APPLYING CONNECTIVISM TO PRODUCT KNOWLEDGE THROUGH GRAPH THEORY AND FILTERING

Joel Johansson*, Fredrik Elgh, Manuel Contero, Pedro Company

ABSTRACT

Mass-customization has forced manufacturing companies to put significant efforts to digitize and automate their engineering and production processes. When new products are to be developed and introduced not only has the production processes to be automated but also the application of knowledge regarding how the product should be designed and produced based on customer requirements. One big academic challenge is to help the industry to make sure that the background knowledge of the automated engineering processes still can be understood by its stakeholders throughout the product life cycle.

The research presented in this paper aims to build an infrastructure to support a connectivistic view on knowledge. Fundamental concepts in connectivism include network formation and contextualization, which here is addressed by the utilization of graph theory together with information filtering techniques. The paper shows how engineering content in spreadsheets, knowledge-bases and CAD-models can be penetrated and represented as filtered graphs to support a connectivistic working approach. Three software demonstrators developed to extract graphs and applying filters to such engineering content are presented and discussed in the paper.

Keywords: Connectivism, knowledge management, knowledge-based engineering, graph theory

1 Introduction

Engineering knowledge refers to the knowledge engineers apply when they are involved in developing products and corresponding production systems. This is a broad definition with emphasis on applying, which means that the knowledge is part of decision-making processes and hence excludes curiosities. Engineering knowledge further refers to any reason for why, how, when, where, what, by whom something is to be done or be constituted. The sum of engineering knowledge formally represented for one product is referred to as product knowledge and may reside in all available representations of the product.

Mass customization has been a steady driving force to capture and automatically utilize such engineering knowledge. Many companies have, for instance, put significant efforts to parametrize (to nearly "automate") CAD-models to quick and accurately respond to changes in product requirements and specifications. This has caused engineers to not only focus on developing single products but product families with wide and flexible design spaces. Parallel to parametrized CAD-models, knowledge management and knowledge-based engineering (KBE) have for decades strived to capture, digitize, and automate the application of this kind of knowledge within product and production development. Even if KBE-systems have gained much attention through the last three decades, industries still found them hard to develop and even harder to maintain over time. Here it is suggested to take a connectivistic view on knowledge to contribute on how it may help industries to maintain their product knowledge continuously.

The paper is organized as follows. First, a frame of reference is presented where we shortly review how product knowledge is formalized, introduce connectivism, graph theory, and information filtering. After that, we include a detailed description of how three of the most common knowledge carriers in manufacturing companies are constituted and how their constituents are connected. Then three examples are described, two where product knowledge was penetrated to gain understanding of how relations in parametric CAD-models connected to KBE-systems or spreadsheets and one where a CAD-model was penetrated to evaluate its modelling quality. Finally, the results are discussed followed by conclusions.

2 Frame of reference

In this section, we will briefly review how product knowledge is commonly represented, followed by a study on connectivism that is a perspective on knowledge that has served as guidance through the development of the proposed working approach that is supported by tools and methods, which are also presented in this paper. The connectivistic perspective is described in the context of knowledge and learning in general. Finally, short introductions to graph theory and filtering are given as support for subsequent sections.

2.1 Formalized product knowledge

Engineering design problems are solved through iterations between synthesis and analysis phases. Design proposals developed through creative processes are evaluated against requirements. During new product development processes, much of the trial and error loops occur as a part of the learning of how to solve the problems connected with the product. The knowledge developed and formalized during these trials is referred to as product knowledge [1]. When the product matures, a set of tested solutions will emerge, a set that is reviewed when customer demands the product for a different

* Author of correspondence, Phone: (+46) 36-101675, Email: joel.johansson@ju.se.

set of requirements. When such a set of solutions exists the synthesis phase gradually turns into a search for existing solutions that can be combined to solve new problems. Also, when the product matures the way of testing the product to requirements is formalized and can sometimes be skipped based on an inductive way of reasoning, i.e., based on experience it can be concluded that the new solution will fit. Proven and formalized knowledge can be handled as a design platform as described in [2]. In the most mature state of a mass-configured product, the processes for developing a variant are well defined and based on user requirements. In such state, computer supports can be utilized in synthesis and analysis phases to a great extent, which is referred to as knowledge-based engineering (KBE).

KBE is a method to synthesize design proposals and has been defined as a technology based on the use of dedicated software able to capture and systematically reuse product and process engineering knowledge, with the final goal of reducing time and costs of product development by automation of repetitive and non-creative design tasks, and support for multidisciplinary design optimization in all the phases of the design process [3]. Engineering problems are typically ill-structured [4] because they lack clear ways of testing any suggested solution, the states of the problem are hard to represent, and the knowledge of the problems is hard to capture and represent. This fundamental property of engineering problems has compelled the use of artificial intelligence to automate engineering design processes. Traditionally KBE is based on knowledge-based-system from artificial intelligence where production rules are the primary carriers of the captured knowledge.

KBE can be seen as an interdisciplinary joint mixture of disciplines, as long as it can be said to be the integration between artificial intelligence and computer-aided engineering, see Figure 1. Artificial intelligence is a set of methods and models from the computer science research field that support flexible modelling of concepts and techniques for logical reasoning, while computer-aided design includes methods and models to model geometry and product structures.

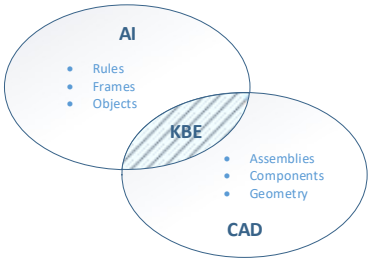

Figure 1: KBE systems from a technical perspective: computer programs containing knowledge and reasoning mechanisms augmented by geometry handling capabilities to provide engineering design solutions. Adapted from [3]

Finally, knowledge-based engineering can be seen in the context of knowledge management further and that KBE is a subset of knowledge engineering which is a subset of knowledge management [3], see Figure 2.

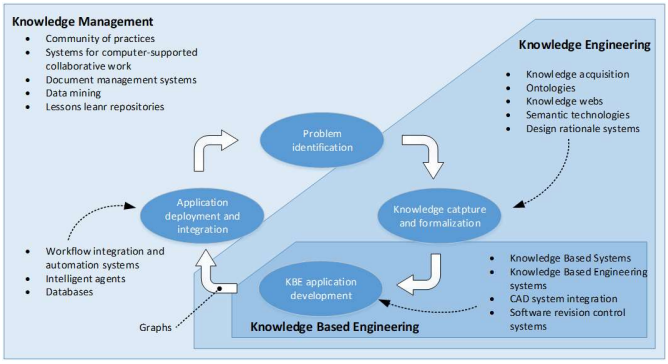

Figure 2: Components of knowledge management including knowledge-based engineering. Adapted from [3].

