Research in Engineering Design (2019) 30:339–361
https://doi.org/10.1007/s00163-018-00306-0

ORIGINAL PAPER

Sharing design definitions across product life cycles

Amar Kumar Behera[1] · Alison McKay[2] · Christopher F. Earl[4] · Hau Hing Chau[2] · Mark A. Robinson[3] · Alan de Pennington[2] · David C. Hogg[5]

Received: 17 May 2017 / Revised: 16 August 2018 / Accepted: 24 December 2018 / Published online: 19 January 2019

Abstract
The research reported in this paper explored the feasibility of embedding multiple design structures into design definitions with a view of sharing design definitions across product life cycles. Two separate case studies using (a) lattice theory and (b) a qualitative data analysis (QDA) software tool were used to illustrate the benefits of embedding. In the first case study, of a robotic arm assembly, lattices in the form of partially ordered sets are used to embed multiple design structures into a given design definition. A software prototype has been built that allows a design bill of materials (BoM) to be extracted from a STEP AP214 file and translated into a lattice that is visualized as a Hasse diagram. This lattice is a sub-lattice of a complete lattice that includes all possible BoM structures for the given collection of component parts in the assembly. New BoM design structures can be defined by selecting the required nodes in the complete lattice and alternative product definitions are then exported as new STEP files. The second case study introduces a collision avoidance robot with associated design structures. It is used to illustrate management of design information using a current technique, design structure matrix (DSM), and compared with how embedding using QDA has the potential to support the establishment of relationships between design structures. Results from these case studies demonstrate that it is feasible to use lattice theory as an underlying formalism and QDA as a means for sharing design definitions.

Keywords Lattice structures · Qualitative data analysis · Bill of materials · Design definitions · BoM management · Robots

1 Introduction

The process of designing using commercial engineering design tools results in multiple design definitions that are shared within product development teams and across supply networks (Gero 1990). Design definitions include shape models, product documentation, and design structures such as bills of materials (BoMs), assembly mating structures, and function structures. In particular, a number of different types of BoMs may be used (Kashkoush and ElMaraghy 2013a) such as those for engineering, manufacturing, packaging, shipping, and servicing. The management of BoMs is

✉ Amar Kumar Behera
a.behera@qub.ac.uk

Alison McKay
a.mckay@leeds.ac.uk

Christopher F. Earl
c.f.earl@open.ac.uk

Hau Hing Chau
h.h.chau@leeds.ac.uk

Mark A. Robinson
m.robinson@lubs.leeds.ac.uk

Alan de Pennington
a.depennington@leeds.ac.uk

David C. Hogg
d.c.hogg@leeds.ac.uk

[1] School of Mechanical and Aerospace Engineering, Queen's University Belfast, Belfast BT9 5AH, UK

[2] School of Mechanical Engineering, University of Leeds, Woodhouse Lane, Leeds LS2 9JT, UK

[3] Leeds University Business School, University of Leeds, Leeds LS2 9JT, UK

[4] Department of Engineering and Innovation, The Open University, Milton Keynes, UK

[5] School of Computing, University of Leeds, Leeds LS2 9JT, UK

crucial to ensure consistent interpretation of product definitions across different organizational departments and across organizations within a supply network, and to maintain interoperability among different engineering systems (Demoly et al. 2013). Current approaches to managing and integrating engineering information, such as that present in separate BoMs, involve using product life cycle management tools that process product-related meta-data (Srinivasan 2011) to enable the coordinated sharing of digital design definitions. Such approaches have included architecting systems using design structure matrices in conjunction with lattices (Engel 2013, 2015) where sensitivity analysis was combined with lattice graphs to create robust system architectures. This involved the use of architecture option (AO) theory for designing systems with life cycle adaptability and indicates that lattice descriptions may be useful and relevant in product data integration. However, despite having a logical, core architecture that defines these meta-models, the heterogeneity of information resources within an organization presents a significant barrier to the use of meta-models for the efficient management of product data (Yoo and Kim 2002).

The success of engineering firms in global markets depends on their ability to design new products that fulfil evolving customer needs within tight constraints of cost and time. Often, this involves modifying existing designs to include new functionalities and forms. A common difficulty here is that of interpreting the previous designer's thought processes underlying the design definition that is available for reuse (Chandrasegaran et al. 2013). Other challenges include managing multiple digital definitions of the same product that have emerged from different departments within the firm (McKay et al. 2015) and the need for engineers to have sufficient knowledge about existing designs to specify search criteria (Vollrath 1998). In addition, the management of engineering design changes has economic consequences for organizations (Kidd and Thompson 2000). Even when an entirely new design is created, it is useful from an organizational perspective to create design definitions that can be reused in the future.

Both situations, reusing existing designs and creating new ones, and the need to support downstream processes such as manufacturing, call for engineers to use design structures within their design definitions (McKay et al. 2015). Examples of design structures include function structures, design grammars, and BoMs. While design structures have been successfully used in existing product development systems, they are usually integral parts of different digital definitions of the same product. This results in a need for careful management of design data and the risks of additional design iterations, which increases product development time and associated costs. The need for design issues in different industrial contexts to be described in a common language calls for interlinked product definitions using design informatics (McMahon 2015). Kim (2006) proposed an assembly design (AsD) ontology for sharing product definitions wherein engineering, spatial, assembly, and joining relations could be easily represented using an assembly relation model that enabled collaborative product development. While it has been argued that concurrent product development processes and integrated product development teams can enhance new product development systems (Ahmad et al. 2013), the optimization of project management practices alone cannot fully overcome technical data management issues. Indeed, as designers spend 14% of their time seeking information (Robinson 2010), readily accessible embedded information of this nature would save substantial time. Within the aerospace sector, for instance, companies develop gas turbine engines within product families—such as Rolls-Royce's Trent range—making incremental changes to a standardized general structure, thereby managing the inherent complexities and sharing best practice (Kerley et al. 2011). Thus, the embedding of functional information into engine designs would substantially increase the efficiency with which subsequent designs could be developed. Furthermore, aeroengine manufacturers have recently changed business models, moving from providing a product to providing a service, through long-term maintenance contracts such as Rolls-Royce's "power-by-the-hour" agreements (Neely 2008). Thus, the capability to embed component costs and in-service performance data into designs would increase efficiency and confer a major competitive advantage.

In this paper, we report a new approach to integrating and managing design information that preserves the heterogeneity of different design definitions while allowing necessary relationships to be defined between them. The relationships are defined using embedding, where one instance of a mathematical structure is contained within another (Stiny 2008), and implemented using lattice theory and qualitative data analysis tools. Embedding is used to relate different design structures to one or more design definitions. We used two separate case studies in this research to illustrate embedding in distinct engineering problem scenarios. Figure 1 illustrates the structure of the case studies while Table 1 lists which design structures are embedded in which design definitions for these case studies.

The paper is structured to convey results from two ways of implementing embedding. The first used a software prototype that enables embedding of BoMs into design definitions using lattice structures. The lattices generated by the software contain the individual parts and sub-assemblies as nodes of the lattices. A user is able to create a new BoM by selecting the required nodes and tracing through the lattice to arrive at the final assembly definition of the product. A case study of a robotic arm, which was discussed in an earlier conference article with preliminary results (Behera et al. 2017), is used to illustrate technical requirements,

Fig. 1 Case study structure for embedding

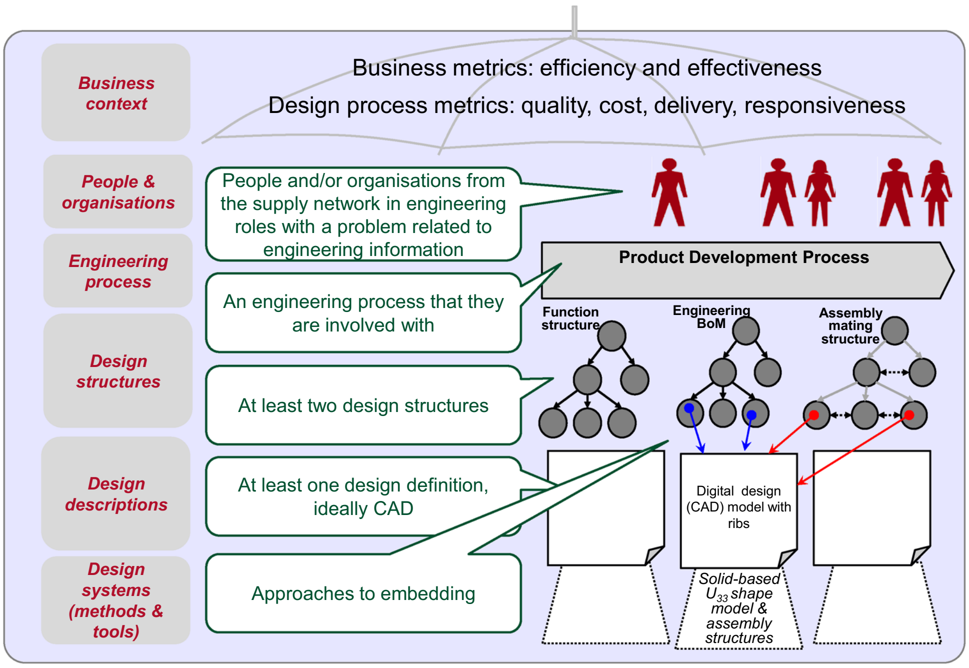

Business context
People & organisations
Engineering process
Design structures
Design descriptions
Design systems (methods & tools)
Business metrics: efficiency and effectiveness
Design process metrics: quality, cost, delivery, responsiveness
People and/or organisations from the supply network in engineering roles with a problem related to engineering information
An engineering process that they are involved with
At least two design structures
At least one design definition, ideally CAD
Approaches to embedding
Product Development Process
Function structure
Engineering BoM
Assembly mating structure
Digital design (CAD) model with ribs
Solid-based U_{33} shape model & assembly structures

Table 1 Details of design structures embedded in design definitions

Case study	Design structures	Design definitions	Tool for embedding
Collision avoidance robot	eBoM, function structure	2D annotated CAD model	NVivo project file
Robot arm	eBoM, purchasing BoM	3D CAD model in STEP AP214 file format	Hyper-dimensional lattice

user perspectives, and implementation requirements for embedding.

In the second approach, the use of a qualitative data analysis tool in implementing embedding is reported. A second case study is introduced with associated design structures, preliminary results of which were reported earlier as a conference article (Behera et al. 2016). The second case study is used to compare and contrast a current technique available to manage design information and implement design changes. The case study concerns the transformation of a robot designed for collision avoidance into one that can follow a marked loop path. Experiments were carried out with annotation of lightweight models and the use of NVivo10 (QSR 2014), a qualitative data analysis commercial software tool, to embed multiple design structures within one or more design definitions. Advantages and disadvantages of this approach were compared to those of a current technique for managing design changes using design structure matrices (DSM). In presenting the results, we aim to (a) help understand how different design structures can be related to each other and the overall product, and (b) differentiate between how design reuse requirements are met by conventional design change management strategies and by embedding. Thus, the paper extends the work discussed in the previous conference articles by not just providing enhanced tools that have been rigorously tested but also by synergizing the results from the two techniques for embedding and contrasting them with the current engineering approaches.

2 Literature review

In this section, three areas of background literature are reviewed. First, a review of literature on design structures is provided as these are the artefacts that get embedded. Next, a summary of design data management techniques and their implementation is provided as this work relates to the integration and management of design structures across engineering life cycles. Finally, prior work on the underlying theory of embedding that is used for the integration is discussed.

2.1 Design structures

Engineering design processes lead to the definition of physical artefacts and associated services and processes. The focus of this paper is on the designed artefact. The core definition of such artefacts includes shape and material information. Design structures are used to support the use of design definitions in manufacturing and life cycle processes. For example, a bill of materials is a design structure well suited for purchasing and procurement activities, whereas a function structure is better suited for use in functional analyses. Design structures are artificial devices that enable transformation of artefacts to carry functional and physical information in a structured manner (McKay et al. 2015).

Design requirements determine the functionality of the product. The function represents the transformation of inputs to outputs based on the flow of energy, materials, and signals that fulfil the design requirements (Pahl and Beitz 2013). Furthermore, the function of a product can be divided into sub-functions, which are less complex than the overall function. This breakdown helps in the search for solutions to design requirements and enables a clear definition of sub-systems required or present in the design. However, what constitutes a function presents a problem for designers and, for this reason, is not straightforward to describe (Eckert 2013). Vermaas (2013) discussed the ambiguity of the different functional descriptions of a product and provided responses to the coexistence of different meanings of function.

