Automatically inferring technology compatibility with an ontology and graph rewriting rules

Roelofs, M. N.; Vos, Roelof

DOI
10.1080/09544828.2020.1860202

Publication date
2020

Document Version
Final published version

Published in
Journal of Engineering Design

Journal of Engineering Design

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/cjen20

Automatically inferring technology compatibility with an ontology and graph rewriting rules

M. N. Roelofs & Roelof Vos

To cite this article: M. N. Roelofs & Roelof Vos (2020): Automatically inferring technology compatibility with an ontology and graph rewriting rules, Journal of Engineering Design, DOI: 10.1080/09544828.2020.1860202

To link to this article: https://doi.org/10.1080/09544828.2020.1860202

JOURNAL OF ENGINEERING DESIGN
https://doi.org/10.1080/09544828.2020.1860202

Taylor & Francis
Taylor & Francis Group

Automatically inferring technology compatibility with an ontology and graph rewriting rules

M. N. Roelofs and Roelof Vos

Faculty of Aerospace Engineering, Delft University of Technology, Delft, The Netherlands

ABSTRACT

In conceptual design of any engineering system, decisions are made regarding which technologies to include and where. One of the first stages of that process is constructing the technology compatibility matrix (TCM), which indicates the compatibility of each pair in a technology set. Rather than constructing a TCM with expert judgment, this study develops a method based on graph transformation rules, allowing for a formal description of technologies. The TCM is then automatically derived. An ontology based on the Basic Formal Ontology is developed to describe systems and technologies, and provides axioms to derive statements about these descriptions. The method is demonstrated with four inference examples, showing how the inferences are made. An industry case study demonstrates the method's ability to mimic human expert reasoning. Although the approach is labour-intensive during setup, it enables knowledge capturing, automated reasoning and can be extended to provide quantitative analysis, to save time and effort.

ARTICLE HISTORY
Received 22 November 2019
Accepted 2 December 2020

KEYWORDS
Ontology; technology selection; technology compatibility matrix; graph transformation

Nomenclature

E	Edges
G	Graph
K	Gluing graph
L	Pattern graph
R	Effect graph
t	Technology
V	Vertices
Operators	
$\forall$	For all
$\exists$	Exists
$\Rightarrow$	Implies
$\Leftrightarrow$	If and only if
$\wedge$	And
$\vee$	Or

CONTACT M. N. Roelofs m.n.roelofs@tudelft.nl Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1 2629HS, Delft, The Netherlands

$\neg$	Negation
$\parallel$	Compatible
$\perp$	Incompatible
$\prec$	Enables

Acronyms

BFO	Basic Formal Ontology
CAD	Computer Aided Design
FD	Functional Decomposition
GUI	Graphical User Interface
PSO	Physics-based Simulation Ontology
TCG	Technology Compatibility Graph
TCM	Technology Compatibility Matrix

1. Introduction

In the conceptual design of engineering systems, it is of the utmost importance to select those components, technologies and configurations that are most likely to achieve the specified requirements, while minimising the risk and cost of development, manufacturing and operations. Technology selection is the decision-making process that establishes a set of technologies, called a technology portfolio, which satisfies all imposed requirements and achieves the set goals.

So what exactly constitutes a technology? In fact, the term technology is rather broad and has enjoyed many different definitions. Marx (1997) even emphasises the vagueness of the term. As an example of a definition, Bush (1981) states:

> 'Technology is a form of human cultural activity that applies the principles of science and mechanics to the solutions of problems. It includes the resources, tools, processes, personnel, and systems developed to perform tasks and create immediate particular, and personal and/or competitive advantages in a given ecological, economic, and social context.'

Although it is a more socio-cultural definition, it expresses the things that a technology encompasses. Furthermore, it highlights that a technology serves a purpose, i.e. has a function or goal. A similar conclusion is obtained from the definition by Merrill (1968). Feibleman (1961) approaches the definition of technology from a broader perspective, including pure science, applied science, technology and engineering. Historically, applied science and technology were quite distinct, being practised by natural philosphers and articans, respectively. However, over the past few centuries, the two have merged to a certain extent. Mitcham and Schatzberg (2009) provides a similar review of, and distinction between pure science, applied science, technology and engineering.

Considering the above definitions, we propose the following definition:

> A technology is a materialised form of knowledge applied to a given system in order to alter the system's behaviour to satisfy certain requirements.

Let us further expand this definition. The *materialised form of knowledge* ranges from implementing a process, incorporating a protocol, or instantiating or modifying a material entity. Thus when the knowledge is only implicit (e.g. exists as a thought), it is not considered materialised. The *system* can be any system, including social, biological and physical systems, as well as engineered systems. For the present work, only engineered systems

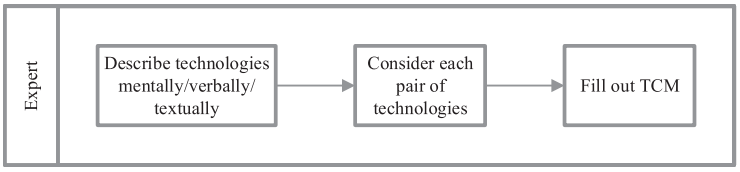


Figure 1. Common practice of constructing a TCM, using only expert judgment.

are considered. The *system behaviour* involves all processes that the system participates in. Finally, a technology has a purpose, as Bush (1981); Merrill (1968) concluded. Therefore, it has to satisfy certain requirements that capture the function or goal of the technology.

Concretely, in the aircraft domain, a technology according to this definition could be a new material for the wingbox that makes the wing more flexible and lighter, to reduce fuel consumption. Another example would be software that enables the airplane to be statically unstable by correcting for it through a controller, allowing the horizontal tailplane to be smaller and reduce fuel burn. Finally, a technology could be an operation procedure where the airspace is more efficiently utilised, such that airplanes can fly more direct routes and have shorter loiter times, reducing the fleet fuel burn at airline level.

In a typical technology evaluation project, different individuals (people or companies) contribute several technologies, which are either expressed through speech or may have been described in writing with perhaps accompanying drawings. From these technology descriptions, a group of experts has to first figure out which combinations of these technologies may be applied to a system of interest. Each valid combination is a technology portfolio and has certain merits and drawbacks on the system performance. Then, each portfolio should be quantified in terms of several quantities of interest. Based on this information, the experts can select which portfolios to consider for further development and implementation.

The issue is that the technologies are not formally expressed, and, therefore, each expert may have a different interpretation of each technology. Consequently, understanding how the technologies interact becomes difficult and requires extensive discussion with other experts. Furthermore, only the final result of these discussions is stored in a technology compatibility matrix (TCM). Afterwards, one cannot deduce from the TCM *why* certain technologies are incompatible. The process is illustrated in Figure 1.

A second issue is that technologies are commonly represented with impact factors (Kirby and Mavris 1999; Cartagena, Rosario, and Mavris 2000; Roth et al. 2001; Utturwar et al. 2002; Gatian and Mavris 2015; Chakraborty and Mavris 2017a, 2017b). These factors are provided by the analysis software used to compute the quantities of interest and do not reflect the technology's intricacies. Nonetheless, Soban and Mavris (2013) argue this is a benefit too, since no commitment to modelling the technology has to be made. We argue that this lack of commitment hampers knowledge capturing and automating (parts of) the technology evaluation and selection process. It should, therefore, be addressed.

Once the impact factors that are affected by the technology are determined and a value is assigned to them, again, the rationale behind this is lost. The rationale behind assigning impact factors may also differ from the rationale used to construct the TCM, because the technologies are interpreted differently in each situation. This happens naturally in human

reasoning; as time passes, our conceptions change, which are often not reflected back on previous decisions.

So how to overcome such issues? As Kitamura (2006) states: 'one of the necessities of ontologies of artefacts for engineering design is […] the lack of explicit description of background knowledge of modelling'. Therefore, we believe an ontology could mitigate the problems in the case of technology evaluation and selection. The ontology enables machine interpretation of the technologies, to automate the construction of a TCM. Automating this task would not only improve the consistency of the technologies' representation but also cuts back in time spent. To illustrate, a TCM contains $(n^2 - n)/2$ entries, where n is the amount of technologies. Assigning each entry by hand quickly becomes a time-intensive task (Amadori, Bäckström, and Jouannet 2017, 2018).

Naturally, efforts to develop an ontology for engineering design have been conducted. The physical systems (PhysSys) ontology (Borst et al. 1994, 1995) comprises of three ontologies addressing the systems layout (component ontology), the physical processes underlying behaviour (process ontology) and the descriptive mathematical relations (Eng-Math ontology) (Gruber and Olsen 1994; Borst, Akkermans, and Top 1997). The former two build on top of three ontologies describing mereology, topology and systems theory. Unfortunately, neither PhysSys nor a detailed description of its workings is available.