As can be concluded from these definitions, CAD-models are part of the represented engineering knowledge. They result from applying the engineering knowledge, and define how the product is to be constituted and embodied. The logic and rules can be embedded into CAD-models but also stored in spreadsheets or knowledge-bases with facts and rules.

2.2 Connectivism

Connectivism is a philosophy of knowledge described by Siemens [5]. It has been defined as "the thesis that knowledge is distributed across a network of connections" [6], and address learning that is located within technology and organizations, a learning that KBE ultimately is intended to support. Connectivism is based on nine principles (described but not numbered by Siemens [5]):

1. Learning and knowledge require a diversity of opinions to present the whole…and to permit selection of best approach.
2. Learning is a network formation process of connecting specialized nodes or information sources.
3. Knowledge rests in networks.
4. Knowledge may reside in non-human appliances, and learning is enabled/facilitated by technology.
5. Capacity to know more is more critical than what is currently known.
6. Learning and knowing are constant, on-going processes (not end states or products).
7. Ability to see connections and recognize patterns and make sense between fields, ideas, and concepts is the core skill for individuals today.
8. Currency (accurate, up-to-date knowledge) is the intent of all connectivistic learning activities.
9. Decision-making is learning. Choosing what to learn and the meaning of incoming information is seen through the lens of a shifting reality. While there is a right answer now, it may be wrong tomorrow due to alterations in the information climate affecting the decision.

Five components are identified within connectivism. Networks are where knowledge resides. Conduit, context, and content together shape the meaning of knowledge and individualized filters to help to focus. These components are briefed in the following subsections.

2.2.1 Networks

Central in the connectivistic view on knowledge is that learning is a network formation process [5]. In the knowledge technologies, as seen in Figure 2, this is realized through the community of practices, systems for computer-supported collaborative work, ontologies and knowledge webs. Interestingly, these technologies are not considered to be a part of knowledge-based engineering system.

2.2.2 Context

Context in the connectivistic view includes elements like emotions, recent experiences, beliefs, and the surrounding environment. Each element possesses attributes, which when considered in a certain light, inform what is possible in the discussion. The object is tied to the nature of the discussion, framework or network of thought. The context-game is the formulation and negotiation of what will be permissible and valued, and the standards to which we will appeal in situations of dispute. The context-game of implementing a new corporate strategy involves individuals, politics, permissible ways of seeing and perceiving, recent events, corporate history, and a multitude of other factors [5]. Context in this broad definition is not typically considered in theories for knowledge management, knowledge engineering, and KBE. However, context influences how the knowledge is implemented in KBE and how it can be understood by stakeholders.

2.2.3 Conduit

Conduits are the mediums through which knower (i.e., experts) and seeker (i.e., knowledge consumers) communicate and through which the known entity finds expression [5]. Conduits are the facilities making the knowledge relevant, current, and available. In manufacturing companies, these conduits today include PLM-systems, intranet, wikis, and shared file servers.

2.2.4 Filters

Siemens [5] briefly reviews the history of how information has been consumed and concludes that we used to go to one source of information to get a thousand points of information (for instance newspapers). Now, we go to a thousand sources of information to create our own view. He continues by saying that we have become the filter, mediator, and the weaver of the networks. A statement that indicates how intervened the concepts in connectivism are.

Since we as humans have a limited possibility to focus our attention (we can only do one or a few things at a time, and we just have a limited time per day) and since the amount of information and knowledge is ever increasing there is a great need for filter the content based on individualized filters and current context (as defined previously).

2.2.5 Content

Content is of course of central importance (even if it is told that the capacity of learning is more important than what we already know). Relevance, however, is not only about the nature of the content. The process of ensuring currency of content/information is critical to managing knowledge growth and function effectively. Content has to blend with conduit and context [5] which means that content should be perceived to be very close. Engineers today put much time to seek for content, but rather the content should seek for the engineers.

2.3 Introduction to graphs and filtering

A graph G(N, E) is a set of nodes (N) and edges (E). The nodes represent entities of interest, and the edges represent how they are connected as tuples of two nodes, where the first one is the source node, and the last one is the target node [7]. When two nodes are connected through an edge, they are said to be neighbors. The degree of one node is defined as the number of neighbors it has, i.e., how many edges are pointing in and out from it. In-degree refers to how many neighbors a node depends on (parent), and out-degree how many neighbors are depending on it (child). Nodes and edges can be labeled indicating what type of entity a node represents and what kind of relation an edge represents. Further, attributes can be assigned to nodes and edges. For nodes, attributes indicate what state the represented entity has, and for edges, the attributes indicate what state the relation between two nodes has.

In the context of this paper, filtering means retrieving graphs by reducing or combing datasets, which is achieved by the application of set theory. The fundamental set of Boolean operations (union, difference, and section) can be applied to nodes and edges and combined so as to produce queries. If the underlying data is stored in a relational database, it is possible to use SQL for filtering. However, a growing number of graph databases and graph querying languages under development support extensive ways of managing and filtering large graphs applying graph theory. The latter method of storing data has proven to outperform relational databases when the data is highly connected [8]. Graph databases are applicable when the connections between the stored entities are equally or more important than the entities themselves. In the examples presented in this paper, Neo4j was used to store the graphs, Cypher [9] was used as querying language, and Gephi [10] was used for visualization (due to the need to cope with big graphs).

3 Dissecting the constituents of engineering knowledge in parametric design

In this study, we focus on parametric CAD-models as such, and parametric CAD-models controlled by spreadsheet applications or by KBE-systems (or both). There are three ways in which a parametric CAD-model (further on it is assumed CAD-models to be parametric) can be controlled. First, spreadsheets can be connected to CAD-models to define family tables, i.e., sets of similar components derived from a single parametric CAD-model. That is an efficient way of handling parametric design of, for instance, fasteners, washers or other standard components. Second, KBE-systems can be connected to CAD-models to execute rules that update the CAD-models. This is useful when handling components and assemblies that are free to change based on customer requirements. A third way is to add formulas into spreadsheets and add them to the geometry rebuild process. In this latter case, the spreadsheet application turns into a KBE-system.