A BoM defines the product structure as a decomposition of a whole into its parts, and this has many applications. For example, Plossl et al. (1994) used it in the context of material requirements planning (MRP). McKay et al. (2004) developed a grid-enabled product data viewer that illustrates this decomposition within a software prototype. In a more recent work, Kashkoush et al. (2013b) have used tree reconciliation, a method from biological sciences, to match BoMs and thereby cluster product variants into families. A BoM can be represented as a hierarchical graph structure, where the arcs indicate the relationship (connections in the graph) from the whole to its individual parts (nodes of the graph), as outlined in the systematic definition for design structures given by McKay et al. (2015). Extending this idea, different design structures can be identified in engineering information and classified into three types, as illustrated in Table 2.

2.2 State of the art: approaches to the management of design information

This section discusses current approaches and evolving paradigms for the management of design information. First, the state of the art in product data management (PDM) and product life cycle management (PLM) systems is reviewed. Next, the use of design structure matrices as a design tool is discussed. Finally, recent work on annotation of designs is covered.

2.2.1 Product data/life cycle management

PDM and PLM system solutions support the management of design information through workflow processes, typically in the form of design objects such as files (Stark 2015). Anticipated benefits of integration across design objects themselves, and advances in design analysis, simulation, and optimisation technologies and systems, have led to simulation-driven (Sandberg et al. 2013) and model-based engineering (Cloutier et al. 2015) solutions. Table 3 lists

Table 2 Framework for the identification of design structures

Node	Type of design structure			
	Functional		Physical	
Arc	Composition	Connection	Composition	Connection
Type of engineering information				
Visual with significant embedded structure	Functional context diagram	Function structure	1D, 2D and 3D BoMs	MCAD models ECAD models LabView simulation module Finite element models Assembly mating conditions
Visual with limited embedded structure	Functional architecture diagram Data flow diagram	Morphological chart Part/assembly/product functionality sketches Function tree	Diagrams in manuals Photographs Digital presentations	Diagrams in manuals Photographs Digital presentations
Text	Requirements specification	Design brief	Design brief Material specification Fastener selection criteria	Design brief

Table 3 A list of some popular PLM tools and their capabilities (KETIV 2014)

PLM software	Vendor	Key customers	Key capabilities
Teamcenter	Siemens	Procter & Gamble, BAE Systems, Astrium	BoM management, community collaboration, compliance management, life cycle visualization
Windchill	PTC	Axeon, John Deere, Xerox	Lightweight design process, cost estimation, direct geometry creation and editing
PLM 360	Autodesk	Porex, Greenpoint, Zep Solar	BoM management, change management, ERP integration
Aras	Aras	Motorola, Lear Corp., Xerox	BoM management, multi-CAD PDM, 3D PDF visualization
Enovia	Dassault	Jaguar, Boeing, Olympus	Collaborative innovation, digital manufacturing enabled
Optiva	Infor	Henkel, Sypris, RPM International	Consumer-grade user interface, label development, industry-specific functionality (food and beverage, chemicals)
IFS PLM	IFS	Colfax, Nestle, SAAB	Real-time data, measurement management, spare part management
MasterControl	Master Control	Nelson Laboratories, BioMimetic Therapeutics	Tailored to specific sectors (biotech, pharma, etc.), automated approval workflows, project planning
EnterpriseIQ	IQMS	Donnelly Manufacturing, Tessy Plastics	Automated workflows with tracking, integrated machine and processing monitoring, smartphone accessible
Arena Cloud PLM	Arena Solutions	SunLink, SiriusXM, Lytro	Central repository for BoMs, cloud-connected content, MCAD, ECAD and ERP integration

some of the commonly adopted PLM tools in industry and their key features.

Architecturally, such solutions involve tight integration of processes and tools around a single design definition (Srinivasan 2011). The feasibility of establishing the digital design definitions needed for full implementation is, however, questionable, especially when associated business risks such as reliability and affordability are considered. A key challenge in establishing such models lies in supporting the multiple viewpoints needed throughout a product's life. In parallel with the development of data exchange standards, such as IGES and STEP, numerous researchers have proposed underlying meta-models for different aspects of these digital design definitions (Srinivasan 2011). Common problems with the adoption of such models include their inflexibility in accommodating changing information requirements and in supporting multiple, sometimes conflicting, viewpoints of different types of user.

Stouffs (2008) proposed a semi-constructive algebraic formalism called "sorts" to construct design representations that can handle emergent information. This formalism allows for matching representational structures and merging data constructs. However, this formalism has found limited applicability in industrial solutions so far. In practical usage, engineers have found the use of PLM tools to hold aggrandizement of unrelated functionalities leading to their use without a roadmap, often increasing costs in the delivery of a product (KETIV 2014). Furthermore, the selection of the right PLM vendor for a firm can be a difficult decision due to the varying capabilities offered by each software product and the technology models they are based on, such as on-premise or cloud-based or SaaS-based (Chen 2011).

Although PLM solutions often provide an integrated set of applications that can eliminate a few of the physical data transmission issues within a firm, and also improve communication between data sets, companies will have requirements that may not be fulfilled by this set of integrated applications. Hence, other applications are often used to manage these requirements. Such situations result in "islands of data" within the company (Stark 2015) due to lack of full integration between data sets. The frequent outcome of this is that information that is inter-related in real life may not be inter-related in the software and informatics systems of the firm, thereby enhancing redundancy in data handling, potentially introducing errors and inconsistencies, and ultimately increasing operational costs (Herzig et al. 2014).

2.2.2 Design structure matrix

Design structure matrix (DSM) is a method used to organize the design of a physical system and capture flows, interactions, and interdependencies between the elements comprising the system (Steward 1981). It helps relate entities of a given kind, such as parts, to each other (Lindemann 2009). Tang et al. (2010) discuss how DSM can be used to assist designers in predicting the impact of changes on current solutions and their reuse on new projects. DSM modelling is used to carry out product decomposition, identification of interfaces between sub-systems and components in a product, and structure analysis to increase the architectural understanding of the system, and thereby support decision-making in redesigning and modularizing products (Habib 2014). While a DSM in its basic form is used to understand the interdependencies between entities of the same type, it may also take the form of linking entities of different types, such as relating functions to components. In such a form, it is then called a domain mapping matrix or DMM (Eppinger and Browning 2012). A further development is that of combining DSM and DMM to form a multi-domain matrix, which provides an overall system model.

Attempts have been made to go beyond just looking at interconnectivities by also adding statistical correlation in the DSM, for instance, to look at communication dependencies in product development (Hepperle et al. 2007). The use of social network analysis (SNA) techniques in conjunction with DSMs has been shown to be effective in new product development (Liberati et al. 2007). Yang et al. (2014) identified techniques to improve the effectiveness of using DSMs by taking into account interaction strength to create an "evolution DSM" and "sensitivity DSM". Two indices, information output time index (IOI) and information receiving time index (IRI), were used to quantify the interactions in terms of refinement of upstream activity and magnitude of coordination and rework of downstream activity if upstream information is changed. As a result of this work, a clustering algorithm for DSMs was proposed that could evaluate the clustering structure based on the interaction strength. This technique was shown to be valuable in designing organization architectures for product development as well. As a management tool, DSM is frequently used in project management to create a representation of the project that takes into account feedback and cyclic task dependencies, thereby improving scheduling and implementation (Danilovic and Browning 2007).

2.2.3 Annotation

Annotation has long been part of engineering practice as a method of communicating about designs in both face-to-face direct interactions and indirect interactions involving sending thoughts and ideas across to stakeholders involved in the product development process (Ding et al. 2009b). Systematic, structured methods of annotation have been proposed such as ones that use markup (Davies and McMahon 2006) followed by organization and manipulation. Further work has recommended the combination of lightweight CAD models together with markup. Various techniques for creating lightweight representations have been proposed. Liu et al. (2015) used algorithms that combined techniques such as components suppression, feature simplification, and hollows seaming to generate lightweight representations. Ding et al. (2009b) developed the Lightweight Model with Multi-layer Annotation (LIMMA) technique for annotating boundary representations (b-rep) of products with the help of a markup language, XML. Ding et al. (2009a) also report the use of Open Archival Information System Reference Model (ISO 14721:2003) in creating annotated representations of both nominal CAD models and lightweight versions using a stand-off technique of layering information that enables markups to be transmitted through the product life cycle independent of the geometry. A number of current approaches that support lightweight representation have been summarized in this work in the form of file format types, which are 3D XML, HSF, JT, PLM XML, PRC, U3D, X3D, and XGL/ZGL. Elsewhere, Song and Chung (2009) report the use of XML data generated using STEP PDM schema and lightweight files created using the use of ACIS kernel with InterOp to create digital mock-ups of disparate CAD models. In summary, while annotation does, to some extent, support multiple viewpoints and changing information requirements, it is limited by the restricted structure of annotation data and an inability to annotate aspects of the design, such as functional information, that do not occur in a CAD model.

2.3 Underlying theory for embedding

Embedding has been documented since the 1930s in the mathematics literature (Tompkins 1939). Descriptions of concrete applications are less common but do occur, for example, in the shape computation literature (Stiny 2008). However, methods to enable the robust implementation of embedding for use in real-world applications remains an open research issue (McKay et al. 2012). The ultimate goal of the research reported in this paper is to explore the use of embedding as a way of allowing engineers to associate multiple design structures with a given design, so that they may be used as and when needed.

Methods of formal concept analysis provide a mathematical means to capture conceptual knowledge within a formal framework (Wille 1992). Lattice theory is one of these methods that can be used to represent concept hierarchies (Wille 2005), where nodes in a lattice represent concepts as sub-concepts and super-concepts (Bělohlávek 2004). In this research, the nodes in a lattice are used to represent parts in a BoM to enable the exploitation of two properties of lattices. First, for a given collection of parts, there is a complete lattice that contains every possible combination of parts (i.e., every possible BoM). Second, given two BoMs, there is a unique biggest lattice (the supremum) that contains both and a unique smallest part (the infimum) that is part of both (Grätzer 2011).

The mapping of shape grammars into graph grammars with an inherent embedding that exploits the multi-dimensional nature of the data structure provided by graphs has previously been shown to be able to generate an alternative representation of shapes enabling visual computations (Grasl and Economou 2011). Graph grammars have also been used to generate a computational and parameterized product model that can encapsulate function structures in a viewpoint-independent manner (Helms et al. 2009). Further, it has been illustrated that a graph-based design language together with a design compiler can be used to translate abstract geometry in an abstract representation scheme into an ad hoc target format (Schmidt and Rudolph 2016). This work further exploits the use of graph grammars in design informatics to illustrate the use of lattice structures, which are essentially graphs, in embedding bills of materials.

2.3.1 Lattice theory

The relevance of lattices as representation schemes for the geometrical modelling of shapes is discussed in a number of early works such as Stouffs (1994) and March (1996). Lattices are appropriate for representing design structures such as BoMs because they are partial order sets in which any two elements have a unique greatest lower bound (also called 'meet' and denoted as '$\wedge$') and a unique least upper bound (also called 'join' and denoted as '$\vee$') (Cameron 1994). The presence of partial order in lattices is useful as it establishes a one-to-one correspondence with the structure in typical product definitions, which consist of parts (sub-assemblies and components), thereby establishing a hierarchy where multiple parts may be present at a specific level of abstraction or design definition.

According to Freese (2013), a lattice $\mathbf{P} = (P, \leq)$ comprises a set P and a partial order relation $\leq$ on P. A unique smallest relation, $\prec$, exists on P that is called as the cover. A lattice can be represented as a 'Hasse diagram', which is a diagram of the acyclic graph $(P, \prec)$ where the edges are straight line segments, such that for two elements a and b in P, where $a < b$, the vertical coordinate for a is less than that of b (Cameron 1994).

In this research, a "complete lattice" is defined as one that contains every possible combination of individual parts in the product's life cycle. All subsets of the complete lattice have a join as well as a meet. An example of a complete lattice for a product consisting of five parts, A, B, C, D, and E is shown as a Hasse diagram in Fig. 2. It can be seen that the nodes of the lattice represent combinations of parts taken 'n' at a time at each level 'n'. It may be noted that a total of 2^m nodes are required to generate the n-hypercube lattice for 'm' parts. A specific BoM can be seen to be a sub-set of this lattice that is generated by selecting specific nodes in the complete lattice. This set inclusion property is referred to as a BoM being "embedded" in the complete lattice in this work.

