# Compositional Assessment of Structural Complexity

Mario Gleirscher*
ENSTA, IP Paris
Paris, France

Christian Colombo
University of Malta
Valetta, Malta

Clare Dixon
University of Manchester
Manchester, United Kingdom

Taylor T. Johnson
Vanderbilt University
Nashville, Tennessee

November 17, 2026

**Abstract**

*Objective:* A fundamental task in design decision making is the assessment of a system's complexity by aggregating corresponding assessments of its parts and by putting system design variants in relation. Complexity can be a proxy for difficulty, opacity, effort, or risk. *Approach:* We use the notion of a *configuration* to model a system's architecture (e.g., its components, connections, properties) in a lean yet versatile and compositional way. We encode configurations as (undirected, weighted, labelled, hierarchical) hypergraphs, their quantitative assessment as a multilinear operator over the (flattened) incidence matrices of the hypergraphs, and relations over configurations via (partial) orderings. *Evaluation:* In an accompanying triple-case analysis [8], we apply this approach to the assessment of *hardness* within the domain of *autonomous systems*. We examine how hardness estimates can be composed from hardness arising from component coupling, an application context, and a system lifecycle. *Contribution:* We propose a concise and flexible method for structural complexity assessment and comparison of system variants.

## Contents

**1 Introduction** **2**

**2 Configurations** **4**

*Corresponding author: mario.gleirscher@ensta.fr

**3 Assessing the Complexity of Configurations** **5**
3.1 Complexity Assessment . . . . . 5
3.2 Attributing Complexity to a Configuration . . . . . 6
3.3 Compositionality . . . . . 8

**4 Ordering Configurations by Structural Complexity** **12**
4.1 Scalar Orders . . . . . 12
4.2 Lexicographical Orders . . . . . 14

**5 Example: Hardness of Autonomous Systems** **14**
5.1 Representing Autonomous Systems as Configurations . . . . . 15
5.2 Structuring Hardness and Integrating its Measurement . . . . . 15
5.3 Comparing and Ordering Systems by Hardness . . . . . 18

**6 Discussion** **19**

**7 Conclusion** **20**

# 1 Introduction

A fundamental task in the decision making for designing a complex system is the *assessment* of its *complexity* by aggregating the corresponding assessments of its parts and by putting comparable systems in relation [10]. This task can recur frequently in the system lifecycle [11]. In systems engineering, complexity is typically treated as a proxy for characteristics, such as difficulty, opacity, effort, power, or risk [12, 24]. Unlike the precise notions of complexity in information theory (e.g., variations of Kolmogorov complexity) and programming (e.g., algorithmic complexity), complexity in *systems engineering* [21, Tab. 3-2] refers to a variety of more or less well measurable phenomena. The latter can include the (structural) size or density of a system [1, 18, p. 7], the difficulty of designing it [3, pp. 183, 211], the lack of insight into its (emergent) properties [35, p. 15], the effort required (i.e., the hardness) to build it [24], the multitude of resources needed to operate it [18, 10, 12], the range of its capabilities [26], the risk of its failure or an underlying lack of controllability [31]. The corresponding measurements can be vague, heterogeneous estimates [14] taken at specific moments, for specific technologies, and for specific elements of a design space. *System models* can be used to describe, assess, and compare such elements.

**Research Hypothesis and Questions (RQs).** Following [10], we assume system complexity to be measurable (i.e., to be an element of a metric space). **RQ1**: How can we co-allocate multiple flavours of system complexity to the elements of a system and compute a consistent overall score using simple (linear) operations? **RQ2**: How can we compare systems and order them by complexity?

**Approach and Application.** We define the notion of a *configuration* to capture a system's architecture (i.e., its components, connections, and properties) in a lean yet versatile and compositional way. Configurations can be obtained as projections of (hierarchical) system models (e.g., [6, 9, 5, 7]). We encode configurations as (undirected, weighted, labelled, hierarchical) hypergraphs. Their quantitative assessment is facilitated via a multilinear operator over the (flattened) incidence matrices of the hypergraphs, and the comparison of configurations via (partial) orderings. In a previous triple-case analysis [8], we derive, evaluate, and illustrate our approach to perform assessments of *hardness* within the domain of *autonomous systems*. In this context, we assess how influences on hardness from component coupling, the application context (e.g., goals derived from a specific technology readiness level, TRL), and the engineering lifecycle (e.g., compliance to a safety standard) can be decomposed using our method.

**Background and Related Work.** A wide variety of research on notions of *complexity* in systems engineering has focused on systematic or modular decomposition [28] and coupling effects [15] as well as on the assessment (i.e., estimation) of engineering effort [23, 34], technical debt [4], artefact quality [13], risk (hazard, fault) proneness [31, 9], structural (e.g., architectural) complexity [21, 2, 17, 37], and engineering complexity [10, 12]. Another large branch of complex systems engineering research deals with design space exploration [25], comparing configurations regarding their power [26], and evolving the latter [36]. Recent surveys [14, 12, 20] conclude there to be no consensus on a common notion of complexity in systems engineering. Nevertheless, they identified research gaps in how to determine, estimate, and compose system complexity, while remaining agnostic of its interpretation.

**Contributions.** We propose a concise and flexible method to attributing complexity to the elements of a system, and performing a quantitative assessment and comparison of system designs. This companion paper adds a technical discussion to our triple-case analysis published in [8]. In particular, we provide proofs of basic properties of our method as well as a compositionality result allowing complexity-preserving model changes. Moreover, our use of hypergraphs and orderings complements and extends similar approaches (e.g., [2]) with more flexibility in complexity attribution, while remaining close to conventional (architectural) abstractions used in engineering management (e.g., design structure matrices for workflow scheduling [33, 11]).

**Overview.** We introduce the notion of a configuration (Section 2) for structuring a system, continue with attributing complexity to the elements of a configuration (Section 3), and examine ways to compare distinct systems by their complexity (Section 4). We apply this method to the assessment of hardness in the domain of autonomous systems (Section 5), examine how hardness estimates can be composed (Sections 5.1 and 5.2) from hardness estimates derived

from component coupling, an application context, and a system lifecycle. We add guidance on using our approach (Sections 5.3 and 6).

**Notation.** $2^X$ signifies the powerset of a set $X$ and $|X|$ its cardinality. For an indexed set $X = \{x_1, \ldots, x_i, \ldots, x_n\} = \{x_i\}_{i=1}^n$ and an $n$-tuple $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_n) = (x_i)_{i=1}^n$, we define $X_{-i} = \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}$ and $\mathbf{x}_{-i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$. $|x|$ is the absolute-value norm of scalar $x$; $\|\mathbf{v}\|_1 = \sum_i v_i$, $\|\mathbf{v}\|_2 = \sqrt{\mathbf{v}^{\mathrm{T}}\mathbf{v}}$, and $\|\mathbf{x}\|_\infty = \max\{|x_i|\}_{i=1}^n$ are the sum, Euclidean, and max norms of vector $\mathbf{x} \in \mathbb{R}^n$, respectively. Given an $n \times m$-matrix $\mathbf{A}$ and $i \in 1..n, j \in 1..m$, $\mathbf{A}_{i*}$ and $\mathbf{A}_{*j}$ denote its $i$th row and $j$th column. $\mathbf{A}_{-i*}$ and $\mathbf{A}_{*-j}$ are the matrices resulting from removing $\mathbf{A}$'s $i$th row and $j$th column. We lift an element-wise map $f$ to a map on sets and vectors via $f(X) = \{f(x_1), \ldots, f(x_n)\}$ and $f(\mathbf{x}) = (f(x_1), \ldots, f(x_n))^{\mathrm{T}}$. $\mathbf{I}_n$, $\mathbf{1}_m$, and $\mathbf{0}_{n,m}$ denote the $n$-dimensional unit matrix, the $m$-dimensional unit vector, and the $n \times m$-matrix with zeroes, respectively. Finally, $\equiv$ indicates a logical equivalence or bi-implication.

## 2 Configurations

Configurations are the central compositional building block of our framework. To allow a flexible interpretation of the elements of a configuration in the context of attributing and quantifying system complexity, we decided to work with weighted, undirected hypergraphs [27].