Other attempts focus more on qualitative physics to describe behaviour of systems. For example, De Kleer and Brown (1984) develop a qualitative physics to describe, predict and explain the behaviour of systems. Using causal analysis and teleological reasoning, based on the qualitative physics, electronic circuits are analysed (De Kleer 1984). Here, function is defined as a causal pattern between variables. Qualitative physics relates structure to behaviour, whilst teleology relates behaviour to function. Other approaches include the process ontology (Forbus 1984) and bond graph theory (Rosenberg and Karnopp 1983; Karnopp, Margolis, and Rosenberg 1990). The latter is used in PhysSys to describe processes (Borst et al. 1995). Hirtz et al. (2002) established a frequently used functional basis to describe the functions of systems and components. The functional basis should be interpreted as a taxonomy and although Hirtz et al. provide descriptions of each class, these are based on natural language and open to interpretation. A functional decomposition (FD) describing aircraft system architectures was developed by Judt and Lawson (2015, 2016), to consecutively enumerate system architectures and search for the best solution regarding some quantity of interest (QoI) with a hybrid heuristic optimisation. Their FD is problem-specific, as is the analysis method, and therefore not easily generalised. The same holds for AirCADia (Guenov et al. 2016), which breaks down the system description into a functional and logical domain and proceeds by mapping functions to means to arrive at a system synthesis. Sen, Summers, and Mocko (2011, 2013b, 2013a) use function–structure graphs to describe the behaviour of a system enabling physics-based reasoning on it. Although effective, it appears to only be applicable to mechanical and electrical engineering domains, while continuum mechanics seem to pose a problem to this approach. The BeCoS tool (Castet et al. 2015; Kaderka et al. 2018) uses an ontology to describe systems semantically rigorous in terms of their behaviour. State-machines describe transitions within the behaviour and, combined with equations, enable analysis of the system. An approach particularly aimed at capturing functional design knowledge with an ontology is presented by Kitamura and Mizoguchi (2004). It complements the device-centric approach PhysSys, because PhysSys has no ontology for functions from the teleological viewpoint.

None of the above work considers technology separately. Therefore, the process of technology evaluation, selection and infusion is not supported. The current work addresses this. An ontology is developed that allows for such an explicit description of technologies. Furthermore, the first issue of automatically deriving the TCM is addressed. The second issue of assigning impact factors will be addressed in future work. The proposed ontology is described in Section 2. This includes the upper ontology and classes, the definition and explanation of technologies as graph transformation rules, rules to infer physical facts, as well as the technology incompatibility and enabling relations. The case studies in Section 3 demonstrate how the ontology may be applied. The paper is concluded by a discussion in Section 4 and reiteration of the method and findings in Section 5.

2. Methodology

The proposed method consists of three parts: an ontology to describe engineering systems, graph transformation rules to represent technologies and rules to infer dependencies between the technologies. The ontology formalises knowledge about engineering systems. That knowledge may be represented as a knowledge graph. The graph transformation rules then reflect parts of knowledge graphs that are modified as a result of a technology. By evaluating what a technology modifies, rules dictate whether that modification is compatible with another. That process is illustrated in Figure 2. Compare it to Figure 1 to see what tasks the support system takes over from experts. Each of these three elements is described in the following sections.

2.1. Engineering systems ontology

The ontology is built with the Basic Formal Ontology (BFO) (Arp, Smith, and Spear 2015; Smith 2012a, 2012b) as upper ontology. The main distinction made in BFO is between continuants and occurrents. Continuants are entities that exist irrespective of time, while occurrents depend on time, and only exist during some portion of it. Continuants can be split into three distinct classes: independent continuants, specifically dependent continuants and generically dependent continuants. The first are entities that exist in virtue of

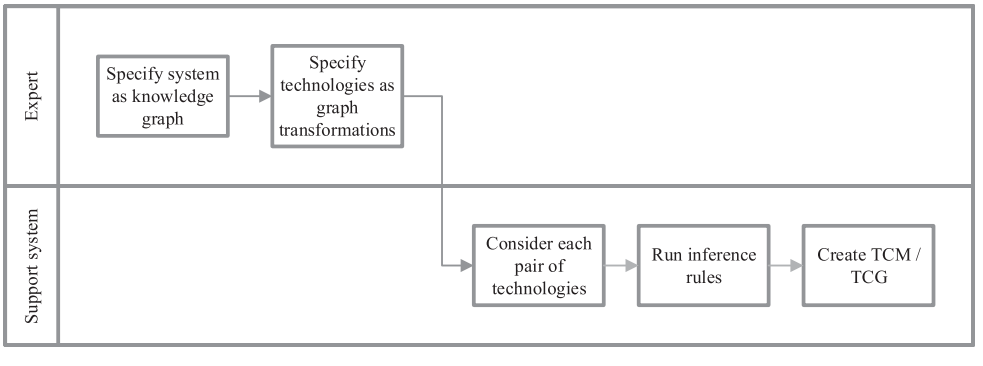


Figure 2. Proposed method for constructing a TCM, using a rule-based system relying on an ontology.

Table 1. Introduced classes and their superclasses (either from Basic Formal Ontology (BFO) or from the proposed ontology).

Class	Superclass
Component	Object (BFO)
Energy	Immaterial entity (BFO)
Fluid	Object (BFO)
Gas	Fluid
Liquid	Fluid
Plasma	Fluid
Interface	Continuant fiat boundary (BFO)
Signal	Generically dependent continuant (BFO)
Solid	Object (BFO)

nothing but themselves. The second are continuants that depend on a certain independent continuant. As such, qualities (e.g. the mass of an object) are specifically dependent continuants. The third class are continuants that only exist in some immaterial form until they are concretised in a specifically dependent continuant. Examples are novels: a specific print of a novel is a concretisation of the novel. Even though there may be multiple copies of the same novel, there is only one such novel.

BFO furthermore defines spatial, temporal and spatiotemporal regions. Together with sites and fiat boundaries, these allow reasoning over the topology of space and time. Finally, the primary class of occurrents used in the present ontology are processes. Processes in BFO occupy a certain spatiotemporal region and have continuant participants, to whom they are a change. Therefore, processes do not change themselves and do not possess qualities (Smith 2012a).

Table 1 shows the classes that are introduced to the current ontology as subclasses of BFO. They are loosely based on works on functional decomposition (Hirtz et al. 2002; Sen, Summers, and Mocko 2011, 2013a, 2013b) and should therefore allow to define engineering components and how these interact. Components, fluids and solids are all BFO objects, as they are material entities that exhibit causal unison. A signal generically depends on a carrier and is merely the interpretation of a physical quality. Interfaces are defined to be continuant fiat boundaries. Finally, energy is classified as an immaterial entity, because it definitely is an independent continuant (it can neither be created nor destroyed, and is equivalent to matter), but is not a material entity.

The interactions between components are represented using processes and various relations between entities. These relations are shown in Table 2. The upper part of this table shows relations from BFO that are used in the present ontology, whereas the bottom part shows newly introduced relations. Table 2 lists the domain, range, reflexivity, symmetry, transitivity and the inverse for each relation. The domain is the set of classes that can act as subject to the relation, while the range indicates the type of object of the relation. Reflexivity entails that an entity has a relationship to itself, i.e. x **relation** x. Symmetry indicates that when x **relation** $y \Rightarrow y$ **relation** x. Reflexivity and symmetry also have inverses: irreflexivity and asymmetry. These mean that the negative of that property holds for the relation. Transitivity propagates a relation through a hierarchical structure; i.e. x **relation** $y \wedge y$ **relation** $z \Rightarrow x$ **relation** z. Finally, the inverse of a relation is simply its counterpart: x **relation** $y \Rightarrow y$ **inverse** x.

Table 2. Relations in present ontology, both taken from BFO (upper part) or introduced (lower part).

Relation	Domain	Range	R	S	T	Inverse
has disposition	independent continuant	disposition				
has function	independent continuant	function				
has part			•	°	•	part of
occupies	material entity ∪ process	spatial region				
occurs in	process	spatial region				
part of			•	°	•	has part
realises	process	realisable entity				realised by
realised by	realisable entity	process				realises
blocking disposition of	disposition	disposition	°			
inhibits	process	process	°			
inhibiting process of	process	process	°			
overlaps			•	•		

Note: The columns R, S and T stand for Reflexive, Symmetric and Transitive, respectively. A • indicates that relation possesses the property, while a ° indicates it possesses the inverse of that property.

The BFO relation **occupies** requires some elaboration. When *m* **occupies** *r*, it means that the material entity (or process) *m* is exactly located in spatial region *r*. Thus, equivalently we could state:

$$\forall\, m, r : m \textbf{ occupies } r \Rightarrow \forall\, r_1 : \text{spatial region}(r_1) \wedge r_1 \textbf{ part of } r \wedge m \textbf{ occupies } r_1 .$$

We took the liberty of adding processes to the domain of **occupies**, because BFO has two relations for this: **occupies spatial region** and **occupies spatiotemporal region**, where the first applies to material entities and the second to processes. A spatiotemporal region then **projects onto** a spatial region at some time. Thus having **occupies** directly between a process and a spatial region is equivalent to stating that the spatiotemporal region that the process occupies, projects onto the same spatial region for any given time instant within the spatiotemporal region.