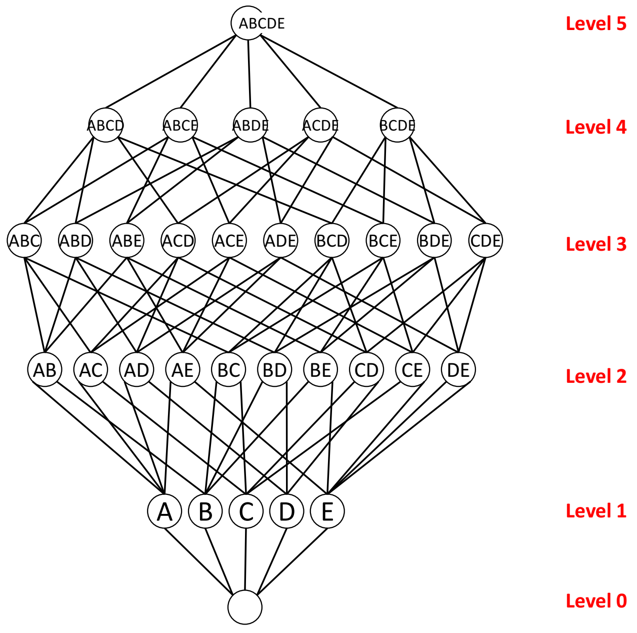


Fig. 2 Complete lattice for a product with five parts

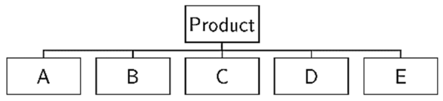
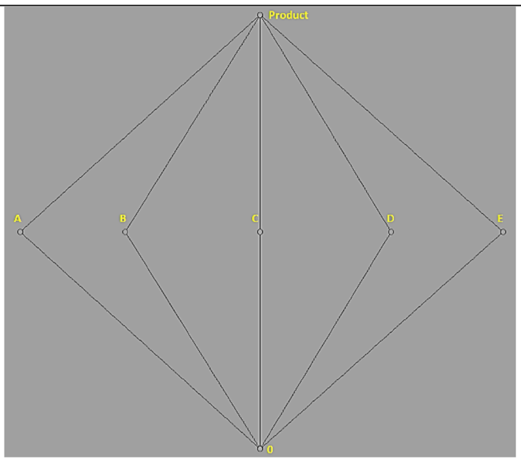
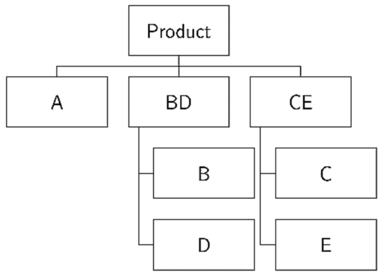
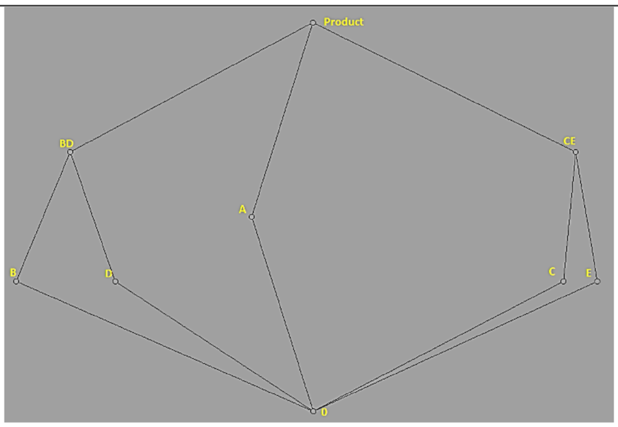
Table 4 illustrates examples of sub-lattices for specific BoMs. It may be noted that the complete lattice provides a space of aggregations of parts, from which parts can be reinterpreted (e.g. a sub-assembly can be dissolved to be considered as a part) or alternative nodes selected, to create new BoMs. This exploits the idea of using embedding of shapes to redefine parts on the fly presented by Stiny (2011). Hence, all possible BoMs can be seen to be embedded in the complete lattice.

For a product with a specific number of parts, it is feasible to construct lattice representations of all possible BoMs. Table 5 shows the different possible configurations for a five-part product. By constructing these lattice configurations, the algorithms required to embed these configurations within the complete lattice were then developed, as the process of embedding involves mapping the nodal configurations of these sub-lattices onto that of the complete lattice.

2.3.2 Qualitative data analysis techniques

Qualitative data analysis techniques enable a systematic search for meaning in data using tools such as method of constant comparison, keywords-in-context, classical content analysis, domain analysis, taxonomic analysis and componential analysis (Leech and Onwuegbuzie 2007). Qualitative methods are useful where the understanding of a particular product or process depends on detailed information and the available data are non-numeric such as textual data or images (Bazeley and Jackson 2013). Several software tools such as HyperRESEARCH, MAXqda, NVivo, QDAMiner,

Table 4 Lattices for different product structures

Product structure	Lattice
	
	

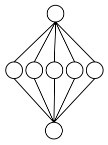
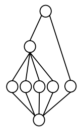
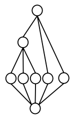
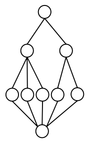
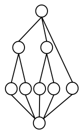
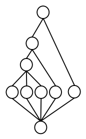
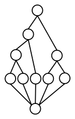
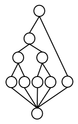
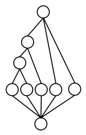
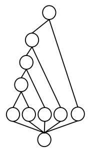
Qualrus, Transana and Weft QDA enable the use of qualitative methods and are primarily centred around providing constructs that help in interpreting and managing data (Corti 2008). These tools usually have a few common features such as (1) importing documents with text data (e.g. TXT, PDF, DOC, HTML file formats), (2) hierarchical tree for characterizing text data with character level coding, (3) retrieving coded information, (4) statistics on coded text, (5) quick text search through an assortment of documents, (6) querying using Boolean operators that integrate coding and search results, (7) identifying and evaluating coding patterns, and (8) export to external file formats such as HTML and CSV (Lewis 2004). To use these tools to support embedding, unstructured attributes need to associate with engineering information available as shape models, free-form text, and images. A systematic way of enabling this association using the commercial qualitative analysis software NVivo was developed within this research, which is discussed in the following sections.

3 Methodology

A software prototype, StrEmbed-3, was built that parses a STEP AP214 file, generates a complete lattice corresponding to the components parts of the product in this file, embeds the BoM corresponding to the file in this complete lattice and, finally, allows alternative BoMs to be defined in terms of the complete lattice. StrEmbed-3 is available online at Chau (2017). For example, if an assembly from a CAD file represents a view as designed, additional BoMs could be created to represent the same product in its “as built”, “as shipped”, or “as in service” forms. StrEmbed-3 was used in experiments to generate BoMs for a case study robotic arm, defined in Sect. 4.1.

To demonstrate embedding using qualitative data analysis, a small, desk-based case study was developed that could be used both to communicate the potential value of embedding to end users and to evaluate the feasibility of implementing embedding using currently available techniques. To this end, a robot was selected as the case study because robotic systems incorporate complexities

Table 5 Different possible BoM structures viewed as lattices (for a product with five parts)

Configurations → Maximum number of arcs in lattice from infimum to supremum ↓	C1	C2	C3	C4
2				
3				
4				
5				

arising from multiple design perspectives, and associated structures in relatively small systems. The engineering scenario used was based on two iterations of the robot, one where the success criterion was collision avoidance, and a second where this criterion changed to following a guide line. The case study is defined in Sect. 4.2. The selection of this case study was done from a past real-world situation and this ensured independence between the case study, which reflects the engineering problem, and technical solution principles. Results from experiments with annotated, lightweight CAD models and NVivo are reported in Sect. 5.2 and compared with evaluations of design structure matrices.

4 Case studies

The two case studies that were used to identify and illustrate the benefits of embedding are discussed below. The first case study involves a robotic arm, where BoMs are represented as lattices and then altered to create new representations, while the second case study involves a collision avoidance robot, whose design has to be changed for a new situation.

4.1 Robotic arm: representation and alteration of BoMs

A case study of a robotic arm assembly, adapted from Zortrax (2016), was used to demonstrate the feasibility of using lattice structures in representing and embedding BoMs. The arm was 3D printed and assembled into a working robot, using SolidWorks models and STL files available from the manufacturer. The arm was manufactured using an Objet1000 3D printer using the material ABS (acrylonitrile butadiene styrene). Two different BoMs, engineering and purchasing, were parsed from their STEP files and then embedded in the complete lattice using StrEmbed-3. Figure 3a shows the CAD model of the arm annotated using open-source viewer, eDrawings, Fig. 3b shows the fabricated and assembled robot, while Fig. 4 shows the engineering and purchasing BoMs. Finally, StrEmbed-3 was used to create a shipping BoM from the engineering BoM, for a simplified five-part robot.

4.2 Collision avoidance robot: change management

The user scenario considered in this study involves a robot with a two-wheel drive, shown in Fig. 5. The robot was

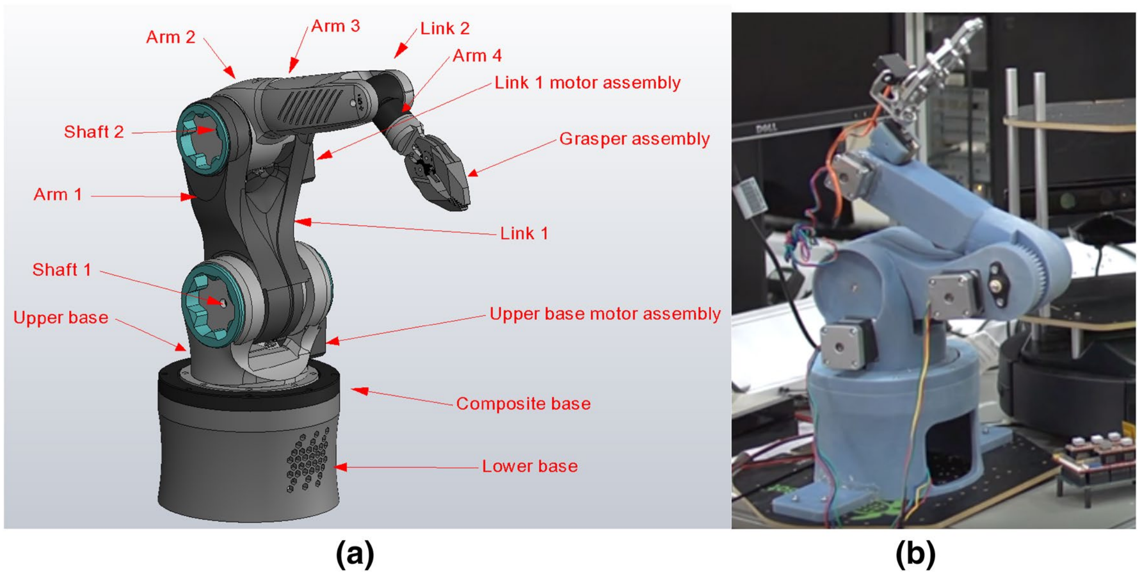


Fig. 3 **a** Annotated robot arm assembly CAD model, **b** robot with mechanical body components fabricated using 3D printing and assembled with electronics, electrical circuitry, and fasteners

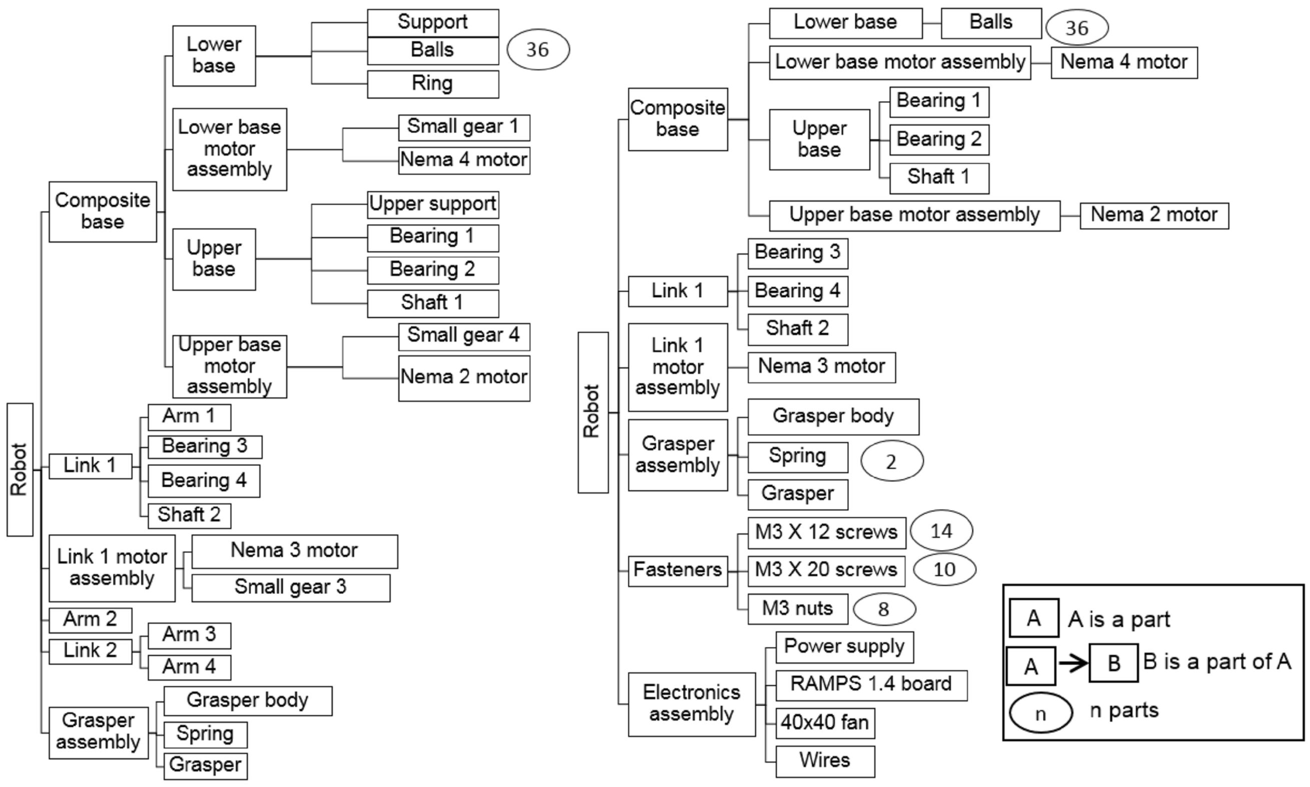


Fig. 4 Engineering BoM for the robot arm assembly (left), purchasing BoM (right)

originally designed to navigate a specified area while avoiding collision with obstacles. Subsequently, it participated in a competition where the goal was to follow a white track and return to the starting point indicated by a magnet. It can be seen from Fig. 5 that there is no direct relationship between the design requirements and the physical design. As a result, if a design requirement changes, expert human intervention is needed to determine the impact of the change on the

