and connect them to the stakeholder requirements domain. Although in Van Beek and Tomiyama (2012), stakeholder requirements are directly connected to leaf nodes of the functional domain, to minimise edge crossings, we propose to use edge bundling (Holten and Van Wijk, 2009) and connect entities only through parent nodes.

As the design progresses, designers add more detailed functions and concrete modules and structural parts. Depending on the domain of interest, whole domains can intently be hidden from the visualisation. This way, design decisions at various levels of detail are constantly discussed, depending on the maturity of the project and technical expertise of the stakeholders.

3.3.2 Improved traceability between design domains

When entities are distant from each other's design domains (e.g. the stakeholder domain and structure domain are three domains apart), relationships between them become hard to track. This problem is especially evident when each domain consists of multiple levels of hierarchy. When the user hovers the mouse over an element of interest, the sequence of edges and nodes of an alluvial diagram accentuates the related entities and connections between them. In this way, non-obvious interactions between elements across all the design domains can be discovered and managed. While general traceability frameworks for design were proposed earlier (Martinec and Pavković, 2014; Pavković *et al*., 2013), the focus of this paper is on the visual traceability across design domains.

Moreover, we note that such a visual representation scheme is not limited to only tree-like structures, as a structural element may realise several functions at once. In this case, connections between structural elements and the related functions are linked through parent nodes.

3.3.3 Shared modelling of the design process

When dealing with complex products, it is hard for designers to keep track of all possible effects of changes on functions, behaviour and components, especially outside of their core expertise. By providing a collaborative design functionality to the tool, design teams can work on their respective subsystems across all domains, so that the system architect can have a bird's eye view on the system, while having access to the detailed views as well. Nodes of a collapsible tree diagram show additional information about system entities on-demand: e.g. documents describing requirements in detail or formulae for physical behaviour or technical drawings of components in the structural domain. Having a common frame of reference guides not only the system architect, but also members of interdisciplinary design teams who may want to explore how other subsystems are constructed.

3.3.4 Design rationale integration

Entities in the structural domain contain not only the descriptions of the structural components but also a rationale about how and why these components were chosen over the other candidates. For instance, suppose that in order to embody the "*to move*" function during the design of a cleaning robot, a designer chose to use wheels instead of continuous tracks due to weight requirements. However, if the requirements change in the future and the robot's ability to drive over obstacles becomes a priority over the robot's weight, system architects can see that continuous tracks are available as an alternative design solution and a justification on why this concept was not chosen in the first place. Similarly, by showing such minified "*history"* of design decisions for each node, designers can later come back and re-evaluate other possible design decisions. Moreover, if such knowledge is available, component entities may indicate whether the chosen component is standard or custom, which helps system architects to assess the overall technological complexity of the proposed solution.

4 CONCLUSION

In the proposed approach, we display product architecture as hierarchically decomposed entities of stakeholders, requirements, functional, behavioural and structural domains. We introduce a visual representation of those domains and exemplify with an example of a cleaning robot using the combination of three information visualisation techniques: collapsible tree diagram, edge bundling, alluvial diagram. First, collapsible tree diagrams enable users to quickly transition between different levels of abstraction. Second, edge bundling techniques reduce visual clutter by grouping together connections and minimising edge crossings. Third, alluvial diagrams allow highlighting relationships between entities that are indirectly connected to each other across multiple domains.
Besides the evaluation of connecting and tracing between design domains, a natural continuation of this work includes intuitive visualisation of other design aspects, for instance, interfaces between objects and modules, information about the manufacturing processes and supply chain, as suggested for further work, e.g. in Mortensen *et al*. (2008). Similarly to other functional modelling approaches, the approach proposed here requires time to build an initial model (Hamraz *et al*., 2012). Nonetheless, once the first model is built, the adaptation of the model to other products in a product family requires much less effort. We also assume the availability of a knowledge-base, which is needed for building such a model.
While further evaluation is necessary, this novel approach is proposed to support shared model building and sense making of complex product architectures. When the number of observed entities in a system is large, the approach allows designers and system architects to trace connections between design domains. Moreover, capturing design rationale enables designers to iterate over the solution space. Compared to other discussed visual representations of product architecture models, this type of visual representation is closer to the conventional mental model of designers and can be conveniently presented to external stakeholders for further discussion and validation of design decisions.