The relations in the second part of Table 2 also require further definition and elaboration. First, the next few rules are necessary to reason about overlapping entities. Because the notion of overlapping is very general (e.g. sets overlap, spatial or temporal regions overlap), it applies to anything, and no class restrictions are included in the rules.

$$\forall\, r_1, r_2 : r_1 \textbf{ overlaps } r_2 \Leftrightarrow \exists r_3 : r_3 \textbf{ part of } r_1 \wedge r_3 \textbf{ part of } r_2 \tag{1}$$

Usually, the **overlaps** relationship applies to spatial regions, in which case CAD or similar software may be used to infer it. For now, a statement is directly added to the ontology for individuals that overlap, such that subsequent inferences can be made. Furthermore, any spatial region that is part of another spatial region, overlaps its parent:

$$r_1 \textbf{ overlaps } r_2 \Leftarrow \forall r_1, r_2 : \text{spatial region}(r_1) \wedge \text{spatial region}(r_2) \wedge r_1 \textbf{ has part } r_2 \tag{2}$$

Another spatial reasoning mechanism that is needed is that any process that is part of another process occurs in the same region:

$$p_1 \textbf{ occurs in } r_1 \Leftarrow \forall p_1, p_2, r_1 : \text{process}(p_1) \wedge \text{spatial region}(r_1) \wedge \text{process}(p_2) \wedge p_2 \textbf{ occurs in } r_1 \wedge p_2 \textbf{ has part } p_1 \tag{3}$$

A significant portion of domain-specific reasoning involves the interaction of processes. Users may wish to define that a process inhibits other processes of some specific type. This is most easily achieved by introducing a class of processes of which all individuals are inhibiting processes of some other class of process:

$$\forall\, p_1, p_2 : p_1 \textbf{ inhibiting process of } p_2 \Rightarrow \begin{array}{l} \text{Inhibiting Process}(p_1) \wedge \\ \text{Inhibited Process}(p_2) \end{array} \tag{4}$$

The relation **inhibiting process of** only informs that its operands have the potential to be inhibitory. However, the involved processes only actually inhibit each other when they occur simultaneously in the same spatial region:

$$\begin{array}{rl} \forall\, p_1, p_2 : p_1 \textbf{ inhibits } p_2 \Leftrightarrow & \text{process}(p_1) \wedge \text{process}(p_2) \wedge \\ & p_1 \textbf{ inhibiting process of } p_2 \wedge \\ & \exists r_1, r_2 : \text{ spatial region}(r_1) \wedge \\ & \text{spatial region}(r_2) \wedge \\ & p_1 \textbf{ occupies } r_1 \wedge p_2 \textbf{ occupies } r_2 \wedge \\ & r_1 \textbf{ overlaps } r_2 \end{array} \tag{5}$$

Dispositions are realised in processes. When two dispositions are realised by inhibitory processes, these dispositions block each other. Therefore, we borrow the idea of a blocking disposition from Goldfain, Smith, and Cowell (2010, 2011):

$$\begin{array}{rl} \forall\, d_1, d_2 : d_1 \textbf{ blocking disposition of } d_2 \Leftrightarrow & \text{disposition}(d_1) \wedge \text{disposition}(d_2) \wedge \\ & \exists p_1, p_2 : \\ & \text{process}(p_1) \wedge \text{process}(p_2) \wedge \\ & p_1 \textbf{ realises } d_1 \wedge p_2 \textbf{ realises } d_2 \wedge \\ & p_1 \textbf{ inhibits } p_2 \end{array} \tag{6}$$

Here, the relation **negatively regulates** is replaced with **inhibits**.

Finally, we introduce the notion of an interface, which is a continuant fiat boundary that has some dispositions. An interface occupies a spatial region. Examples of interfaces are connection points of pipes or cables, or the surface of an object.

$$\begin{array}{rl} \text{interface}(i) \equiv & i \in \text{ continuant fiat boundary} \wedge \\ & \exists r : i \textbf{ occupies } \text{spatial region}(r) \wedge \\ & \exists e : i \textbf{ part of } \text{material entity}(e) \wedge \\ & \exists d : i \textbf{ has disposition } \text{disposition}(d) \end{array} \tag{7}$$

Interfaces are used to express interactions between physical (material and immaterial) entities, through dispositions and functions defined on them.

2.2. Technologies as graph transformations

Using an ontology to provide semantics to data, the data itself can be represented as a graph. A graph is a tuple $G = (V, E)$ where V are the nodes and $E \in V \times V$ the edges, each of which is a tuple (s, t) where s is the source node and t the target node, in the case of a directed edge. If the edge is undirected, there is no distinction between s and t. A type graph defines types of nodes and edges and specifies how those nodes may be connected by those edges. An ontology can be mapped onto such a type graph, allowing information adhering to the ontology to be represented as a knowledge graph.

Specifying changes to graphs formally is done with graph transformations. Graph transformations in the form of graph transformation rules are part of graph grammars, which have been around since the 1970s and have seen some use in design automation and synthesis (Helms, Shea, and Hoisl 2009; Irani and Rudolph 2003; Stavrev 2011). Inspired by this, the present work uses graph transformation rules to describe the effect of technologies on engineering systems.

A graph transformation $r : (L \leftarrow K \rightarrow R)$ is a construct consisting of three graphs: the pattern L, the gluing graph K and the replacement graph or effect graph R. The pattern L is to be matched inside a certain graph (the system in this case), while the replacement graph R replaces the matched instance of L, if it exists. Determining what elements to remove or add is done through the gluing graph K, which contains the corresponding nodes and edges of L and R. Note that K can also be empty when L and R have no common substructure. In that case, all elements in L are removed from a graph along with any edges connected to the removed nodes. Then R is substituted in, but will have no connections to the rest of the graph, resulting in a disconnected graph. If K is non-empty, the difference between L and K, denoted by $L-K$, are the nodes and edges which are removed by the graph transformation. Similarly, the difference between R and K, i.e. $R-K$ are the nodes and edges which are added by the graph transformation.

The concept of graph matching is more formally explained using graph morphisms. Let G be the system graph that the rule r is to be applied to. When there is a match of the pattern L in G, there is a graph morphism $m : L \mapsto G$. A graph morphism m consists of two functions $f_V : V_L \mapsto V_G$ and $f_E : E_L \mapsto E_G$, such that $s_G \circ f_E = f_V \circ s_L$ and $t_L \circ f_E = f_V \circ t_L$ (Ehrig et al. 2015). Here V denotes the vertices of a graph and E its edges. Furthermore, s is a function that retrieves the source node of an edge, and t a function that retrieves the target node.

2.2.1. Independence

Graph transformation rules may interfere with one another when applied to the same graph. Therefore, two types of independence between these rules are defined: parallel independence and sequential independence (Ehrig et al. 2015). The former implies that two transformations can be applied simultaneously. The latter implies that the transformations can be applied in any order and produce an identical end-result.

There may be additional constraints on both these independence definitions, when application conditions are included in the graph transformation rule, as is done by Ehrig et al. (2015). However, for the current implementation these conditions are not included. The notion of parallel dependence is used in Section 2.3.1 to infer incompatibility

statements about pairs of technologies. Sequential dependence is used in Section 2.4.1 to infer enabling relationships between technologies.

2.3. Technology incompatibility

Presently, the purpose of the ontology is to infer whether any two technologies are incompatible with one another. We assume that if no reason is found for the pair to be incompatible (denoted by $\perp$), it is compatible (denoted by $\parallel$). Thus

$$t_1 \parallel t_2 \Leftrightarrow \neg(t_1 \perp t_2) \tag{8}$$

There are three causes for incompatibility between technologies: the graph transformation rules, physics and functionality. Each of these is treated separately in the following sections.

2.3.1. Transformation incompatibility

Technology compatibility only requires the transformation rules to be parallel independent. That ensures that neither technology offsets any effect of the other. Note that transformation incompatibility only applies when the technologies are introduced to an overlapping portion of the system graph G.

Parallel independence is checked as follows. First the maximum common (induced) subgraph $K_{1,2}$ of L_1 and L_2 is found as

$$K_{1,2} = \text{argmax}_{m_1,m_2} |V_{K_{1,2}}| : K_{1,2} \overset{m_1}{\longmapsto} L_1, K_{1,2} \overset{m_2}{\longmapsto} L_2 \tag{9}$$

Two technologies are parallel independent, when neither t_1 removes something t_2 uses nor vice versa. That means that their removal graphs may not contain any node or edge that the pattern of the other technology requires. Conversely, if that is the case, the technologies are concluded to be incompatible:

$$\begin{aligned} t_1 \perp t_2 \Leftarrow\ & (L_1 - K_1 \cap K_{1,2} \neq \emptyset) \vee \\ & (L_2 - K_2 \cap K_{1,2} \neq \emptyset) \end{aligned} \tag{10}$$

2.3.2. Physical incompatibility

The main reasoning mechanism for incompatibility of technologies is when their simultaneous application would result in a physically inconsistent situation. The following rules capture several of such inconsistent situations.