**Definition 1 (Configuration)** Let $L$ be the universe of labels. A *configuration* $c$ is a labelled, edge-weighted, and undirected hypergraph $c = (C, E, w)$ with $n$ vertices $C = \{c_i\}_{i=1}^n$, $m$ labelled hyperedges $E = \{e_i\}_{i=1}^m \subseteq L \times 2^C$, and a map $w\colon E \to \mathbb{R}$ assigning weights to the edges in $E$. □

An undirected hypergraph with only bilateral edges can be translated and extended into a labelled directed graph with $E' \subseteq C \times L \times C$ containing $E$ such that $(l, \{c, c'\}) \in E \implies (c, l, c'), (c', l, c) \in E'$.

**Definition 2 (Sub-configuration)** Given two configurations $c$ and $c'$, the element $c = (C, E, w)$ is called a *sub-configuration* of $c'$, written $c \subseteq c'$, iff $C \subseteq C'$, $E \subseteq E'$ with $E \subseteq L \times 2^C$, and $w = w'|_E$. $c$ is a *strict* sub-configuration of $c'$, written $c \subset c'$, iff $c \subseteq c'$ and $C \subset C'$ or $E \subset E'$. □

Each element $c_i \in C$ induces a sub-configuration $(\{c_i\}, \emptyset, \emptyset)$ of $c$. $c_i$ is a configuration itself. Hence, $c$ has a tree structure. $c$ is said to be *atomic* if $c = (\emptyset, \emptyset, \emptyset)$. We use $|c| = |E| + \sum_{c_i \in C} |c_i|$, with $|c| = 1$ if $c$ is atomic, as a notion of the *size* of $c$. Let $\mathcal{C}$ be the class of all configurations and $\mathcal{C}_c$ be the class of sub-configurations of $c$. By definition, $c \in \mathcal{C}_c$ and, from $c \in \mathcal{C}_{c'}$, we have $c \subseteq c'$.

Figure 1 illustrates two relations on $\mathcal{C}$: *owns*-links reflect $C$ in Definition 1, and *couples*-links correspond to $E$ and $w$ in Definition 1. In addition, *sub-configuration-of*-links are implicit and reflect Definition 2, and *corresponds-to* relates distinct sub-configurations (Definition 5 below). Figure 3 on page 15 puts these relationships into a meta-modelling context.

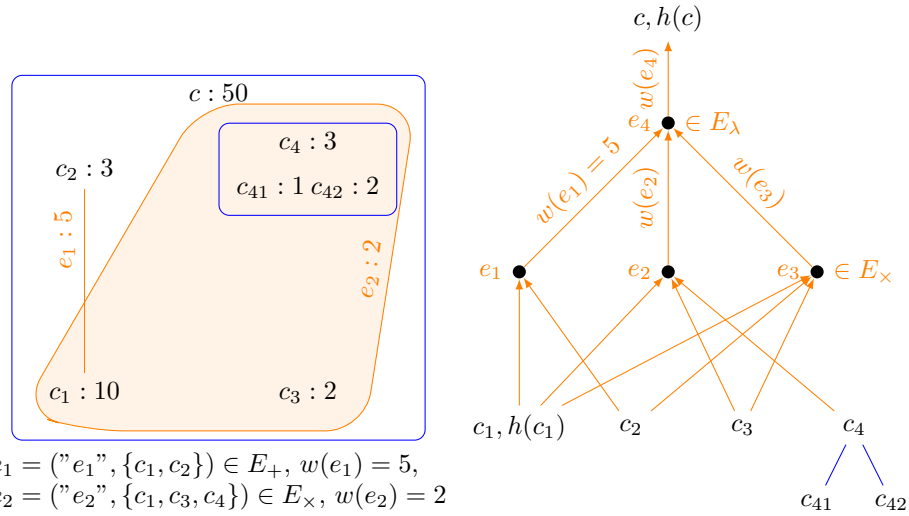


Figure 1: Bird's eye view of a configuration $c$ (left) with atomic configurations $c_1$, $c_2$, and $c_3$, a compound configuration $c_4$ with two further configurations $c_{41}$ and $c_{42}$, and two hyperedges $e_1$ and $e_2$. Tree view of $c$ (right) with a multiplicative hyperedge $e_3 \in E_\times$ and a hyperedge $e_4$ part of the scaling facet $E_\lambda$.

# 3 Assessing the Complexity of Configurations

Addressing **RQ1**, this section discusses how a configuration can be assessed by using a multilinear operator (Definition 4) based on the incidence matrix of the hypergraph (Definition 1) associated with the configuration. To allow compositional assessment, we work with sub-configurations (Definition 2, Figure 1). Complexity assignment happens either individually or via hyperedges. Different flavours of complexity can be assigned and one can control whether the assigned quantity has a *multiplicative* or *additive* effect (Section 3.2).

## 3.1 Complexity Assessment

**Incidence Matrices of Hypergraphs.** Given a configuration $c = (C, E, w)$, $C$ and $E$ induce a (counter)incidence matrix $\mathbf{E} \in \{-1, 0, 1\}^{n \times m}$, each element $(c, e)$ specifying whether $(\pm 1)$ or not $(0)$ vertex $c$ and edge $e$ are (counter)incident. Each column of $\mathbf{E}$ specifies all the vertices (counter)incident to a particular hyperedge. From $w$, we obtain an edge weighting vector

$$\mathbf{w} = (w(e_1), \ldots, w(e_m))^{\mathrm{T}} \in \mathbb{R}^m .$$

**Complexity Aggregation.** We define *complexity* $h(c)$ as a vector-valued map $h \colon \mathcal{C}^d \to \mathbb{R}^d_{\geq 0}$ computing vectors of dimension $d \in \mathbb{N}$. Interpreting $C$ as a vector

$\mathbf{c} = (c_1, \ldots, c_n)^{\mathrm{T}}$, we obtain the *local complexity* vector of $c$ by

$$h(\mathbf{c}) = (h(c_1), \ldots, h(c_n))^{\mathrm{T}} \in \mathbb{R}^n_{\geq 0}\,. \tag{1}$$

Apart from a vector decomposing the configuration vector (1), we can also obtain a vector decomposing complexity measures

$$\mathbf{h}(c) = (h_1(c), \ldots, h_n(c))^{\mathrm{T}} \in \mathbb{R}^n_{\geq 0}\,,$$

or a combination of both by

$$\mathbf{h}(\mathbf{c}) = (h_1(c_1), \ldots, h_n(c_m))^{\mathrm{T}} \in \mathbb{R}^{n\times m}_{\geq 0}\,.$$

In the remainder, we will focus on (1). Looking at $c$ as a whole, we can obtain a complexity scalar with

**Definition 3 (Structural Complexity)** The *structural complexity* $h(c) \in \mathbb{R}_{\geq 0}$ of a configuration $c$ is given by

$$h(c) = \max\{h(\mathbf{c})^{\mathrm{T}}\mathbf{E}\mathbf{w}, 0\}\,. \qquad \square$$

$h(c) < 0$, a potential result of counterincident edges and negative weights, is avoided through $\max\{\cdot, 0\}$. Although not detailed here, one can interpret negative complexity as "less complex than expected".

If $|e| = 2$ for any $e \in E$ and without labels,[1] $E$ and $w$ induce a weighted symmetric adjacency matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$ with cells $\mathbf{W}_{cc'} = \mathbf{W}_{c'c} = \frac{1}{2}w(e)$ for $e = (l, \{c, c'\}) \in E$ and $\mathbf{W}_{cc} = 1$ for any $c$.[2] This specific case suggests an alternative to Definition 3 for simple graphs:

$$h(c) = \|h(\mathbf{c})^{\mathrm{T}}\mathbf{W}\|_2. \tag{2}$$

## 3.2 Attributing Complexity to a Configuration

Hyperedges allow one to attribute complexity to a configuration locally, globally, or in a freely distributed manner.