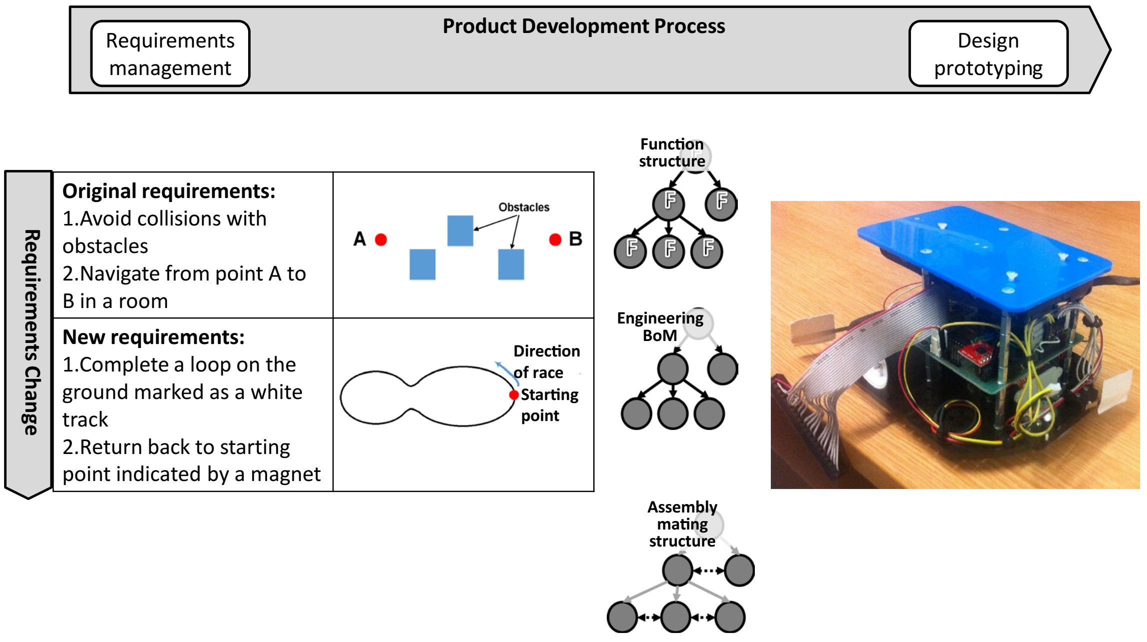

Product Development Process
Requirements management
Design prototyping
Requirements Change
Original requirements:
1.Avoid collisions with obstacles
2.Navigate from point A to B in a room
A
B
Obstacles
New requirements:
1.Complete a loop on the ground marked as a white track
2.Return back to starting point indicated by a magnet
Direction of race
Starting point
Function structure
Engineering BoM
Assembly mating structure

Fig. 5 Case study product, associated design structures and change in requirements

product itself and, therefore, downstream processes such as manufacturing. For these reasons, Rinderle and Suh (1982) identified a need for mappings between representations of a design in both functional and physical domains to help interpret functional couplings. In this study, we show how, by relating design requirements and a function structure to each other, the relationships needed to support design decisions become available. After creating the necessary design structures, as discussed below, the qualitative data analysis tool NVivo was then used to build these mappings within a single project and this approach was then contrasted with a current technique of managing design changes using DSMs.

4.2.1 A function structure for the robot

Function structures were built by breaking down the robot's overall function into sub-functions. For the collision avoidance robot, a function structure has been created as shown in Fig. 6. The overall function of the robot is to navigate from a start position 'A' to a target end position 'B', while avoiding obstacles. Assembly mating conditions connecting the components are necessary to create the assembled robot.

The basic function of navigation is implemented by supplying electric power from batteries. This power is converted to rotation of the shafts of the direct current (DC) motors. The motion of the shaft is transmitted to the axle connecting the wheels of the robot after passing through a gearbox with a specific transmission ratio. While this is the basic function, the robot also needs to avoid collisions. To achieve this, it collects data from proximity sensors and micro-feeler switches. These data are integrated and analysed to determine whether or not there is an obstacle on the path. A collision-free path is thus determined and used to navigate the robot. It may be noted that building the function structure in Fig. 6a required relationships to the physical domain (in the form of the exploded assembly shown in Fig. 6b). This kind of mapping between domains is key to any design process (Kim et al. 1991) and this example shows how the presence of relevant design structures can assist this process.

It is noteworthy that the design of the robot was driven by the overall functional requirement of collision avoidance. Other sub-functions such as navigation were then analysed to determine what parts would make up the robot. Proximity sensors and micro-feeler switches are provided to achieve the function of collision avoidance. The motors and batteries are there to move the robot on the floor. The gearbox is present to provide a transmission ratio. It is an internal assembly and not visible in the exploded view shown, and hence not labelled, but you can see it in the BoM shown in Fig. 7. The function structure and exploded parts (BoM) are linked, and hence have relationships with each other.

4.2.2 Bill of materials for the robot

The robot was decomposed into its engineering BoM as shown in Fig. 7. Note that although wires and fasteners are included in the BoM, they are not always considered a part of the CAD model and, as a result, may not feature in the same. In particular, wires are usually not rigid elements and, hence, are often omitted from 3D CAD models.

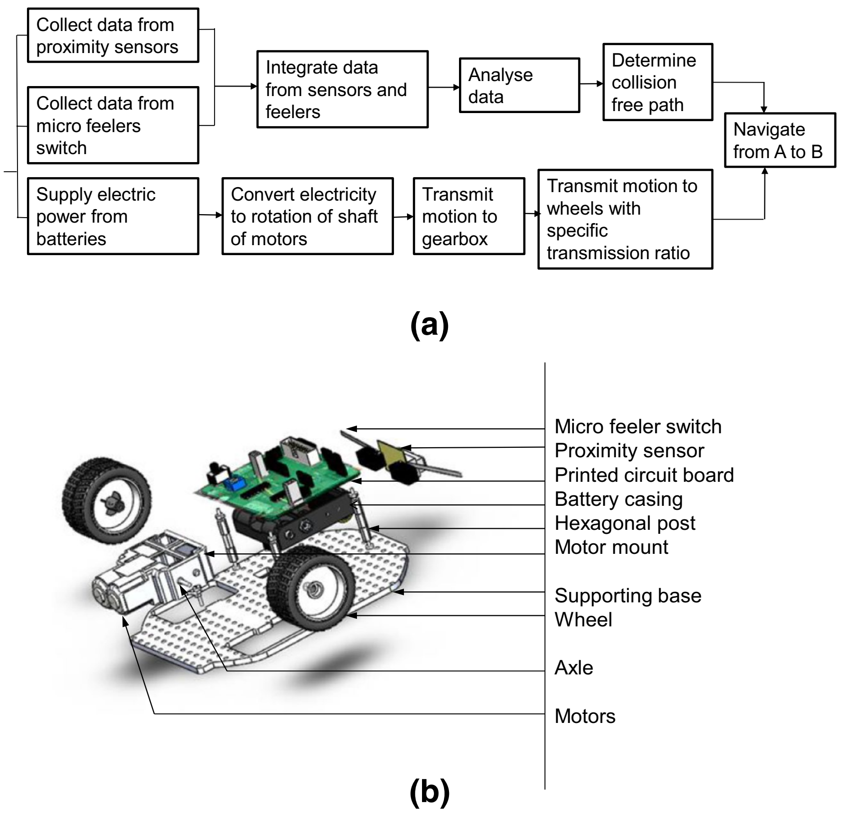

Collect data from proximity sensors
Collect data from micro feelers switch
Supply electric power from batteries
Integrate data from sensors and feelers
Analyse data
Determine collision free path
Navigate from A to B
Convert electricity to rotation of shaft of motors
Transmit motion to gearbox
Transmit motion to wheels with specific transmission ratio
(a)
Micro feeler switch
Proximity sensor
Printed circuit board
Battery casing
Hexagonal post
Motor mount
Supporting base
Wheel
Axle
Motors
(b)

Fig. 6 Function structure for the collision avoidance robot: **a** relates to the individual components in the exploded view of the assembly shown in **b**