A trivial inconsistency arises when two material entities overlap. Thus if two material entities occupy the same space, they cannot co-exist. Then any two technologies introducing material entities that cannot co-exist are incompatible:

$$\begin{aligned} t_1 \perp t_2 \Leftarrow\ & \exists c_1, c_2, r_1, r_2 : \text{material entity}(c_1) \in (R - K)_1 \wedge \\ & \text{material entity}(c_2) \in (R - K)_2 \wedge \\ & \text{spatial region}(r_1) \wedge \text{spatial region}(r_2) \wedge \\ & c_1 \textbf{ occupies } r_1 \wedge c_2 \textbf{ occupies } r_2 \wedge \\ & r_1 \textbf{ overlaps } r_2 \end{aligned} \tag{11}$$

Introducing a process that inhibits the process introduced by another technology leads to incompatibility:

$$\begin{aligned} t_1 \perp t_2 \Leftarrow\ & \exists p_1, p_2 : \text{process}(p_1) \in (R-K)_1 \wedge \\ & \text{process}(p_2) \in (R-K)_2 \wedge \\ & (p_1 \textbf{ inhibits } p_2 \vee p_2 \textbf{ inhibits } p_1) \end{aligned} \tag{12}$$

2.3.3. Functional incompatibility

Often, it is undesirable to have two technologies introduce the same functionality in the same region. Therefore, one might wish to define incompatibility between such technologies. This is the case when they introduce processes that have an equivalent effect.

For this, a form of equivalence between processes has to be defined. For now, the equivalence is based on the effect of the process. This is inferred when two processes realise the same type of disposition:

$$\begin{aligned} p_1 \textbf{ has equivalent effect } p_2 \Leftarrow\ & \forall p_1, p_2 : \text{process}(p_1) \wedge \text{process}(p_2) \wedge \\ & \exists d_1, d_2 : \text{disposition}(d_1) \wedge \text{disposition}(d_2) \wedge \\ & p_1 \textbf{ realises } d_1 \wedge p_2 \textbf{ realises } d_2 \wedge \\ & d_1 \subseteq d_2 \end{aligned} \tag{13}$$

According to this definition, even when a process has multiple dispositions, only one of which is equivalent with one from another process, the processes have an equivalent effect. The equivalent effect is, therefore, only a subset of the full effect of the process. To obtain a stronger notion of equivalence between the processes, the existential quantifier on the dispositions should be replaced with a universal quantifier. Depending on one's viewpoint, that may be the more correct way forward.

The technology incompatibility statement that when the technologies introduce processes that have an equivalent effect, then reads:

$$\begin{aligned} t_1 \perp t_2 \Leftarrow\ & \exists p_1, p_2, r_1, r_2 : \\ & \text{process}(p_1) \in (R-K)_1 \wedge \text{process}(p_2) \in (R-K)_2 \wedge \\ & p_1 \textbf{ has equivalent effect } p_2 \wedge \\ & \text{spatial region}(r_1) \wedge \text{spatial region}(r_2) \wedge \\ & p_1 \textbf{ occupies } r_1 \wedge p_2 \textbf{ occupies } r_2 \wedge \\ & r_1 \textbf{ overlaps } r_2 \end{aligned} \tag{14}$$

Only when the processes overlap in space are they considered incompatible, because in theory, the one technology could introduce a process on the moon and the other on Earth, which obviously have nothing to do with one another. This rule does not capture the situation where the regions r_1 and r_2 are situated such that the processes have the same effect in the same region nonetheless. This could happen, for example, when the processes are flames on two sides of a metal plate. Clearly, they occur in distinct regions, but their effect – heating the plate – is equivalent.

2.4. Technology enabling

Besides incompatibility, statements regarding technologies enabling one another have to be inferred. When technology t_1 enables t_2, this is written as $t_1 \prec t_2$. Technology enabling occurs through two mechanisms: the graph transformation rules and physics-based rules in the ontology. Each of these rules is overruled if any incompatibility statement fires for the pair of technologies. This means that:

$$t_1 \prec t_2 \Rightarrow t_1 \parallel t_2 \tag{15}$$

and, despite what any of the following rules might suggest:

$$t_1 \perp t_2 \Rightarrow t_1 \not\prec t_2 \wedge t_2 \not\prec t_1 \tag{16}$$

2.4.1. Transformation enabling

Technology transformation rules may be sequentially dependent. In that case, the technology that sequentially depends on another is enabled by the latter. Nonetheless, they still have to be parallel independent. There are two situations that need to be taken into account: one when the transformations by themselves are sequentially dependent and one when the transformations are sequentially dependent when applied to a system graph.

In the first case, the pattern of the enabled technology should appear in the effect of the enabling technology, but should not be present in the latter's pattern:

$$t_1 \prec t_2 \Leftarrow L_2 \subseteq R_1 \wedge \neg(L_2 \subseteq L_1) \tag{17}$$

Thus the requirements for t_2 are completely resolved by the effect of t_1, while those requirements were not fulfilled before application of t_1, because L_2 is not a subgraph of L_1.

When applied to a system graph G, the pattern L_2 does not have to be a subgraph of R_1, but rather a subgraph of G after application of t_1. Similarly, L_2 should not appear in G, instead of L_1:

$$t_1 \prec t_2 \Leftarrow L_2 \nsubseteq G \wedge G \overset{t_1}{\longmapsto} H \wedge L_2 \subseteq H \tag{18}$$

These rules fire when, for example, a technology is the addition of a component to the system, and the second technology only works for such a component. The last rule also works when only part of a system is modified, say for example a material is changed from metal to composite, and the other technology requires a composite material to work.

2.4.2. Physical enabling

A technology enables another when it removes an interface or process or disposition that inhibits one that the other introduces. The converse case, when a technology adds a process or interface that another requires, has to be handled as well.

Consider the flame in a jet engine combustor. For it to work, the airflow into the combustor must be relatively laminar. So a technology that laminarises the inflow of air enables the combustion process. In other words, the enabling technology removes the turbulence process that inhibited the flame. The second case can be illustrated by a technology that adds a pumping process to force the fuel into the injector nozzle.

The challenge is that when this type of enabling occurs, there may be other aspects that still prevent the enabled technology to be fully applicable. For example, this happens when there are two processes that the technology relies on, which are both inhibited by some existing process in the system. If another technology only resolves the inhibition of one of these processes, the other process still prevents application of the technology in question. Therefore, we need to check if all inhibiting aspects are resolved by the enabling technology. Note as well that the following rules are based on the application of a technology to a system graph.

The above description is captured in the following rule. After applying $t_1 : G \mapsto H_1$ and $t_2 : H_1 \mapsto H_2$, this rule infers physical enabling through removal of inhibiting processes or blocking dispositions:

$$\begin{aligned} t_1 \prec t_2 \Leftarrow \forall\, \text{disposition}(d) \ \in (R-K)_2 :\ & \{\nexists\, \text{disposition}(d_G) \in H_1 : \\ & d_G \textbf{ blocking disposition of } d\} \wedge \\ & \forall\, \text{process}(p) \ \in (R-K)_2 :\ \{\nexists\, \text{process}(p_G) \in H_1 : p_G \textbf{ inhibits } p\} \wedge \\ & ([\exists\, \text{disposition}(d) \ \in (R-K)_2 :\ \{\exists\, \text{disposition}(d_G) \in G : \\ & d_G \textbf{ blocking disposition of } d\}] \vee \\ & [\exists\, \text{process}(p) \ \in (R-K)_2 :\ \{\exists\, \text{process}(p_G) \in G : p_G \textbf{ inhibits } p\}]) \end{aligned} \tag{19}$$

The other mechanism by which two technologies may enable one another is when they resolve dependencies imposed on each other's interfaces. This is simply the case when, for example, two electrical conductors are connected, such that a current may flow through. (A battery alone does not produce current, nor an electromotor by itself. If the two are connected, however, a current flows and the function of the electromotor is realised. The battery already fulfills its function by storing chemical energy. Depending on your point of view, it also has a function to convert chemical energy to electric energy and/or to provide it to some other device. In that case, the battery has these functions realised by the complementing interface.)

For this mechanism we require the notion of complementary and collective dispositions (Goldfain, Smith, and Cowell 2010). The idea is that an object aggregate C can have a disposition D that is formed by the mereological[1] sum of its constituent dispositions d_i. Thus $C = \sum c_i$, and $\forall c_i : c_i$ **has disposition** d_i, such that $D = \sum d_i$. The latter sum describes parthood between dispositions, which is not clearly defined. Instead, the process aggregate P that realises D can be described as the sum of its constituents: $P = \sum p_i : P$ **realises** D. Then, for each of these constituent processes, there must exist a part of C that manifests it: $\forall p_i (\exists c_i \in C : c_i$ **has disposition** d_i **realised by** $p_i)$. This works, because parthood of processes is a better defined concept.