**Additive, Multiplicative, and Scaling Facets.** One can provide complexity estimates for Definition 3 in three ways, called the scaling, additive, and multiplicative facets. These facets form a partition $E = E_\lambda \uplus E_+ \uplus E_\times$ with a corresponding split of $w$, $\mathbf{E} = (\mathbf{E}_\lambda, \mathbf{E}_+, \mathbf{E}_\times)$, and $\mathbf{w} = (\mathbf{w}_\lambda, \mathbf{w}_+, \mathbf{w}_\times)^{\mathrm{T}}$. Then, we obtain

**Definition 4 (Faceted Structural Complexity)** The *faceted structural complexity* $h(c)$ of a configuration $c$ is given by

$$h(c) = \max\Big\{ \underbrace{\lambda_c}_{\text{scaling facet}} \Big( \underbrace{\mathbf{E}_+\mathbf{w}_+}_{\text{additive facet, } h(\mathbf{c})} \Big)^{\mathrm{T}} \underbrace{\mathbf{E}_\times\mathbf{w}_\times}_{\text{multiplicative facet}}, 0 \Big\}$$

where $\lambda_c = \prod_{e\in E_\lambda} \lambda_e$ and $\lambda_e \in \mathbb{R}$. $\square$

[1] $E$ only has unlabelled, undirected edges with cardinality two.

[2] Self-adjacency can be used to aggregate complexity in a degenerate graph with $E = \varnothing$.

To simplify investigation and focus on key properties, we occasionally apply

**Proposition 1 (Non-negativity, Irreducibility)** *Structural complexity is a* non-negative *and* irreducible *quantity, that is,* $h(\mathbf{c}) \in \mathbb{R}^n_{\geq 0}$, $\mathbf{E} \in \mathbb{R}^{n\times m}_{\geq 0}$, $\mathbf{w} \in \mathbb{R}^m_{\geq 0}$, *and* $w(e), \lambda_e \geq 1$ *for* $e \in E_\lambda \cup E_\times$. □

For example, the operator variants for $h(c)$ defined in the Definitions 3 and 4 are linear when Proposition 1 applies.

Via the *additive facet* $E_+$, the local complexity $h(c_i)$ of $c_i \in C$ results from adding the weights of all *additive hyperedges* in $E_+$ that $c_i$ is involved,

$$\begin{aligned} h(c_i) &= \mathbf{E}_{+i*}\mathbf{w}_+ \\ &= \textstyle\sum_{j=1}^{|E_+|} \mathbf{E}_{+ij} w(e_{+j})\,. \end{aligned} \tag{3}$$

Via the *multiplicative and scaling facets* $E_\times$ and $E_\lambda$, the overall complexity $h(c)$ results from summing $n$ products, each involving $h(c_i)$ and the sum of the weights of all *multiplicative hyperedges* in $E_\times$ that $c_i$ is involved,

$$\begin{aligned}
h(c) &= \lambda_c h(\mathbf{c})^{\mathrm{T}} \mathbf{E}_\times \mathbf{w}_\times && \text{(by Def. 4)} \\
&= \lambda_c (\mathbf{E}_+ \mathbf{w}_+)^{\mathrm{T}} (\mathbf{E}_{\times i*} \mathbf{w}_\times)_{i=1}^n && \text{(add. \& mult. hyperedges of } c_i\text{)} \\
&= \lambda_c ((h(c_i))_{i=1}^n)^{\mathrm{T}} (\mathbf{E}_{\times i*} \mathbf{w}_\times)_{i=1}^n && \text{(local complexity of } c_i\text{)} \\
&= \lambda_c ((\mathbf{E}_{+i*} \mathbf{w}_+)_{i=1}^n)^{\mathrm{T}} (\mathbf{E}_{\times i*} \mathbf{w}_\times)_{i=1}^n && \text{(by (3))} \\
&= \lambda_c ((\textstyle\sum_{j=1}^{|E_+|} \mathbf{E}_{+ij} w(e_{+j}))_{i=1}^n)^{\mathrm{T}} (\sum_{j=1}^{|E_\times|} \mathbf{E}_{\times ij} w(e_{\times j}))_{i=1}^n && \text{(by (3), anal.)} \\
&= \lambda_c \textstyle\sum_{i=1}^n \big( \sum_{j=1}^{|E_+|} \mathbf{E}_{+ij} w(e_{+j}) \big) \big( \sum_{j=1}^{|E_\times|} \mathbf{E}_{\times ij} w(e_{\times j}) \big) && (4) \\
&= \lambda_c \textstyle\sum_{i=1}^n h(c_i) \sum_{j=1}^{|E_\times|} \mathbf{E}_{\times ij} w(e_{\times j}) && \text{(local complexity } h(c_i) \text{ re-isolated)} \\
&= \textstyle\sum_{i=1}^n \lambda_c \mathbf{E}_{+i*} \mathbf{w}_+ \mathbf{E}_{\times i*} \mathbf{w}_\times && \text{(*, complexity projected onto } c_i\text{)} \\
&\stackrel{\text{def}}{=} \textstyle\sum_{i=1}^{|C|} h(c)|_{c_i} && \text{(summarised, recall } n = |C|\text{)}
\end{aligned}$$

The derivation leads to the observation (*) that $h(c)$ can be represented as the sum of the *projections* of all hyperedges onto the members of $C$. From this observation, we extract

**Corollary 1** *Given a configuration* $c = (C, E, w)$ *and* complexity projections $h(c)|_{c_i} \stackrel{\text{def}}{=} \lambda_c \mathbf{E}_{+i*} \mathbf{w}_+ \mathbf{E}_{\times i*} \mathbf{w}_\times$ *for* $c_i \in C$, *we rephrase structural complexity as*

$$h(c) = \textstyle\sum_{i=1}^{|C|} h(c)|_{c_i}\,.$$ □

**Local Complexity.** $E_+$ allows local complexity to be split in pieces (Definition 4). $h(\mathbf{c})$ (Definition 3) can also be given in terms of singleton hyperedges via $\mathbf{w}_+^\ell = \mathbf{w}_+^C = h(\mathbf{c})$ with $\mathbf{E}_+^\ell = \mathbf{E}_+^C = \mathbf{I}_n$. A uniformly distributive hyperedge is a hyperedge $e = (l, C) \in E_+$, where a *complexity effect* $\hat{w}$ is distributed uniformly across all members of $C$ by using $w(e) = \hat{w}/|C| = \hat{w}/n$ in $\mathbf{w}$.

Following a single-point-of-information philosophy, singleton hyperedges enable all complexity estimates to be kept in $w$ (and, thus, $\mathbf{w}$). Any combination of additive hyperedges can be brought into a form with only singleton additive hyperedges. This observation leads to

**Corollary 2 (Additive Normalform)** *Given an additive complexity vector of the form* $h(\mathbf{c}) = \mathbf{E}_+\mathbf{w}_+ = (\mathbf{E}_+^{-\ell}, \mathbf{E}_+^{\ell})(\mathbf{w}_+^{-\ell}, \mathbf{w}_+^{\ell})^{\mathrm{T}}$*, we define* $h(\mathbf{c}) = \mathbf{E}_+^{\ell}\mathbf{w}'_+ = \mathbf{I}_n\mathbf{w}'_+ = \mathbf{w}'_+$ *where* $\mathbf{w}'_{+i} = \mathbf{w}_{+i}^{\ell} + \sum_{j\in 1..|E_+^{-\ell}|} \mathbf{E}_{ij}^{-\ell}\mathbf{w}_{+j}^{-\ell}$. □

**Global Complexity.** With $E_\lambda$, one can add complexity to $c$ as a whole. Consequently, $e \in E_\lambda$ has the form $e = (l, C)$ (i.e., it covers $c$) and is part of the multiplicative facet with a column vector $\mathbf{e} = \mathbf{1}_n$ added to $\mathbf{E}_\times$.

**Distributing Complexity.** With $E_\times$, one can distribute complexity estimates using hyperedges of the form $e = (l, C')$ with $C' \neq C$. To include local complexity, for each $c_i$, we add an auto-labelled self-loop $(c_i, \{c_i\})$ to $E_\times$ and set $w(c_i, \{c_i\}) = 1$. This way, we obtain $E_\times^{\ell} \subseteq E_\times$ as well as $\mathbf{E}_\times^{\ell}$ and $\mathbf{w}_\times^{\ell}$ as part of $\mathbf{E}_\times$ and $\mathbf{w}_\times$. Consequently, we have that $\mathbf{E}_\times^{\ell} = \mathbf{I}_n$ and $\mathbf{w}_\times^{\ell} = \mathbf{1}_n$. Multiplicative self-loops allow a relaxation of Proposition 1, such that $w(e) \in \mathbb{R}_{\geq 0}$ for $e \in E_\times^{-\ell}$.