We will take a close look at three types of engineering content, CAD-models, spreadsheets, and KBE-systems to find

their fundamental constituents and to see how they are connected. Then we apply graph theory and filters to make a foundation to further studies in knowledge and complexity management.

3.1 CAD-models

The information model of a CAD-model can be constituted in many ways and differs between CAD-systems (therefore neutral CAD-formats are needed). Still, in Figure 3 a schematic information model shows the commonly agreed fundamental constituents. An **assembly** is, as seen in the figure, composed of **instances** of **parts** which can either be **components** or **assemblies** (the terminology used in the literature differs somewhat between what is a part and what is a component, here a component is a piece made from one material, a part can be either one component or a composition of components). **Components** are made up from at least one **feature** while **assemblies** may contain **features** or not. **Components** and **assemblies** may comprise **parameters** which are carriers of base type data such as Booleans, integers, doubles or text values packed with a name. **Assemblies** and **components** may comprise **equations**, which are mathematical expressions involving **parameters**. The most common type of **feature** in CAD-models is **geometrical features**. **Geometrical features** are the composition of **entities** which may be one-, two- or three-dimensional geometrical elements, such as points, lines, curves, planes, and surfaces. **Geometrical dimension or constraints** are a particular type of **parameters** that make reference to geometrical **entities** to control their definitions.

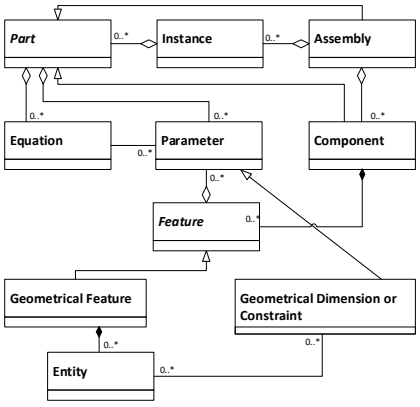

Figure 3: Information model of CAD-model.

There are several types of relations in a CAD-model. As seen from Figure 3, components are related to assemblies as "part-of" relations. Features, parameters, and equations are also "part-of" components, and entities are "part-of" features. These easy-to-understand relationships are often visualized in the CAD-system through a "model tree". A typical model tree is shown in Figure 4, which contains a top assembly (Assembly1) having three instances of parts (2 SubAssembly1 and 1 SubAssembly2). SubAssembly1, in turn, is constituted by two instances of Component1, which is made up from Feature1 and Feature2 (Note that not all CAD-systems show the instance-level in the model tree, i.e., the nodes SubAssembly1 (1), SubAssembly1 (2), and SubAssembly2 (1)).

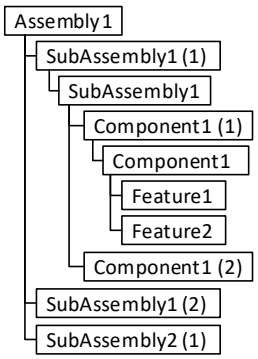

Figure 4: A typical model tree in CAD-systems only show "part-of" relations.

Other than "part-of" relations exist in CAD-models which are not that obvious but are interesting to engineers when developing and maintaining the models. An example of such type of relationship is the references between geometrical entities conveyed through geometrical dimensions or constraints. The geometrical engines efficiently manage local references but crossed references between entities belonging to different features result in hierarchical dependencies named as parent/child relations (where, for instance, deleting the parent feature will cancel the child one). Another example of relationships important to engineers involves the intent of the CAD-model. Most CAD-systems allow adding logic to the CAD-model through equations, which may govern the parameters (like the number of instances of a repetitive pattern), the geometrical dimensions (that govern the size or even the topology of the CAD-model), or other geometrical relationships (like Boolean flags that may govern the parallelism or perpendicularity between two entities). We name them as mathematical relations. These relations could also be viewed as parent/child relations if they are unary expressions. The relationships are modeled as edges in the graphs per Table 2.

3.2 Spreadsheets

Spreadsheets are frequently used within engineering design to store and manage information regarding the product and are a part of the product model. Spreadsheets may, as mentioned, be connected to CAD-models as design tables or as a part of the geometrical build process as an "analysis" feature. The reason for adding spreadsheets as a part of the product model is the flexibility to model information. The central concept in spreadsheet applications is the **cell**. In Figure 5 a schematic information model of a spreadsheet is drawn. **Cells** reside, as seen in Figure 5, in **worksheets** and **worksheets** reside in **workbooks**. A cell may contain a **formula** that refers to other cells. Formulas act as functions with several possible input cells but with one output only, which is displayed in the cell to which it belongs.

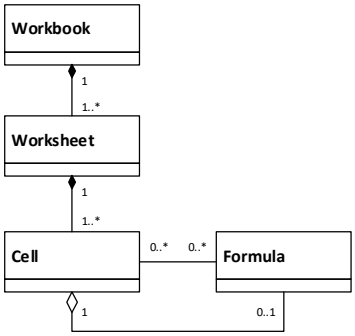

Figure 5: Information model of spreadsheets.

When connected to a CAD-model we can view cell values as facts, and formulas as rules in a KBE-system according to the definition of KBE in [3], where the inference engine is then realized by the spreadsheet application (which automatically updates the cell and displays the result when the linked cells are modified).

3.3 Knowledge-bases

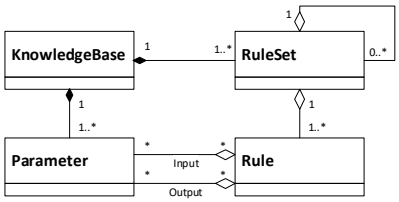

Figure 6: Information model of knowledge-bases.

KBE-systems gained attention during the last decade and have been integrated to several CAD-system. Standalone KBE-systems also exist. The most interesting class in a KBE-system is the **Rule** class. **Rules** reside, as seen in Figure 6, in **rulesets**, while **rulesets** reside in **KnowledgeBase**. **Rules** consist of **parameters** and act as functions with one or several input **parameters** and one or several output **parameters**. Note that the parameters in Figure 3 are a subclass of parameters in Figure 6 since the parameter class in a knowledge-base tends to be more general than in a CAD-model.