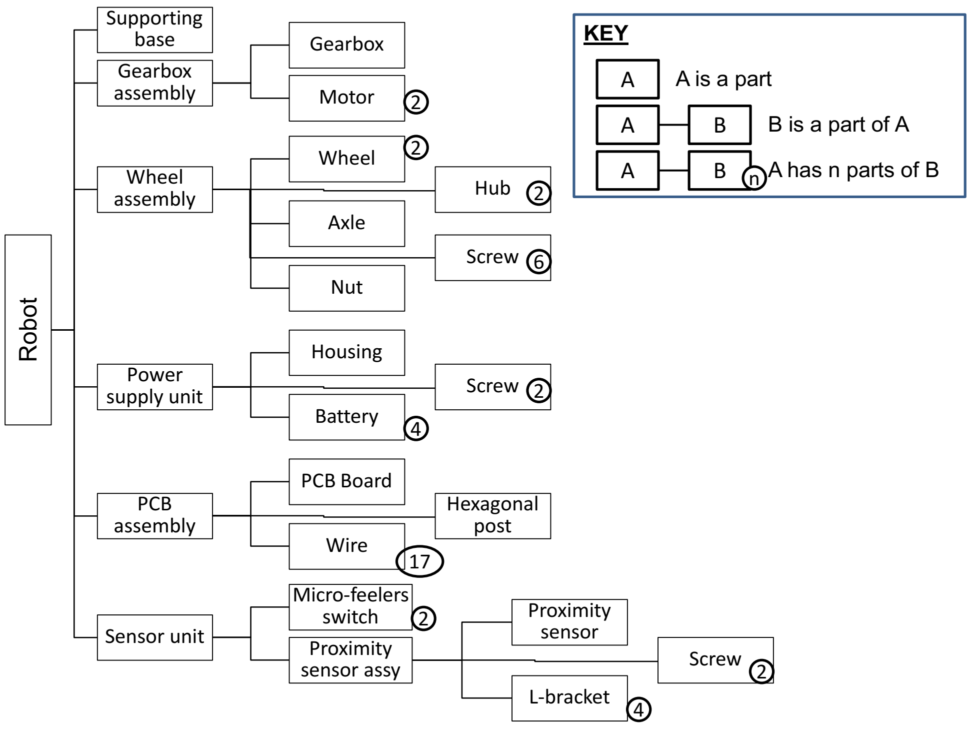


Fig. 7 Physical structure of the robot's engineering BoM

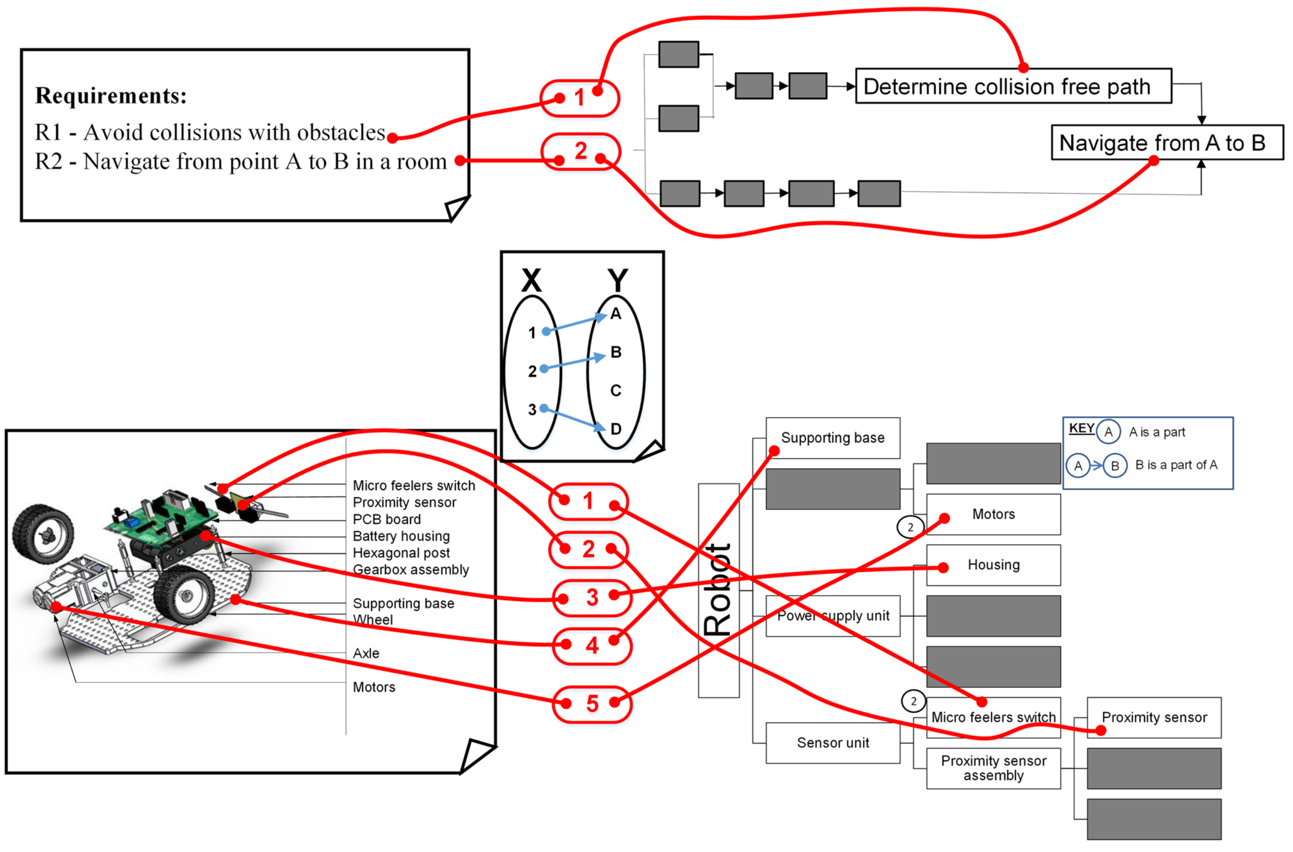


Fig. 8 Relationships between different design definitions for the robot

4.2.3 Relationships between the design structures

The four different definitions for the robot design (requirements, function structure, shape model, and BoM) were mapped to each other as shown in Fig. 8. The top half shows how requirements are related to the function structure while the bottom half illustrates how the shape model is related to the bill of materials. In mathematical terms, this kind of mapping can be classified as an embedding of one definition within another (Stiny 2008). The mapping is injective (inset in Fig. 8) and structure preserving. It represents a topological isomorphism between two sets and is a continuous function with a continuous inverse. Such a function preserves distinctness in the sense that every element in the co-domain of the function has only at the most one image in the domain (Bartle 1964).

4.2.4 CAD lightweight experiment

The experiments with a lightweight CAD model used a SolidWorks model that was exported as a 3DXML file and imported to eDrawings, a free SolidWorks viewer provided by Dassault Systèmes. Different annotated markups were created showing individual components of the robot, and NVivo was used to incorporate image files of the BoM and function structure. These annotations were exported and stored as .markup files, for later use within eDrawings. The lightweight model together with the markups was also separately stored as a *.easm file. This approach makes it feasible to send markup files to project collaborators, with change details mentioned, without having to send an entire CAD model through, making change management a fast and visual process. The markup file can be used with any lightweight model derived from the original model, but is useful only if the absolute locations of the individual components have not been altered: the markup file has no intelligence about the CAD model from which it was generated. Functional information was captured by annotating relevant parts of the lightweight model using free-form text. Images of the annotated model were imported within NVivo to implement embedding.

5 Results

Results from applying the different approaches to embedding to the case studies are reported in this section.

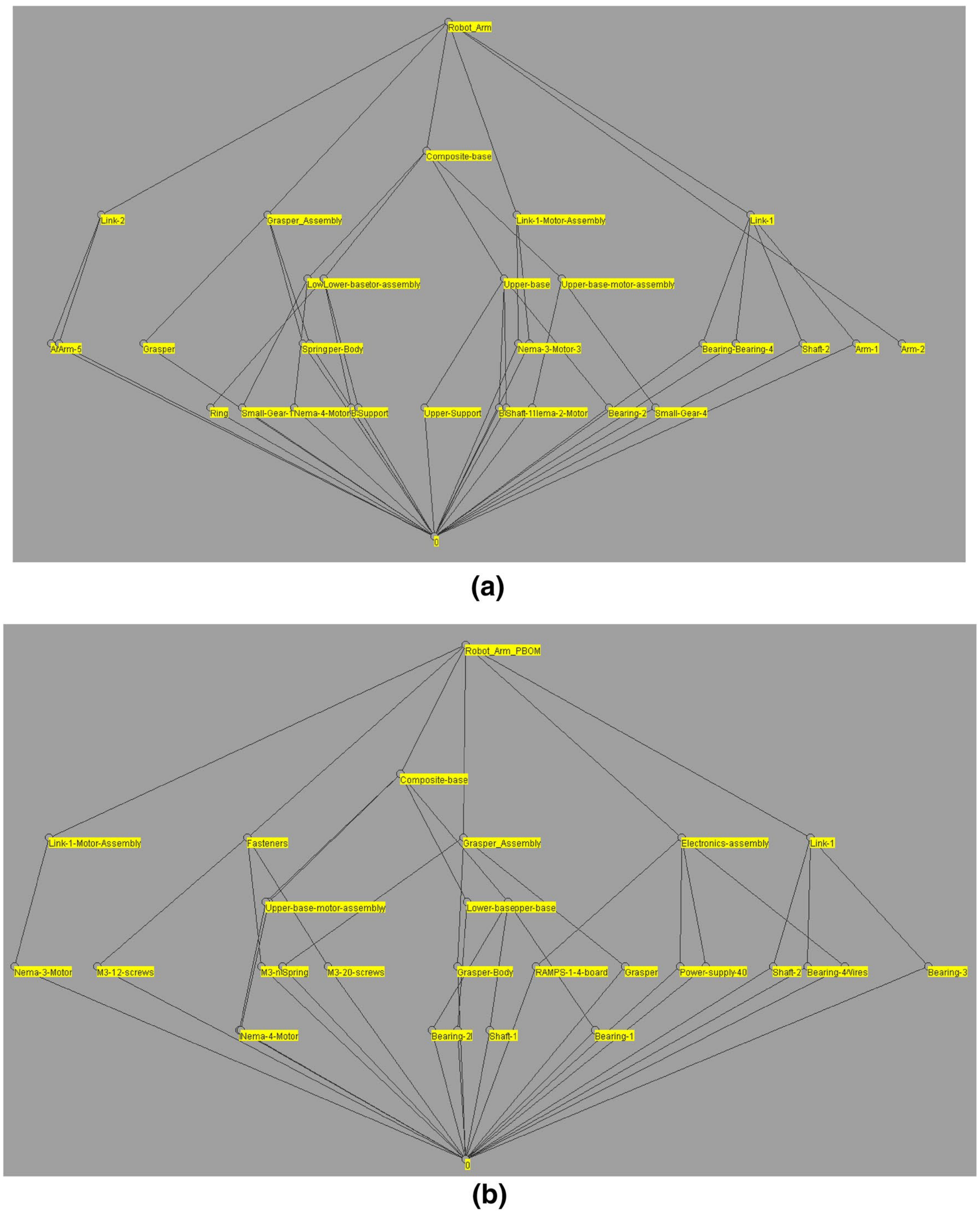


Fig. 9 Annotated lattices for engineering (**a**) and purchasing (**b**) BoMs

5.1 Embedding using lattice structures

Lattices for the two BoMs for the robotic arm were generated using an earlier software prototype StrEmbed-1 (Chau 2016) and visualized using a lattice visualization software, LatDraw (Freese 2013), yielding the results shown in Fig. 9. As the engineering BoM has more parts than the purchasing BoM, the Hasse diagram for the engineering BoM lattice (Fig. 9a) has more overlapping nodes (5) compared to the purchasing BoM (2) (Fig. 9b). The overlapping of labels in the engineering BoM is also higher than in the purchasing BoM. It may also be noted that the purchasing BoM is not

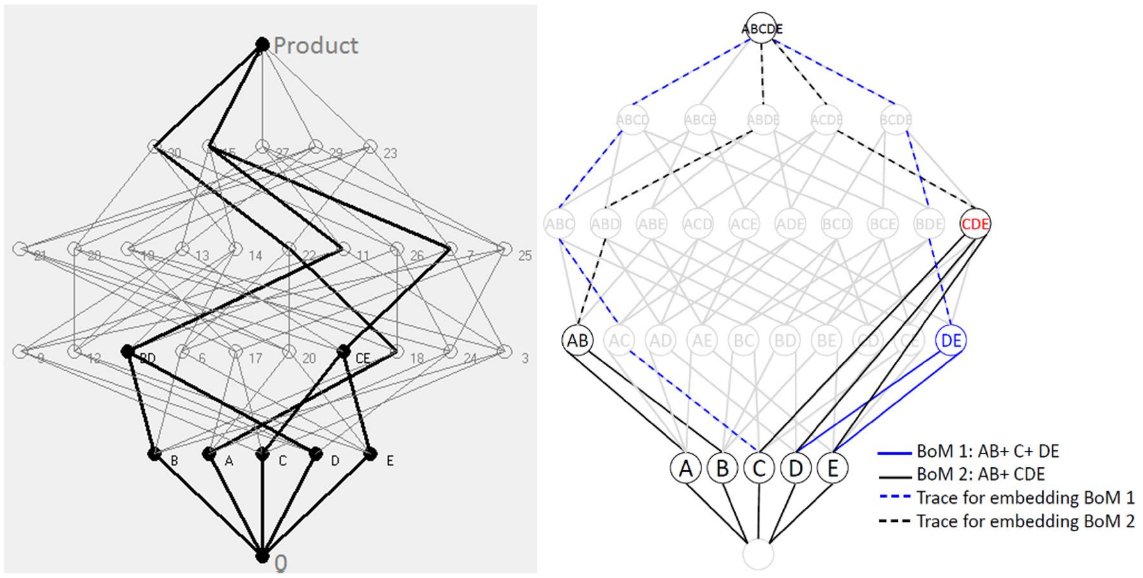


Fig. 10 **a** Embedding a specific BoM within the complete lattice of a product with five parts, **b** creating new design structures by moving from one embedding to another

a sub-set of the engineering BoM, as it has two additional sub-assemblies in fasteners and electronics assembly, which were not present in the CAD model of the robot and hence, did not show up in the engineering BoM.

Attempts to create new design definitions using StrEmbed-3 to first embed a given design definition in a complete lattice Fig. 10, and then altering it to create new design definitions, were successful as illustrated for the robotic arm (simplified to 5 parts) in Fig. 11. It may be noted, however, that only the assembly structure is altered by the developed software prototypes; the physical locations of the parts are changed using CAD software. The use of the complete lattice to create new BoMs can be achieved by addressing user interaction challenges, such as those discussed by Pattison and Ceglar (2014). Based on the individual nodes selected by the user at the instant of creating a new BoM, only the feasible ones are then available for further selection. This type of dynamic generation of BoMs requires updating a linked list dynamically within the code while at the same type handling the visualization effects on the user's display interface. This functionality was tested in StrEmbed-3, using both a command line interface and within the StrEmbed-3 graphical user interface (GUI), which was successfully used to generate new product structures.

5.2 Embedding using a qualitative data analysis tool

In this section, techniques used for the second case study and the results of using these techniques are discussed. First, the systematic use of NVivo as a qualitative data analysis tool to implement embedding is discussed. Then the use of DSM to carry out change management is analysed.

