TRACING THE EMERGENCE OF DESIGN PROBLEMS AND THEIR IMPACTS ON THE COMPLEXITY OF ENGINEERING SOLUTIONS

Beernaert, Torben (1);
Etman, Pascal (2);
De Bock, Maarten (3);
Classen, Ivo (1);
De Baar, Marco (1)

1: Dutch Institute For Fundamental Energy Research (DIFFER), The Netherlands;
2: Eindhoven University of Technology, The Netherlands;
3: ITER Organization, France

ABSTRACT

The design of ITER, a large-scale nuclear fusion reactor, is intertwined with profound research and development efforts. Tough problems call for novel solutions, but the low maturity of those solutions can lead to unexpected problems. If designers keep solving such emergent problems in iterative design cycles, the complexity of the resulting design is bound to increase. Instead, we want to show designers the sources of emergent design problems, so they may be dealt with more effectively. We propose to model the interplay between multiple problems and solutions in a problem network. Each problem and solution is then connected to a dynamically changing engineering model, a graph of physical components. By analysing the problem network and the engineering model, we can (1) derive which problem has emerged from which solution and (2) compute the contribution of each design effort to the complexity of the evolving engineering model. The method is demonstrated for a sequence of problems and solutions that characterized the early design stage of an optical subsystem of ITER.

Keywords: Large-scale engineering systems, Complexity, Problem solving, Functional modelling

Contact:
Beernaert, Torben
DIFFER
PEPD
Netherlands, The
t.f.beernaert@differ.nl

Cite this article: Beernaert, T., Etman, P., De Bock, M., Classen, I., De Baar, M. (2021) 'Tracing the Emergence of Design Problems and Their Impacts on the Complexity of Engineering Solutions', in *Proceedings of the International Conference on Engineering Design (ICED21),* Gothenburg, Sweden, 16-20 August 2021. DOI:10.1017/pds.2021.584

1 INTRODUCTION

Climate change is expected to dramatically impact our lives. If we want to avoid an ecological breakdown, we urgently need clean and sustainable energy sources. Nuclear fusion is a candidate technology that could provide virtually unlimited energy. The technology is in the research and development (R&D) stage and it is unclear how large-scale energy production through nuclear fusion can be made technologically or economically viable. A global team of scientists and engineers is currently designing ITER, the next experimental nuclear fusion reactor (ITER, n.d.).

ITER's success depends on its designers' capabilities to solve problems that have never been addressed before. Critical subsystems of ITER are burdened with the challenges of measuring detailed plasma properties under extreme conditions. As there are no standard solutions, ITER highly depends on new materials and technologies through research and development.

Many of the solutions that are generated by an R&D process are still on a conceptual level; They are not reliable and functional enough to be directly employed in an industrial setting. There are often open problems or risks that still need attention. Some of these problems emerge as a direct consequence of the designed solution. R&D usually addresses such emergent problems with more R&D. Hence the notion of an R&D *cycle*. But the design evolves with every iteration in this cycle of problems and solutions. It drifts further away from the original problem and is likely to grow in complexity. This is exactly where R&D clashes with project management. Repetitive design cycles are a nightmare for budget limitations and project deadlines. The complexity of the resulting design can also be a source of integration risks.

So the question is: How can designers create a mature solution with minimum complexity in as few design iterations as possible? One elegant approach would be to reduce the amount of emergent problems by reconsidering earlier solutions. But therefore it is necessary to make designers aware of the impact of their design solutions. As a first step, we present a method that aims to support designers in effectively solving problems throughout iterative design cycles.

We propose to track the dynamic design development in a directed network of problems and solutions. This network shows which problem is solved by which solution, and which solution has given which problem. An important assumption is that problems and solutions can be related to engineering elements of the design, such as functions, components or variables. This assumption allows us to do two things: we can logically determine dependencies between problems and solutions that may have otherwise been overlooked, and we can compute the contribution of each solution to the complexity of the design.

Figure 1 explains the method, considering the simplified design of an optical diagnostic system during two consecutive iterations. Each iteration incrementally evolves the design, represented by a graph of physical components. The initial design problem p_1 is related to three components. In the first iteration, a solution s_1 is proposed to solve p_1, introducing new components to the design. But at the end of this iteration, a new problem p_2 is identified. Because p_2 is related to a component that was introduced by s_1, we say that p_2 has emerged from s_1! If s_1 would change or be removed, then so would p_2. In the second iteration, the emergent p_2 is solved by s_2, which involves the addition of yet another component. The complexity of the design, computed from the graph structure, increases throughout the design iterations.