3.4 Graphs and filters

The constituents of CAD-models, spreadsheets, and knowledge-bases; and their relations together form networks that can be represented as graphs. These graphs can be filtered, and visualized to gain insight to the model. They can also be used as a foundation for navigating rationale as described in [11]. To achieve all that, it is useful to add attributes to nodes and edges, to make the graph meaningful. Attributes attached to the nodes and edges in this paper are listed in Table 1.

The **EdgeType** attribute is what distinguishes the graphs presented in the paper from the usual model-trees in CAD-systems, as it makes it possible to model how the entities are connected. When reviewing the class diagrams in Figure 3, Figure 5, and Figure 6 five different types of couplings are found, these are listed in Table 2.

The values in the first column of that table are used as valid values for the **EdgeType** attribute, as much as labels for the edges.

Table 1: Four attributes were added to develop the graphs in this paper.

Name	Applies to	Description
URI	Nodes	Unique Resource Identifier. Includes file path and the internal path to the represented entity.
Label	Nodes, Edges	Text to show in the graph.
EntityType	Nodes	Type of entity
EdgeType	Edges	Type of relation as in Table 2

Table 2: Five types of relations are identified within CAD-models (the two marked by * are the only relation types that are undirected). Connected constituents are defined in Figure 3 and Figure 5.

Relation type	Connected constituents	Realized by
Part-of	Entity → Feature	Feature entities
Part-of	Feature → Component	Component features
Part-of	Component → Assembly	Assembly instances
Kind-of	Instance → Part	Instance
Connection	Entity → Feature	References in feature
Mathematical	Parameter → Parameter, Cell → Cell	Expressions in equation, Formula
Connection*	Entity ↔ Entity, Part ↔ Part	Geometrical constraints
Spatial*	Entity ↔ Entity	Location

Filtering of information in the graphs is what realizes the contextualization of knowledge, and—due to the innovative nature of design—there will never be a set of universal filters; however, some filters may be more general or more frequently used than others. Here we suggest three sets filters. The first set of filters is the combination of retrieving nodes of types **Geometrical Dimension**, **Parameter** and **Cell** (i.e., nodes representing entities that control the design) and that are of degree 1. The entities represented by the retrieved node of such filters are the entry points for the CAD-~~,~~ model: changing any of their values impacts the design. In everyday language, we call these entities design parameters.

The second suggested set of filters involves filtering on **Part-Of**, and **Kind-Of** edges which yields the model tree as represented in the CAD-systems (it is a combination of the "Assembly-tree" and "Part-tree") which can be used as a support to evaluate the complexity of CAD-models.

The third set of filters involves filtering edges on **External connections** yields the interfaces between the CAD-model and design tables. Adding edges of types "Mathematical" gives the entire set of logic for the CAD-model, which is the third suggested set of filters. The nodes resulting from that filter represents the logical part of the product model.

3.5 Rendering graphs

To get a seamless overview of the engineering content of CAD-models and their connected spreadsheets and knowledge-bases, they can be analyzed using the theory in the previous

sections to render graphs. In our implementation, there are three algorithms used to do the analysis, the first one traverses the CAD-model, and its sub-models to extract the nodes and edges. The second algorithm analyses any spreadsheet connected to the CAD-models to extract nodes and edges within them and in connection to the CAD-model. The third one loops through all rules and connected parameters to extract nodes and edges.

A simple example demonstrates the output from these algorithms. Let say we have a CAD-model containing a box with a hole through (see left of Figure 7). The model-tree contains merely two extrude features, one for the box and one for the hole. The two features are based on their corresponding parametrized sketches, which are shown in the model tree (see the center of Figure 7). The dimensions of the box are controlled by a design table, so that height, width, and length are controlled by the cells in a connected spreadsheet. Two configurations (or product instances) are defined (Default and Config1). The set of values for the first configuration is in turn controlled by a "base value" that is a sort of "scale" that applies to all the three parameters and is stored in a cell in another spreadsheet (see the right side of Figure 7).

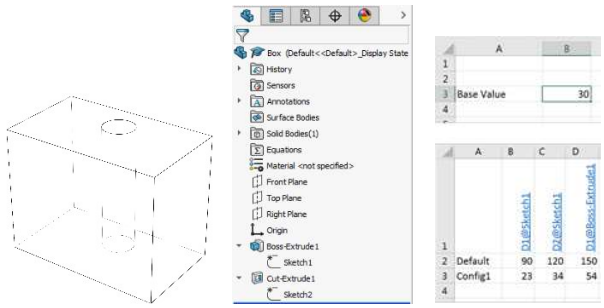

Figure 7. Simple CAD-model of a box with a hole with a corresponding model tree (left and center) and a connected spreadsheet, base value controls Default configuration (right).

Applying the algorithms to the simple box-with-hole example yields a graph with 22 nodes and 28 edges. The graph is visualized in Figure 8 with the Yifan Hu layout algorithm [12]. There are seemingly duplicate nodes (for instance two D1) this is because of using short names on the nodes (Label attribute). The nodes are unique with the URI, thus, displaying the full names would solve this apparent duplicity.

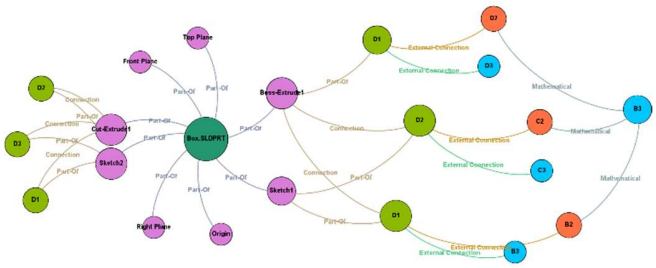

Figure 8: Graph illustrating the relations in the CAD-model, design table and in between. 22 Nodes and 28 relations.

Applying a filter to retrieve design parameters results in 7 design parameters: the nodes D1, D2 and D3 (green) to the left and B1, B2, B3 and the other B3 (blue) to the right in Figure 8. This is interesting as it indicates that every value in a design table is indeed a design parameter. Applying a filter to retrieve part-of relations yields two graphs one with the CAD-model and one with the spreadsheet. Using a filter to extract the mathematical model yields a graph with nodes B2, C2, D2 (orange) and B3 (blue), which represent the logical part of the product model.