5.2.1 The implementation of embedding using NVivo

The use of NVivo is illustrated in Fig. 12. Lightweight representations of design definitions created in computer-aided design tools are annotated within the CAD software and then exported as images (Step 1 in figure). The available design structures such as BoMs, assembly mating conditions, and function structures are also generated as separate text or image files. These design representations are then imported within NVivo as "internal sources" (Step 2). Next, nodes representing "design definitions" and "design structures" are created and expanded as sub-nodes to include the parts and sub-assemblies under design definitions and specific design structures (Step 3). Each internal source is then linked to these nodes and sub-nodes by selecting the area of text or image data that specifically links up with a particular node or sub-node (Step 4). Thus, the sections of each source are linked to the specific part or design structure creating a cross-reference. Once this is done, the nodes within the NVivo project have a number of embedded references that are linked to specific parts of the design structures that have been imported (Step 5). This enables people within the organization to easily look up the NVivo project to understand the design in detail and where individual components are located within the product hierarchy in terms of sub-assemblies (Step 6). Furthermore, it enables easy identification of all the components

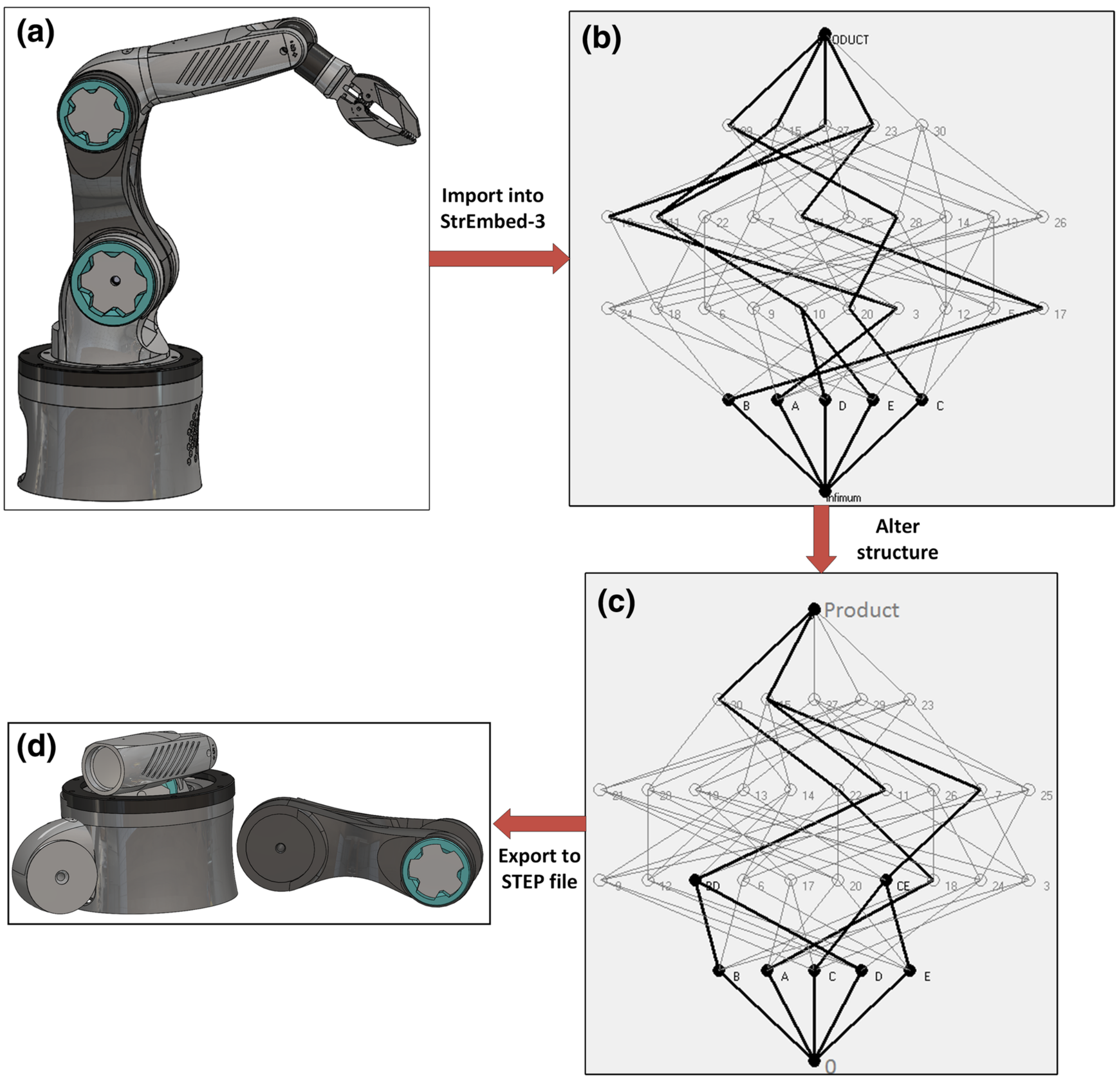


Fig. 11 Results illustrating the use of StrEmbed-3 to create new design definitions **a** robotic arm as designed, **b** complete lattice with engineering BoM embedded in it, **c** alternate product structure for shipping created using StrEmbed-3 and **d** shipping CAD model as visualized in CAD software

and functionalities that would be affected in the event of a design change being implemented.

5.2.2 Results of embedding using NVivo

Image files from the lightweight experiment were imported into NVivo. Nodes representing elements of the design definitions from the case study were created and used to link individual elements of the BoM and function structure to the robot definition. This process enabled the embedding of information from design structures into the design definition so that, if changes to the design were reflected in the imported data, consequences of changes made to the robot design would be visible. Figure 13 shows how multiple design structures were linked up within the NVivo project environment to enable this. Figure 13a shows a screenshot of the project showing nodes representing the definition and design structures in red boxes. The sub-node for the proximity sensor shows three embedded relationships to an engineering BoM (b), an element in function structure (c), and the sensor highlighted by NVivo in an annotated image of the robot (d).

5.2.3 DSM to support change management

A component-based DSM was drawn up for the collision avoidance robot. The new requirement, to complete a loop by tracking a white line on a dark background, was met by placing

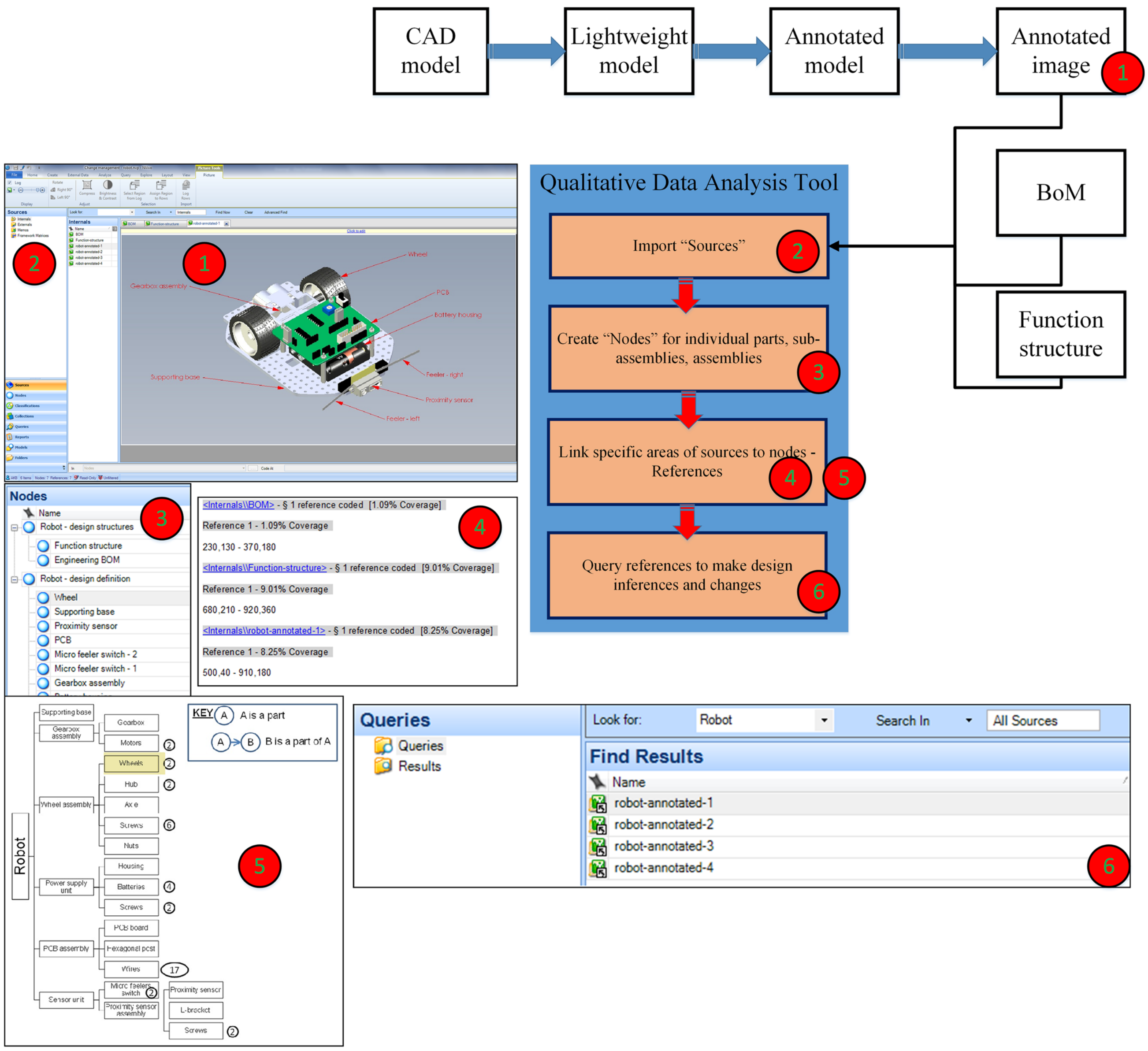


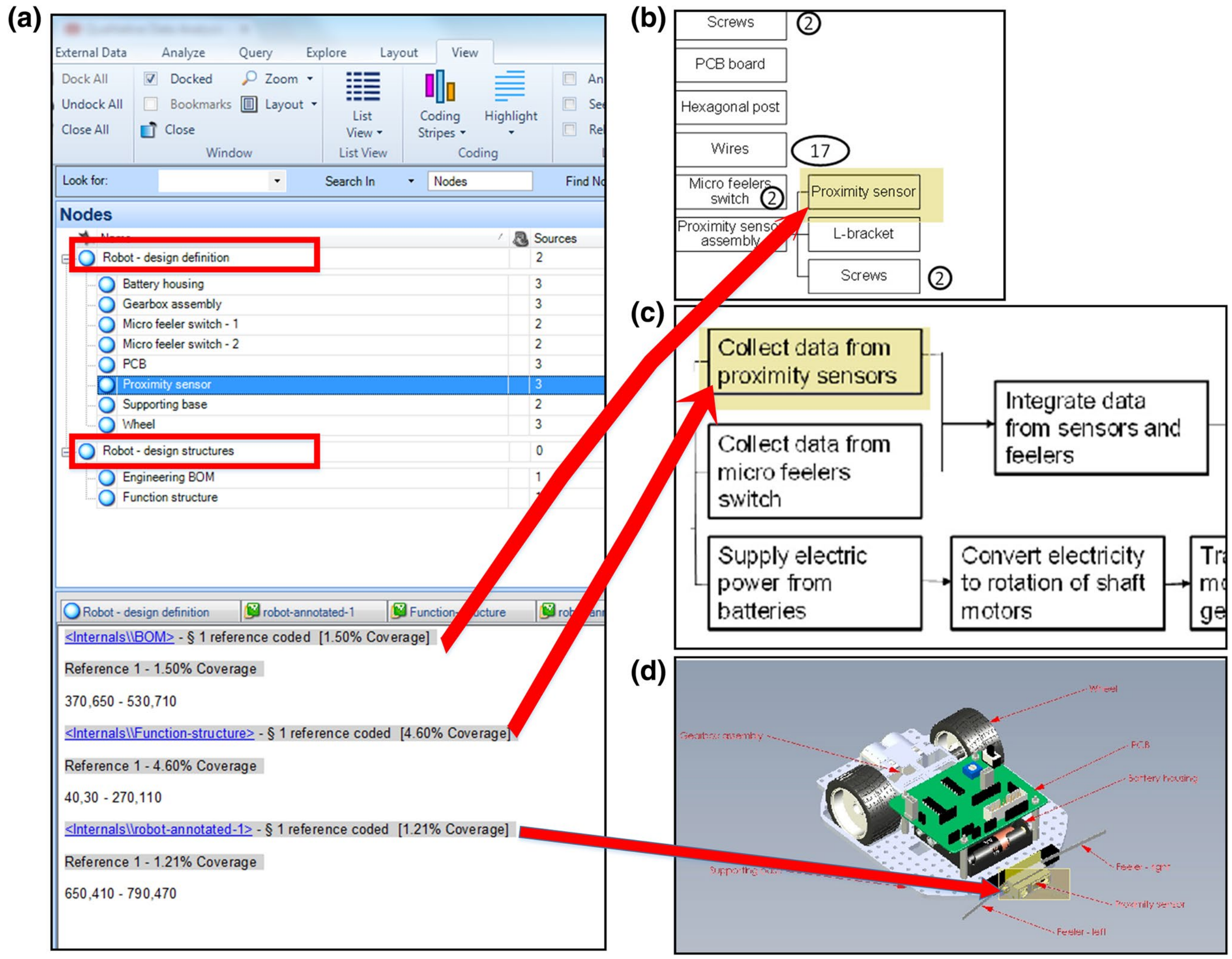
Fig. 12 Illustration of the procedure for embedding using NVivo: screenshots with circled buttons refer to specific steps in the procedure

a white line sensor on the bottom of the robot. Likewise, the requirement to return to the starting point indicated by a magnet was met using a Hall effect sensor. These additions were wired to the printed circuit board (PCB) assembly so that their outputs could be used in the logic required to meet the new requirements. This led to the DSM for the ‘follow the line’ robot in Fig. 14. The shaded columns/rows in blue are additional for the ‘follow the line’ robot and the green boxes are the new interdependencies vis-à-vis the collision avoidance robot. It may be noted that sub-assemblies not involved in the change are shown as parts in this figure and fasteners are omitted in the illustration.

