Automatically Generating the Technology Compatibility Matrix Using Graph Transformations

Roelofs, Martijn; Vos, Roelof; Amadori, Kristian; Jouannet, Christopher

DOI
10.2514/6.2019-2886

Publication date
2019

Document Version
Final published version

Published in
AIAA Aviation 2019 Forum

AIAA AVIATION Forum
17-21 June 2019, Dallas, Texas
AIAA Aviation 2019 Forum

10.2514/6.2019-2886

Automatically Generating the Technology Compatibility Matrix Using Graph Transformations

Martijn N. Roelofs* and Roelof Vos†
Delft University of Technology, Delft, Zuid-Holland, 2600AA, The Netherlands

Kristian Amadori‡ and Christopher Jouannet§
Advanced Design and Survivability, Saab Aeronautics, Linköping, Sweden

One of the first stages during technology evaluation and selection is constructing the technology compatibility matrix, which indicates the compatibility of each pair in a technology portfolio. Construction of the technology compatibility matrix usually involves subjective expert judgment, which inhibits assessing each pair consistently and storing the decision rationale. Therefore, this study develops a method based on graph transformations, allowing for a formal theoretical description of technologies. In order to automate this process, two algorithms use these graph transformations to deduce technology compatibility and dependencies between technologies. The first checks whether changes made by a pair of transformations are compatible with each other. The second defines dependencies for each technology and attempts to resolve these using other technologies. The method is put into practice for a technology portfolio within Saab Aeronautics and the automatically generated compatibility matrix is compared to a pre-existing expert-judgment-based matrix for the same portfolio. Roughly 90% of the entries in the algorithm-based TCM are identical to the expert-based TCM. Therefore, the automated method is capable of capturing much of human reasoning in this context. However, in order to fully agree with human expertise, physics and design reasoning should be included to expand the scope of inference.

Nomenclature

d	=	Dependency
G	=	System Graph
K	=	Gluing Graph
L	=	Pattern Graph
R	=	Replacement Graph
t	=	Technology
FD	=	Functional Decomposition
IRL	=	Integration Readiness Level
MCS	=	Maximum Common Subgraph
MoE	=	Measure of Effectiveness
SoI	=	System of Interest
SRL	=	System Readiness Level
TCM	=	Technology Compatibility Matrix
TRL	=	Technology Readiness Level
QoI	=	Quantity of Interest

*PhD Candidate, Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1 2629HS, Delft, The Netherlands
†Assistant Professor, Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1 2629HS, Delft, The Netherlands, Associate Fellow AIAA
‡PhD, Advanced Design and Survivability, Member AIAA
§PhD, Advanced Design and Survivability, Member AIAA

I. Introduction

Early in the conceptual design of engineering systems it is of utmost importance to select those components, technologies and configurations that are most likely to achieve the specified requirements, while reducing risk and cost of development, manufacturing and operations. The term technology encompasses tools, techniques, processes, and the resulting products. Therefore, this study views any change to an existing system as a technology, including the development and manufacturing processes. Technology selection is then the decision making process that establishes a set of technologies, called a technology portfolio, which satisfies all imposed requirements and achieves the set goals. Previously, a literature review [1] identified a plethora of challenges with technology selection of which two are addressed in this paper: (1) modeling of technologies is usually avoided and replaced by impact factors and (2) extensive use of expert judgment. Modeling technologies as impact factors fails to capture specific knowledge of what the technology represents. Furthermore, when only few impact factors are available, technologies cannot be differentiated properly. Expert judgment is not a problem by itself, but introduces the challenges of subjectivity and inconsistency. Therefore, relying on expert judgment extensively counteracts minimizing risk and cost of the development effort.

Conventionally, technology selection is performed by collecting technology readiness (TRL) and integration readiness (IRL) levels, describing their effects in an impact matrix and their compatibility in another matrix. Such an approach was taken by Amadori et al. [2, 3]. The data collection phase was essentially carried out completely using expert judgment, reducing traceability, objectivity and consistency. The technology compatibility matrix (TCM) from those studies is used in this work, and while reassessing its entries some inconsistencies were found or the original rationale had to be recalled.

In previous work [4], a formal modeling approach based on an ontology and graph theory was developed for defining engineering systems. It is based on funcitonal decomposition, where the functional basis by Hirtz et al. [5] was taken as a starting point. Other efforts have been made towards similar goals. A functional decomposition (FD) describing aircraft system architectures was developed by Judt and Lawson [6, 7], to consecutively enumerate architectures and search for the best solution with a hybird heuristic optimization. Their FD is problem-specific, as is the analysis method, and therefore not easily generalized. The same holds for AirCADia [8], 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. Where others start off with a FD, Yuan et al. [9] developed a method to automate the FD process itself, which may be useful for end-users to specify their intentions when modeling a system. Sen et al. [10–12] used function–structure graphs to describe the behavior of a system enabling physics-based reasoning on it. Although effective, it appears to only be applicable to mechanical and eletrical engineering domains, while continuum mechanics seem to pose a problem to this approach. The BeCoS tool developed by Kaderka et al. [13, 14] uses an ontology to describe systems semantically rigorous in terms of their behavior. State-machines describe transitions within the behavior and, combined with equations, enable analysis of the system. Most of these studies focus on design synthesis or only on knowledge capturing and assessment of a specific system. None are specifically focused on modeling technologies or enabling technology selection. Nonetheless, they provide meaningful insights and methods to describe engineering systems (and hence technologies) in terms of functions and behaviors. By combining these into a graph data structure, and including means to represent technologies, the challenges presented above are addressed.

In the present work, graph transformations are introduced as the workhorse for specifying technologies. Graph transformations are part of graph grammars, which have been around since the 1970s and have seen some use in design automation and synthesis [15–17]. How they enable automated deduction of the technology compatibility matrix — one of the first steps in a technology evaluation process — is explained in Section II. A previously analyzed technology portfolio, for which the compatibility matrix was constructed using expert judgment, is compared to the automatically generated version obtained with the present method in Section III. This way, the applicability and advantages of this theoretical framework are evaluated, and concluded in Section IV. That section also clarifies how this method enables a semantic description of technologies, avoiding the use of impact factors as descriptors. Furthermore, it addresses how, although expert judgment is still needed, expert subjectivity is reduced, but cannot be fully avoided.

II. Methodology

This methodology builds upon the idea to represent systems and technologies with graphs, following a specific ontology [4]. This section first recaps baisc graph theory notions, and continues with a summary of the ontology that is used. It then proceeds by describing the graph transformations that are used to represent technologies. Finally, the two algorithms to deduce the technology compatibility matrix automatically are introduced and explained.

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. Therefore, it is analogous to an ontology. A typed graph (G, type_G) is a graph adhering to a type graph, which in this paper follows the ontology from Ref. [4]. An attributed graph adds three elements to the tuple describing the graph: (V_A, E_{VA}, E_{EA}). V_A are attribute nodes, typically consisting of a name–value pair, while E_{VA} are the node–attribute edges and E_{EA} the edge–attribute edges.