Generally, additive or multiplicative self-loops in $\mathbf{E}_+$ and $\mathbf{E}_\times$ are optional. However, to benefit from some algebraic properties when manipulating hierarchical hypergraphs as discussed below, self-loops become essential.

## 3.3 Compositionality

First, we identify the conditions under which scaling corresponds to adding a multiplicative hyperedge. Second, we show how recursive complexity computation and hypergraph expansion commute. Given a configuration $c = (C, E, w)$, we only consider the case of $h(c) \geq 0$ in the following.

**Compositionality via the Scaling Facet.** Definition 4 raises the question of when is adding a hyperedge $e$ to a configuration $c$ equivalent to multiplying its complexity $h(c)$ with a scalar factor $\lambda_e$? One may ask for the equivalence

$$\lambda_e h(c) \overset{\text{Def. 3}}{=} \lambda_e h(\mathbf{c})^{\mathrm{T}}\mathbf{E}\mathbf{w} = h(\mathbf{c})^{\mathrm{T}}\mathbf{E}'\mathbf{w}'\,.$$

Let $e = (l, C')$ with label $l \in L$, $C' \subseteq C$, and weight $w(e) \in \mathbb{R}$. $e$ induces a vector $\mathbf{e}$ with incidence 1 for all $c_i \in C'$. We can then define $c' = (C, E', w')$ with $\mathbf{E}' = (\mathbf{E}, \mathbf{e})$ and $\mathbf{w}' = (\mathbf{w}, w(e))^{\mathrm{T}}$. For $C' = C$, we obtain $\mathbf{e} = \mathbf{1}_n \in \mathbf{E}_\times$ and, thus, $\mathbf{E}' = (\mathbf{E}, \mathbf{1}_n)$, and $e$ belongs to the scaling facet. Hence,

$$\begin{aligned}
h(\mathbf{c})^{\mathrm{T}}\mathbf{E}'\mathbf{w}' &=^{\text{def}} h(\mathbf{c})^{\mathrm{T}} \begin{bmatrix}\mathbf{E} & \mathbf{1}_n\end{bmatrix} \begin{bmatrix}\mathbf{w} \\ w(e)\end{bmatrix} && \text{(keep } h(\mathbf{c}) \text{ for later)} \\
&= h(\mathbf{c})^{\mathrm{T}}\big(\mathbf{E}\mathbf{w} + \mathbf{1}_n w(e)\big) \\
&= h(\mathbf{c})^{\mathrm{T}}\mathbf{E}\mathbf{w} + h(\mathbf{c})^{\mathrm{T}}\mathbf{1}_n w(e) && \text{(by Def. 3)}
\end{aligned}$$

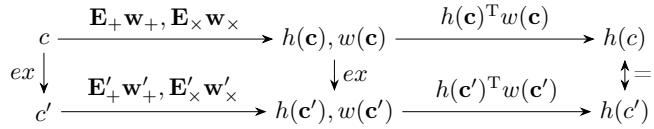


Figure 2: Complexity attribution and configuration tree expansion commute

$$\lambda_e h(c) = h(c) + w(e) h(\mathbf{c})^\mathrm{T} \mathbf{1}_n \qquad \text{(*, divide by } h(c)\text{)}$$

$$\lambda_e = 1 + w(e) \frac{h(\mathbf{c})^\mathrm{T} \mathbf{1}_n}{h(c)} \qquad \text{(**, if } w(e) \text{ is known)}$$

$$w(e) = \frac{(\lambda_e - 1) h(c)}{h(\mathbf{c})^\mathrm{T} \mathbf{1}_n} = \frac{(-1 + \prod_{r \in R} \lambda_r) h(c)}{\sum_{c'' \in C} h(c'')} \qquad \text{(***, if } \lambda_e \text{ is known)}$$

Note that $h(\mathbf{c})^\mathrm{T} \mathbf{1}_n = \sum_{c'' \in C} h(c'')$. Equation (*) highlights how multiplying $h(c)$ with a factor $\lambda_e$ corresponds to adding to $h(c)$ the sum of all local measurements weighted by some $w(e)$ chosen according to equation (**). Finally, equation (***) interprets $e$ as a situation where $\lambda_e = \prod_{r \in R} \lambda_r$ is the compound factor arising from the scaling complexity of some set $R$ of requirements.

**Corollary 3** *Given a configuration $c = (C, E, w)$, a factor $\lambda_e$ can be encoded as a hyperedge $e = (l, C)$ by extending $c$ to $c' = (C, E \cup \{e\}, w \cup \{e \mapsto w(e)\})$ based on (***). Vice versa, any hyperedge of the form $e = (l, C)$ can be encoded as a factor $\lambda_e$ and removed from $c$, leading to $c' = (C, E \setminus \{e\}, w|_{E \setminus \{e\}})$.* □

Corollary 3 indicates that Definition 3 offers sufficient instrumentation to allow practical uses of the framework via the scaling facet.

**Complexity Computation Commutes with Hypergraph Expansion.** Given a configuration $c$, we can compute $h(c)$ either by

(i) recursively applying Definition 4, starting with the leaves of the configuration tree, or

(ii) *flattening $c$'s hypergraph* by recursively expanding non-atomic configurations into a configuration $c' = ex(c)$ and then applying Definition 4 to $c'$.

As, by Corollary 3, the scaling facet is multiplicative, we focus on the additive and multiplicative facets in the remainder. Figure 2 and Theorem 1 capture this relationship, its accompanying proof provides an algorithm for (ii).

**Theorem 1 (Attribution and Expansion Commute)** *Given a configuration $c = (C, E, w)$, the expansion $c' = ex(c)$ yields the equality $h(c) = h(c')$.* □

Proof (Theorem 1) The proof is by constructing a map $ex_1 \colon \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ for expanding an arbitrary non-atomic element $c_i \in C$ with $c_i = (C_i, E_i, w_i)$ into an expanded configuration $c' = (C', E', w')$, such that $h(c) = h(c')$. An invocation of $c' = ex_1(c, c_i)$ yields $C' = (C \setminus \{c_i\}) \cup C_i$. The contruction of $E'$

and $w'$—the incidence matrix $\mathbf{E}$ and weight vector $\mathbf{w}$ segmented according to Section 3.2—needs closer attention. First, we extract the parts related to $c_i$:

$$
\begin{aligned}
h(c) &= h(c') && \text{(by assumption)}\\
h(\mathbf{c})^{\mathrm{T}} w(\mathbf{c}) &= h(\mathbf{c}')^{\mathrm{T}} w(\mathbf{c}') && \text{(by Def. 3)}\\
(h(\mathbf{c}_{-i})^{\mathrm{T}}, h(\mathbf{c}_i))(w(\mathbf{c}_{-i}), w(\mathbf{c}_i))^{\mathrm{T}} &= (h(\mathbf{c}'_{-i})^{\mathrm{T}}, h(\mathbf{c}'_i)^{\mathrm{T}})(w(\mathbf{c}'_{-i}), w(\mathbf{c}'_i))^{\mathrm{T}} \\
&&& \text{(by (1) and } c' = ex_1(c, c_i) \text{ on LHS)}\\
h(\mathbf{c}_{-i})^{\mathrm{T}} w(\mathbf{c}_{-i}) + h(\mathbf{c}_i) w(\mathbf{c}_i) &= h(\mathbf{c}'_{-i})^{\mathrm{T}} w(\mathbf{c}'_{-i}) + h(\mathbf{c}'_i)^{\mathrm{T}} w(\mathbf{c}'_i)\\
h(\mathbf{c}_i) w(\mathbf{c}_i) &= h(\mathbf{c}'_i)^{\mathrm{T}} w(\mathbf{c}'_i)\,. && (*, \text{ by } \mathbf{c}_{-i} = \mathbf{c}'_{-i})
\end{aligned}
$$