6 Discussion

6.1 Lattices for embedding: pros and cons

The results of implementing the software prototype for lattice generation of individual BoMs and complete lattices with all possible combinations of parts show that, for a given assembly, it is feasible to embed new design structures into a complete lattice generated from the original assembly. However, there is a limitation on the dimensionality of the hypercube lattices that can be optimized for visualization using both the in-house software developed at Leeds, StrEmbed-3, and external software



<Internals\\BOM> - § 1 reference coded [1.50% Coverage]

Reference 1 - 1.50% Coverage

370,650 - 530,710

<Internals\\Function-structure> - § 1 reference coded [4.60% Coverage]

Reference 1 - 4.60% Coverage

40,30 - 270,110

<Internals\\robot-annotated-1> - § 1 reference coded [1.21% Coverage]

Reference 1 - 1.21% Coverage

650,410 - 790,470

Fig. 13 **a** Experiment within NVivo illustrating, **b** embedding of a BoM, **c** a function structure and **d** an annotated image of a CAD model within a single project file

			SB	GA	WA	MFS	PS	LB	WLS	HES	PSU	PBP	W
Base		SB		X		X		X	X	X	X	X	
Gearbox assembly		GA	X		X								
Wheel assembly		WA		X									
Sensor unit	Micro feelers switch	MFS	X										X
	Proximity sensor	PS						X					X
	L-bracket	LB	X				X						
	White line sensor	WLS	X										X
	Hall effect sensor	HES	X										X
Power supply unit		PSU	X										
PCB assembly	PCB board & post	PBP	X										X
	Wires	W				X	X		X	X		X	

Fig. 14 DSM for the ‘follow the line’ robot (blue columns/rows green boxes indicate changes from the ‘collision avoidance’ robot DSM; the green boxes highlight the new interconnections resulting from introducing the new components). (Color figure online)

programs such as LatDraw, which can currently only display up to 7-hypercube lattices without overlapping nodes. In addition, the computational time needed to generate large lattices grows exponentially with the number of parts in the assembly, making it impractical to generate whole complete lattices. Two avenues for further research are, therefore, to explore ways in which users might limit the number of parts of interest, for example, using methods such as holophraxis from human–computer interaction (Ashford et al. 2011), or to be more parsimonious when generating the complete lattice so that only sub-lattices of potential interest are computed. Eventually, the lattice structure can simply serve as a background data structure, with only tree edits performed by the user that are visualized to carry out the necessary design changes. Another strategy to address computational complexity in complex products could be to use high-performance computing (HPC) tools (Hager and Wellein 2010) in generating and manipulating product structures using lattices.

Lattice drawing (LatDraw) is based on a combination of mathematical rank function to determine the height of the elements in the lattice and a modification of the forces method of graph theory (Freese 2013). The use of the force method involves applying a strong repulsive force to the nodes followed by a strong attractive force and finally balanced forces. In addition, each node in the diagram can be manually pulled horizontally to emphasize a certain structure that a user desires. Each node can also be pulled vertically to the extent that partial orders in the diagram are not changed. While this approach is useful in generating static lattices, dynamically generated lattices need better programming approaches.

The purchasing BoM did not have a complete CAD model associated with it; for example, it had electronic components and fasteners which had not been designed by the manufacturer. The absence of these components meant that there were two solutions: (1) create a new CAD model for the purchasing BoM, or (2) generate the lattice file manually. Furthermore, the absence of fasteners in the CAD model meant that the number and type of fasteners had to be manually interpreted. Typically, in many industrial designs, mechanical and electronic components may be designed in separate environments. Hence, a single STEP file may not be readily available for use within the developed prototypes. Furthermore, large products in firms may often create situations where different sub-assemblies may be modelled separately. So, even if all the components are of one type, such as mechanical, it may not be feasible to bring them together to generate a singular definition, which can then be exported into a readily interpretable file format such as STEP AP214. The purchasing BoM highlighted the need to accommodate such practicalities in developing a solution for embedding.

6.2 Comparison of the approaches to management of design information

To analyse the pros and cons of embedding using qualitative data analysis, change management using a conventional design structure matrix approach was contrasted with (a) the use of qualitative data analysis (QDA) tools, and (b) lattice structures. Table 6 captures the key differences in these approaches as mapped against typical requirements for design reuse and change. The requirements against which the comparisons were done were identified through a survey of the latest state of the art in engineering change management.

The addition of two new components to the robot resulted in eight new interdependencies to be analysed. However, despite analysing these interdependencies, there were issues with reusing the original design. The DSM analysis did not reveal an important functional aspect of adding the white line sensor. The white line sensor had to be placed at an optimal distance on the bottom of the supporting base to maintain a specific clearance from the ground that enabled it to detect the white track. This functional requirement could have been captured by embedding a new function structure for the ‘follow the line’ robot within the design definition and/or an annotated CAD model within the NVivo project file, which is linked to individual parts affected by this requirement (Fig. 15). DSM does not have a framework to do this unless a domain mapping matrix (DMM) had been used that could relate components to functions. However, the implementation of embedding achieved its goal of allowing potential benefits to be evaluated. One such important benefit of embedding, if implemented effectively, is that it allows shape models to be used to mediate different design structures that have been created and are used for different purposes. This could provide significant improvements for the planning and management of change management processes because managers would be able to visualize the impacts of changes. It may be noted that while lattice structures provide a method to agglomerate all possible product structures within a single visual representation, they do not yet provide the ability to integrate functional information seamlessly. Hence, integrating QDA tools together with lattices could provide the necessary means to ensure that most of the decisions that product data management necessitates can be facilitated using embedding.

7 Conclusions

The results from the experiments on embedding using lattice structures and qualitative data analysis tools indicate the feasibility of allowing engineers to associate multiple design structures with a given design. It was illustrated that lattice theory can be used as an underlying formalism for this purpose while tools like NVivo provide a unified space where multiple structures can be aggregated and linked to the relevant design definitions. Although there are established representations for BOMs available in the literature, they are not embeddable within design definitions such as CAD models because of inconsistencies across the underlying meta-models. Hence, current representations do not provide the ability to minimize duplication across design definitions within a firm or to enhance integration of CAD data between different divisions of a firm.

We have demonstrated that it is possible to represent and visualize a BoM as a lattice, and embed a given lattice into a complete lattice generated from the same assembly model. Using lattices, multiple BOMs can be embedded to a given number of design definitions thereby enabling a designer to implement and manage changes more robustly. In addition, alternate valid product structures can be easily generated from the lattice representation. Tools to enable the creation

Table 6 Comparison of techniques used in implementing design reuse and change

Change management requirements	Design structure matrix	Embedding using QDA tools	Embedding using lattice structures
Ease of creation/implementation	High for few components but decreases for complex assemblies	Facilitated by creation of new nodes corresponding to new parts and removal of old nodes corresponding to old parts; requires time to set up for initial product but easy to implement changes	Requires interpretation of CAD files; lattice generation time fast for small number of parts, but increases rapidly for larger number. Altering product structures to carry out design changes shown to be feasible in this study
Support for automated visualization of changes	Moderate; DSMs do not usually carry change information but such a feature may be supported by writing programs that highlight new parts and connections	High; immediate snapshot of changes visible within project file; supports importing annotated design changes and function structures	Physical product structure changes can be implemented, but technique to illustrate the changes not yet available. Embedding of function structures is not immediately evident using this approach
Data management issues arising from solution	Leads to a new definition which needs storage and management	Creates limited unified definitions which are easy to manage	Lattice provides a tool to agglomerate different product structures
Ease of interpretation	Need to look-up each combination; cumbersome for large matrices	Easy to interpret as multiple design structures are available within same definition	Visual interpretation of parts and sub-assemblies; large number of parts requires tree editing operations for ease of interpretation
Support for change validation	Requires validation of new matrix to be done for all new elements	Easy validation as embedded data will throw exceptions for non-physical change	Product structure changes easily validated as the lattice only allows valid new product definitions, e.g. a single part may not be part of multiple sub-assemblies
Ability to query revised design for implications of changes	Depends on number and complexities of changes implemented; only small changes may be easy to query and resolve	Enables querying by linking text to design definition; integrates cross-departmental design structures leading to smooth change approval	Lattices by themselves do not provide the ability to query designs for implications of changes—integration with QDA tools necessary
Capturing hidden functional information	Possible but needs adding by engineers during matrix creation	Provides framework for embedding hidden design data	Lattices do not provide this ability yet; integration with QDA tools necessary
Enabling reuse of design definition data	New matrices need to be created for each change request and existing connections may need revalidation	Simple and easy to reuse—nodes can be removed and new nodes added; new sources can be easily imported within existing project file	Simple and easy to reuse—nodes in lattices can be selected and de-selected to create new definitions; lattices can be re-initiated with a different number of parts

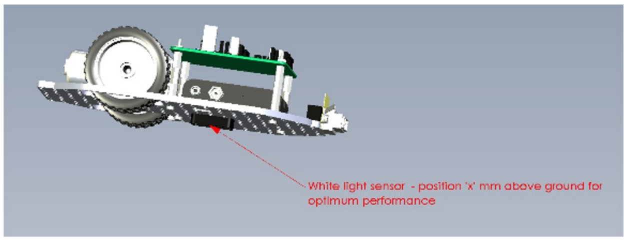

White light sensor - position 'x' mm above ground for optimum performance

Fig. 15 Additional functional requirement captured using annotation

of new BoMs by selecting paths through the complete lattice have been tested. Challenges at this stage involve finding ways to generate new BoMs dynamically from a complete lattice and overcoming the issues related to visualization of large lattices.

This study has also demonstrated that current methods for reusing design definitions, such as DSM, become limited when the granularity of design definitions increase. Furthermore, with rapidly reducing product life cycles, to stay competitive, product development companies need to be able to implement design changes quickly and reliably, ensuring that consequences of a change have been considered. Current matrix-based techniques can enable this, but with high storage requirements of complex designs and their definitions, it becomes costly to maintain multiple digital definitions of the same product in different parts of the company. Embedding design structures within a smaller collection of design definitions provides a potential solution by allowing shape models to mediate across design structures. Furthermore, capturing functional information, and relating it to design definitions, is critical for designing. The example of the gap between the white line sensor and the ground provides a simple example of how shape models such as 3D CAD files alone provide insufficient information for future designers.

This paper has reported research based on two design case studies and explored the potential benefits of embedding using a new software prototype based on lattices and a currently available qualitative data analysis software tool. The results illustrate that embedding offers significant advantages regarding change visualization and validation, data management, interpretation, querying, incorporation of hidden functionalities, design reuse, and automation of change management. However, to realize this potential, significant further work is needed on the implementation of embedding and its integration with existing design solutions. Techniques such as geometric hashing can facilitate pattern discovery in BoMs and matching them to store them in a minimal space. Resolving different data structures in multi-CAD environments is essential to handling multiple design structures simultaneously. Furthermore, new approaches to embedding of function structures are needed to build a robust architecture for faster and reliable new product development.