The type graph for this method is shown in Figure 1. The ontology consists of components, behaviors and flows, which together form a system description. A behavior is classified as either transformation, transmutation or design, based on how it affects the flows that it interacts with. Transformations only modify attributes of the flows, while transmutations change the type of flow through a process. The design behavior was included to signify modifications to a system during development. A flow is a physical entity, classified as material, energy or signal (each of which can be further subdivided), or a combination thereof. Sometimes a flow represents a component (e.g. in a manufacturing process), but more often it describes the electricity through a cable, the airflow around a wing or the fuel in a tank. Components are nodes that get their definition from the context of the graph; more specifically, which behaviors they execute. Problem-specific attributes can be defined on the components to further detail them. A component executes one or multiple behaviors, which is indicated using the corresponding edge. Furthermore, a component can be described as being equivalent to a flow, when it is necessary to define its material representation. As explained above, behaviors act on flows. Edges between a flow and a behavior then specify the direction of the flow with respect to the behavior as inflow or outflow. Finally, each of the three node types can be hierarchically decomposed using parent–child relationships with nodes of the same type.

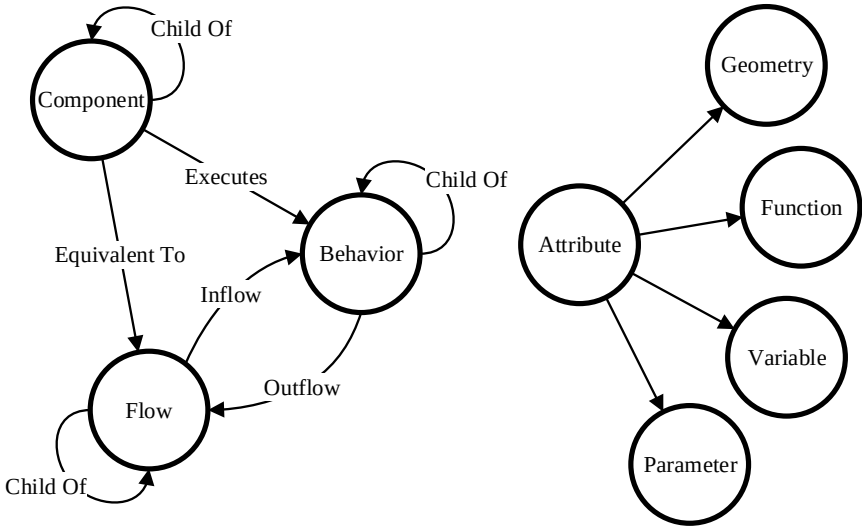
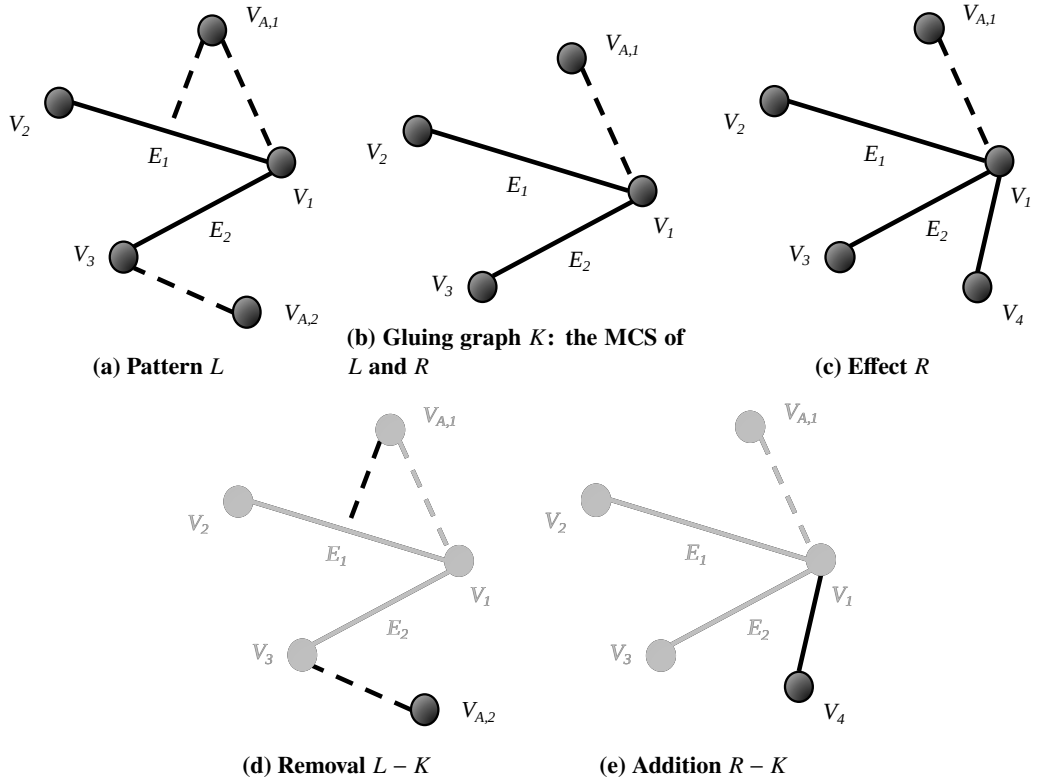


Fig. 1 The type graph that represents the ontology.

A multitude of attributes can be defined on either nodes or edges (also shown in Figure 1). The geometry attribute holds a qualitative description of a shape, and has means to compare itself to other geometries. The function attribute should be applied to edges, and points to a particular attribute of the target node. It specifies how this attribute is affected: it either increases, decreases, is kept constant, or is optimized. Variables and parameters are both numerical attributes, including a quantity type. The variable allows for different values in different states of the system, while a parameter is constant.

The method presented so far only allows describing the behavior of an engineering system, which enables quantitative analysis of it. To extend this method, technologies are now included in the form of graph transformations, which are a formal means to specify a change to a graph. A graph transformation $t : (L \leftarrow K \rightarrow R)$ is a construct consisting of three graphs: the pattern L, the gluing graph K and the replacement graph or effect graph R. An abstract example is shown in Figure 2. Essentially, the pattern L (Figure 2a) is to be matched inside a certain graph (the system in this case), while the replacement graph R (Figure 2c) replaces the matched instance of L, if it exists. The means of doing this is by first finding a maximum common subgraph (MCS) of L and R, which is the largest subgraph that is present in both these graphs, as visualized in Figure 2b. Note this can also be empty when L and R have no common substructure. We call

this MCS a gluing graph K, since the items in this graph are untouched by the application of the graph transformation and form the bridge between the pattern and effect graphs. The difference between L and K, denoted by $L - K$, are the nodes and edges which are removed by the graph transformation (Figure 2d). Similarly, the difference between R and K, i.e. $R - K$ are the nodes and edges which are added by the graph transformation (Figure 2e). Using this formal technique, technologies can be defined to alter attributes of components, introduce new components, add or remove behaviors from components, etc. Moreover, a technology can be defined as generic or specific as intended by the analyst. A subgraph isomorphism (pattern matching) algorithm finds all the areas where the technology can be applied to a system, making the technology definition reusable across multiple studies and systems.