Equation (*) relates the complexity fraction of the unexpanded $\mathbf{c}_i$ and the vectorial complexity fraction of $\mathbf{c}'_i$. Without loss of generality, we assume $c_i$ to be in additive normalform (Corollary 2). Recalling $\mathbf{c}_i \equiv c_i \equiv C_i \equiv \mathbf{c}'_i$ and unfolding the structure of $h(\mathbf{c}_i)$, $w(\mathbf{c}_i)$, $h(\mathbf{c}'_i)$, and $w(\mathbf{c}'_i)$ according to Definition 4 yields

$$
\begin{aligned}
h(\mathbf{c}_i) &= (\mathbf{E}_{i+}\mathbf{w}_{i+})^{\mathrm{T}} \mathbf{E}_{i\times}\mathbf{w}_{i\times} \in \mathbb{R}\,,\\
w(\mathbf{c}_i) &= \underbrace{\mathbf{E}^{-\ell}_{\times i*}\mathbf{w}^{-\ell}_{\times}}_{\text{multipl. facet}} + \underbrace{\mathbf{E}^{\ell}_{\times i-i}\mathbf{w}^{\ell}_{\times -i}}_{0 \text{ (cf. self-loops)}} + \underbrace{\mathbf{E}^{\ell}_{\times ii}\mathbf{w}^{\ell}_{\times i}}_{1 \text{ (cf. self-loops)}} \in \mathbb{R}\,, && (5)\\
h(\mathbf{c}'_i) &= \underbrace{\mathbf{E}'^{-\ell}_{+i*}\mathbf{w}'^{-\ell}_{+}}_{?\text{ (to be id.)}} + \underbrace{\mathbf{E}'^{\ell}_{+i-i}\mathbf{w}'^{\ell}_{+-i}}_{\mathbf{0} \text{ (cf. self-loops)}} + \underbrace{\mathbf{E}'^{\ell}_{+ii}\mathbf{w}'^{\ell}_{+i}}_{\mathbf{E}_{i+}\mathbf{w}_{i+}} \in \mathbb{R}^{|C_i|}\,, \quad \text{and} && (6)\\
w(\mathbf{c}'_i) &= \underbrace{\mathbf{E}'^{-\ell}_{\times i*}\mathbf{w}'^{-\ell}_{\times}}_{?\text{ (to be id.)}} + \underbrace{\mathbf{E}'^{\ell}_{\times i-i}\mathbf{w}'^{\ell}_{\times -i}}_{\mathbf{0} \text{ (cf. self-loops)}} + \underbrace{\mathbf{E}'^{\ell}_{\times ii}\mathbf{w}'^{\ell}_{\times i}}_{\mathbf{E}_{i\times}\mathbf{w}_{i\times}} \in \mathbb{R}^{|C_i|}\,, && (7)
\end{aligned}
$$

which, via substitution into (*), results in

$$
\begin{aligned}
h(\mathbf{c}_i)(\mathbf{E}^{-\ell}_{\times i*}\mathbf{w}^{-\ell}_{\times} + 1) &= (\mathbf{E}'^{-\ell}_{+i*}\mathbf{w}'^{-\ell}_{+} + \mathbf{E}_{i+}\mathbf{w}_{i+})^{\mathrm{T}}(\mathbf{E}'^{-\ell}_{\times i*}\mathbf{w}'^{-\ell}_{\times} + \mathbf{E}_{i\times}\mathbf{w}_{i\times})\,,\\
h(\mathbf{c}_i)\mathbf{E}^{-\ell}_{\times i*}\mathbf{w}^{-\ell}_{\times} + h(\mathbf{c}_i) &= (\mathbf{E}'^{-\ell}_{+i*}\mathbf{w}'^{-\ell}_{+})^{\mathrm{T}}\mathbf{E}'^{-\ell}_{\times i*}\mathbf{w}'^{-\ell}_{\times} + (\mathbf{E}_{i+}\mathbf{w}_{i+})^{\mathrm{T}}\mathbf{E}'^{-\ell}_{\times i*}\mathbf{w}'^{-\ell}_{\times}\\
&\quad + (\mathbf{E}'^{-\ell}_{+i*}\mathbf{w}'^{-\ell}_{+})^{\mathrm{T}}\mathbf{E}_{i\times}\mathbf{w}_{i\times} + h(\mathbf{c}_i)\,,\\
h(\mathbf{c}_i)\mathbf{E}^{-\ell}_{\times i*}\mathbf{w}^{-\ell}_{\times} &= (\mathbf{E}'^{-\ell}_{+i*}\mathbf{w}'^{-\ell}_{+})^{\mathrm{T}}\mathbf{E}'^{-\ell}_{\times i*}\mathbf{w}'^{-\ell}_{\times} + (\mathbf{E}_{i+}\mathbf{w}_{i+})^{\mathrm{T}}\mathbf{E}'^{-\ell}_{\times i*}\mathbf{w}'^{-\ell}_{\times}\\
&\quad + (\mathbf{E}'^{-\ell}_{+i*}\mathbf{w}'^{-\ell}_{+})^{\mathrm{T}}\mathbf{E}_{i\times}\mathbf{w}_{i\times}\,,\\
h(\mathbf{c}_i)\mathbf{E}^{-\ell}_{\times i*}\mathbf{w}^{-\ell}_{\times} &= (\mathbf{E}'^{-\ell}_{+i*}\mathbf{w}'^{-\ell}_{+} + \mathbf{E}_{i+}\mathbf{w}_{i+})^{\mathrm{T}}\mathbf{E}'^{-\ell}_{\times i*}\mathbf{w}'^{-\ell}_{\times}\\
&\quad + (\mathbf{E}'^{-\ell}_{+i*}\mathbf{w}'^{-\ell}_{+})^{\mathrm{T}}\mathbf{E}_{i\times}\mathbf{w}_{i\times}\,,\\
(\mathbf{E}_{i+}\mathbf{w}_{i+})^{\mathrm{T}}\mathbf{E}_{i\times}\mathbf{w}_{i\times}\mathbf{E}^{-\ell}_{\times i*}\mathbf{w}^{-\ell}_{\times} &= (\mathbf{E}_{i+}\mathbf{w}_{i+})^{\mathrm{T}}\mathbf{E}'^{-\ell}_{\times i*}\mathbf{w}'^{-\ell}_{\times}\,, && (\text{use } \mathbf{E}'^{-\ell}_{+i*} = \mathbf{0})\\
\underbrace{\mathbf{E}_{i\times}\mathbf{w}_{i\times}}_{\in \mathbb{R}^{|C_i|}}\underbrace{\mathbf{E}^{-\ell}_{\times i*}\mathbf{w}^{-\ell}_{\times}}_{\in \mathbb{R}} &= \underbrace{\mathbf{E}'^{-\ell}_{\times i*}\mathbf{w}'^{-\ell}_{\times}}_{\in \mathbb{R}^{|C_i|}}\,. && (8)
\end{aligned}
$$

Note that the simplification gained by choosing $\mathbf{E}'^{-\ell}_{+i*} = \mathbf{0}$ relies on $c$ being in additive normalform (Corollary 2) but does not reduce the applicability of Theorem 1. With (8), we can simplify (5), (6), and (7) in order to obtain

$$
w(\mathbf{c}_i) = \mathbf{E}^{-\ell}_{\times i*}\mathbf{w}^{-\ell}_{\times} + 1\,,
$$

$$h(\mathbf{c}'_i) = \mathbf{E}_{i+}\mathbf{w}_{i+}\,, \quad \text{and}$$
$$w(\mathbf{c}'_i) = \mathbf{E}_{i\times}\mathbf{w}_{i\times}(\mathbf{E}^{-\ell}_{\times i*}\mathbf{w}^{-\ell}_{\times} + 1) = \mathbf{E}_{i\times}\mathbf{w}_{i\times}w(\mathbf{c}_i)\,.$$