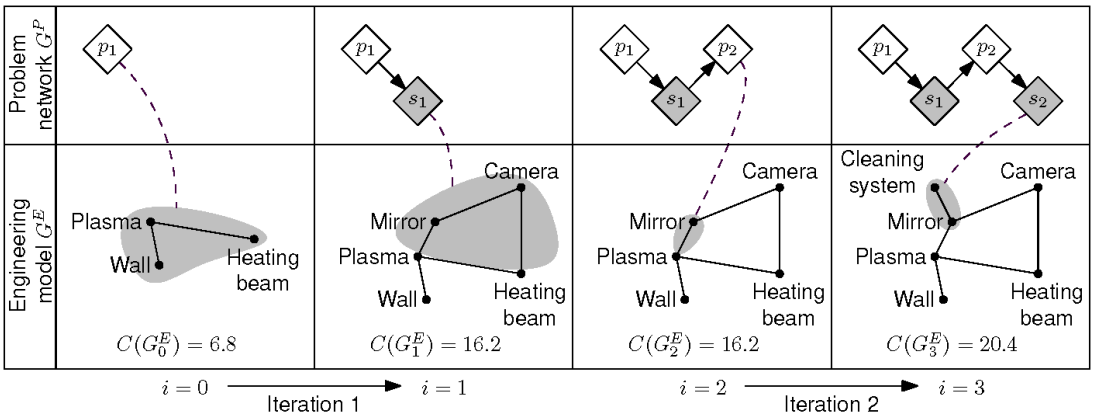

Figure 1. Four snapshots at times i of an early design effort. The top row shows causal relations between encountered problems and designed solutions in a problem network G^P. The bottom row shows how a corresponding engineering model G^E grows in complexity.

https://doi.org/10.1017/pds.2021.584 Published online by Cambridge University Press

The outline of the paper is as follows. Section 2 contains relevant literature regarding nuclear fusion engineering and engineering design. In Section 3, we introduce the concept of a problem network (the top half of Figure 1) to trace how design problems emerge from solutions. In Section 4, problems and solutions are related to an engineering model (the bottom half of Figure 1). Finally, Section 5 describes a demonstration of the method, considering the early design of the Visible Spectroscopy Reference System (VSRS), an optical diagnostic subsystem of ITER. We identify dependencies between tracked problems and solutions, and observe the growing complexity of the associated engineering model.

2 DEFINITIONS AND RELEVANT LITERATURE

Summers & Shah (2010) recognize three main concepts in engineering design: the design problem, the design process and the design product. The design problem is a statement of requirements, needs, functions or objectives. It is the input of a design process, the search for a solution. The output of the design process is the design product, but we use the term 'solution' to denote this output.

Although Summers & Shah (2010) define the design problem as a structured representation of a specific question or situation, in practice design problems are ill-defined and ill-structured (Jonassen *et al.*, 2006). Often they are even interpreted, structured and formulated by the designers themselves (Daly *et al.*, 2012). This is most notably the case in design processes that rely heavily on R&D.

The design of ITER is a perfect example. The hostile environment inside the reactor poses huge problems to the diagnostic subsystems (Feder *et al.*, 2015). A ubiquitous problem for optical diagnostics is the decrease in performance of plasma-facing mirrors suffering from this environment (Litnovsky *et al.*, 2019). Hopes are that an active cleaning system, which is currently in development (Ushakov *et al.*, 2020), can effectively mitigate this problem. But given the low maturity of this technology, it is likely that new problems emerge during its development. These would have to be addressed in subsequent iterations, extending the chain of problems and solutions. In order to converge on a final solution, the amount of emerging problems has to be lower than the amount of problems that are being solved.

Are emergent problems always bad? No, not at all! In fact, they may be helpful milestones in a complex problem solving process. Systematic approaches often consider some form of subsequent problem-solution pairs (Pahl *et al.*, 2007). See for example function-means analysis (Burge, 2006) or multi-stage decision making (Høyland & Wallace, 2001). Even the systems engineering V-model may be regarded as a translation from a complex design problem to multiple simpler ones (Forsberg *et al.*, 2005).

However, we focus on the case where problems are undesired and emerge unintentionally. New product development typically involves learning during the design process (Simpson *et al.*, 1998). Therefore, new problems could emerge because an existing problem is better understood or because of new insights about a designed solution. Any emergent problem is likely to lead to another design iteration and another solution, increasing the complexity of the overall design process and the overall solution.

Engineering design revolves around various elements, such as requirements, functions, components and variables (Pahl *et al.*, 2007). Model-Based Systems Engineering (MBSE) is an engineering methodology where these elements and their interdependencies are captured in models (Wymore, 2018). In the context of MBSE, a solution is a model of the physical realisation of a design. We consider it as an arrangement of interconnected physical components. Such a solution can be interpreted as a graph $G = (V,E)$, where vertices V are components and edges E are dependencies between them. Figure 1 displays an example. During subsequent design iterations, the graph evolves as components are added or removed.

Complexity is an attribute of a solution that is commonly approached via its graph representation. The graph of a complex solution features many components that are tightly coupled. Such a solution is generally assumed to be a significant source for unexpected costs, time delays and technical risk. To gain insight into the complexity of a solution, one could simply plot it as a network or block diagram (e.g. the bottom row in Figure 1). However larger graphs are better depicted in a matrix form, such as an adjacency matrix, N^2-chart or a Design Structure Matrix (DSM) (Eppinger & Browning, 2012). Alternatively, complexity can be computed from the mathematical properties of the graph. Sinha & de Weck (2013) propose the following formula to determine complexity C of graph G:

$$C(G) = C(G;\alpha,\beta) = \sum_{i=1}^{N} \alpha_i + \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \beta_{ij} A_{ij} \right) \frac{E(A)}{N} \tag{1}$$

https://doi.org/10.1017/pds.2021.584 Published online by Cambridge University Press

where N is the number of vertices, α_i is the internal complexity of vertex i, β_{ij} is the complexity of edge (v_i, v_j), A is the binary adjacency matrix of the graph, and $E(A)$ is the matrix energy, which can be obtained through singular value decomposition. Sinha & de Weck (2013) posit that development costs increase super-linearly with C. Raja *et al.* (2019) used Equation 1 to determine the complexity of integrated load-carrying structures in an aerospace application. For a recent overview of network-based metrics for engineering systems, we refer to Giota *et al.* (2017).