(a) Pattern L **(b) Gluing graph K: the MCS of L and R** **(c) Effect R**

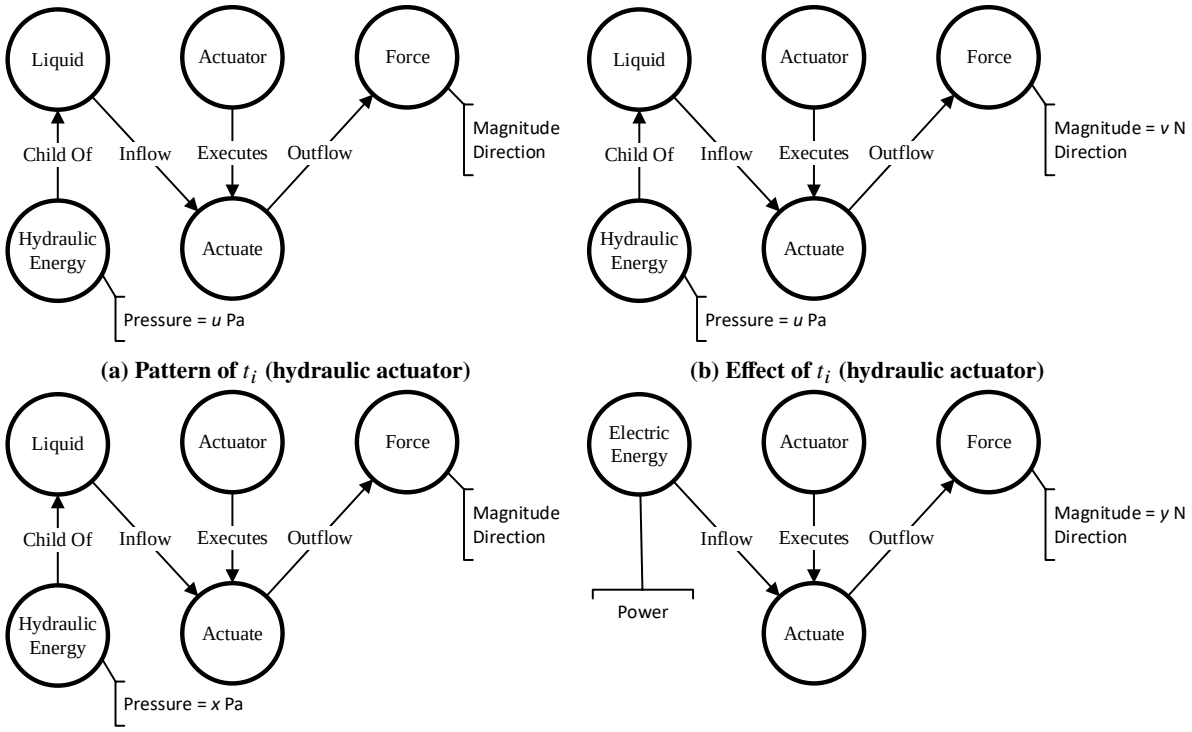
(d) Removal $L - K$ **(e) Addition $R - K$**

Fig. 2 Principles of a graph transformation rule, specified over a pattern L and effect R. The gluing graph K is the maximum common subgraph of L and R. The gray edges and nodes in the removal and addition graphs are not present in those graphs, but visualize the difference with L and R, respectively.

As a running example, think of two actuator technologies, as depicted in Figure 3. Suppose the baseline is a hydraulically driven actuator. The first technology only specifies (in its pattern) that the actuator has an input pressure of u Pa. It adds an attribute to the output force, setting the magnitude to v N. The second technology replaces the hydraulic power with electric power. Furthermore, it does this for actuators with an input pressure of x Pa. It results in a force of y N.

With the technology definitions established, the aim is to form portfolios, i.e. sets of technologies, which may be applied to the system of interest and consecutively observe the performance of that portfolio on quantities of interest. However, enumerating all 2^n possible combinations is prohibitive. As such, one means to reduce the search space is the technology compatibility matrix (TCM), which is a symmetrical matrix that shows for each pair of technologies their level of compatibility. Typically, a binary indication is used, but here a more elaborate classification is used:

$$(t_i, t_j) \mapsto \begin{cases} -1 & \text{incompatible} \\ 0 & \text{compatible/independent} \\ 1 & \text{possible synergy} \\ 2 & \text{required combination} \end{cases} \tag{1}$$



(a) Pattern of t_i (hydraulic actuator) **(b) Effect of t_i (hydraulic actuator)**

(c) Pattern of t_j (electric actuator) **(d) Effect of t_j (electric actuator)**

Fig. 3 Example of graph transformations representing two actuator technologies.

This classification captures more information about how pairs of technologies are expected to interact, such that creating portfolios from the TCM becomes more efficient and effective. When a pair of technologies (t_i and t_j) is considered, the

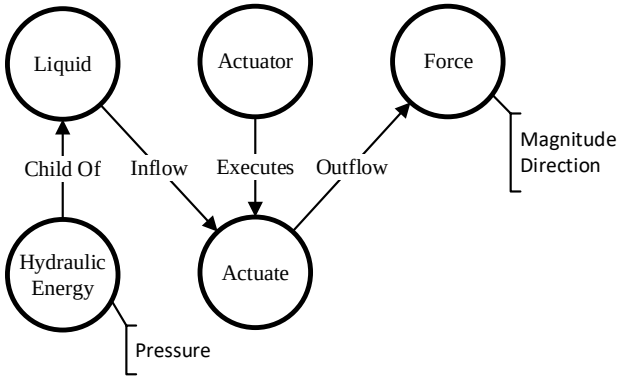


Fig. 4 The common subgraph between the patterns of the technologies in Figure 3. Note that only the value of the pressure attribute is missing with respect to each pattern graph.

principles of parallel and sequential independence (of graph transformations) are applicable. It is deemed beyond the scope of this text to explain them in detail (interested readers can consult Ref. [18]), but the first three steps as explained below represent these principles. Each step is exemplified using the actuator technologies shown in Figure 3. The pseudo-code for the algorithm deciding compatibility of a pair of technologies is given in Algorithm 1. First the MCS between the pattern graphs of both technologies is found, denoted by $K_{i,j}$. For the example case, this graph is shown in Figure 4. If it is not empty, i.e. there is overlap between the technology patterns, the following four cases are considered:

1) Lines 3 to 8. If there are overlapping parameters in the patterns, yet the values that are required differ, the patterns

are incompatible, and as such, the technologies can not be applied simultaneously. All attributes in both patterns are considered. Attributes with equal name, for corresponding nodes or edges (which are mapped by $K_{i,j}$), are compared (by value). For the actuator technologies, the overlapping parameter in the patterns is the hydraulic pressure. When $u \neq x$, the technologies are incompatible, because different actuators (based on input hydraulic pressure) are targeted.