The next two sections describe three case examples where product knowledge was elicited using the tools and methods presented so far. The first case example illustrates how graph representations may help to find where to reduce the complexity of parametrized models, thus preventing catastrophic failures and/or unacceptable calculation delays. The second case example illustrates the complexity of this problem, since even an apparently simple part may convey a massive amount of design information, which will result in a very complex graph that would require powerful yet easy to use filtering capabilities. The third case example illustrates the eliciting knowledge about the CAD model quality from the complexity analysis of the parent/child graph in CAD models. This type of graph is representative of one subset of the five types of relations identified by our method within CAD-models and was used to devise engineers how to develop CAD-models that can be efficiently reused and controlled by spreadsheets and KBE-systems.

4 Eliciting knowledge about relations in parametric design using graphs

To verify the concepts presented in the previous sections a prototype software was developed and applied to two real-life examples: hot runners for injection molding, and truss joints in earthquake-proof buildings. These cases illustrate how graph rendering and filtering can be applied to gain knowledge about CAD-models that are complex regarding variants and regarding numbers of rules.

4.1 Hot runners for injection molding

The first company where the software was tested develops and manufactures heated runner systems for injection molding of plastic components. The company has automated their engineering processes to some extent as described in [13]. One reason to analyze the company's CAD-models was that the product is suitable for design automation, in spite that every produced hot runner system is unique. The runner systems differ in layout. See for instance Figure 9, where an example of the H-shaped layout is shown. There are also X-shaped layouts, circular layouts, and custom layouts. The runners are connected to the tooling cavity through in-gates. The number of in-gates is up to 48 for a single system. There are five series of in-gates that all can have two different types of bushings (with lengths ranging up to 600 mm), and have up to 9 different types of end caps. This led to the automation of the CAD-models based on parametrized families managed through interconnected spreadsheets used as design tables. Due to the combinatorial complexity of the product, mathematical equations were added

as equations in the CAD-models as well as in the spreadsheets. This approach made the CAD-system unstable so that when adding several instances of the in-gates (the protruding parts in Figure 9 and CAD-model in Figure 10) it eventually crashed after a long time (sometimes up to 40 minutes) of number crunching. It did not always crash, so the engineers tended to wait and hope for it to go through. The crashing problems were eliminated when reforming the CAD-models as described in [13]. In the subsections, we describe the CAD-models, spreadsheets and the results of rendering and filtering graphs.

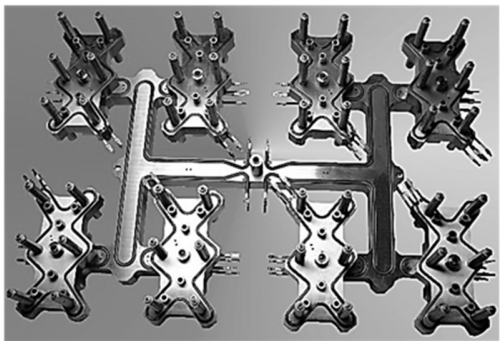

Figure 9: Hot runner system with 48 gates in X-layout [13].

Figure 10: This seemingly small and easy CAD-model caused the CAD-system to crash repeatedly.

4.1.1 CAD-models

The CAD-model targeted is created in Solidworks and consists of five sub-parts of which only one is an assembly (which in turn consist of three components). In total, we are talking about seven components in two assemblies, which seems very little to make a CAD-system to collapse. The vast number of variants for each component is what makes it so difficult. Besides, the product is to operate in much higher temperature than it is produced, which resulted in several temperature configurations for each component (although this was not considered in the test reported here). The top-level assembly contained 42 equations controlling features on all levels and was connected to a spreadsheet as a design table.

4.1.2 Spreadsheet

The spreadsheet connected to the CAD-model as a design table was a Microsoft Excel spreadsheet and included all possible combinations of the part components. The combinations were added using formulas in the spreadsheet so that design table updates when changing certain cell values (this method works for engineer-to-order products). In total, the design table contained 1248 cells, and there were additionally 46 cells with formulas to adjust the values in the design table.

4.1.3 Graph extraction and content filtering

The prototype software, which was developed in the Microsoft .NET platform, took 2 minutes to generate the entire graph. The output data was stored in Neo4j graph-database. This database was queried using the Cypher querying language for graphs. The resulting graphs from the queries (the three filters mentioned previously) were exported as Graph Modelling Language (.GraphML) which is a general, XML-based language, to store graphs in a standardized way [14]. To visualize the graphs several freely available software applications were tested. Due to the size of the graph Gephi [15] with the Yifan Hu [12] and Force Atlas 2 [10] layout routines proved to work. The graph contains 3932 nodes (47% formula, 28% geometrical dimension, 17% feature, 2.9% instance, 2.2% cell, 0.7% Component, 0.6% Assembly, 0.2% parameter) connected in 11321 relations.

When applying the filters, it showed that there are 1165 entry points of which the majority are of Geometrical Dimension type, further filtering shows that there are 12 parameters and 86 cells. Although the academic experiment proved the usefulness of the approach, for industrial use, these entities should be managed, and the information regarding them should be made accessible to retrieve to engineers. Filtering for product structure made it possible to automatically retrieve a product-variant-master [16]. Seven subgraphs occurred when filtering for mathematical relations four subgraphs represented equations in the CAD-model and three represented formulas in the spreadsheet.

By rendering, visualizing, and filtering graphs from the CAD-model, it was possible to identify all entry points, simplifying the product structure and structuring the represented knowledge in equations in the CAD-model and its connected spreadsheet. Thus, the graph proved useful to give a clue of why the parametrized model became so difficult to handle.

4.2 Truss joints in earthquake-proof buildings

Another prototype software was developed and applied to an industrial company that develops and manufactures power plant solutions. The power plants partially consist of steel trusses, which must be earthquake proof. The company tested to automate the construction of truss joints to cut lead-time. The result of the tests was a set of CAD-models with such a huge number of rules embedded into them that it was perceived hard to manage their design content. In this study one CATIA CAD-model consisting of one component was targeted. This seems very little and easy to overview. Still, the vast number of rules for each component (stored in the CAD-model) makes it difficult to manage the CAD-model. The CATIA Knowledgeware KBE-rules connected to the CAD-model as formulas, checks, and design tables included all possible combinations of the component.

4.2.1 Graph extraction and content

In total, the knowledge-base consists of 82 formulas, seven rules, four checks and three design tables. The prototype software, developed in the Microsoft .NET platform, took 469 seconds to generate the entire graph. The output from the routine was stored in Neo4j graph-database. This database was

queried using the Cypher querying language for graphs. The resulting graphs from the queries (the three filters mentioned previously) were exported as Graph Modelling Language (.GraphML) which is a general, XML-based language, to store graphs in a standardized way [14]. Gephi was used to visualize the graphs.