In conclusion, there seems to be much knowledge about solving constant design problems and formalizing the complexity of static engineering models. But in R&D and new product development, the design problem and the design solution are typically uncertain. Overlooking the dynamics of problem that emerge from this uncertainty will lead to a long design process resulting in a complex solution. In order to reconcile the exploratory nature of such design processes with the need to implement a timely and cost-effective solution, we set out to investigate the emergence of new design problems and how they affect the continuously evolving design solution.

3 THE EMERGENCE OF NEW DESIGN PROBLEMS

We need a way to interpret the relations between design problems and solutions through iterative cycles. We picture problems and solutions in a problem network, showing a snapshot of the design at a moment in time. New solutions and problems are added as the design evolves, creating an extended network. Figure 2 displays a problem network, the accumulated design problems and the accumulated design solutions. This figure will serve as an example for the following clarifications.

The problem network is a directed graph $G^P(\{P,S\},\{D_d,D_e,D_p,D_s\})$, where vertices represent a set of problems $P = \{p_1,p_2,...\}$ and solutions $S = \{s_1,s_2,...\}$. Edges represent four possible kinds of relations between them. We distinguish design dependencies D_d $(p \rightarrow s)$, emergence dependencies D_e $(s \rightarrow p)$, problem dependencies D_p $(p \cdots p)$ and solution dependencies D_s $(s \cdots s)$. The descendants $\mathbb{D}(v)$ of vertex v are defined as the collection of all vertices that can be reached by following directed edges starting from v. In the shown network, we find $\mathbb{D}(s_1) = \{p_2,p_4,s_3\}$ and $\mathbb{D}(p_3) = \{p_4,s_3,s_4\}$.

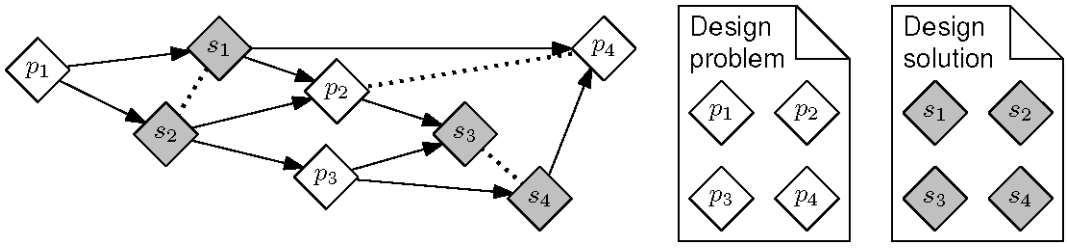

Figure 2. Problems p and solutions s are connected in a problem network. Problems p_2, p_3 and p_4 have emerged from solutions that were designed to solve p_1.

Design and emergence dependencies signify cause-effect relationships: The vertex at an arrow's head exists as a consequence of the vertex at its tail. For example, problem p_1 has led to solution s_1 by design, and solution s_1 has led to problem p_2 by emergence. A problem can emerge from a combination of multiple solutions, such as problems p_2 and p_4. Similarly, a solution can contribute to solving multiple problems, such as s_3. A problem may be solved by multiple solutions at the same time. If they could not exist independently, they should be regarded as a single solution. For example, we could reduce contamination on an optical mirror by adding both a protective shutter and an active cleaning system.

Problem and solution dependencies signify general overlaps between the scopes of problems or solutions. These overlaps could be beneficial or troublesome, but designers should definitely take them into account. Two problems are dependent if they are contradicting, where any solution that mitigates one of the problems is likely to worsen the other. A problem dependency may also indicate alignment between problems when they are potentially solved with a single solution. Two solutions are dependent if they could or need to be integrated together. In some cases it may be less costly to develop two solutions together rather than independently, but in other cases this leads to new challenges.

Figure 2 shows that a designer has attempted to solve problem p_1 via two solutions s_1 and s_2. However, it has become clear that problems p_2, p_3 and p_4 have emerged from these solutions. So s_3 and s_4 have been implemented to deal with the emerged problems. Let it be clear that s_3 and s_4 do not contribute to the

https://doi.org/10.1017/pds.2021.584 Published online by Cambridge University Press

original problem p_1 directly. They merely solve the problems that emerged from s_1 and s_2. No solution has yet been designed for problem p_4, so the design in Figure 2 may need another design iteration.
The designer could now solve p_4 with a new design effort. But the graph shows that it may be worth reconsidering to change the design of s_1 or s_4 in such a way that p_4 is suppressed or simplified. If it would even be possible to remove s_1 completely, all its descendants would disappear too. Then the remaining problems would reduce to $P = \{p_1, p_3\}$ and the necessary solution would reduce to $S = \{s_2, s_3, s_4\}$.