2) Lines 9 to 11. If $K_{i,j}$ intersects with the removal graph of either one of the technologies, they are considered incompatible. This is because one of the technologies removes part of the pattern of the other. For example, if $u = x$, the patterns for both actuator technologies are equal, and Step 1 does not indicate incompatibility. However, when the second technology is applied, the hydraulic input power has been replaced with electrical power. Hence, the first technology cannot be applied to that actuator anymore, and therefore they are incompatible.

3) Lines 12 to 19. If both technologies add an equivalent attribute to the same element of the graph, but with differing value, their effects are incompatible. This is evaluated similarly to item 1. The relation between the effects of both technologies is shown in Figure 5. So elements in R_i can be mapped to R_j through K_i of the graph transformation describing technology i, then $K_{i,j}$ and finally the transformation's gluing graph K_j of technology j. Evidently, the inverse mapping is also possible. Either way, the parameters changed by K_i and K_j are taken, which will be two maps from L to R of their respective technology. These maps are then intersected based on the map $K_{i,j}$, to find those attributes that change on elements which are considered equal between both pattern graphs L_i and L_j. Finally, for these parameters, their value in their corresponding effect graph is considered, and checked for equality. When at least one is found, the technologies are incompatible. Both actuator technologies in the example add a specification of the output force. However, when $v \neq y$ different values are specified, and thus the technologies are incompatible.

4) Lines 20 to 26. Whenever two equivalent behaviors are combined in the same location, incompatibility is asserted. This conclusion is based on engineering judgment, as it leads to redundant behavior. As explained in section III, this step should, in future developments, be replaced with a more elaborate algorithm that performs physics reasoning.

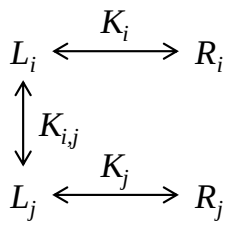


Fig. 5 Relation between effect graphs R of two technologies i and j. The arrows map elements of two related graphs through the indicated gluing graph K.

The first three steps are applicable to graph transformations in general. The step in lines 28 to 32 is performed even when the technology patterns do not overlap. When both technologies affect the same parameter in the same way, a possible synergy is indicated. For example, when both decrease mass (not necessarily of the same component), the total reduction in mass is at least the sum of the two, but the two individual effects may combine into an even more pronounced effect. Finally, when none of the five conditions are met, the pair is considered independent.

Since, the TCM is symmetric, only bidirectional relationships can be represented. However, asymmetric or directed relationships between pairs of technologies are equally possible. The first is the enabling relationship, which indicates a certain technology enables another. Conversely, the second is the disabling relationship. A third relationship, co-enabling, is symmetric, but still treated separate from the TCM, since it is inferred using the same algorithm as the former two. Algorithm 2 shows this inference process, which is based on dependencies that are established for each technology. The results from this algorithm can be represented as a graph, as shown in Figure 6. Although only pairs of technologies are considered presently, this graph could show groups of technologies enabling another technology, using hyper-edges. Such higher-dimensional relationships are impossible to show in a TCM. The inference process would be

Algorithm 1 Pseudo-code for TCM algorithm

Require: Technologies $t_i : (L_i \leftarrow K_i \rightarrow R_i)$ and $t_j : (L_j \leftarrow K_j \rightarrow R_j)$

1: $K_{i,j} \leftarrow$ GetMaximumCommonSubgraph(L_i, L_j)
2: **if** $K_{i,j} \neq$ NULL **then**
3: $\mathcal{P} \leftarrow$ ParameterChanges($K_{i,j}$)
4: **for all** $(p_n, p_m) \in \mathcal{P}$ **do**
5: **if** p_n and p_m are not commensurate **then**
6: **return** -1
7: **end if**
8: **end for**
9: **if** $(R_i - K_i \cap K_{i,j} \neq \emptyset)$ or $(R_j - K_j \cap K_{i,j} \neq \emptyset)$ **then**
10: **return** -1
11: **end if**
12: $\mathcal{P}_i \leftarrow$ ParameterChanges(K_i)
13: $\mathcal{P}_j \leftarrow$ ParameterChanges(K_j)
14: $C \leftarrow \mathcal{P}_i \cap K_{i,j} \cap \mathcal{P}_j$
15: **for all** $(c_n, c_m) \in C$ **do**
16: **if** c_n and c_m are not commensurate **then**
17: **return** -1
18: **end if**
19: **end for**
20: $\mathcal{B}_i \leftarrow$ BehaviorsIn(R_i)
21: $\mathcal{B}_j \leftarrow$ BehaviorsIn(R_j)
22: **for all** $(b_n, b_m) \in \mathcal{B}_i \times \mathcal{B}_j$ **do**
23: **if** location $b_n \equiv$ location b_m and $b_n \equiv b_m$ **then**
24: **return** -1
25: **end if**
26: **end for**
27: **end if**
28: $\mathcal{F}_i \leftarrow$ FunctionAttributesIn($R_i - K_i$)
29: $\mathcal{F}_j \leftarrow$ FunctionAttributesIn($R_j - K_j$)
30: **if** $\mathcal{F}_i \cap \mathcal{F}_j$ **then**
31: **return** 1
32: **end if**
33: **return** 0

identical to what is done here, albeit in a prohibitively large search space, for which no efficient algorithm has been conceived, yet.

As explained above, Algorithm 2 relies on dependencies a technology introduces. Currently, only one type of dependency is considered: the removal or addition of flows in the graph transformation. Flows that serve as inflow, outflow, or both form such dependencies. For example: one technology adds an electricity inflow, then it requires another technology to add an electricity outflow, such that the two can be connected. Each dependency d_i contains the functions ResolvesWith(d_j) and Disables(d_j), which both return a boolean value. The first returns true when the dependencies d_i and d_j resolve with one another. The second returns true when d_i has the opposite effect of resolving with d_j. For example, d_i is the removal of a pneumatic power outflow, while d_j represents the addition of a pneumatic power inflow, it is clear that d_i disables d_j. However, since multiple dependencies may be present for each technology, each dependency may be resolved with another dependency. The technologies are then still compatible. Therefore, the algorithm only checks for disabling after it establishes that no other relationships are present between the technologies.

The ResolvesWith(d_j) function is used in Algorithm 2 during the constraint satisfaction problem (CSP) to compute pairs of resolving dependencies between two technologies (see line 4). A CSP is given a set of variables (the dependencies of d_j in this case) and a value domain for each variable (the dependencies of d_i that resolve with that dependency of d_j). It proceeds to find a value for each variable, while satisfying any constraints defined. In this case, the only constraint is that each variable should be assigned a unique value. As such, a bijection (one-to-one map) is established between

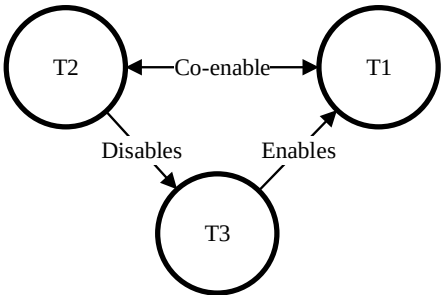


Fig. 6 Enabling graph relationships. This graph is an extension to the TCM, showing directed (asymmetric) relationships. The co-enabling relationship is symmetric, but shown in this graph, since it is the output of Algorithm 2.