To make this work in practice, inferences are required that establish how interfaces enable one another's functions. This can be done for two interfaces with the following rule:

$$\begin{aligned} I_1 \textbf{ complements } I_2 \Leftarrow\ & \text{interface}(I_1) \ \wedge\ \text{interface}(I_2) \wedge \\ & \forall p_0 : I_2 \textbf{ has function } f \textbf{ realised by } p_0 \\ & [(\exists p_2 : p_0 \textbf{ has part } p_2 \wedge \\ & I_2 \textbf{ has disposition } d_2 \textbf{ realised by } p_2) \wedge \end{aligned}$$

$$\forall p_1 : p_0 \textbf{ has part } p_1 \wedge$$
$$\neg(I_2 \textbf{ has disposition } d_2 \textbf{ realised by } p_1)$$
$$(\exists d_1 : I_1 \textbf{ has disposition } d_1 \textbf{ realised by } p_1)] \tag{20}$$

This rule states that for each function of interface I_2 it performs at least one of the sub-processes involved in fulfilling that function. Furthermore, for each part of p_0 that is not manifested by interface I_2, there should be a disposition in interface I_1 that does manifest it.

Looking back at the definition from Goldfain, Smith, and Cowell (2010), it should be clear that p_0 is the collective process P. Furthermore, the interfaces I_1 and I_2 are the parts c_i that manifest the sub-processes p_i (as p_1 and p_2). This rule also works when $p_0 = p_1 = p_2$, because **has part** is both transitive and reflexive (see Table 2).

Now it has been established that these two interfaces form a collective that realises the function intended by one of them, we can bring it back to technology level and infer technology enabling:

$$t_1 \prec t_2 \Leftarrow \exists\, \text{interface}(I_2) \in (R-K)_2\ (\exists\, \text{interface}(I_1)\ \in (R-K)_1 : I_1 \textbf{ complements } I_2)s \tag{21}$$

Thus a technology enables another when it complements one or more interfaces of the other. To explain why this is taken as the enabling condition, rather than requiring each interface to be complemented by the other technology, consider a pump. A pump has two interfaces: inflow of non-pressurised fluid and outflow of pressurised fluid. A pump is only fully enabled when both ports are connected. However, no single object would simultaneously satisfy both these interfaces. Therefore, only one interface has to be complemented by an enabling technology, even though that may mean that the enabled technology remains with unresolved interfaces.

3. Application and results

As explained in the introduction, one of the first steps in technology selection is defining a technology compatibility matrix (TCM), which is used to prune out combinations of technologies that should not be considered together. In this section, several technologies are modelled to showcase the inferences the presented method enables and how these can be leveraged to automatically construct a technology compatibility graph (TCG), which is a generalisation of the TCM.

After that, we show how a complete TCM is generated for an industry technology set. It is compared to the one specified by experts to investigate how well equipped the proposed method is in mimicking expert reasoning.

3.1. Implementation

Although the method is independent of the implementation, it hinges on the presence of two components: graph rewriting rules and a rule-based inference engine. The system is described as a knowledge graph in any format. The technologies are represented as a triple of graphs: the pattern, gluing and effect graph. These graphs should be parsed by the graph

library, which carries out the graph rewriting operation $G \overset{t}{\longmapsto} H$ for each technology. Then all pairs of the graphs H are fed into the inference engine, which applies the rules detailed herein and outputs the TCG relations.

Most of the method has been implemented in Protégé, by including the BFO-OWL ontology and specifying SWRL rules to reflect the rules stated in Section 2. However, some rules, e.g. Equations (19) or (20), cannot be expressed in SWRL, which is why the entire approach was rebuilt with custom code written in C#. A small graph library is built to work on knowledge graphs and perform graph matching. The input graphs that specify the systems and technologies are supplied in a custom XML format. On top of it, a very naive reasoner is used, which continues to execute all rules until no further inferences are made. The rules are written in code. A GUI allows the user to open the input files and start the inference process. The GUI also shows the resulting TCG.

3.2. Test case description

The first case study (see Sections 3.3 and 3.4) revolves around an aerodynamic surface, such as an aircraft wing. The following technologies are considered: a vortex generator, a plasma actuator, natural laminar flow, conformal antennas and conductive structure. All of these, except for the conductive structure, have the function to reduce the friction drag of the wing. The conductive structure is meant to remove the need for cables and can provide electricity distributed over a surface. In current practice, experts would now discuss these technologies and establish a TCM by hand. That is, they draw up a matrix and fill out each cell by discussing whether that pair of technologies is incompatible or has any other dependency.

Instead, the proposed approach is applied here. The graph transformation rules (representing technologies) are shown as knowledge graphs with the addition and removal subgraphs. As such, knowledge about what a technology constitutes is captured. Then, we show what inferences the computer algorithm would make in what order to illustrate the approach. Rather than doing so for each pair of technologies, one of the compatibility rules from Section 2.3 or one of the enabling rules from Section 2.4 is illustrated with one pair of the five technologies.

The second case study (see Section 3.6) takes a set of technologies from the aviation industry and applies the full approach to obtain a TCM. This TCM is compared to one created using expert judgment. Thus we show the applicability of the method to practical use cases and its reasoning accuracy.

3.3. Compatibility reasoning

Two of the incompatibility rules from Section 2.3 are illustrated: inhibitory processes (see Section 3.3.1) and functional equivalence of processes (see Section 3.3.2).

3.3.1. Physical incompatibility: inhibiting process

A body moving through air experiences friction from the air particles colliding with the object's surface. When the airflow around the body is laminar, the air particles move in layers that hardly interact. Therefore, the particles close to the surface move tangentially to it, which reduces the friction drag, because less collisions occur. Conversely, when the flow is

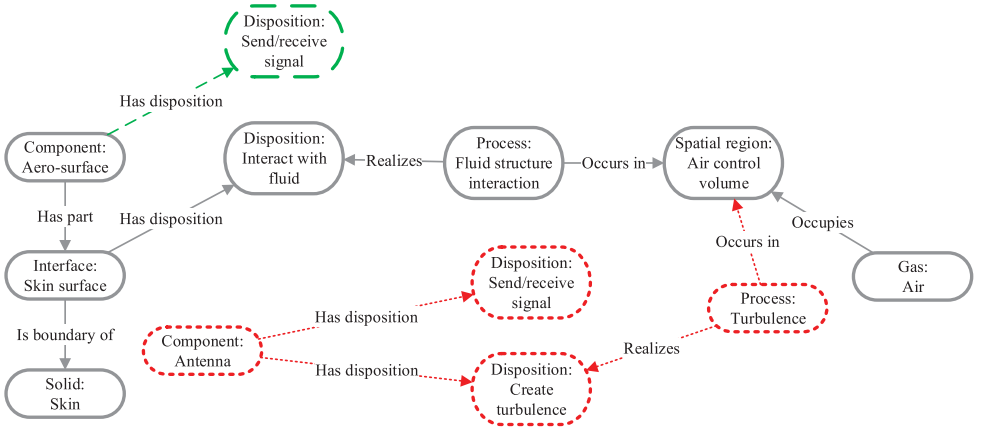


Figure 3. Graph transformation of conformal antenna. Grey represents the pattern, green (dashed) the added nodes and edges, and red (dotted) the removed nodes and edges.

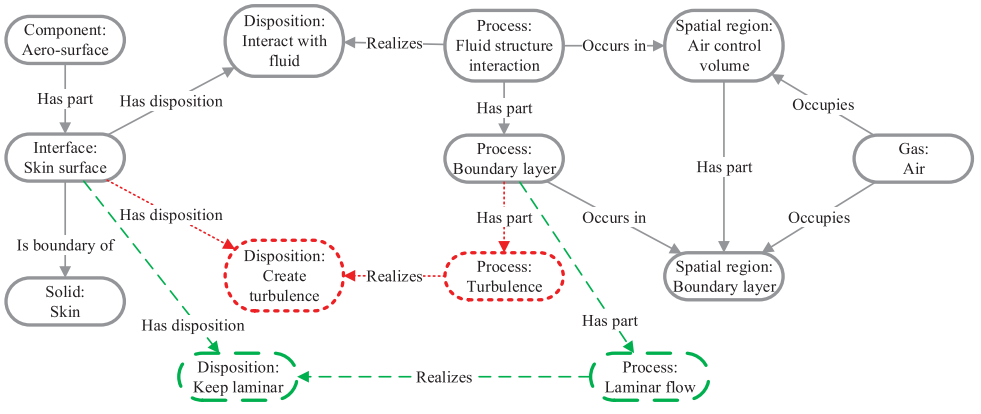


Figure 4. Graph transformation of laminar flow. Grey represents the pattern, green (dashed) the added nodes and edges, and red (dotted) the removed nodes and edges.

turbulent, the air moves chaotically, with more friction as a result. The portion of air that experiences friction from the body's surface is called the boundary layer. Air outside the boundary layer is usually regarded as laminar, while the boundary layer itself starts of as laminar flow, but quickly transitions into turbulent flow.

On any aerodynamic surface, protrusions introduce turbulence, which inhibits laminar flow. Thus when a technology introduces natural laminar flow ,[2] while another introduces turbulence, they are incompatible, according to Equation (12). When taking the pattern from Figure 3 (i.e. the grey and red parts of the graph) and the effect of Figure 4 (i.e. the grey and green parts of the graph), both a turbulent and laminar flow process are present.

To infer incompatibility between these two technologies, several rules are used. First, Equation (4) is used to define that turbulence inhibits laminar flow. Then, Equation (3) establishes that the laminar flow process in Figure 4 occurs in the boundary layer spatial region.