3.1 Problem severity and solution effectiveness

The definition of the problem network as a graph allows us to introduce weights to its vertices and edges. Problems and solutions are attributed a 'severity'-weight and an 'effectiveness'-weight, respectively. The severity of problem p is denoted as $|p|$ and can be defined upon its identification. It should be the objective of any connected solution to minimize $|p|$. The effectiveness of solution s is denoted as $|s|$ and is defined by how well it reduces the severity of connected problems.
Consider the optical mirror that is contaminated by isotopes emitted from a fusion plasma —problem p_2 in Figure 1. The severity of this problem may be defined as the build-up of the contamination layer over the lifetime of the mirror. An initial estimate could give $|p_2| = 50$ nm. If s_2 is an active cleaning system that can remove 30 nm of build-up, then $|s_2| = 30$ nm and $|p_2| = 20$ nm. If we would add a protective shutter s_3 that can prevent 10 nm of build-up, $|p_2|$ will further reduce to 10 nm.
Problem p_2 is solved for 60% by s_2 and for 20% by s_3. These percentages can be interpreted as weights of the design dependencies, indicating how heavily a problem relies on a specific solution. In this case, $w_{(p_2,s_2)} = 0.6$ and $w_{(p_2,s_3)} = 0.2$. The designer now has to decide whether to make the existing solutions more effective, to introduce new solutions or to accept the remaining problem severity $|p_2| = 10$ nm.
For now we consider only linear problems with a scalar severity, meaning that the effectiveness of a solution is independent of any other solution. However in practice, problems and solutions can be coupled in a more complex manner. Consider the situation where it becomes progressively more difficult to remove material from the mirror surface. Then it matters in which order solutions are applied and it becomes difficult to isolate each contribution. We advise to normalize the severity of problems in any situation, so that solutions can be easily compared.
Finally, a solution may also contribute to the mitigation of multiple problems. In this case, we sum the contribution to each problem, acquiring solution effectiveness as:

$$|s| = \sum w_{(p_i,s)} |p_i| \quad \forall\, p_i \mid (p_i, s) \in D_d \tag{2}$$

where $w_{(p_i,s)}$ is the weight of a design dependency between s to p_i.

4 THE PROBLEM NETWORK AND THE ENGINEERING MODEL

In the previous sections we have introduced a way to visualize problem emergence. Now we are in need of a mechanism that explains which problem has emerged from which solution. In this section we propose such a mechanism via the interplay between two dynamically evolving graphs: the problem network and the engineering model, depicted in the top and bottom half of Figure 1 respectively. This interplay is exploited to identify dependencies in the problem network and to compute the impact of each solution on the complexity of the engineering model.
Problems and solutions are identified and implemented at discrete steps i in time, corresponding to columns in Figure 1. Hence, we denote the time-dependent graphs of the problem network and the engineering model as G_i^P and G_i^E, respectively. At time i, problems $P_i = \{p_{i1}, p_{i2}, p_{i3}, \dots\}$ are identified and solutions $S_i = \{s_{i1}, s_{i2}, s_{i3}, \dots\}$ are implemented.
We assume that problems and solutions can be represented by a set of physical components and interactions. That is, each problem p_{ij} and solution s_{ij} is related to a subgraph of G_i^E, denoted by $G_{p_{ij}}$ and $G_{s_{ij}}$. For example, the contaminated mirror problem p_2 relates to $G_{p_2} = (\{v_1, v_2\}, (v_1, v_2))$, where v_1 is the plasma, v_2 is the mirror and (v_1, v_2) is the particle flux between them. The cleaning solution s_2 relates to $G_{s_2} = (\{v_2, v_3\}, (v_2, v_3))$, where v_3 is the cleaning system and (v_2, v_3) is the electrical power between cleaning system and mirror.

https://doi.org/10.1017/pds.2021.584 Published online by Cambridge University Press

Problems can only relate to components of the engineering model graph that already existed at the previous time step, but solutions can introduce new components to the engineering model. We denote the subset of $G_{s_{ij}}$ that is new in the model as $G^+_{s_{ij}}$. In our example $G^+_{s_2} = (\{v_3\},(v_2,v_3))$.
The initial engineering model is given by the union of the root problems: $G^E_0 = \bigcup\{G_{p_{01}}, G_{p_{02}}, G_{p_{03}}, ...\}$. As new components are introduced by solutions, the graph of the engineering model grows according to $G^E_{i+1} = \bigcup\{G^E_i, G_{s_{i1}}, G_{s_{i2}}, G_{s_{i3}}, \dots\}$.

4.1 Deriving dependencies

Design dependencies D_d are usually well known by designers. That is, they know which solution is designed to mitigate which problem. But it can be more challenging to see where a specific problem comes from (emergence dependency $d_e(s,p)$), which solutions should be designed together (solution dependency $d_s(s_i,s_j)$) and which problems could contain overlaps or contradictions (problem dependency $d_p(p_i,p_j)$). For given sets of identified problems $\{P_0,P_1,P_2,\dots\}$ and implemented solutions $\{S_0,S_1,S_2,\dots\}$, we propose to logically derive D_e, D_s and D_p. We do this by analysing the subgraphs of problems and solutions in the engineering model domain.
Firstly, we derive emergence dependencies by the following logic:

$$D_e = \left\{(s,p) \quad \forall\, s,p \quad \mid G^+_s \cap G_p \neq \emptyset\right\} \tag{3}$$