Algorithm 2 Pseudo-code for enabling graph algorithm

Require: Technologies $\mathcal{T}$

1: **for all** $(t_i, t_j) \in \mathcal{T} \times \mathcal{T}$ **do**
2: $\quad \mathcal{D}_i \leftarrow$ GetDependencies(t_i)
3: $\quad \mathcal{D}_j \leftarrow$ GetDependencies(t_j)
4: $\quad s \leftarrow$ ConstraintSatisfactionProblem($\mathcal{D}_j$, $\mathcal{D}_i$)
5: $\quad$ **if** $s \neq$NULL **then**
6: $\quad\quad$ **if** $|\mathcal{D}_i| = |\mathcal{D}_j|$ **then**
7: $\quad\quad\quad$ **return** (t_i, t_j) co-enable
8: $\quad\quad$ **else**
9: $\quad\quad\quad$ **return** t_i enables t_j
10: $\quad\quad$ **end if**
11: $\quad$ **else if** $L_j \subseteq R_i$ and not $L_j \subseteq L_i$ **then**
12: $\quad\quad$ **return** t_i enables t_j
13: $\quad$ **else**
14: $\quad\quad$ **for all** $(d_n, d_m) \in \mathcal{D}_i \times \mathcal{D}_j$ **do**
15: $\quad\quad\quad$ **if** d_n disables d_m **then**
16: $\quad\quad\quad\quad$ **return** t_i disables t_j
17: $\quad\quad\quad$ **end if**
18: $\quad\quad$ **end for**
19: $\quad$ **end if**
20: **end for**

the dependencies. Alternatively, there may be no solution. If there is a solution and the number of dependencies for both technologies are equal, all dependencies are resolved, so a co-enabling relationship is found (see lines 6 to 10). Otherwise, all of t_j's dependencies are resolved by t_i (recall that $\mathcal{D}_j$ are used as variables to the CSP, whereas $\mathcal{D}_i$ are the values for those variables). The Disables(d_j) function is used when no solutions to this problem have been found, and all dependencies of both technologies are checked for disabling in lines 13 to 18. Finally, there is a dependency-independent enabling relationship when the pattern L_j of t_j is found as a subgraph of the effect R_i of t_i (see lines 11 and 12).

Even though the relationships resulting from Algorithm 2 are asymmetric, they are indicated in the TCM, to allow proper comparison with the manually defined TCM. Each disabling relationship is classified as -1 in the TCM, while each co-enabling relationship is classified as a 2, following exactly the definition in Equation 1. Finally, an enabling relationship is classified as 1, because there is no strict requirement to implement the enabling technology (otherwise, a 2 would be appropriate), but using it probably provides a more beneficial solution than using some other component or implementing the other technology as-is (hence, a 1 is more appropriate than a 0).

III. Results and Discussion

A technology portfolio of 26 technologies from Saab Aeronautics was implemented to demonstrate the method and validate the derived compatibility matrix (TCM). The original set of technologies, and manually derived TCM were published earlier by Amadori et al. [3]. Due to confidentiality concerns, the technologies nor their graph transformations are presented here. Only the resulting TCM is shown with the names of the technologies left out.

For each technology a transformation rule was defined, reflecting both its physical implementation and functionality. Consecutively, each pair of technologies was assessed using algorithms 1 and 2. The results from the two algorithms were combined as described at the end of the previous section (i.e. representing the enabling graph in the TCM).

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24	T25	T26
T1		0	1	0	1	0	0	1	-1	0	0	0	0	0	0	0	0	0	1	0	0	-1	1	1	-1	1
T2			1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T3				-1	0	0	0	0	0	0	0	0	0	0	0	0	0	1, 2	1, 0	0	0	0	0	0	0	0
T4					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T5						0, 1	1	1	0	0	-1	0	0	0	0	0	0	2	1, 0	0	0	0	1, 0	0	0	0
T6							1	1, 0	0	0	-1	1, 0	0	0	0	1, 0	0	0	0	0	0	0	0	0	0	0
T7								1, 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T8									0	0	-1	0	0	0	0	0	0	2	1, 0	1, 0	0	0	1, 0	0	0	0
T9										-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T10											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T11												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T12													1	1	1	1	2, 0	1	1	1	1	0	0	0	0	0
T13														1	1	1	0	1	1	1	2, 1	0	0	0	0	0
T14															1	1	0	1	1	1	2, 1	0	0	0	0	0
T15																1	0	1	1	1	1	0	0	0	0	0
T16																	0	1	1	1	1	0	0	0	0	0
T17																		2, 0	1, 0	1, 0	1, 0	0	0	0	0	0
T18																			0, 1	1	1	0	0, 1	0, 1	0	0, 1
T19																				1	1	0	0	0	0	0
T20																					1	0	0	0	0	0
T21																						0	0	0	0	0
T22																							-1, 0	-1, 0	2, 0	0
T23																								0, -1	-1, 1	0
T24																									0, 1	0
T25																										0
T26																										

2	Required pair
e, a	Difference (expert-based, algorithm-based)
1	Compatible, possible synergy
-1	Incompatible
0	Independent

Fig. 7 Automatically derived TCM, compared with the manually defined one. Green cells marked with 2 indicate a mandatory combination. Green cells marked with 1 indicate compatibility and possible synergy. Cells marked with 0 indicate independence, whereas red cells marked with -1 indicate incompatibility. Last, yellow cells show differences between the manually defined TCM (mark before comma) and the automatically derived one (mark after comma).

The results are summarized in Figure 7, which shows the TCM as derived manually and automatically. Differences are marked in yellow, where the number before the comma was manually defined, and number after the comma is the result from the automated method. A summary of the entry counts in the TCM is given in Table 1. In total there were 325 pairs of technologies evaluated, of which 88 contained a non-zero entry (i.e. there is a dependency). Of these entries, 55 were identified identical to the manually defined TCM (63%), whereas 33 were different (38%).