ACKNOWLEDGEMENTS

This research has partially been funded by the H2020 EU Framework Programme for Research and Innovation through the EURITO project under Grant Agreement n° 770420. We would also like to thank Pelle Willumsen and the anonymous reviewers for their insightful comments.

REFERENCES

Ahmad, N., Wynn, D. and Clarkson, P. (2010), "Development and Evaluation of a Tool to Estimate the Impact of Design Change", Design 2010, pp. 105–116, https://doi.org/10.1186/s13018-017-0583-2.

Akao, Y., King, B. and Mazur, G. (1990), "Quality Function Deployment: Integrating Customer Requirements into Product Design", Productivity press., Cambridge, MA.

Alvarez Cabrera, A.A., Woestenenk, K. and Tomiyama, T. (2011), "An architecture model to support cooperative design for mechatronic products: A control design case", *Mechatronics, Pergamon*, Vol. 21 No. 3, pp. 534–547, https://doi.org/10.1016/J.MECHATRONICS.2011.01.009.

Aurisicchio, M. and Bracewell, R. (2013), "Capturing an integrated design information space with a diagram-based approach", *Journal of Engineering Design*, Taylor & Francis, Vol. 24 No. 6, pp. 397–428, https://doi.org/10.1080/09544828.2012.757693.

Van Beek, T.J., Erden, M.S. and Tomiyama, T. (2010), "Modular design of mechatronic systems with function modeling", *Mechatronics, Pergamon*, Vol. 20 No. 8, pp. 850–863, https://doi.org/10.1016/j.mechatronics.2010.02.002.

Van Beek, T.J. and Tomiyama, T. (2008), "Connecting views in mechatronic systems design, a function modeling approach", 2008 IEEE/ASME International Conference on Mechatronics and Embedded Systems and Applications, MESA 2008, Vol. 31, pp. 164–169, https://doi.org/10.1109/MESA.2008.4735676.

Van Beek, T.J. and Tomiyama, T. (2012), "Structured workflow approach to support evolvability", *Advanced Engineering Informatics*, Elsevier Ltd, Vol. 26 No. 3, pp. 487–501, https://doi.org/10.1016/j.aei.2012.05.003.

Booch, G., Jacobson, I. and Rumbaugh, J. (1999), "The Unified Modeling Language Reference Manual", https://doi.org/10.1017/CBO9781107415324.004.

Bracewell, R., Wallace, K., Moss, M. and Knott, D. (2009), "Capturing design rationale", *CAD Computer Aided Design*, Elsevier, Vol. 41 No. 3, pp. 173–186, https://doi.org/10.1016/j.cad.2008.10.005.

Bruun, H.P.L. and Mortensen, N.H. (2012), "Visual product architecture modelling for structuring data in a PLM system", *IFIP AICT - Advances in Information and Communication Technology*, Vol. 388, pp. 598–611, https://doi.org/https://doi.org/10.1007/978-3-642-35758-9_54.

Card, S.K., Mackinlay, J. and Shneiderman, B. (1999), "Readings in Information Visualization: Using Vision to Think", Morgan Kaufmann Publishers.

Chandrasegaran, S.K., Ramani, K., Sriram, R.D., Horváth, I., Bernard, A., Harik, R.F. and Gao, W. (2013), "The evolution, challenges, and future of knowledge representation in product design systems", *CAD Computer Aided Design*, Vol. 45 No. 2, pp. 204–228, https://doi.org/10.1016/j.cad.2012.08.006.

Clarkson, J., Simons, C. and Eckert, C. (2004), "Predicting change propagation in complex design", *Journal of Mechanical Design*, Vol. 126 No. 5, pp. 788–797.

Crawley, E., Cameron, B. and Selva, D. (2015), "System Architecture: Strategy and Product Development for Complex Systems", Prentice Hall Press.

Cross, N. (2008), "Engineering Design Methods: Strategies for Product Design", Wiley, https://doi.org/10.1016/0261-3069(89)90020-4.

Eppinger, S.D. and Browning, T.R. (2012), "Design Structure Matrix Methods and Applications", Vol. 1, MIT Press.