A problem p_{ij} emerges from a solution s_{ij} if s_{ij} has introduced a new component that is a part of the representation of p_{ij}. Referring to our contaminated mirror example; If we identify a new problem p_3 with the cleaning system ($v_3 \in G_{p_3}$), Equation 3 will identify emergence dependency $d_e(s_2,p_3)$ because the cleaning system was introduced by s_2 and therefore $G^+_{s_2} \cap G_{p_3} = \{v_3\}$.
With these emergence dependencies known, it is possible to construct the causal structure of the problem network and to define the descendants of each problem and solution. This definition is necessary to identify problem and solution dependencies as:

$$D_p = \left\{(p_1,p_2) \quad \forall\, p_1,p_2 \neq p_1 \;\middle|\; \begin{array}{l} G_{p_1} \cap G_{p_2} \neq \emptyset \\ p_1 \notin \mathbb{D}(p_2) \wedge p_2 \notin \mathbb{D}(p_1) \end{array}\right\} \tag{4}$$

$$D_s = \left\{(s_1,s_2) \quad \forall\, s_1,s_2 \neq s_1 \;\middle|\; \begin{array}{l} G_{s_1} \cap G_{s_2} \neq \emptyset \\ s_1 \notin \mathbb{D}(s_2) \wedge s_2 \notin \mathbb{D}(s_1) \end{array}\right\} \tag{5}$$

There exists a dependency between any two solutions or problems in the problem network if (1) their definitions overlap in the engineering model, i.e. if they share the same component or function, and if (2) they did not indirectly cause each other. This logic only identifies the existence of a relation. Its weight and consequences would have to be determined differently.

4.2 Complexity impact

Implemented solutions always introduce new components to an engineering model. These new components and their new interactions contribute to the complexity of the overall design. As such, solutions have a direct impact on the complexity of the graph representation of the engineering model. We define the local and global impact of solution s_i that is implemented in G^E_i:

$$\begin{aligned} \Delta C_L(s_i) &= C\left(\bigcup\{G^E_i, G^+_{s_i}\}\right) - C\left(G^E_i\right) \\ \Delta C_G(s_i) &= C\left(\bigcup\{G^E_i, G_{s_i}, \quad G_{s_j}\ \forall\, s_j \in \mathbb{D}(s_i)\}\right) - C\left(G^E_i\right) \end{aligned} \tag{6}$$

Where $C(G^E)$ is the complexity of graph G^E, see Equation 1. The local complexity impact is equal to the difference between the complexity of the graph before and after implementation of s_i, and reflects the increase due to any new components. However at a later stage, unanticipated problems that have emerged from s_i will have to be solved by the additional solutions in $\mathbb{D}(s_i)$. The additional effect of these solutions is accounted for in the global complexity impact. In Figure 1, $\Delta C_L(s_1) = C(G^E_1) - C(G^E_0) = 9.4$ and $\Delta C_G(s_1) = C(G^E_3) - C(G^E_0) = 13.6$.

https://doi.org/10.1017/pds.2021.584 Published online by Cambridge University Press

5 DEMONSTRATION FOR ITER

We have introduced the concepts of a problem network and how it can be related to components in an engineering model. Now, we demonstrate the application of the presented method on the early architectural design of the VSRS, where a functional model is updated while design decisions are made. The functional model is specified using the Elephant Specification Language (Wilschut *et al.*, 2017), and comprises a rich structure of components, functions, requirements and variables over multiple layers of decomposition. However, in this demonstration we reduce the model to a simplified set of components and the interfaces between them on a single layer of decomposition.

Over the course of several weeks of design, design problems were identified and preliminary solutions were implemented. Table 1 shows these problems and solutions as they appeared over time. Column 4 shows the normalized severity of an identified problem $|p|$ or the normalized contribution $w_{(p,s)}$ of a solution to a problem. Design dependencies, which solutions contribute to which problems, are implicitly specified by $w_{(p,s)}$. Columns 5 and 6 contain the definition of the problem or solution in terms of components (graph vertices) and interfaces (graph edges) of an engineering model. The name of each component can be read from Figure 3.

Table 1. The input information for the demonstrated method: identified problems p and implemented solutions s in the early architectural design of the VSRS.

i	Object	Description	$\|p\|$ or $w_{(p,s)}$	Components	Interfaces
0	p_1	The heating beam may damage the wall if the plasma becomes too thin.	$\|p_1\| = 3$	v_1, v_2, v_3	(v_1, v_2), (v_2, v_3)
1	s_1	Measure plasma properties via visible spectroscopy. Shut down heating beam if shinethrough detected.	$w_{(p_1,s_1)} = 0.9$	v_1, v_4, v_5, v_6, v_7, v_8	(v_2, v_4), (v_3, v_6), (v_4, v_5), (v_5, v_7), (v_6, v_4), (v_7, v_8), (v_8, v_1)
2	p_2	Plasma contaminates the mirror.	$\|p_2\| = 2$	v_4	(v_2, v_4)
3	s_2	Protect mirror with shutter.	$w_{(p_2,s_2)} = 0.3$	v_9	(v_2, v_9), (v_9, v_4)
4	p_3	Optical properties of instruments may change, reducing accuracy.	$\|p_3\| = 1$	v_4, v_5	(v_4, v_5), (v_5, v_7)
5	s_3	Use a calibrated light source to regularly determine optical properties.	$w_{(p_2,s_3)} = 0.1$, $w_{(p_3,s_3)} = 0.9$	v_9, v_{10}	(v_9, v_4), (v_4, v_{10}), (v_4, v_5), (v_5, v_7)
6	p_4	Thermal expansion of vacuum wall causes optical misalignment.	$\|p_4\| = 1$	v_3, v_6, v_4	(v_3, v_6), (v_6, v_4)
7	s_4	Measure misalignment and move mirror accordingly.	$w_{(p_4,s_4)} = 1$	v_{11}, v_{12}, v_6	(v_{11}, v_{12}), (v_{12}, v_4)
8	s_5	Add an active cleaning system.	$w_{(p_2,s_5)} = 0.5$	v_{11}, v_{13}	(v_{11}, v_{13}), (v_{13}, v_4)