Table 1 Entry counts of joined TCM from Figure 7

	Count	% of Non-zero	% of Total
Cells	325	N/A	100
Zeros	237	N/A	73
Non-zeros	88	100	27
Matches	55	63	17
Differences	33	38	10

All differences ending with a 0 indicate that the algorithm was unable to find a dependency between that pair of

technologies. That means insufficient information was available, or the reasoning mechanism lacked inference rules. In most cases, the manually defined dependencies are based on interaction of physical processes. The reasoning mechanism is not yet capable of representing these processes in sufficient detail, nor able to make inferences about their interactions. Mainly in row T18, differences following a 0,1 pattern are present. Therefore, the algorithm shows a dependency where the experts do not. In these particular cases, there is indeed a dependency between the involved technologies, such that T18 enables T19, T23, T24 and T26. However, T18 is not necessarily required. Whether or not it is better to indicate a 0 or 1 in these cases is arguable. Nonetheless, it shows the algorithm is more consistent in its reasoning than the human experts. Each cell with 2,1 (i.e. the experts indicated a 2, but the algorithm a 1), the method was capable of detecting a dependency, although it failed to capture the significance of these dependencies. This again is a result of the absence of in-depth physics reasoning. The difference in cell (T5, T6) is likely due to an incorrect indication in the manually defined TCM, since physically and technically there is no reason why these two technologies would be incompatible. Conversely, cell (T23, T24) should show a 0, but here a bug in the algorithm leads to the faulty -1. Cells (T23, T25) and (T24,T25) show differences due to a lack in modeling capabilities of the present method. Essentially, these technologies revolve around aircraft-wide systems, rather than location-specific technologies. In fact, one could say T23 and T25 are design philosophies, something about which the reasoning algorithm cannot make inferences.

In order to improve the assessments made by the algorithms, two reasoning mechanisms should be included: physics and design reasoning.

Physics reasoning Many dependencies are a result of coupled physical processes, or understanding of material properties and behavior. As of yet, the ontology is lacking sufficient means to describe the processes on which a technology relies and how it interacts with these. Furthermore, no reasoning mechanism is in place to infer dependencies between them. As a solution, a graph could be constructed with cause–effect relationships between the processes, which include not only physical processes (e.g. aerodynamic phenomena, heat transfer or structural failure), but also development and operational processes (e.g. design, manufacturing, or maintenance). These should be referenced by the behaviors implemented by technologies, after which an inference mechanism can traverse the cause–effect graph to infer dependencies between technologies.

Design reasoning Inspecting the differences closely reveals that the ontology lacks the ability to capture systems integration. To elaborate, it is capable of representing changes to a specific component, but less so to a complete system. One possible solution is to include a multiplicity parameter for components, explicating a set is present in a system, rather than just an individual. This ties in with another issue; several technologies are not so much a change to the system, but more of a design philosophy. Enabling reasoning about this would at least require specifying the functions of flows themselves, such that alternative solutions for the same intended functions may be found, which can be captured as design rationale.

Concerning the TCM itself, an even more elaborate scoring should be considered:

$$(t_i, t_j) \mapsto \begin{cases} -2 & \text{incompatible} \\ -1 & \text{compatible, detrimental} \\ 0 & \text{compatible/independent} \\ 1 & \text{possible synergy} \\ 2 & \text{required combination} \end{cases} \tag{2}$$

In this case, a detrimental relationship is included. It was found that many technologies that are indicated as incompatible are, in fact, compatible, if purely physics-based reasoning is applied. However, combining these technologies would have a detrimental effect on either one's performance, using engineering judgment. Essentially, this relationship indicates bad engineering practice, and during portfolio creation will be treated identically to being incompatible. Inclusion of the extra relation is, therefore, not mandatory, but may refine the TCM.

The TCM is used to eliminate unfeasible technology portfolios from the total set of possible combinations. Therefore, the differences from the expert-based TCM with the automatically generated TCM lead to differences in creating the technology portfolios and consequently in prioritizing the technologies. With the new TCM, 1.67 million possible technology portfolios are created, as opposed to only 340 thousand for the pre-existing TCM. This is roughly a five-fold increase, although one should keep in mind that the total number of possibilities (without TCM) is $2^{26} = 67.1$ million. Hence, the TCM still succeeds in reducing the amount of feasible technology portfolios.

Investigating the effect of all technology portfolios on a system characteristic yields a thumb plot as depicted in Figure 8, sorted by SRL in this case. A Pareto front of portfolios results, which trade offs SRL with the effect on a QoI (i.e. a system characteristic). Comparing the thumb plots for the expert-based TCM (Figure 8a) with the one for the algorithm-based TCM (Figure 8b) shows that the maximum absolute effect on the system characteristic has increased, so the added portfolios included better performing ones as well. This indicates the pre-existing TCM was too restricting, although, conversely, the new TCM may contain infeasible portfolios, because several incompatibilities were not detected. On another critical note, the effects here are estimated by projecting from the sensitivities calculated for the simulation environment. Therefore, the actual magnitude of the system characteristic may shift when a recalculation of the simulation is performed for each portfolio. Despite this shift, it is expected that the overall benefit is retained.

Studying the frequency of the individual technologies occurring in portfolios on the Pareto front produces the bar plots in Figure 9. Comparing these provides insight into how the changes in the TCM reflect in the relative importance of the technologies. Most notably, technology 2 occurs in none of the former portfolios, while appearing in approximately 70% of the current portfolios. Conversely, technology 11 had some frequency formerly, but has dropped from all portfolios with the new TCM. Surprisingly, for both these technologies, all entries in the TCM are unchanged, as shown in Figure 7. Therefore, this result is attributed to shifting relationships between other technologies and the added portfolios. This also shows that no direct relation exists between changes in the TCM and the shifting in technology frequency on the Pareto front. Several other technologies have an increased frequency in the new Pareto front: technologies 6, 18, 19, 23 and 24. In some cases the expert-based TCM contains more zeros than the algorithm-based TCM, while for others the opposite is true. Again, no direct conclusion can be drawn from the changes in the TCM. Nonetheless, the automatically generated TCM allowed to right some erroneously defined relations between technology pairs and since the technology prioritization changes as a result, it shows these changes are significant. More importantly, as consistency is increased, a more correct set of technologies may now be selected that includes promising technologies left out otherwise. In other words, the confidence in the results has increased.

IV. Conclusion

This study extends an existing ontology by applying the concept of graph transformations to model the effect of technologies on a system. Compatibility between technologies is deduced through the parallel and sequential independence rules, which are generically defined for graph transformations. Two additional rules are based on the applied ontology: redundant behaviors in the same location result in incompatibility, and affecting an equivalent parameter identically may result in a synergy. Furthermore, an enabling graph is deduced which indicates whether a technology enables or disables another technology and when two technologies are required to be applied simultaneously. This graph is constructed using the dependencies of a technology and how these resolve with another technology's dependencies.

Comparing the expert-based and the algorithm-based TCM, approximately 63% of the non-zero entries are evaluated identically, or 90% if the non-zero entries are included. The differences are mainly due to lack of physics and design reasoning in the current algorithms. Several differences indicate inconsistent judgment in the manual TCM. Therefore, the automated approach fulfills its goal of being a more structured way to assess dependencies between pairs of technologies. Thus, it reduces subjectivity, while the graph transformation approach enables a semantically meaningful technology definition, which is impossible with the common approach that uses impact factors as descriptors. Therefore, the first challenge (regarding knowledge capturing) set out in the introduction is tackled, too. Comparing the resulting technology portfolios reveals that the potential benefit increased. Moreover, the technology frequencies on the Pareto front shifted, so different technologies would be selected based on the algorithm-based TCM. Therefore, the changes due to the presented method have significant implications, in which more confidence can be put.