Fiorineschi, L., Rotini, F. and Rissone, P. (2016), "A new conceptual design approach for overcoming the flaws of functional decomposition and morphology", *Journal of Engineering Design*, Vol. 27 No. 7, pp. 438–468, https://doi.org/10.1080/09544828.2016.1160275.

Furnas, G.W. (1986), "Generalized Fisheye Views", *CHI*, https://doi.org/10.1145/22627.22342.

Gero, J.S. and Kannengiesser, U. (2004), "The situated function-behaviour-structure framework", *Design Studies*, Vol. 25, pp. 373–391, https://doi.org/10.1016/j.destud.2003.10.010.

Ghoniem, M., Fekete, J.D. and Castagliola, P. (1997), "On the readability of graphs using node-link and matrix-based representations: A controlled experiment and statistical analysis", *Information Visualization*, Vol. 4 No. 2, pp. 114–135, https://doi.org/10.1057/palgrave.ivs.9500092.

Gu, X. (2010), "Toyota Recalls : Revealing the Value of Secure Supply Chain", *Massachusetts Institute of Technology*.

Habib, T. and Komoto, H. (2014), "Comparative analysis of design concepts of mechatronics systems with a CAD tool for system architecting", *Mechatronics, Pergamon*, Vol. 24 No. 7, pp. 788–804, https://doi.org/10.1016/j.mechatronics.2014.03.003.

Hamraz, B., Caldwell, N.H.M. and John Clarkson, P. (2012), "A Multidomain Engineering Change Propagation Model to Support Uncertainty Reduction and Risk Management in Design", *Journal of Mechanical Design*, Vol. 134 No. 10, pp. 100905–100905–14, https://doi.org/10.1115/1.4007397.

Hamraz, B., Caldwell, N.H.M., Ridgman, T.W. and Clarkson, P.J. (2014), "FBS Linkage ontology and technique to support engineering change management", *Research in Engineering Design*, https://doi.org/10.1007/s00163-014-0181-9.