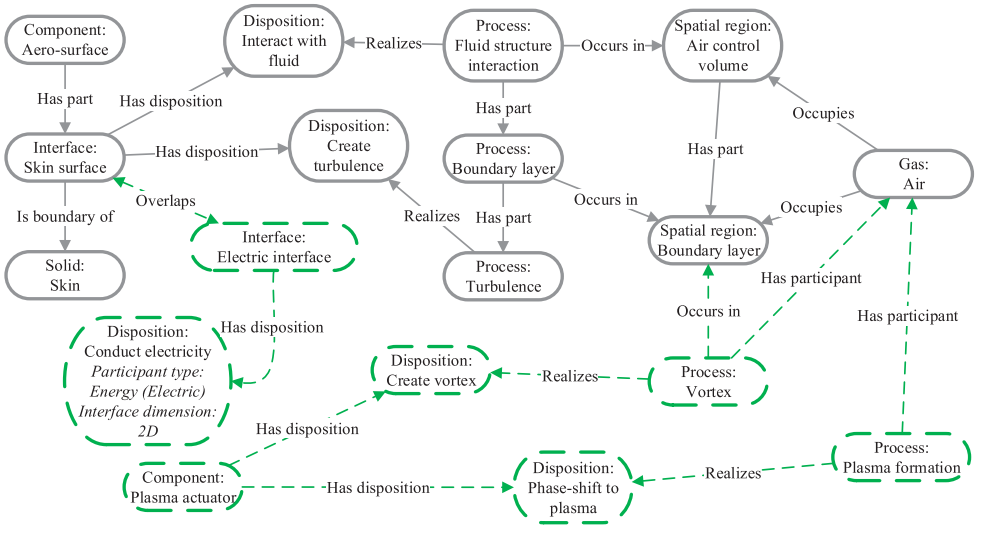


Figure 5. Graph transformation of plasma actuator. Grey represents the pattern and green (dashed) represents added nodes and edges.

This enables Equation (5) to infer that the turbulence process in Figure 3 inhibits the laminar flow process. Finally, that is the information needed for Equation (12) to deduce that the antenna is incompatible with the natural laminar flow technology.

3.3.2. Functional incompatibility: process equivalence

Although a laminar boundary layer produces less friction than a turbulent one, it is also more prone to a process called separation. Separation is the detachment of the boundary layer from the surface, as a result of the flow reversing direction close to the surface. This results in a lot of additional aerodynamic drag, and, therefore, is highly undesired. Several technologies exist that aim to turn the boundary layer turbulent in a controlled fashion, such that it will not separate. One such technology is a plasma actuator, which exploits ionisation of air to turn it into plasma, locally. This has the effect of creating small vortices, making the boundary layer turbulent. Similarly, a vortex generator has the same effect, but is simply a small vane.

A plasma actuator is modelled as shown in Figure 5. It adds the plasma actuator component (in green) to an existing aerodynamic surface (in grey). As a second technology, consider the vortex generator, which would have a similar graph, except for the disposition and process of creating plasma, and without the electric interface.The incompatibility of these two technologies follows directly from Equation (14). Because the technologies are defined similarly, the fact that p_1 **has equivalent effect** p_2 follows from Equation (13).

3.4. Enabling reasoning

Again, two rules are highlighted to showcase the enabling relationship inferences the ontology enables. First, we look at technologies that remove an inhibiting process to enable another technology in Section 3.4.1. Second, the idea of complementing interfaces is exemplified in Section 3.4.2.

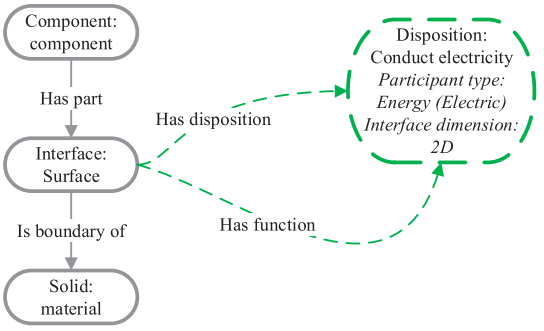


Figure 6. Graph transformation of conductive structure. Grey represents the pattern and green (dashed) represents added nodes and edges.

3.4.1. Physical enabling: removal of inhibiting process

Consider conformal antennas (see Figure 3) and natural laminar flow (see Figure 4). A conformal antenna is entirely embedded in a surface, such that the surface has no protrusions. In order to infer that the conformal antenna enables natural laminar flow, Equation (19) is used. The second part of the rule, which states that an inhibitory process or blocking disposition should be present when only natural laminar flow is applied, is true, as already examined in Section 3.3.1. After application of the conformal antenna rule, the turbulence caused by the antenna is removed. Application of the natural laminar flow rule itself causes the turbulence from the aerodynamic surface to disappear. Now, no process exists that inhibits the laminar flow process, and the first half of Equation (19) is also satisfied.

3.4.2. Physical enabling: complementing interfaces

As Figure 5 shows, the plasma actuator contains an interface that has the disposition to conduct electricity. However, there is no process that realises it, which prevents Equation (20) from firing. If the conductive structure shown in Figure 6 is present, however, such a conduction process could be instantiated between the two interfaces introduced by each technology. In order to do that, a separate rule is added to the domain ontology, that creates a transfer process whenever two interfaces overlap that have the disposition to transmit a certain type of entity. It reads:

$$
\begin{aligned}
&\exists i_1, i_2, d_1, d_2, t, id : \\
&\text{interface}(i_1) \ \wedge \ \text{interface}(i_2) \ \wedge \ i_1 \ \textbf{overlaps} \ i_2 \ \wedge \\
&i_1 \ \textbf{has disposition} \ d_1 \ \wedge \ i_2 \ \textbf{has disposition} \ d_2 \ \wedge \\
&\text{disposition}(d_1) \ \in \ \text{ToTransmit} \ \wedge \ \text{disposition}(d_2) \ \in \ \text{ToTransmit} \ \wedge \\
&d_1 \ \textbf{participant type} \ t \ \wedge \ d_2 \ \textbf{participant type} \ t \ \wedge \\
&d_1 \ \textbf{interface dimension} \ id \ \wedge \ d_2 \ \textbf{interface dimension} \ id \\
&\quad \Rightarrow \exists p_0 : \ \text{Transfer}(p_0) \ \wedge \ p_0 \ \textbf{realises} \ d_1 \ \wedge \ p_0 \ \textbf{realises} \ d_2
\end{aligned}
\tag{22}
$$

Conduction is such a transfer process, when the type of entity to transmit is electric energy.

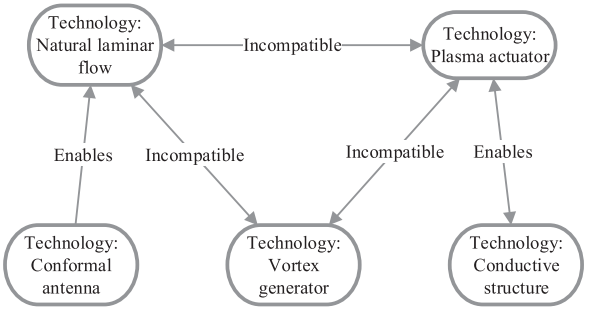


Figure 7. The technology compatibility graph (TCG), a generalisation of the TCM. Absence of a relation between two technologies indicates compatibility.

With these two technologies, Equation (22) instantiates the process that Equation (20) requires to infer that the interfaces i_1 (the skin surface) and i_2 (the electric interface) complement each other. That, in turn, causes Equation (21) to hold true, which concludes that a conductive structure enables the plasma actuator. Note that the inverse is also true, because Equation (20) infers that the two interfaces are complementary in either direction.

3.5. The technology compatibility graph

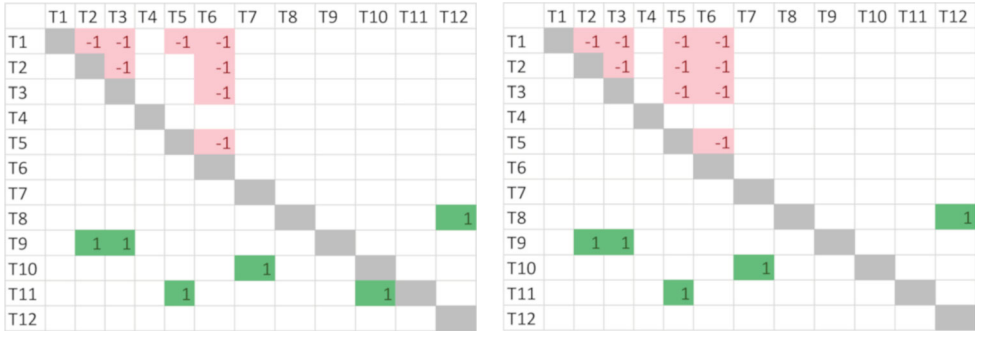
Instead of a TCM, a technology compatibility graph (TCG) is automatically constructed using the above approach. The TCG is shown in Figure 7 for the five technologies investigated in this section. Note that Figure 7 omits the **compatible** relation, because that unnecessarily clutters the graph. Thus, when no relation is depicted between two technologies, they are compatible.