To derive the segments of the matrices $\mathbf{E}'_+$, $\mathbf{w}'_+$, $\mathbf{E}'_\times$, and $\mathbf{w}'_\times$, we first construct $\mathbf{E}'^{-\ell}_{\times i*} \in \mathbb{R}^{|C_i|\times|E_\times|-|C|}$ from $\mathbf{E}^{-\ell}_{\times i*} \in \mathbb{R}^{1\times|E_\times|-|C|}$, satisfying (8). Recall that the expansion does not introduce new hyperedges in $\mathbf{E}'^{-\ell}$. In (8), the non-local multiplicative complexity of an element $c_u \in C_i$ amounts to

$$\sum_{v\in E^{-\ell}_{\times i*}} \mathbf{E}'^{-\ell}_{\times i*,uv}\mathbf{w}'^{-\ell}_{\times,v} = \underbrace{\sum_{v\in E_{i\times}} \mathbf{E}_{i\times,uv}\mathbf{w}_{i\times,v}}_{w(c_u)} \sum_{v\in E^{-\ell}_{\times i*}} \mathbf{E}^{-\ell}_{\times i*,1v}\mathbf{w}^{-\ell}_{\times,v}$$

$$\sum_{v\in E^{-\ell}_{\times i*}} \mathbf{E}'^{-\ell}_{\times i*,uv}\mathbf{w}^{-\ell}_{\times,v} = \sum_{v\in E^{-\ell}_{\times i*}} w(c_u)\mathbf{E}^{-\ell}_{\times i*,1v}\mathbf{w}^{-\ell}_{\times,v} \qquad (\text{by } \mathbf{w}'^{-\ell}_{\times} = \mathbf{w}^{-\ell}_{\times})$$

$$\mathbf{E}'^{-\ell}_{\times i*,uv} = w(c_u)\mathbf{E}^{-\ell}_{\times i*,1v} \qquad (\text{by comparison of coefficients})$$

$$\mathbf{E}'^{-\ell}_{\times i*} = \mathbf{E}_{i\times}\mathbf{w}_{i\times}\mathbf{E}^{-\ell}_{\times i*}\,. \qquad (\text{lifted to } \mathbb{R}^{|C_i|\times|E_\times|-|C|})$$

The last equation clarifies that, under reasonable side conditions, $\mathbf{E}'^{-\ell}_{\times i*}$ simply arises from (8) after division by $\mathbf{w}^{-\ell}_{\times}$.

Recall that $h(c') = h(\mathbf{c}')^{\mathrm{T}}w(\mathbf{c}') = (\mathbf{E}'_+\mathbf{w}'_+)^{\mathrm{T}}\mathbf{E}'_\times\mathbf{w}'_\times$. With the considerations for $c_i$ reflected in equations (5) to (8), we can define the expansion via[3]

$$\mathbf{E}'_+ = \begin{bmatrix} \mathbf{E}^{-\ell}_{+-i*} & \mathbf{E}^{\ell}_{+-i-i} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{E}_{i+} \end{bmatrix}, \qquad \mathbf{w}'_+ = (\mathbf{w}^{-\ell}_+, \mathbf{w}^{\ell}_{+-i}, \mathbf{w}_{i+})^{\mathrm{T}},$$

$$\mathbf{E}'_\times = \begin{bmatrix} \mathbf{E}^{-\ell}_{\times-i*} & \mathbf{E}^{\ell}_{\times-i-i} & \mathbf{0} \\ \mathbf{E}_{i\times}\mathbf{w}_{i\times}\mathbf{E}^{-\ell}_{\times i*} & \mathbf{0} & \mathbf{E}_{i\times} \end{bmatrix}, \text{ and } \quad \mathbf{w}'_\times = (\mathbf{w}^{-\ell}_\times, \mathbf{w}^{\ell}_{\times-i}, \mathbf{w}_{i\times})^{\mathrm{T}}.$$

$C'$, $E'$, and $w'$ can be derived from $h(\mathbf{c}')$, $\mathbf{E}'$, and $\mathbf{w}'$. With $ex_1$, any non-atomic configuration can be expanded and embedded into a higher-level configuration. To close the proof, we define $ex$ as an iterative map using $ex_1$ to derive $c'$ as a least fixed point $c' = \mu c\colon ex_1(c, \mathrm{sel}(C))$, where the map $\mathrm{sel}(S)$ selects an arbitrary element from the set $S$ or $\perp$ if $S = \varnothing$. $c = ex_1(c, \perp)$ covers the terminal case. As the number of nodes in a configuration tree is finite, it is easy to see that $ex$ will terminate while preserving Theorem 1. ■

Successive expansion of non-atomic configurations leads to a single flattened hypergraph for the considered top-level configuration. Complexity calculation can then be done by a single application of Definition 4.

**Corollary 4** *The restrictions $h(c) \geq 0$ and $h(\mathbf{c}) = \mathbf{E}_+\mathbf{w}_+$ to achieve compositionality imply the equivalence of Definition 3 and Definition 4.* □

[3] $\mathbf{E}^{-\ell}$ and $\mathbf{w}^{-\ell}$ refer to $\mathbf{E}$ and $\mathbf{w}$ without self-loop components, and the generic zero matrix $\mathbf{0}$ has appropriate dimensions.

Nevertheless, compositionality allows a mix of Definitions 3 and 4 and (2) across a configuration tree. For example, one can use Definition 4 for the top level of the tree and (2) at a lower level.

Theorem 1 and Corollary 4 enable the conversion of a hierarchical hypergraph (i.e., one with non-atomic nodes) into a flat hypergraph (i.e., one with only atomic nodes) and, thus, offer a flexible design tool that can be used, for example, for restructuring a system architecture based on complexity clusters.

# 4 Ordering Configurations by Structural Complexity

Addressing **RQ2**, this section introduces some instrumentation to compare and, consequently, order the domain of configurations $\mathcal{C}$ by structural complexity.

**Relating Configurations.** A comparison of configurations requires relating their structures, for example, a relation between the elements of an autonomous submarine and an autonomous car, or a similarity of variants of a specific system family (e.g., two fire-fighting drones differing in only a few characteristics).

**Definition 5 (Commonality)** A *commonality* $\sim^r \subseteq \mathcal{C} \times \mathcal{C}$ is an equivalence relation over configurations. □

$\sim^r$ is expected to be the result of constructing the reflexive, symmetric, transitive closure $\sim_u^{r*}$ of some $\sim_u^r$ specified by the user, perhaps derived from a domain-specific taxonomy [30, 22]. Transitive closure requires configurations to be compared at an appropriate level. For example, given $c_1$ (camera only), $c_2$ (camera, lidar), and $c_3$ (radar, lidar), a comparison regarding "has common sensor" ($\sim^{\text{com}}$) violates transitivity, while a comparison of "has optical sensing" ($\sim^{\text{opt}}$) forms an equivalence class $\{c_1, c_2, c_3\}$.

Given two configurations $c_i$ and their classes of sub-configurations $\mathcal{C}_i \subseteq \mathcal{C}$, $i = 1, 2$, we use the projection $\sim^r_{c_1c_2} = \sim^r \cap (\mathcal{C}_1 \times \mathcal{C}_2) \cup (\mathcal{C}_2 \times \mathcal{C}_1)$. $\sim^r_{c_1c_2}$ inherits the property of an equivalence relation from $\sim^r$. Below, we omit the subscript $c_1c_2$ if it is clear from the context.

## 4.1 Scalar Orders

Let us now consider the comparison of two configurations, $c_i$ for $i = 1, 2$, in the context of an order $\preceq \subseteq \mathcal{C} \times \mathcal{C}$. Below, we examine two linear orders and two partial orders, closing by indicating some relationships among them.

As a first linear order, we look at the *direct black-box comparison* of $c_1$ and $c_2$, stating that $c_2$ "is at least as complex (to design, assure, operate) as" $c_1$ if

$$c_1 \preceq_{\mathsf{bb}} c_2 \equiv h(c_1) \leq h(c_2)\,.$$

As a second linear order, we consider the *selective comparison*, which we define as the sum of signed differences of $c_2$ and $c_1$ in specific regards (e.g., modulo all

pairs $(c_1', c_2')$ of sub-configurations in $\sim^r_{c_1 c_2}$) being positive, formally,

$$c_1 \preceq^r_{\mathsf{sel}} c_2 \equiv 0 \leq \sum_{c_1' \sim^r c_2'} h(c_2') - h(c_1')\,.$$

**Lemma 1** *$\preceq_{\mathsf{bb}}$ and $\preceq^r_{\mathsf{sel}}$ are linear orders.* □

Proof It is easy to see that $\preceq_{\mathsf{bb}}$ is an order and that it is linear.