Future work will include the design and physics reasoning mechanisms. Furthermore, the portfolio creation process should be improved to more efficiently and selectively produce promising portfolios. Uncertainty quantification will be included to reflect input and model uncertainty.

Acknowledgments

This research is sponsored by the European Commission under the CleanSky II research program as part of the project MANTA with grant agreement number 724558. The authors would furthermore like to thank Erik Bäckström from Saab Aeronautics.

References

[1] Roelofs, M., and Vos, R., "Technology Evaluation and Uncertainty-Based Design Optimization: A Review," *2018 AIAA Aerospace Sciences Meeting*, AIAA, Kissimmee, 2018.

[2] Amadori, K., Bäckström, E., and Jouannet, C., "Selection of Future Technologies during Aircraft Conceptual Design," *55th AIAA Aerospace Sciences Meeting*, AIAA, 2017, pp. 1–11. doi:10.2514/6.2017-0233.

[3] Amadori, K., Bäckström, E., and Jouannet, C., "Future Technologies Prioritization for Aircraft Conceptual Design," *2018 AIAA Aerospace Sciences Meeting*, AIAA, Kissimmee, 2018.

[4] Roelofs, M., and Vos, R., "Formalizing Technology Descriptions for Selection During Conceptual Design," *2019 AIAA Aerospace Sciences Meeting*, AIAA, San Diego, 2019.

[5] Hirtz, J., Stone, R. B., Mcadams, D. A., Szykman, S., and Wood, K. L., "A functional basis for engineering design : Reconciling and evolving previous efforts," *Research in Engineering Design*, Vol. 13, 2002, pp. 65–82. doi:10.1007/s00163-001-0008-3.

[6] Judt, D. M., and Lawson, C. P., "Application of an automated aircraft architecture generation and analysis tool to unmanned aerial vehicle subsystem design," *Journal of Aerospace Engineering*, Vol. 229, No. 9, 2015, pp. 1690–1708. doi:10.1177/0954410014558691.

[7] Judt, D. M., and Lawson, C., "Development of an Automated Aircraft Subsystem Architecture Generation and Analysis Tool," *Engineering Computations*, Vol. 33, No. 5, 2016, pp. 1327–1352. doi:10.1108/EC-02-2014-0033.

[8] Guenov, M. D., Molina-Cristobal, A., Voloshin, V., Riaz, A., and Van Heerden, A. S. J., "Aircraft Systems Architecting – a Functional - Logical Domain Perspective," *16th AIAA Aviation Technology, Integration, and Operations Conference*, AIAA, Washington, 2016. doi:10.2514/6.2016-3143.

[9] Yuan, L., Liu, Y., Sun, Z., Cao, Y., and Qamar, A., "A hybrid approach for the automation of functional decomposition in conceptual design," *Journal of Engineering Design*, Vol. 27, No. 4-6, 2016, pp. 333–360. doi:10.1080/09544828.2016.1146237.

[10] Sen, C., Summers, J. D., and Mocko, G. M., "A protocol to formalise function verbs to support conservation-based model checking," *Journal of Engineering Design*, Vol. 22, No. 11-12, 2011, pp. 765–788. doi:10.1080/09544828.2011.603295.

[11] Sen, C., Summers, J. D., and Mocko, G. M., "Physics-Based Reasoning in Conceptual Design Using a Formal Representation of Function Structure Graphs," *Journal of Computing and Information Science in Engineering*, Vol. 13, 2013, pp. 1–12. doi:10.1115/1.4023488.

[12] Sen, C., Summers, J. D., and Mocko, G. M., "A Formal Representation of Function Structure Graphs for Physics-Based Reasoning," *Journal of Computing and Information Science in Engineering*, Vol. 13, 2013, pp. 1–13. doi:10.1115/1.4023167.

[13] Castet, J., Rozek, M., Ingham, M., Rouquette, N., and Chung, S., "Ontology and Modeling Patterns for State-Based Behavior Representation," *2018 AIAA Infotech*, AIAA, Kissimmee, 2015.

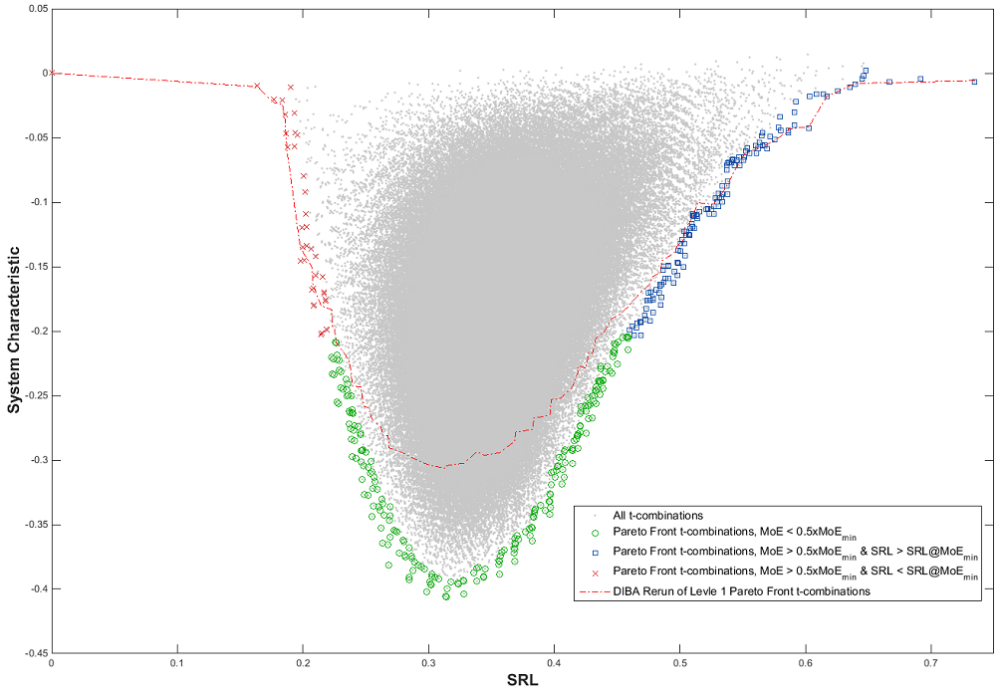
[14] Kaderka, J., Rozek, M., Arballo, J., Wagner, D., and Ingham, M., "The Behavior, Constraint and Scenario (BeCoS) Tool: A Web-Based Software Application for Modeling Behaviors and Scenarios," *2018 AIAA Aerospace Sciences Meeting*, AIAA, Kissimmee, 2018.