Problem p_1 is the original design problem that cannot be influenced. But is this also true for problems p_2, p_3 and p_4? And how did each designed solution contribute to the overall complexity of the engineering model? The information in Table 1 will be analysed according to the method in order to provide answers. We present the dependencies between problems, solutions and the engineering model in a Multi-Domain Matrix (MDM), Figure 3. By convention, an off-diagonal entry in the matrix indicates that row element y depends on column element x, i.e. that there is an edge $x \rightarrow y$ in the equivalent graph.

The top left matrix represents the engineering model as a conventional component DSM. This matrix can be compiled from columns 5 and 6 in Table 1 without additional analysis. As five subsequent solutions were implemented, the model has expanded from three to thirteen components. We emphasized the sections of this DSM to show the graphs $\{G_0^E, ..., G_8^E\}$ at different timesteps.

5.1 Problem severity and solution effectiveness

Column 4 in Table 1 quantifies the severity of each problem and the contribution of each solution to mitigating a problem. From Equation 2, we can compute the effectiveness of each solution. The initial

https://doi.org/10.1017/pds.2021.584 Published online by Cambridge University Press

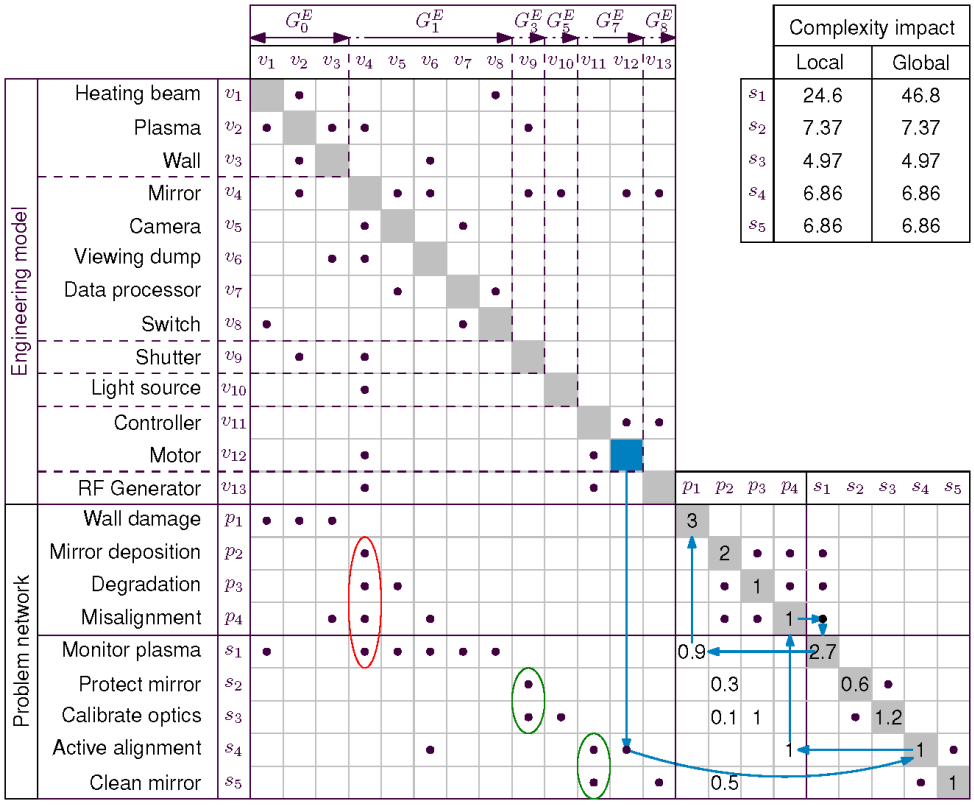

Figure 3. A Multi-Domain Matrix displays the engineering model, the problem network and the interactions between them. The red circle highlights the source for emergence and problem dependencies. The green circles signify the source of two solution dependencies.

problem severity and the computed solution effectiveness are printed on the diagonal of the problem network DSM. One can easily identify how well each problem is addressed by going through its design dependencies in the $P \rightarrow S$ matrix. For example, p_2 is mitigated by s_2, s_3 and s_5 for a total of 90%.