The graph contains 435 nodes connected in 824 relations. When applying the filters, there are 15 entry points of which the 12 are of Geometrical Dimension type, 2 String parameters and 1 Integer parameter. Again, although the academic experiment proved the usefulness of the approach, for industrial use, these entities should be managed, and the information regarding them should be made easy to retrieve to engineers. However, the problem is challenging, as the logical model consists of 129 nodes and 125 mathematical connections. However, when applying the Force Atlas 2 [10] layout, it was seen that the logical model could be grouped into 16 clusters of which 5 clusters where non-sense since they only contain rules for the transition of KBE parameter to CAD dimension.

By applying the algorithms to render and filter graphs from the CAD-model and its connected knowledge-base, it was possible to identify its entry points and to restructure the knowledge-base to a more intuitive organization and to move its central parts out from the CAD-system.

5 Eliciting knowledge about the CAD model quality from the parent/child graph

So far in this paper, we have seen how spreadsheets and knowledge-bases connected to CAD-models which can be penetrated to render graphs that are subsequently filtered to gain knowledge regarding the logical models that control the CAD-model. However, for the logical model rules and formulas to be able to control the CAD-model, the CAD-model itself needs to be sound and of good quality. In this section, we show by an example how the quality of CAD-models can be evaluated based on graph theory and filtering.

5.1.1 CAD-model quality

The theoretical model of quality is based on the linguistic model by Contero et al. [17], which defines three levels. The morphological quality level is related to the geometrical and topological correctness of the CAD model. The syntactic quality level is linked to the proper use of modeling conventions such as naming rules for features, datum, part, assembly, drawings, and layouts; layer structure and function and part/assembly parameters and attributes. The semantic/pragmatic quality level takes into account the CAD model capability for reuse and modification. CAD users have an abundant variety of modeling procedures for shaping their designs. However, experience shows that certain procedures provide better solutions than others. To provide a score of model quality, and approach inspired in the concept of rubrics has been applied.

On one hand, learning rubrics are scoring guides, constructed of descriptors or evaluative criteria (usually arranged in a table format) to set up the specifications to assess [18]. Summative rubrics are useless to determine CAD models' quality, as they produce a final summative or global score (so they aim to sort subjects into those who pass and those who fail the evaluation), while formative rubrics are worth considering, as they provide feedback about the performance [19]. In fact, a specific approach to check the quality of CAD models by way of suitable formative rubrics was proved helpful to disclose the different dimensions of CAD models quality [20], although was also proved difficult to apply by using traditional "static" rubrics [21].

Computer science has been reported helpful to provide e-rubric forms with enhanced capabilities, including "anchors" (written descriptions, examples that illustrate the various levels of attainment, or work samples [22]) or the capability to become adaptable and provide metadata about the evaluation process. This is the case of the Annota e-rubric platform [23] that we developed to deal with the management of complex rubrics as those used to derive the score used in this study.

On the other hand, the capability of a model to be reused is linked to a proper modeling sequence. Different modeling strategies lead to different Parent/Child graph structures (This is the same as filtering on product structure "kind-of" and "part-of" relations). To obtain a complexity metric of these graphs, we "reused" in this context, the *size* and *cyclomatic complexity* (CC) parameters used in software engineering as indicators of source code complexity [24] where *Size* is represented by the number of nodes in the model, while *CC* represents the number of independent paths through the graph and is calculated as:

$$CC = e + i + u - n$$

Where:

e is the number of edges, that is, the number of paths between nodes.

i is the number of inputs, that is, the number of nodes without parents.

u is the number of outputs, that is, the number of nodes without children.

n is the total number of nodes. It is the same as "size".

The hypothesis was that both *size* and *CC* correlate inversely with the CAD model quality they decrease when model quality grows and vice versa, using as indicator of model quality the score obtained using the Annota e-rubrics platform.

5.1.2 Graph extraction and quality assessment

An analysis software tool was developed to process in batch mode a set of Solidworks CAD models, to extract their parent/child dependency graphs, to calculate the complexity parameters of these graphs and integrate the quality score of the analyzed models from an external spreadsheet. This tool was developed using Microsoft Visual Basic .NET and the Solidworks application programming interface (API). While in batch mode, it can successively process every model contained in a folder and its subfolders. The tool extracts the two parameters *size* and *cyclomatic complexity* (*CC*) from every model.

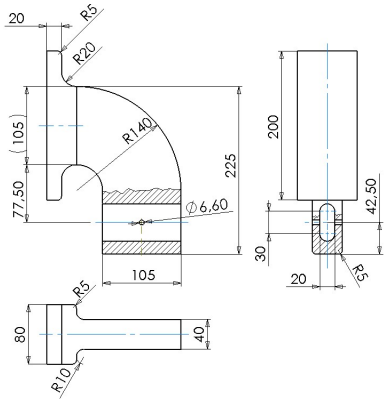

Figure 11: Part used in the experimental study.

5.1.3 Applying the method

To test the hypothesis that both *size* and *cyclomatic complexity* (*CC*) correlate inversely with the CAD model quality, a study involving 47 engineering students that modeled the part defined by the drawing presented in Figure 11, was conducted. Applying the methodology explained in the previous section, the CAD models were scored on a ten-point scale (range [0, 10], where 0 means that none valid model was produced, while ten means that a good quality model was obtained). Then the mean and standard deviation were tabulated for the three parameters (Table 3), and their correlations were analyzed (Table 4).

Table 3: Descriptive analysis

Parameter	Mean	Standard deviation
Size	16.09	2.59
CC	21.74	5.44
Score	6.86	1.84

Table 4: Correlation results (alpha = 0.01)

	Size	Cyclomatic complexity
Spearman rho	-0.418	-0.405
p-value	0.003	0.005

As displayed in Figure 12 and Figure 13, both *size* and *CC* parameters show a rough negative slope when plotted against the *score*, as we hypothesized. Before conducting the data correlation analysis, a Shapiro-Wilk test was used to ensure that studied data set does not follow a normal distribution. Then, a non-parametric Spearman correlation test with a significance level of 0.01 (alpha = 0.01) was applied. Descriptive statistics and correlation results presented in Figure 12 and Figure 13 validate our hypothesis, as they show that p-values for both parameters are less than 0.01, and negative.