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

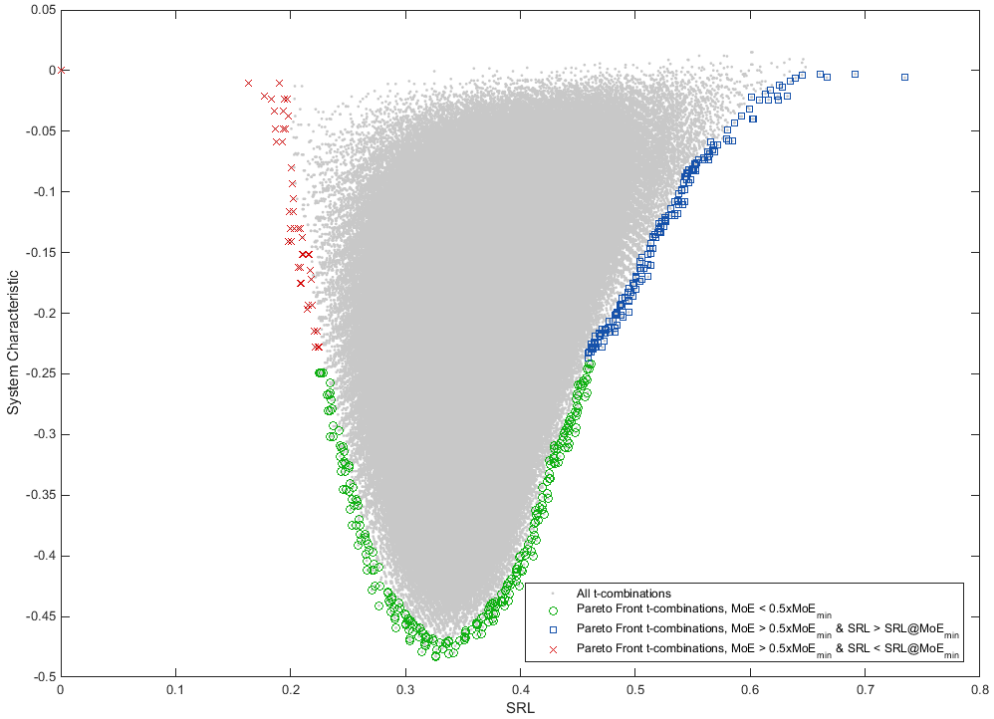
[16] Irani, M., and Rudolph, S., "Design Grammars for Conceptual Designs of Space Stations," *54th International Astronautical Congress of the International Astronautical Federation, the International Academy of Astronautics and the International Institute of Space Law*, AIAA, Bremen, 2003.

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

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

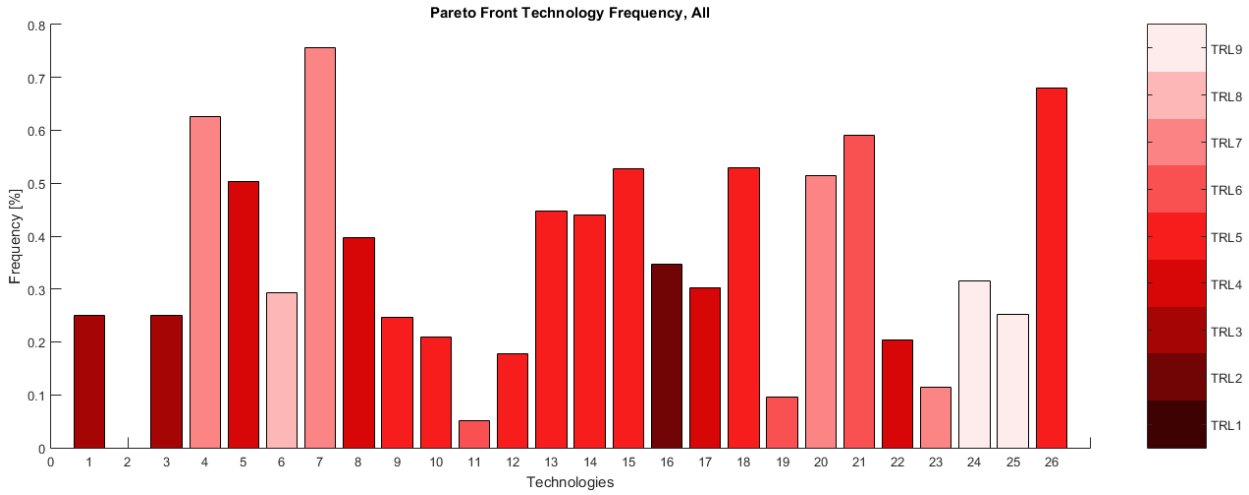


(a) Thumb plot for the expert-based portfolios [2]

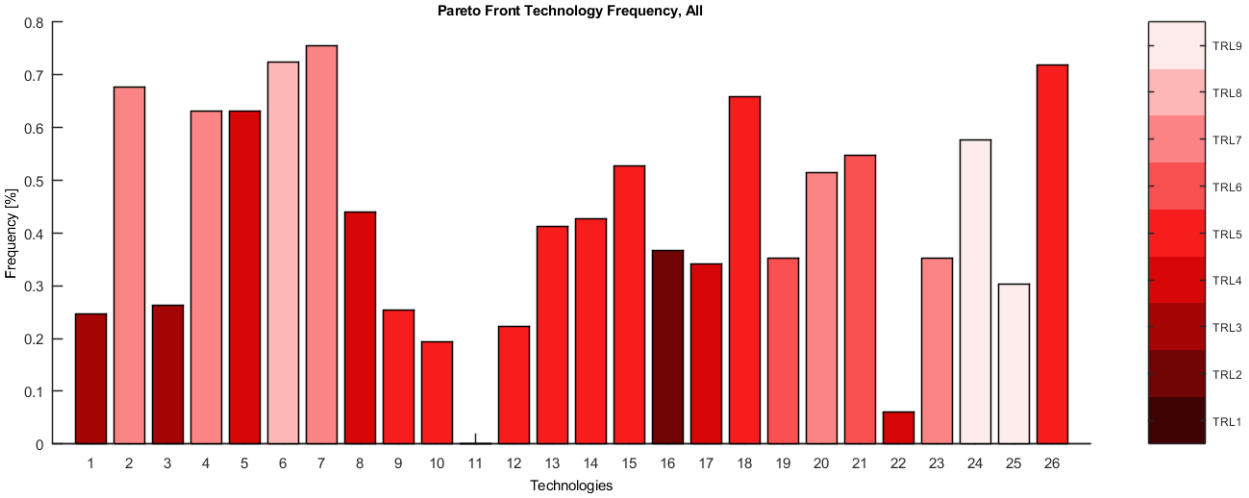


(b) Thumb plot for the algorithm-based portfolios

Fig. 8 The thumb plots for the technology portfolios resulting from (8a) the expert-based TCM and (8b) the algorithm-based TCM. Each thumb plot shows all portfolios sorted by SRL and their effect on a certain system characteristic. The colored markers indicate the Pareto front and are divided into three categories: portfolios with low SRL and low effect (red cross); portfolios with average SRL and high effect (green circle); and portfolios with high SRL and low effect (blue square).



(a) Frequency plot for the expert-based TCM [2]



(b) Frequency plot for the algorithm-based TCM

Fig. 9 Pareto front technology frequencies. The color scale indicates the TRL.