5.2 Deriving dependencies

The bottom left matrix shows the mapping between the components in the engineering model (column 5) and the problems and solutions in the problem network (column 2). Here, Equation 3 finds three emergence dependencies. Problems p_2, p_3 and p_4 all involve the mirror v_4. Because this component was introduced by s_1, we say that p_2, p_3 and p_4 emerged from s_1. If the design of the mirror in s_1 would change, then these problems would be affected too. The emergence dependencies are indicated in the $S \rightarrow P$ matrix. When read per row, this matrix shows us the sources of a specific problem. When read per column, it shows us the consequences of a specific solution.

Because p_2, p_3 and p_4 are all related to the mirror, Equation 4 tells us that these problems are related by an entry in the $P \rightarrow P$ matrix. We also see that s_2 and s_3 both make use of the shutter, and that s_4 and s_5 both depend on the controller. Equation 5 identifies the entries in the $S \rightarrow S$ matrix.

5.3 Complexity impact

All solutions introduce new components to the model and increase its complexity in a sequence of graphs $G_0^E, \ldots, G_8^E$. Figure 3 shows a table with the local and global complexity impact of each solution. These values are computed from Equations 1 and 6, where $\alpha = 1$ and $\beta = 1$.

The initial engineering model contains three components, leading to a complexity of $C(G_0^E) = 6.8$. Solution s_1 has added five new components, expanding the engineering model to G_1^E with $C(G_1^E) = 31.4$. The local complexity impact of s_1 is therefore $31.4 - 6.8 = 24.6$.

The global complexity impact of s takes into account the impact of follow-up solutions that deal with problems that emerged from s. Therefore, it can only be computed after emergence dependencies are derived. From the $S \rightarrow P$ matrix, we can see that p_2, p_3 and p_4 have emerged from s_1. These problems have led to solutions s_2 to s_5. The components that are introduced by these solutions are considered in

https://doi.org/10.1017/pds.2021.584 Published online by Cambridge University Press

the global complexity impact of s_1. In this case, we use Equation 6 to calculate $\Delta C_G(s_1) = C(G_8^E) - C(G_0^E) = 46.8$. So, at $i = 8$ it turned out that s_1 is responsible for more complexity than thought initially. Figure 4 shows the complexity of the engineering model over time, mapped to a spread out problem network. Note that this view does not contain any new data respective to Figure 3, but merely presents it in a different way. To some it may be easier to trace dependencies in this graph, although the MDM is more compact and can be better scaled. We can easily see the difference between the local and global impact on complexity due to s_1. No problems have emerged from s_2, and therefore $\Delta C_G(s_2) = \Delta C_L(s_2)$.

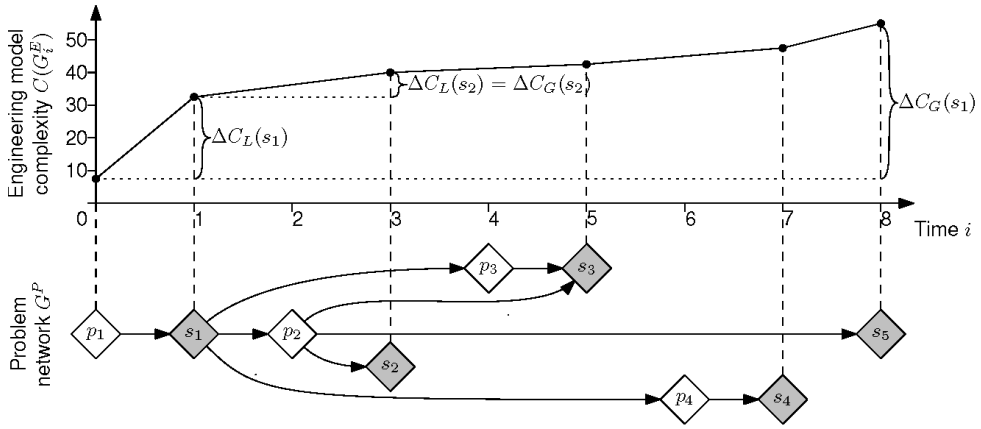

Figure 4. The complexity of the engineering model grows (top) as it is impacted by solutions s_i in the problem network (bottom). At $i = 8$, we learn that s_1 has more impact on complexity than initially expected: $\Delta C_G(s_1) > \Delta C_L(s_1)$.

5.4 Risk analysis

The MDM does not only show the needs and consequences for specific design solutions. It can also be used as a risk analysis tool, as it contains paths that tell us which problems may occur when a certain component fails during operation. If we are interested in the consequences of a motor failure, we follow the blue arrows starting at v_{12}. The motor is a part of the active alignment system solution s_4. A failure of this solution leads to misalignment of the VSRS p_4. This means that the VSRS cannot properly monitor the plasma s_1 any more and that the wall of ITER may be damaged p_1. As such, we have established a causal chain from the motor component to the integrity of the wall.

We have demonstrated the proposed method in the context of the early design of the VSRS. Therefore, all problems and solutions were within the scope of a single design team. But we expect that the phenomenon of problem emergence plays a greater role as the scale of the design effort increases. After all, the VSRS in itself is simply one intermediate solution in the enormous problem network of ITER, spanning a design process of multiple decades and thousands of design and research teams.