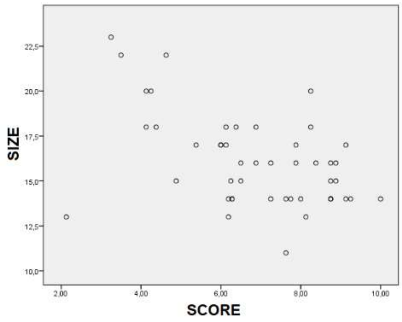

Figure 12: Size vs. quality score chart

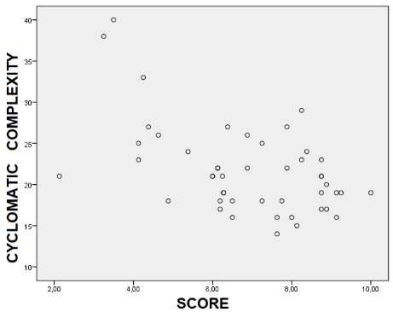

Figure 13: CC vs. quality score chart

The result gives insight into how to analyze the performance of CAD-models so that they can be reused and controlled by spreadsheets or knowledge-bases.

6 Discussion

Product quality is a polysemic concept understood and managed in different ways by the distinct stakeholders of a designed product. This is why, in our view, the collaborative design of a new industrial product must progress from an interdisciplinary activity (where interdisciplinarity implies a joint mixture of disciplines), passing through multidisciplinary until shifting up to transdisciplinary (which implies the fusion between the disciplinary knowledge and the know-how of lay people). However, the transition stages require consolidation. In particular, this paper is a starting point on how to apply the connectivistic view of knowledge [5] to the knowledge represented in KBE-systems, spreadsheets, and CAD-models.

The 3D Model definition is one necessary stage in collaborative product creation. In fact, CAD models are the primary view of the product (at least in the claim of the Model-Based Enterprise paradigm [25]). Thus, quality of CAD models is of capital importance for the quality of products. However, sectoral approaches to control and improve quality of CAD models have proved incomplete thus far. In the third case example, some parameters of the graphs were extracted to gain knowledge regarding how to develop CAD-models that can be reused and efficiently controlled by KBE-systems and spreadsheets.

When reviewing the nine principles of connectivism, it can be concluded that they can be implemented at an industrial level as follows (same numbering as in the frame of reference):

1. Any stakeholder adding geometry, facts, or rules to CAD-models, spreadsheets, or knowledge-base contributes to the knowledge flow. However, since the knowledge is represented in CAD-models, spreadsheets, and knowledge-bases, it may be hard to understand and should be made more accessible by the possibility to add rich comments.
2. PDM-systems serve as infrastructure to access CAD-models, spreadsheets, and knowledge-bases. However, a PDM-system only manages files and meta-data regarding the files. Adding routines to extract graphs on check-in events and storing graphs next to the files in the PDM system would serve as an infrastructure, to connect nodes of information sources.
3. The digitized knowledge stored in the KBE-system, CAD-models, and spreadsheet inherits its network from processes in knowledge management and knowledge engineering. The visualization of the network makes this more apparent.
4. The KBE-system, CAD-models, and spreadsheets facilitate storage of digitized engineering knowledge.
5. As seen from the second case example it was possible to reduce the complexity of the logical model just by viewing the filtered graph. One factor that made it easy was the lay-outing algorithm that was applied in a real-time manner so that the nodes and edges were animated when applying the layout. This way of creating contextual overviews of the product models supports the organizational learning.
6. Viewing knowledge development as a continuously ongoing process can be enabled through versioning control of the knowledge. This is not considered in this paper. It would change the picture in Figure 2 so that there would be no barriers between knowledge management, knowledge engineering, and knowledge-based systems; they would be continuously sub-processes of the organizational learning.
7. The visualization of network and context through graphs makes it possible to see the connections and patterns of the knowledge. By filtering the graphs, it is possible to see patterns that have been hard to grasp before.
8. Currency has not been targeted in this paper. However, by introducing graphs in PDM-system as mentioned in bullet two would make it possible to have accurate and up-to-date graphs for every CAD-model, KBE-rule, and spreadsheet readily available at every moment.
9. Viewing decision-making as a learning process is in line with viewing product development as a learning process, in which engineers indeed find themselves in a very shifting reality. Through the visualization of the network and its context, and the interactive navigation of knowledge—with easy to add knowledge content forms—an agile KBE development platform is at place supporting the flow of information.

The low amount of knowledge re-use in the industry is thought to be caused by the low level of standardization in the formalization of knowledge created by the design engineers. Report files content varies a lot from engineer to engineer, and it is not certain that the enterprise can safely re-use them in new projects. A more standardized way to formalize this knowledge could result in a higher re-use of the knowledge which could both save time and ensure the quality of the produced products [26]. This is supported by the rendering and filtering of graphs from CAD-models, spreadsheets and knowledge-bases.

The broad and diffuse, definition of context within the connectivism may be surpassed through the development of ontologies or domain-specific languages suited to the needs of engineers. At this end, stakeholders must agree on a basic terminology useful for engineers to develop and share their concepts, models, and context. Thus, allowing for other engineers to change these models, to adapt to their context or needs. It can further be said that the KBE-system treats the elements of knowledge stored in the knowledge-based on context as well. Sometimes a UDF is treated as a logical element, another time it is viewed as a geometrical element being part of a drawing, and yet another time it is used in a CNC or CMM process. The term polymorphism in computer programming reflects this very technical view of context, even if the connectivistic term is much broader.

When reviewing the three case examples presented in this paper it can be concluded that taking a connectivistic view on the engineering knowledge represented by CAD-models, spreadsheets and knowledge-bases made it possible to:

1. Increase the understanding of the product model by identification of entry points (design parameters).
2. Making it possible to examine the product structure and extract a product variant master.
3. Gain better understanding and possibility to retrace the logical model entangled into knowledge-base, CAD-model, and spreadsheet.
4. Devise engineers how to develop CAD-models so that they can be reused and efficiently controlled by KBE-systems and spreadsheets.