If $r$ is non-empty, $\preceq^r_{\mathsf{sel}}$ is reflexive (because $0 \leq 0$), antisymmetric (because $r$ is symmetric), and transitive (by the transitivity of $r$ and $(\mathbb{R}, \leq)$). Hence, $\preceq^r_{\mathsf{sel}}$ is an order. If $r$ is empty, the convention $\sum_{\varnothing} f = 0$ renders uncommon configurations to be complexity-equivalent. $\preceq^r_{\mathsf{sel}}$ is total, hence, linear. ■

For some family of configurations $\mathcal{C}^r \subseteq \mathcal{C}$, let $\sim^r$ be the reflexive hull of a bijection $r\colon \mathcal{C}_1 \cap \mathcal{C}^r \to \mathcal{C}_2 \cap \mathcal{C}^r$. This way, we obtain a partial order called *flat glass-box comparison*, defined as

$$c_1 \preceq^r_{\mathsf{fgb}} c_2 \equiv \sim^r \neq \emptyset \wedge \forall (c_1', c_2') \in \sim^r \colon c_1' \preceq_{\mathsf{bb}} c_2'\,.$$

The domination order in [32, p. 132] can be seen as a specialisation of $\preceq^r_{\mathsf{fgb}}$.

**Lemma 2** *$\preceq^r_{\mathsf{fgb}}$ is a partial order.* □

Proof If $r$ is non-empty and all pairs are ordered, $\preceq^r_{\mathsf{fgb}}$ is reflexive and antisymmetric (because $\preceq_{\mathsf{bb}}$ is reflexive and antisymmetric), and transitive (because $\sim^r$ is assumed to be the reflexive hull of a bijection $r$ on $\mathcal{C}^r \subseteq \mathcal{C}$ and two bijections establish another bijection) and, hence, a partial order. If $r$ is empty or not all pairs are ordered, $\preceq^r_{\mathsf{fgb}}$ is undefined, hence, partial. ■

Furthermore, we have the *recursive glass-box comparison*, given as

$$c_1 \preceq^r_{\mathsf{rgb}} c_2 \equiv \sim^r \neq \emptyset \wedge \forall (c_1', c_2') \in \sim^r \colon c_1' \preceq^{r'}_{\mathsf{rgb}} c_2'\,, \tag{9}$$

where $r'$ is the injection associated with $c_1'$ and $c_2'$, and $c_1' \preceq^{r'}_{\mathsf{rgb}} c_2' \equiv c_1' \preceq_{\mathsf{bb}} c_2'$ if $c_1$ or $c_2$ is atomic and $c_1' \sim^{r'} c_2'$.

Because the two glass-box comparisons are $\sim$-relative, they fail to take into account differences in $|c_1|$ and $|c_2|$. Hence, in situations where $c_1$ is a large system with low complexity and $c_2$ is a small one with high complexity, the extreme case $|\sim^r| = 1$ could well lead to $c_1 \preceq^r c_2$, which, because of $\sim^r \neq \emptyset$, is allowed but might turn out to be counterintuitive.

A remedy could be to generalise $\preceq^r_{\mathsf{sel}}$ to a *$\sim$-normalised* order

$$\begin{aligned} c_1 \preceq^r_{\mathsf{nsel}} c_2 \equiv 0 &\leq \sum_{c_1' \sim^r c_2'} \frac{h(c_2')}{|c_1|} - \frac{h(c_1')}{|c_2|} \\ &\equiv 0 \leq \frac{1}{|c_1|} \sum_{c_1' \sim^r c_2'} h(c_2') - \frac{1}{|c_2|} \sum_{c_1' \sim^r c_2'} h(c_1')\,. \end{aligned}$$

If $|c_1| = |c_2|$ and $\sim^r$ is a bijection, we obtain $\preceq^r_{\mathsf{sel}}$. The structural difference between $c_1$ and $c_2$ is abstracted away in both variants.

**Corollary 5** *Given two configurations $c_1$ and $c_2$, if $r' \subseteq r$ with $r' \neq \emptyset$, then*

$$c_1 \preceq^{r}_{\mathsf{[f|r]gb}} c_2 \implies c_1 \preceq^{r'}_{\mathsf{[f|r]gb}} c_2 \,.$$ □

The corollary applies to $\preceq^{r}_{\mathsf{rgb}}$ because the $\sim$-relations in the lower-level configurations are fixed in the context of the comparison.

Provided Proposition 1 (i.e., $\mathbf{E} \in \mathbb{R}^{n\times m}_{\geq 0}$, $\mathbf{w} \in \mathbb{R}^{m}_{\geq 0}$, $\lambda$ and $\mathbf{w}_\times \geq 1$), complexity is monotonic under the sub-configuration relation (Definition 2). This observation is expressed in

**Lemma 3** *For $c, c' \in \mathcal{C}$, $c \subseteq c' \implies h(c) \leq h(c')$.* □

Proof Given $c' = (C', E', w')$, let $c'' = (C', E'', w'|_{E''})$ with $E'' = E' \setminus \{e\}$. From (4), we observe that the removal of edge $e$ leads to $h(c'') \leq h(c')$. More generally, the removal of $E' \subseteq E$ from $c$ maintains $h(c) \leq h(c')$. By Definition 2, removing $c_i \in C$ from $C'$ removes all edges using $c_i$, which is already covered by edge removal. ■

Provided the corresponding assumptions, Lemma 3 yields

**Corollary 6** *For $c_i, c_i' \in \mathcal{C}$, $i = 1, 2$, if $c_1' \subseteq c_1$ and $c_2 \subseteq c_2'$, we have*

$$c_1 \preceq_{\mathsf{bb}} c_2 \implies c_1' \preceq_{\mathsf{bb}} c_2' \,.$$ □

### 4.2 Lexicographical Orders

Definition 4 can be applied to any atomic or sub-configuration, thus, any combination of facets. As a result, one can flexibly compute complexity vectors (1) in addition to complexity scalars. Based on such vectors, we can order configurations *lexicographically*, for example, using $\preceq_{\mathsf{bb}}$.

To simplify notation, we use $c^h$ and $c^t$ for the head (first element) and tail (all but the first elements) of $h(\mathbf{c})$ (or $\mathbf{h}(c)$). Given a (user-defined) linear order $\preceq_i \subset C_i \times C_i$ to prioritise the characteristics in $C_i$ and a local complexity vector $h(\mathbf{c}_i)$ for $c_i$ with $h(c_i^h)$-components in ascending order according to $\preceq_i$, we recursively define

$$c_1 \preceq_{\mathsf{lx}} c_2 \equiv c_1^h \prec_{\mathsf{bb}} c_2^h \vee \left(c_1^h =_{\mathsf{bb}} c_2^h \wedge c_1^t \preceq_{\mathsf{lx}} c_2^t\right)$$

where, for an empty tail $c^t = []$, we define $[] \preceq_{\mathsf{lx}} []$, $[] \preceq_{\mathsf{lx}} c$, and $c \not\preceq_{\mathsf{lx}} []$ for any non-empty configuration $c$.

**Lemma 4** *$\preceq_{\mathsf{lx}}$ is a linear order.* □

Proof The linearity of $\preceq_i$ and $\preceq_{\mathsf{bb}}$ implies the linearity of $\preceq_{\mathsf{lx}}$. ■

## 5 Example: Hardness of Autonomous Systems

The concepts discussed in the Sections 2, 3, and 4 were developed for and motivated by a previous triple-case analysis in the domain of autonomous systems [8]. This section provides additional material to that analysis.