3.6. Full technology set TCM

To test the applicability of the method to a practical test case, a set of technologies from a study in aircraft industry is taken and evaluated using the proposed method. The purpose of this demonstration is to verify whether the diverse set of technologies that may be encountered in industry can be modelled using the graph transformation approach, and whether the inference rules apply to these cases. The particular set of technologies considered here apply to an aircraft. They are distinct from the ones shown in the previous test case. However, due to confidentiality issues, the technologies cannot be explicitly displayed, and are replaced by generic identifiers. As Figure 8 shows, there are only few differences between the automated method and the TCM as established by a group of experts.

A pair of experts considered each pair of technologies and established the TCM in Figure 8(a) through discussion. None of the technologies were explicitly defined, and each expert had their own interpretation of it. The process involved some iteration, as viewpoints on what makes technologies incompatible or enable another changed along the way.

Based on further deliberation with the same experts, graph transformation rules were created that capture the technologies as best as possible. Because it is cumbersome to capture all details of a technology, emphasis was put on the aspects that could interfere with



	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
T1		-1	-1		-1	-1						
T2			-1			-1						
T3						-1						
T4												
T5						-1						
T6												
T7												
T8												1
T9		1	1									
T10							1					
T11					1					1		
T12												

(a) TCM by experts

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
T1		-1	-1		-1	-1						
T2			-1		-1	-1						
T3					-1	-1						
T4												
T5						-1						
T6												
T7												
T8												1
T9		1	1									
T10							1					
T11					1							
T12												

(b) TCM by method

Figure 8. TCM comparison between experts and current method. A −1 denotes incompatibility, while a 1 denotes enabling and is a directed relationship (thus, the 1 to the far right reads T8 enables T12, whereas the 1 to the far left reads T9 enables T2). (a) TCM by experts. (b) TCM by method.

other technologies. These transformation rules, along with a system graph that was constructed to accommodate all the technologies, were input to the method outlined in this paper. The resulting TCM is shown in Figure 8(b).

The first difference is the incompatibility between technologies T2 and T5, and T3 and T5. Both pairs are assessed as incompatible by the current method, due to an inhibiting process, with Equation (12). However, the reason the experts do not agree is because of the scale of the two processes. The inhibiting process does occur in the same region as the other process, but is much smaller and does not have a significant effect on it. A more comprehensive understanding of physics by the inference engine is required to deal with such intricacies. As long as that is not possible, it would be necessary for experts to explicitly model inapplicability constraints for technologies or have the ability to overrule conclusions drawn by the inference algorithm.

The second difference is the enabling relation between T11 and T10. The inability of the current method to detect this, is a result of a lack in modelling capabilities. Concretely, the graph rewriting rule implementation does not allow for optional patterns to be specified, which is required for the correct implementation of T10 that allows inference of the missing enabling relation. An optional pattern is an alternative pattern, or extension thereof, that allows application of the technology when present in a source graph.

4. Discussion

As set out in the introduction, technologies are usually modelled only as impact factors. That has the advantage of being a simple approach, yielding results quickly. However, it fails to capture knowledge of what the technologies actually are and inferences as presented herein are not possible. The present approach offers this possibility, at the expense of additional effort to define the technologies as graph transformations. In future work, this approach can be extended to offer simulation capabilities. For larger technology sets, this approach might then become less labour-intensive, because analysis methods can be automatically assigned to the technologies, as well as the values for the input variables. This reduces the need for experts to identify and assign impact factors.

For the current approach to work well, the graph transformation patterns should correctly match to the system graphs. However, when symmetries are present in either, ambiguities result as a graph matching algorithm identifies multiple matches. Concretely, this may happen when a node in the pattern is considered equivalent with multiple nodes in the system graph. When less information is present, this issue becomes more prevalent. Therefore, equivalence determination is one of the cornerstones of this approach. Further research has to be conducted to establish robust ways to determine equivalence for dispositions and processes.

In addition, a designer or analyst must have a good understanding of the ontology and how to represent systems and technologies in it. Because different individuals have different viewpoints and conceptions, their description of identical entities may differ. Furthermore, the formal language of an ontology may make it difficult to describe physical entities and phenomena properly, especially for people with less training in its use (Benjamin et al. 1996). This issue is pervasive in ontology research and application, and there seems to be a lack of sound solutions. To alleviate this problem, ontologies should be built up of multiple, general ontologies, such as a mereology, topology, geometry and physical process ontology (Benjamin et al. 1996). The modelling process should follow several steps: defining the components, defining the processes and behaviour, and defining the mathematics (Borst, Akkermans, and Top 1997). It would then be evident what to model at each step and when to transition from one step to another. This process should be supported by well-written and accessible instructions. Another option is to construct a library of systems and processes that a designer can use, rather than defining these themselves.

There may be different ontologies suitable for describing engineering systems and the physics surrounding them. The proposed ontology can be easily replaced by such other ontology. Only the rules presented in this paper have to be rewritten in terms of those ontologies, for the approach to work. In the future, for example, the PSO ontology (Cheong and Butscher 2019) appears to be a suitable candidate (partly because it also is based on BFO).

Regarding the presented ontology, we should emphasise that the categorisation under BFO (see Table 1) is debatable. For example, some or all fluids may not be regarded BFO objects, because they should be maximally causally unified. However, why would a solid piece of material be considered an object, while a gaseous portion of material is not? An atmosphere is a causally unified portion of gas molecules, so could be considered an object. We reason this is true for any (portion of) gas, and, by extension, for any fluid. The classification of energy as an independent continuant may also excite opposition. It is indeed common to regard energy as a dependent continuant, as it is usually described as a quantitative property of an object. However, due to the mass–energy equivalence, we argue energy should be treated on an equal level as matter. Considering, for example, radiant energy, it can either be described as the energy carried by photons, or as an electromagnetic wave that oscillates electric and magnetic fields. The views are equivalent, according to the wave–particle duality. This again suggests that energy and matter should be treated equivalently. Nonetheless, these classifications are not concrete and more work from the community is required to assert or reject them.

Rather than having to specify everything by hand, several relations can be determined automatically using more sophisticated software. For example, the **overlaps** relation can be computed using CAD software. Furthermore, the interaction between processes could

be determined by simulating them and observing some key qualities. Through principal component analysis and other statistical methods, the behaviour of groups of processes can be deduced, which then helps to establish the **inhibiting process of** relationships. This possibility enables the method to extend existing systems engineering practices. However, it requires a rather detailed description of the technologies and systems; something that may not be available in the conceptual design phase.

5. Conclusion

An ontology is developed based on Basic Formal Ontology to describe engineering systems and technologies affecting them. The systems can be represented as graphs onto which the ontology is mapped. The concept of graph transformation rules is applied to model the effect of technologies on a system. Rules are defined to deduce compatibility and enabling relations between technologies from these graph transformations and the ontology. Several applications show how different graph transformations trigger the execution of the rules, causing an inference chain to start and conclude whether a technology is compatible or incompatible with another, or enables another. An industry test case shows that the system is capable of mimicking human expert reasoning.

The method hinges on the ability to determine equivalence between entities in the ontology, and, especially for processes and dispositions, this equivalence definition needs to be further defined. Nonetheless, the approach offers a means to formally describe technologies and provides knowledge capturing, which comes at the expense of modelling effort. The extra modelling effort is offset by the ability to trace back why decisions are made and to explicate each technology. Future work will introduce the means to simulate the technologies for quantitative assessment.

Notes

1. Mereology is the study of parts and the wholes they form in philosophy and mathematical logic.
2. Natural laminar flow occurs when a body is shaped such that the flow around it remains laminar.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This research is sponsored by the European Commission under the CleanSky II research program as part of project MANTA with grant agreement number 724558.

References

Amadori, Kristian, Erik Bäckström, and Christopher Jouannet. 2017. "Selection of Future Technologies during Aircraft Conceptual Design." In *55th AIAA Aerospace Sciences Meeting*, 1–11. AIAA.

Amadori, Kristian, Erik Bäckström, and Christopher Jouannet. 2018. "Future Technologies Prioritization for Aircraft Conceptual Design." In *2018 AIAA Aerospace Sciences Meeting*. Kissimmee: AIAA.

Arp, Robert, Barry Smith, and Andrew D. Spear. 2015. *Building Ontologies with Basic Formal Ontology*. Cambridge: MIT Press.

Benjamin, J., P. Borst, H. Akkermans, and B. Wielinga. 1996. "Ontology Construction for Technical Domains." In *International Conference on Knowledge Engineering and Knowledge Management*, 98–114. Springer.

Borst, P., J. Akkermans, A. Pos, and J. Top. 1995. "The PhysSys Ontology for Physical Systems." In *Working Papers of the Ninth International Workshop on Qualitative Reasoning*, 11–21. Amsterdam: University of Amsterdam.

Borst, P., H. Akkermans, and J. Top. 1997. "Engineering Ontologies." *International Journal of Human-Computer Studies* 46 (2-3): 365–406.