6 CONCLUSION

Designers are problem solvers. But due to the conceptual level of solutions in research and development processes, new problems may emerge unintentionally. Since it is difficult to track which problem emerged from which solution, the straightforward response is to solve emergent problems in subsequent design iterations. However, these may incur even more problems and more design features.
This way of problem solving leads to numerous design iterations. The resulting design, an arrangement of interconnected physical components, is bound to be overly complex. We believe that a profound understanding of the origins of emergent problems will lead to better designs in fewer design iterations. We present a method to investigate the interplay between problems and solutions through iterative design cycles. A network of problems and solutions shows us that instead of solving an emergent problem, it may be suppressed by changing an earlier solution.
Problems and solutions in this network are related to the components of a growing engineering model. This link allows us to derive which problem has emerged from which solution and to compute the contribution of each solution to the complexity of the design.

https://doi.org/10.1017/pds.2021.584 Published online by Cambridge University Press

The method is demonstrated for the early design iterations of an optical diagnostic system in plasma fusion applications. The input for our method are problems and solutions as they were identified and implemented over time, each described in terms of physical components. We could derive which problem has emerged from which solution and observe the evolution of an engineering model. Finally, the method showed the contribution of each solution to the complexity of the engineering model.
We encourage designers to be aware of the problems and solutions in their design, by giving them the tools that point at the *right* problems. Sometimes, avoiding a problem is better than solving it.

DISCLAIMER

The views and opinions expressed herein do not necessarily reflect those of the ITER Organization.

REFERENCES

Burge, S. 2006. Function Means Analysis (FMA): Alias Morphological Analysis. In: *The Systems Engineering Tool Box*.

Daly, S. R., Adams, R. S., & Bodner, G. M. 2012. What Does it Mean to Design? A Qualitative Investigation of Design Professionals' Experiences. *Journal of Engineering Education*, **101**(2), 187–219.

Eppinger, S. D., & Browning, T. R. 2012. *Design Structure Matrix Methods and Applications*. 1 edn. MIT Press.

Feder, R., Zhai, Y., Johnson, D., Zolfaghari, A., Wood, R., Reichle, R., DeBok, M., Graves, V., Klepper, C., Biewer, T., Rowan, B., & Phillips, P. 2015. Engineering challenges for ITER diagnostic systems. *Pages 1–7 of: 2015 IEEE 26th Symposium on Fusion Engineering (SOFE)*. Austin, TX, USA: IEEE.

Forsberg, K., Mooz, H., & Cotterman, H. 2005. *Visualizing Project Management*. 3 edn. New York, NY: John Wiley & Sons.

Giota, P., Alex, D., Caroline, V., & Malcolm, R. 2017. System Architectures Assessment Based On Network Metrics. 11.

Høyland, K., & Wallace, S. 2001. Generating Scenario Trees for Multistage Decision Problems. *Management Science*, **47**(2), 295–307.

ITER. *ITER - the way to new energy*. http://www.iter.org.

Jonassen, D., Strobel, J., & Lee, C. B. 2006. Everyday Problem Solving in Engineering: Lessons for Engineering Educators. *Journal of Engineering Education*, **95**(2), 139–151.

Litnovsky, A., Voitsenya, V.S., Reichle, R., Walsh, M., Razdobarin, A., Dmitriev, A., Babinov, N., Marot, L., Moser, L., Yan, R., Rubel, M., Widdowson, A., Moon, S., Oh, S.G., An, Y., Shigin, P., Orlovskiy, I., Vukolov, K.Yu., Andreenko, E., Krimmer, A., Kotov, V., Mertens, Ph., & Specialists Working Group on First Mirrors of the ITPA Topical Group on Diagnostics. 2019. Diagnostic mirrors for ITER: research in the frame of International Tokamak Physics Activity. Nuclear Fusion, **59**(6), 066029.

Pahl, G., Beitz, W., Feldhusen, J., & Grote, K. H. 2007. *Engineering design: a systematic approach*. 3 edn. London: Springer.

Raja, V., Kokkolaras, M., & Isaksson, O. 2019. A simulation-assisted complexity metric for design optimization of integrated architecture aero-engine structures. *Structural and Multidisciplinary Optimization*, **60**(1), 287–300.

Simpson, T. W., Rosen, D., Allen, J. K., & Mistree, F. 1998. Metrics for Assessing Design Freedom and Information Certainty in the Early Stages of Design. *Journal of Mechanical Design*, 120(4), 628-635.

Sinha, K., & de Weck, O. L. 2013. A network-based structural complexity metric for engineered complex systems. *Pages 426–430 of: 2013 IEEE International Systems Conference (SysCon)*. Orlando, FL: IEEE.

Summers, J. D., & Shah, J. J. 2010. Mechanical Engineering Design Complexity Metrics: Size, Coupling, and Solvability. *Journal of Mechanical Design*, **132**(2), 021004.

Ushakov, A., Verlaan, A., Stephan, U., Steinke, O., de Bock, M., Maniscalco, M. P., & Verhoeff, P. 2020. ITER visible spectroscopy reference system first mirror plasma cleaning in radio-frequency gas discharge – circuit design and plasma effects. *Fusion Engineering and Design*, **154**(May), 111546.

Wilschut, T., Etman, L. F. P., Rooda, J. E., & Vogel, J. A. 2017. Multi-level function specification and architecture analysis using ESL: A lock renovation pilot study. In: *Proceedings of the ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*.

Wymore, A. W. 2018. *Model-based systems engineering*. Vol. 3. CRC press.

https://doi.org/10.1017/pds.2021.584 Published online by Cambridge University Press