7 Conclusions

This paper is a starting point of applying the connectivistic view of knowledge to engineering knowledge as it is represented in product models. It was shown that by scanning the elements within knowledge-bases in KBE-systems, CAD-models, and spreadsheets, it is possible to render and filter graphs to support the connectivistic knowledge philosophy. Five areas were covered: network, filters, context, content, and conduits. Even if connectivism is an abstract philosophy of knowledge, this research proves that it is possible to adopt such a mindset to enhance further the way engineering knowledge is treated and to keep it alive and up-to-date. The proposed methods are far from ready and much work must be done to make them readily available to the engineers in global companies. However, it is already possible to apply the algorithms and methods presented in this paper to increase the understanding of product models by identification of entry

points (design parameters), making it possible to examine the product structure and extract a product variant master, gain better understanding and possibility to retrace the logical model entangled into knowledge-base, CAD-model and spreadsheet, and to devise engineers how to develop CAD-models so that they can be reused and efficiently controlled by KBE-systems and spreadsheets. This helps manufacturing companies to manage and reuse engineering knowledge as valuable assets. Which according to Kennedy [1] is critical to stay competitive in the long term.

8 Acknowledgments

The work presented has evolved during the IMPACT project, funded by the Swedish Knowledge Foundation, and has been partly presented on three conferences [27–29]. The work has also been partially supported by the Universitat Politècnica de Valencia (Spain) through the ANNOTA2 project.

9 References

[1] Kennedy M, Harmon K, Minnock E. Ready, Set, Dominate: Implement Toyota's Set-Based Learning for Developing Products and Nobody Can Catch You. Oaklea Press; 2008.

[2] André S, Elgh F, Johansson J, Stolt R. The design platform – a coherent platform description of heterogeneous design assets for suppliers of highly customised systems. J Eng Des 2017. doi:10.1080/09544828.2017.1376244.

[3] Rocca G La. Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design. Adv Eng Informatics 2012;26:159–79. doi:10.1016/j.aei.2012.02.002.

[4] Simon HA. The structure of ill structured problems. Artif Intell 1973;4:181–201. doi:10.1016/0004-3702(73)90011-8.

[5] Siemens G. Knowing knowledge. Winnipeg, MB]: [G. Siemens]; 2006.

[6] Downes S. Connectivism and connective knowledge. Essays Mean Learn Networks Natl Res Counc Canada 2012.

[7] Diestel R. Graph theory. 2. ed. . New York: Springer; 2000.

[8] Guia J, Soares VG, Bernardino J. Graph databases: Neo4j Analysis. ICEIS 2017 - Proc. 19th Int. Conf. Enterp. Inf. Syst., vol. 1, 2017.

[9] Holzschuher F, Peinl R. Performance of graph query languages: Comparison of cypher, gremlin and native access in Neo4j. ACM Int. Conf. Proceeding Ser., 2013. doi:10.1145/2457317.2457351.

[10] Jacomy M, Venturini T, Heymann S, Bastian M. ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software. PLoS One 2014;9:e98679.

[11] Poorkiany M, Johansson J, Elgh F. Capturing, structuring and accessing design rationale in integrated product design and manufacturing processes. Adv Eng Informatics 2016;30. doi:10.1016/j.aei.2016.06.004.

[12] Hu Y. Efficient, High-Quality Force-Directed Graph Drawing. Math Journal 2005;10:37.

[13] Johansson J. Howtomation© suite: A novel tool for flexible design automation. Adv. Transdiscipl. Eng., vol. 2, 2015, p. 327–36. doi:10.3233/978-1-61499-544-9-327.

[14] GraphML Team. The GraphML File Format 2016. http://graphml.graphdrawing.org/ (accessed February 16, 2017).

[15] Gephi.org. The Open Graph Viz Platform n.d. https://gephi.org/ (accessed February 16, 2017).

[16] Hvam L, Mortensen NH, Riis J. Product customization. Springer Science & Business Media; 2008.

[17] Contero M, Company P, Vila C, Aleixos N. Product data quality and collaborative engineering. IEEE Comput Graph Appl 2002;22. doi:10.1109/MCG.2002.999786.

[18] Reddy YM, Andrade H. A review of rubric use in higher education. Assess Eval High Educ 2010;35. doi:10.1080/02602930902862859.

[19] Panadero E, Jonsson A. The use of scoring rubrics for formative assessment purposes revisited: A review. Educ Res Rev 2013;9. doi:10.1016/j.edurev.2013.01.002.

[20] Company P, Contero M, Otey J, Plumed R. Approach for developing coordinated rubrics to convey quality criteria in MCAD training. CAD Comput Aided Des 2015;63. doi:10.1016/j.cad.2014.10.001.

[21] Company P, Otey J, Contero M, Agost M-J, Alminana A. Implementation of adaptable rubrics for CAD model quality formative assessment. Int J Eng Educ 2016;32.

[22] Jonsson A, Svingby G. The use of scoring rubrics: Reliability, validity and educational consequences. Educ Res Rev 2007;2. doi:10.1016/j.edurev.2007.05.002.

[23] Company P, Contero M, Otey J, Camba JD, Agost M-J, Pérez-López D. Web-Based system for adaptable rubrics case study on CAD assessmend. Educ Technol Soc 2017;20.

[24] Möller K-H, Paulish DJ. Software metrics. A practitioner's guide to improved product development. Chapman Hall Comput. Ser. London Chapman Hall,| c1993, 1993.

[25] Camba J, Contero M, Company P. CAD reusability and the role of modeling information in the MBE context 2017. doi:10.13140/RG.2.2.25072.66567.

[26] Hjertberg T, Stolt R, Poorkiany M, Johansson J, Elgh F. Implementation and management of design systems for highly customized products - state of practice and future research. Transdiscipl. lifecycle Anal. Syst. Proc. 22nd ISPE Inc. Int. Conf. Concurr. Eng., 2015, p. 165–74.

[27] Johansson J. Exploring Design Content In Cad-Models And Knowledge Bases Using Graph Theory And Filtering. Methods Tools CAE - concepts Appl., Bielsko Biała, Poland: 2017.

[28] Johansson J, Elgh F. Applying Connectivism to Engineering Knowledge to Support the Automated Business. 24th ISPE Int Conf Transdiscipl Eng

Singapore, 10 July to 14 July, 2017 2017:621–8. doi:10.3233/978-1-61499-779-5-621.
[29] Johansson J. Analysing Engineering Knowledge in CAD-models and Spread Sheets using Graph Theory and Filtering. 24th ISPE Int. Conf. Transdiscipl. Eng. Transdiscipl. Eng. A Paradig. Shift, 2017, p. 629–38. doi:10.3233/978-1-61499-779-5-629.