Acknowledgements The authors would like to thank the Engineering and Physical Sciences Research Council (EPSRC) for Grant number EP/N005694/1 which supported this research, Dr. Jordan Boyle for organizing a national level robotics workshop where the robotic arm was designed and manufactured, and Muhannad Al-Omari and Peter Hayward for help with CAD models of the robots.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Appendix

Definitions of some key terms used in this paper are given below.

Design definition: A design definition is a representation that defines the design of a product. Examples are 2D shape drawings on paper or computer, 3D models as physical prototypes or as a visual representation on computer, and pictures of a product or its prototype.

Design structure: A design structure is an artificial sense-making device used to make physical artefacts (or their definitions) easier to use in engineering processes. Examples are bills of materials, shape grammars, assembly mating conditions, and function structures.

Design model: A design model is a prototype of the design of a product, typically as a 3D representation in either physical or virtual form, either scaled or of nominal dimensions.

References

Ahmad S, Mallick DN, Schroeder RG (2013) New product development: impact of project characteristics and development practices on performance. J Prod Innov Manag 30:331–348

Ashford J, Churcher N, Irwin W (2011) Dynamic visualisation of software state. In: Proceedings of the 34th Australasian computer science conference, Australian Computer Society, Inc., vol 113, pp 127–136

Bartle RG (1964) The elements of real analysis, vol 2. Wiley, New York

Bazeley P, Jackson K (2013) Qualitative data analysis with NVivo. Sage Publications Limited, New Delhi

Behera AK, McKay A, Chau HH, Robinson MA (2016) Embedding multiple design structures into design definitions: a case study of a collision avoidance robot. In: 14th international conference on design, DESIGN 2016, Dubrovnik, Croatia, 16–19 May 2016

Behera AK, McKay A, Chau HH, de Pennington A, Robinson MA (2017) Embedding design descriptions using lattice structures: technical requirements, user perspectives and implementation. In: 6th international conference on research into design. Springer, Singapore, pp 557–566

Bělohlávek R (2004) Concept lattices and order in fuzzy logic. Ann Pure Appl Logic 128:277–298

Cameron PJ (1994) Combinatorics: topics, techniques, algorithms. Cambridge University Press, Cambridge

Chandrasegaran SK, Ramani K, Sriram RD, Horváth I, Bernard A, Harik RF, Gao W (2013) The evolution, challenges, and future of knowledge representation in product design systems. Comput Aided Des 45:204–228

Chau HH (2016) StrEmbed-1—embedding project first prototype. http://www.personal.leeds.ac.uk/~menhhc/embedding/. Accessed 07 Apr 2016

Chau HH (2017) StrEmbed-3—structure embedding (StrEmbed) version 3 release A. https://doi.org/10.5281/zenodo.232162. Accessed 15 Mar 2017

Chen KC (2011) Using general system approach for product lifecycle management software selection and evaluation Rev Bus Inform Syst (RBIS) 13

Cloutier R, Sauser B, Bone M, Taylor A (2015) Transitioning systems thinking to model-based systems engineering: systemigrams to SysML models. IEEE Trans Syst Man Cybern Syst 45:662–674

Corti L (2008) JISC final report: data exchange tools and utilities (DExT) repositories and preservation tools programme. UK data archive. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.680.4085&rep=rep1&type=pdf. Accessed 17 May 2017

Danilovic M, Browning TR (2007) Managing complex product development projects with design structure matrices and domain mapping matrices. Int J Project Manag 25:300–314

Davies D, McMahon CA (2006) Multiple viewpoint design modelling through semantic markup. In: ASME 2006 international design engineering technical conferences and computers and information in engineering conference, 2006. American Society of Mechanical Engineers, pp 561–571

Demoly F, Dutartre O, Yan X-T, Eynard B, Kiritsis D, Gomes S (2013) Product relationships management enabler for concurrent engineering and product lifecycle management. Comput Ind 64:833–848. https://doi.org/10.1016/j.compind.2013.05.004

Ding L, Ball A, Matthews J, McMahon C, Patel M (2009a) Annotation of lightweight formats for long-term product representations International. J Comput Integr Manuf 22:1037–1053

Ding L, Davies D, McMahon CA (2009b) The integration of lightweight representation and annotation for collaborative design representation. Res Eng Des 20:185–200

Eckert C (2013) That which is not form: the practical challenges in using functional concepts in design artificial intelligence for engineering design. Anal Manuf 27:217–231

Engel A, Reich Y (2015) Advancing architecture options theory: six industrial case studies. Syst Eng 18(4):396–414

Engel A, Reich Y (2013) Architecting systems for optimal lifetime adaptability. In: DS 75-4: proceedings of the 19th international conference on engineering design (ICED13), design for harmonies, vol 4: Product, service and systems design, Seoul, 19–22 Aug 2013

Eppinger SD, Browning TR (2012) Design structure matrix methods and applications. MIT Press, New York

Freese R (2013) Lattice drawing. http://www.math.hawaii.edu/~ralph/LatDraw/. Accessed 01 Apr 2016

Gero JS (1990) Design prototypes: a knowledge representation schema for design. AI Mag 11:26

Grasl T, Economou AGRAPE (2011) Using graph grammars to implement shape grammars. In: Proceedings of the 2011 symposium on simulation for architecture and urban design. Society for Computer Simulation International, pp 21–28

Grätzer G (2011) Lattice theory: foundation. Springer, New York

Habib T (2014) Multidisciplinary product decomposition and analysis based on design structure matrix modeling. In: Brunoe TD, Nielsen K, Joergensen KA, Taps SB (eds) Proceedings of the 7th world conference on mass customization, personalization, and co-creation (MCPC 2014), Aalborg, 4th–7th Feb 2014. Twenty years of mass customization—towards new frontiers. Springer, Cham, pp 409–423. https://doi.org/10.1007/978-3-319-04271-8_35

Hager G, Wellein G (2010) Introduction to high performance computing for scientists and engineers. CRC Press, Boca Raton

Helms B, Shea K, Hoisl F (2009) A framework for computational design synthesis based on graph-grammars and function-behavior-structure. In: ASME 2009 international design engineering technical conferences and computers and information in engineering conference, 2009. American Society of Mechanical Engineers, pp 841–851

Hepperle C, Maier AM, Kreimeyer M, Lindemann U, Clarkson PJ (2007) Analyzing communication dependencies in product development using the design structure matrix. In: DSM 2007: proceedings of the 9th international DSM conference, Munich, 16–18 Oct 2007

Herzig SJI, Qamar A, Paredis CJJ (2014) An approach to identifying inconsistencies in model-based systems engineering. Proc Comput Sci 28:354–362

Kashkoush M, ElMaraghy H (2013a) Product design retrieval by matching bills of materials. J Mech Des 136:011002–011002. https://doi.org/10.1115/1.4025489

Kashkoush M, ElMaraghy H (2013b) Matching bills of materials using tree reconciliation. Proc CIRP 7:169–174

Kerley W, Wynn DC, Eckert C, Clarkson PJ (2011) Redesigning the design process through interactive simulation: a case study of life-cycle engineering in jet engine conceptual design. Int J Serv Oper Manag 10:30–51

KETIV I (2014) Top 10 product lifecycle management software comparision. Business-Software.com. http://ketiv.com/files/Top%2010%20PLM%20Report%202014_0.pdf. Accessed 11 Apr 2017

Kidd M, Thompson G (2000) Engineering design change management. Integr Manuf Syst 11:74–77

Kim S-J, Suh NP, Kim S-G (1991) Design of software systems based on axiomatic design. Robot Comput Integr Manuf 8:243–255

Kim K-Y, Manley DG, Yang H (2006) Ontology-based assembly design and information sharing for collaborative product development. Comput Aided Des 38(12):1233–1250

Leech NL, Onwuegbuzie AJ (2007) An array of qualitative data analysis tools: a call for data analysis triangulation. Sch Psychol Q 22:557

Lewis RB (2004) NVivo 2.0 and ATLAS. ti 5.0: a comparative review of two popular qualitative data-analysis programs. Field Methods 16:439–464

Liberati M, Munari F, Racchetti P, Splendiani T (2007) Social network techniques applied to design structure matrix analysis. The case of a new engine development at Ferrari. In: DSM 2007: proceedings of the 9th international DSM conference, Munich, 16–18 Oct 2007

Lindemann U (2009) Design structure matrix. Institute of Product Development, Technische Universität München. http://www.dsmweb.org/. Accessed 19 Nov 2015

Liu W, Zhou X, Zhang X, Niu Q (2015) Three-dimensional (3D) CAD model lightweight scheme for large-scale assembly and simulation. Int J Comput Integr Manuf 28:520–533. https://doi.org/10.1080/0951192X.2014.880811

March L (1996) The smallest interesting world? Environ Plan 23:133–142

McKay A, Hagger DN, Dement CW, de Pennington A, Simons P (2004) Relationships in product structures. In: DS 56: proceedings of the 7th workshop on product structuring–product platform development, Chalmers University, Göteborg, 24–25 Mar 2004

McKay A, Chase S, Shea K, Chau HH (2012) Spatial grammar implementation: from theory to useable software artificial intelligence for engineering design. Anal Manuf 26:143–159

McKay A, Stiny GN, de Pennington A (2015) Principles for the definition of design structures. Int J Comput Integr Manuf 1–14

McMahon CA (2015) Design informatics: supporting engineering design processes with information technology. J Indian Inst Sci 95(4):365–377

Neely A (2008) Exploring the financial consequences of the servitization of manufacturing. Oper Manag Res 1:103–118

Pahl G, Beitz W (2013) Engineering design: a systematic approach. Springer, New York

Pattison T, Ceglar A (2014) Interaction challenges for the dynamic construction of partially-ordered sets. In: CLA, 2014, pp 23–34

Plossl GW, Orlicky J (1994) Orlicky's material requirements planning. McGraw-Hill Professional, New York

QSR (2014) NVivo for windows: getting started. http://download.qsrinternational.com/Document/NVivo10/NVivo10-Getting-Started-Guide.pdf. Accessed 01 Dec 2015

Rinderle JR, Suh NP (1982) Measures of functional coupling in design. J Eng Ind 104:383–388. https://doi.org/10.1115/1.3185846

Robinson MA (2010) An empirical analysis of engineers' information behaviors. J Am Soc Inf Sci Technol 61:640–658

Sandberg S, Lundin M, Näsström M, Lindgren L-E, Berglund D (2013) Supporting engineering decisions through contextual, model-oriented communication and knowledge-based engineering in simulation-driven product development: an automotive case study. J Eng Des 24:45–63

Schmidt J, Rudolph S (2016) Graph-based design languages: a Lingua Franca for product design including abstract geometry. IEEE Comput Graph Appl 36:88–93

Song I-H, Chung S-C (2009) Synthesis of the digital mock-up system for heterogeneous CAD assembly. Comput Ind 60:285–295

Srinivasan V (2011) An integration framework for product lifecycle management. Comput Aided Des 43:464–478

Stark J (2015) Product lifecycle management. In: Product lifecycle management. Springer, New York, pp 1–29

Steward DV (1981) The design structure system: a method for managing the design of complex systems. IEEE Trans Eng Manag 71–74

Stiny G (2008) Shape: talking about seeing and doing. The MIT Press, New York

Stiny G (2011) What rule (s) should I use? Nexus Netw J 13:15–47

Stouffs R (1994) The algebra of shapes. Carnegie Mellon University, Pittsburgh

Stouffs R (2008) Constructing design representations using a sortal approach. Adv Eng Inform 22:71–89

Tang D, Zhu R, Tang J, Xu R, He R (2010) Product design knowledge management based on design structure matrix. Adv Eng Inform 24:159–166

Tompkins C (1939) Isometric embedding of flat manifolds in Euclidean space. Duke Math J 5:58–61

Vermaas PE (2013) The coexistence of engineering meanings of function: four responses and their methodological implications artificial intelligence for engineering design. Anal Manuf 27:191–202

Vollrath I (1998) Reuse of complex electronic designs. In: Advances in case-based reasoning. Springer, New York, pp 136–147

Wille R (1992) Concept lattices and conceptual knowledge systems. Comput Math Appl 23:493–515

Wille R (2005) Formal concept analysis as mathematical theory of concepts and concept hierarchies. In: Formal concept analysis. Springer, New York, pp 1–33

Yang Q, Yao T, Lu T, Zhang B (2014) An overlapping-based design structure matrix for measuring interaction strength and clustering analysis in product development project. IEEE Trans Eng Manag 61:159–170

Yoo SB, Kim Y (2002) Web-based knowledge management for sharing product data in virtual enterprises. Int J Prod Econ 75:173–183. https://doi.org/10.1016/S0925-5273(01)00190-6

Zortrax SA (2016) Zortrax—case studies—3D printed dancing robots. https://zortrax.com/case-studies/. Accessed 08 Apr 2016