Borst, P., A. Pos, J. Top, and J. Akkermans. 1994. "Physical Systems Ontology." In *Working Papers European Conference on Artificial Intelligence ECAI 1994 Workshop on Implemented Ontologies*, 4–80. Amsterdam.

Bush, C. L.. 1981. "Taking Hold of Technology: Topic Guide for 1981–1983." *American Association of University Women*.

Cartagena, M. A., J. E. Rosario, and D. N. Mavris. 2000. "A Method for Technology Identification, Evaluation, and Selection of Aircraft Propulsion Systems." In *36th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, Huntsville: AIAA.

Castet, J., M. L. Rozek, M. D. Ingham, N. F. Rouquette, and S. H. Chung. 2015. "Ontology and Modeling Patterns for State-Based Behavior Representation." In *2018 AIAA Infotech*, Kissimmee: AIAA.

Chakraborty, Imon, and Dimitri N. Mavris. 2017a. "Assessing Impact of Epistemic and Technological Uncertainty on Aircraft Subsystem Architectures." *Journal of Aircraft* 54 (4): 1388–1406. http://arc.aiaa.org/doi/10.2514/6.2016-3144.

Chakraborty, Imon, and Dimitri N. Mavris. 2017b. "Integrated Assessment of Aircraft and Novel Subsystem Architectures in Early Design." *Journal of Aircraft* 54 (4): 1268–1282. http://dx.doi.org/10.2514/6.2016-0215%http://arc.aiaa.org/doi/10.2514/6.2016-0215.

Cheong, Hyunmin, and Adrian Butscher. 2019. "Physics-Based Simulation Ontology: An Ontology to Support Modelling and Reuse of Data for Physics-Based Simulation." *Journal of Engineering Design*30 (10-12): 655–687.

De Kleer, Johan. 1984. "How Circuits Work." *Artificial Intelligence* 24 (1-3): 205–280.

De Kleer, Johan, and John Seely Brown. 1984. "A Qualitative Physics Based on Confluences." *Artificial Intelligence* 24 (1-3): 7–83.

Ehrig, H., C. Ermel, U. Golas, and F. Hermann. 2015. *Graph and Model Transformation*. 1st ed. Heidelberg: Springer.

Feibleman, James K. 1961. "Pure Science, Applied Science, Technology, Engineering: An Attempt At Definitions." *Technology and Culture* 2 (4): 305–317.

Forbus, Kenneth D. 1984. "Qualitative Process Theory." *Artificial Intelligence* 24: 85–168.

Gatian, Katherine N., and Dimitri N. Mavris. 2015. "Enabling Technology Portfolio Selection through Quantitative Uncertainty Analysis." In *15th AIAA Aviation Technology, Integration and Operations Conference*, 1–28. Dallas: AIAA.

Goldfain, Albert, Barry Smith, and Lindsay G. Cowell. 2010. "Dispositions and The Infectious Disease Ontology." In *Formal Ontology in Information Systems*, Frontiers in Artificial Intelligence and Applications, 400–413. Amsterdam: IOS Press.

Goldfain, Albert, Barry Smith, and Lindsay G. Cowell. 2011. "Towards An Ontological Representation of Resistance: The Case of MRSA." *Journal of Biomedical Informatics* 44 (1): 35–41. Ontologies for Clinical and Translational Research.

Gruber, T. R., and G. R. Olsen. 1994. "An Ontology for Engineering Mathematics." In *Principles of Knowledge Representation and Reasoning*, 258–269. San Mateo: Morgan Kaufmann.

Guenov, Marin D., Arturo Molina-Cristobal, Vitaly Voloshin, Atif Riaz, and Albert S. J. Van Heerden. 2016. "Aircraft Systems Architecting – A Functional – Logical Domain Perspective." In *16th AIAA Aviation Technology, Integration, and Operations Conference*. Washington: AIAA.

Helms, B., K. Shea, and F. Hoisl. 2009. "A Framework for Computational Design Synthesis Based on Graph-Grammars and Function-Behavior-Structure." In *International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*. San Diego: ASME.

Hirtz, Julie, Robert B. Stone, Daniel A. Mcadams, Simon Szykman, and Kristin L. Wood. 2002. "A Functional Basis for Engineering Design : Reconciling and Evolving Previous Efforts." *Research in Engineering Design* 13: 65–82.

Irani, M. R., and S. Rudolph. 2003. "Design Grammars for Conceptual Designs of Space Stations." In *54th International Astronautical Congress of the International Astronautical Federation, the International Academy of Astronautics and the International Institute of Space Law*. Bremen: AIAA.

Judt, David M., and Craig P. Lawson. 2015. "Application of An Automated Aircraft Architecture Generation and Analysis Tool to Unmanned Aerial Vehicle Subsystem Design." *Journal of Aerospace Engineering* 229 (9): 1690–1708.

Judt, David M., and Craig P. Lawson. 2016. "Development of An Automated Aircraft Subsystem Architecture Generation and Analysis Tool." *Engineering Computations* 33 (5): 1327–1352.

Kaderka, J. D., M. L. Rozek, J. K. Arballo, D. A. Wagner, and M. D. Ingham. 2018. "The Behavior, Constraint and Scenario (BeCoS) Tool: A Web-Based Software Application for Modeling Behaviors and Scenarios." In *2018 AIAA Aerospace Sciences Meeting*. Kissimmee: AIAA.

Karnopp, D. C., D. L. Margolis, and R. C. Rosenberg. 1990. *System Dynamics: A Unified Approach*. 2nd ed. New York: John Wiley & Sons.

Kirby, Michelle R., and Dimitri N. Mavris. 1999. "Forecasting Technology Uncertainty in Preliminary Aircraft Design." In *World Aviation Conference*, 14. San Francisco: AIAA. http://hdl.handle.net/1853/6325.

Kitamura, Yoshinobu. 2006. "Roles of Ontologies of Engineering Artifacts for Design Knowledge Modeling.In Proc. of the 5th International Seminar and Workshop Engineering Design in Integrated Product Development (EDIProD 2006).Gronów, Poland.

Kitamura, Yoshinobu, and Riichiro Mizoguchi. 2004. "Ontology-Based Systematization of Functional Knowledge." *Journal of Engineering Design* 15 (4): 327–351.

Marx, Leo. 1997. "Technology: The Emergence of a Hazardous Concept." *Social Research* 64 (3): 965–988.

Merrill, Robert S. 1968. "The Study of Technology." *International Encyclopedia of the Social Sciences*15: 576–589.

Mitcham, Carl, and Eric Schatzberg. 2009. "Defining Technology and The Engineering Sciences." In *Philosophy of Technology and Engineering Sciences*, 27–63. Amsterdam: Elsevier.

Rosenberg, R. C., and D. C. Karnopp. 1983. *Introduction to Physical System Dynamics*. New York: McGraw-Hill, Inc.

Roth, B. A., B. J. German, D. N. Mavris, and N. I. Macsotai. 2001. "Adaptive Selection of Engine Technology Solution Sets From a Large Combinatorial Space." In *37th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*. Salt Lake City: AIAA. http://smartech.gatech.edu/handle/1853/6408.

Sen, Chiradeep, Joshua D. Summers, and Gregory M. Mocko. 2011. "A Protocol to Formalise Function Verbs to Support Conservation-Based Model Checking." *Journal of Engineering Design* 22 (11–12): 765–788.

Sen, Chiradeep, Joshua D. Summers, and Gregory M. Mocko. 2013a. "A Formal Representation of Function Structure Graphs for Physics-Based Reasoning." *Journal of Computing and Information Science in Engineering* 13: 1–13.

Sen, Chiradeep, Joshua D. Summers, and Gregory M. Mocko. 2013b. "Physics-Based Reasoning in Conceptual Design Using a Formal Representation of Function Structure Graphs." *Journal of Computing and Information Science in Engineering* 13: 1–12.

Smith, Barry. 2012a. "Classifying Processes: An Essay in Applied Ontology." *Ratio* 25 (4): 463–488.

Smith, Barry. 2012b. "On Classifying Material Entities in Basic Formal Ontology." In *Interdisciplinary Ontology: Proceedings of the Third Interdisciplinary Ontology Meeting*, 1–13. Tokyo: Keio University Press.

Soban, Danielle S., and Dimitri N. Mavris. 2013. "Assessing the Impact of Technology on Aircraft Systems Using Technology Impact Forecasting." *Journal of Aircraft* 50 (5): 1380–1393. http://arc.aiaa.org/doi/abs/10.2514/1.C031871.

Stavrev, V.. 2011. "A Shape Grammar for Space Architecture – Part II. 3D Graph Grammar – An Introduction." In *41st International Conference on Environmental Systems*. Portland: AIAA.

Utturwar, Aditya, Sriram Rallabhandi, Daniel DeLaurentis, and Dimitri Mavris. 2002. "A Bi-Level Optimization Approach for Technology Selection (For Aircraft Design)." In *9th AIAA/ISSMO Symposium and Exhibit on Multidisciplinary Analysis and Optimization*. Atlanta: AIAA.