Hansen, C.T. and Andreasen, M.M. (2002), "Two approaches to synthesis based on the domain theory", *Engineering Design Synthesis*, Springer London, London, pp. 93–108, https://doi.org/10.1007/978-1-4471-3717-7_6.
Heer, J. and Card, K.S. (2004), "DOITrees Revisited: Scalable, Space-Constrained Visualization of Hierarchical Data", *Advanced Visual Interfaces*, pp. 421–424, https://doi.org/10.1145/989863.989941.
Hehenberger, P. (2014), "Perspectives on hierarchical modeling in mechatronic design", *Advanced Engineering Informatics*, Elsevier, 1 August, https://doi.org/10.1016/j.aei.2014.06.005.
Holten, D. and Van Wijk, J.J. (2009), "Force-Directed edge bundling for graph visualization", *Computer Graphics Forum*, Vol. 28 No. 3, pp. 983–990, https://doi.org/10.1111/j.1467-8659.2009.01450.x.
Keller, R., Eckert, C.M. and Clarkson, P.J. (2005), "Multiple views to support engineering change management for complex products", CMV 2005, Vol. 2005, IEEE, pp. 33–41, https://doi.org/10.1109/CMV.2005.11.
Keller, R., Eckert, C.M. and Clarkson, P.J. (2009), "Using an engineering change methodology to support conceptual design", *Journal of Engineering Design*, Vol. 20 No. 6, pp. 571–587, https://doi.org/10.1080/09544820802086988.
Koh, E.C.Y. (2017), "A study on the Requirements to Support the Accurate Prediction of Engineering Change Propagation", *Systems Engineering*, Vol. 20 No. 2, pp. 147–157, https://doi.org/10.1002/sys.21385.
Komoto, H. and Tomiyama, T. (2010), "A system architecting tool for mechatronic systems design", *CIRP Annals - Manufacturing Technology*, Vol. 59 No. 1, pp. 171–174, https://doi.org/10.1016/j.cirp.2010.03.104.
Maier, A.M., Baltsen, N., Christoffersen, H. and Störrle, H. (2014), "Towards Diagram Understanding: A Pilot-Study Measuring Cognitive Workload Through Eye-Tracking", *Proceedings of International Conference on Human Behaviour in Design 2014*, No. October, pp. 1–6.
Maier, M. and Rechtin, E. (2000), "The Art of Systems Architecting", CRC Press.
Martinec, T. and Pavković, N. (2014), "Visualization of information traceability in product development", *Proceedings of International Design Conference, DESIGN*, Vol. 2014–Janua, pp. 1831–1842.
Maurer, M. and Lindemann, U. (2008), "The application of the Multiple-Domain Matrix: Considering multiple domains and dependency types in complex product design", *2008 IEEE International Conference on Systems, Man and Cybernetics, IEEE*, pp. 2487–2493, https://doi.org/10.1109/ICSMC.2008.4811669.
Mortensen, N.H., Hvam, L., Pedersen, R. and Kvist, M. (2008), "Modelling and visualising modular product architectures for mass customisation", *Int. J. Mass Customisation*, Vol. 2, pp. 216–239.
Novick, L.R. and Hurley, S.M. (2001), "To Matrix, Network, or Hierarchy: That Is the Question", *Cognitive Psychology*, https://doi.org/10.1111/ajph.12083.
Pavković, N., Štorga, M., Bojčetić, N. and Marjanović, D. (2013), "Facilitating design communication through engineering information traceability", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 27 No. 2, pp. 105–119, https://doi.org/10.1017/S0890060413000012.
Peak, R.S., Burkhart, R.M., Friedenthal, S.A., Wilson, M.W., Bajaj, M. and Kim, I. (2007), "Simulation-Based Design Using SysML - Part 1: A Parametrics Primer", *INCOSE Intl. Symposium*.
Pimmler, T.U. and Eppinger, S.D. (1994), "Integration analysis of product decompositions", *ASME 6th Design Theory and Methodology Conference*, pp. 343–351, https://doi.org/10.1016/j.ijpe.2011.01.023.
Plaisant, C., Grosjean, J. and Bederson, B.B. (2002), "SpaceTree: Supporting exploration in large node link tree, design evolution and empirical evaluation", *IEEE Symposium on Information Visualization, IEEE Comput. Soc*, pp. 57–64, https://doi.org/10.1109/INFVIS.2002.1173148.
Qian, L. and Gero, J.S. (1996), "Function–behavior–structure paths and their role in analogy-based design", *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, Vol. 10 No. 04, p. 289, https://doi.org/10.1017/S0890060400001633.
Rosvall, M. and Bergstrom, C.T. (2010), "Mapping change in large networks", edited by Rapallo, F. *PLoS ONE, Public Library of Science*, Vol. 5 No. 1, p. e8694, https://doi.org/10.1371/journal.pone.0008694.
Shannon, R. and Johannes, J.D. (1976), "Systems Simulation: The Art and Science", *IEEE Transactions on Systems, Man, and Cybernetics*, https://doi.org/10.1109/TSMC.1976.4309432.
Simon, H.A. (1965), "The architecture of complexity", *General Systems*, Vol. 10, pp. 63–76.
Steward, D. V. (1981), "Design Structure System: a Method for Managing the Design of Complex Systems.", *IEEE Transactions on Engineering Management*, Vol. EM-28 No. 3, pp. 71–74, https://doi.org/10.1109/TEM.1981.6448589.
Suh, N.P. (1990), "The principles of design", *Oxford University Press on Demand*, Vol. 6.
Umeda, Y., Ishii, M., Yoshioka, M., Shimomura, Y. and Tomiyama, T. (1996), "Supporting conceptual design based on the function-behavior-state modeler", *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, Vol. 10 No. 04, p. 275, https://doi.org/10.1017/S0890060400001621.
Umeda, Y., Takeda, H., Tomiyama, T. and Yoshikawa, H. (1990), "Function, behaviour, and structure", *Applications of Artificial Intelligence in Engineering V*, Vol. 1, pp. 177–193, https://doi.org/10.2307/1235767.
Zhou, H., Xu, P., Yuan, X. and Qu, H. (2013), "Edge bundling in information visualization", *Tsinghua Science and Technology*, Vol. 18 No. 2, pp. 145–156, https://doi.org/10.1109/TST.2013.6509098.