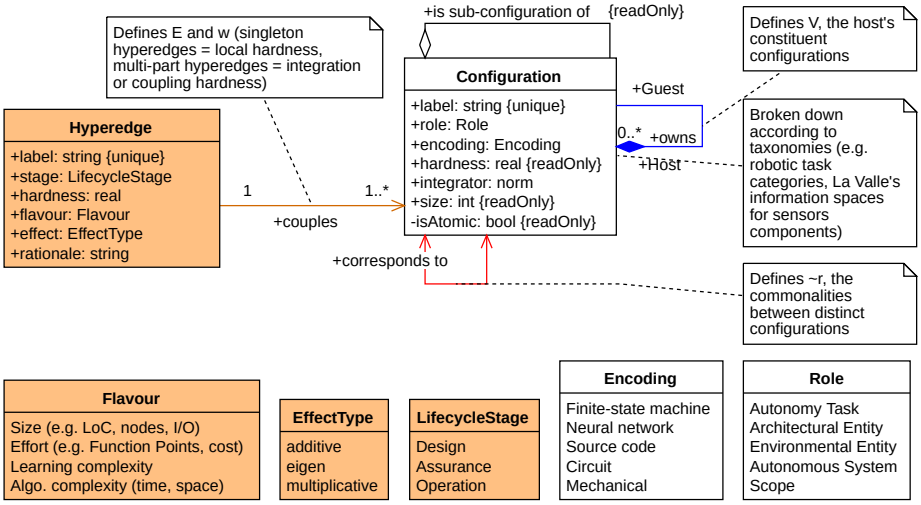


Figure 3: Meta-model of configurations and complexity attribution

## 5.1 Representing Autonomous Systems as Configurations

Figure 3 rephrases the concepts, formally introduced in Section 2, as a meta-model in a SysML class diagram. In the domain of autonomous systems, a configuration $c = (C, E, w)$ can represent the structure of an

- *autonomy task* (e.g., capabilities [22], functions [30]) describing behaviour,
- *architectural entity* (e.g., mechanical, electrical/electronic, software),
- *autonomous system* (i.e., one or more under consideration),
- *environmental entity* (e.g., humans, other systems, infrastructure), and a
- *scope* to form a generic collection of configurations.

$C$ can be partitioned accordingly. Roles, encodings, and lifecycle stages are non-exhaustive and informal. In the remainder, our top-level configuration $c$ will be a single autonomous system.

## 5.2 Structuring Hardness and Integrating its Measurement

We draw inspiration from [32] and graph complexity measures defined over $\mathcal{C}_c$ to inform the choice of the structure of $E$. To assess and compare configurations, we attribute hardness estimates (e.g., envisaged difficulties in $c$'s engineering lifecycle, incl. design, operation, decommissioning) to configurations and compute an overall hardness score. Hyperedges can be grouped according to their lifecycle stage, for example, into a *design*, *assurance*, and *operational* facet.

**Design, Assurance, and Operational Hardness.** Hyperedges $E$ (see the *couples* association in Figure 3) can be used to describe

- *design hardness* related to the construction of an autonomous system (the individual tasks or entities and their integration or coupling),
- *assurance hardness* (e.g., according to a TRLs) related to the verification and validation (V&V) of the desired features or properties of an autonomous system, and
- *operational hardness* resulting from operating an autonomous system in a specific (e.g., complex, hostile, or regulated) environment.

Let $f \in \{d, a, o\}$ be a design, assurance, or operational lifecycle stage of $c$ and let $c^f = (C, E^f, w^f)$ be $c$'s sub-configuration representing $f$. Design and assurance hardness can reflect the complexity of constructing $c$ or achieving the desired assurance level[4] for $c$, derived from $E$ and $w$.

Alluding to TRLs, $h(c^a)$ can be an aggregate of the hardness of assuring the *properties* (e.g., correctness, safety, effectiveness, energy-efficiency) *required* of $c$. In particular, $E^a$ can reflect a requirements list or document.

Based on these lifecycle stages and Section 3.2, we assume $E_\circ$ with $\circ \in \{\lambda, +, \times\}$ to be structured into design, assurance, and operational hyperedges, leading to $E_\circ = E_\circ^d \uplus E_\circ^a \uplus E_\circ^o$ with $\mathbf{E}_\circ = (\mathbf{E}_\circ^d, \mathbf{E}_\circ^a, \mathbf{E}_\circ^o)$ and $\mathbf{w}_\circ = (\mathbf{w}_\circ^d, \mathbf{w}_\circ^a, \mathbf{w}_\circ^o)^{\mathrm{T}}$. For Corollary 4, we then use $\mathbf{E}^{-\ell} = \mathbf{E}^{d,a,o}$.

**Local, Integration, and Vectorial Hardness.** For $f$, hardness assessment can be performed by looking at

- the *local hardness* $h(c_i)$ of $c_i \in C^f$ (provided as a constant if $c_i$ is atomic, otherwise computed recursively),
- the *integration hardness* $h(c^f)$ (i.e., the complexity of coupling the elements of $C^f$ via interfaces and derived from $E^f$ and $w^f$), and
- some *hardness vector*, for example, $h(c^d, c^a) = (h(c^d), h(c^a))^{\mathrm{T}}$.

For example, we can use $\|h(\mathbf{c})\|_2 + h(c^a)$ to combine an aggregate of local hardness values of $C$ with assurance hardness of $c$.

**Facets versus Local Hardness.** Given an *operational condition* (e.g., open environment) or an *assurance requirement* (e.g., safety), hardness can be assigned to a context $C' \subseteq C$, leading to an edge $e = (l, C')$, belonging to the *assurance or operational facet* $E_\times^a \uplus E_\times^o$, and a corresponding weight $w(l, C')$. It can also be useful to interpret $w(l, C') \in (0, w_{\max}]$ as a percentage with $w_{\max} = 1$.

Table 1 provides hyperedge examples for each facet. One thereof indicates how to connect an autonomy task (e.g., navigation) and an environmental entity

[4] In the sense of quality, integrity, confidence, or readiness.

| **Label** $l \in L$ | **Context** $C' \in C$ | **Weight** $w(l, C')$ | **Description (Rationale)** |
|---|---|---|---|
| open env. | $\{c_5, \ldots, c_7\}$ | 0.5 | $\in E^o_\times$: *Performing* the tasks $c_5, \ldots, c_7$ in an open environment increases the overall local hardness of $c_5, \ldots, c_7$ by 50 %. |
| sensor safety | $\{c_1, \ldots, c_4\}$ | 1 | $\in E^a_\times$: *Assuring* requirements on sensor safety of the architectural components $c_1, \ldots, c_4$ doubles the overall local hardness of $c_1, \ldots, c_4$. |
| wire-less com. | $\{c_2, c_3\}$ | 0.2 | $\in E^d_\times$: *Integrating* the architectural entities $c_2$ and $c_3$ via wireless communication increases the overall local hardness of $c_2, c_3$ by 20 %. |

Table 1: Examples of hyperedges in specific facets and their description

(e.g., open environment) and use $w$ to encode operational hardness (e.g., navigating in an open environment). Nevertheless, instead of using facets to add hardness, one can distribute corresponding fractions of local hardness values $h(c_i)$ with $c_i \in C'$ across the considered context $C'$.

**Metrics to Compute Hardness.** The estimate $h(c)$ of an atomic configuration $c$ or the weight $w(e)$ of some hyperedge $e$ can be the result of a specific complexity metric (Section 1). For example, if $c$ is an autonomy task encoded as a state machine, we can use some notion of state machine complexity. If $c$ is a neural network, we might want to apply a definition of learning complexity. If $c$ is an architectural entity (e.g., a software component), we can use function points as a customer- or business-value-based estimate of the design or assurance effort. We can, moreover, use $w(e)$ of $e$ belonging to the assurance scaling facet $E^a_\lambda$ to associate function points to a sub-configuration subjected to V&V.

In summary, Definition 4 offers a variety of ways to combine hardness estimates by considering adjacent node and edge weights.

**Interpreting Local Hardness.** Depending on the *category* (e.g., movement task, sensor fusion sub-system) and *representation* (e.g., finite-state machine or FSM, Markov model [2], convolutional neural network) of $c$ (e.g., waypoint following), *local*, *global*, or *integration hardness* can be obtained from existing complexity estimates. For example, if $c$ is

- an *obstacle detection task* implemented as a neural network, one can use a measure of its nodes, features, connections, I/O dimensions to determine the local hardness in terms of the expected training effort,
- a *software component* written in a high-level procedural language (e.g., C, Java), one could use a translation of function points $\mathsf{fp}(c)$, cyclomatic complexity $\mathsf{cyc}(c)$, or lines of code $\mathsf{loc}(c)$ to determine the local hardness in terms of the expected effort of constructing $c$, or

- a *programmable electronic component* modelled as an FSM along with an assurance requirement $\varphi$, one could use measures of the state space and transition relation $h(c) = \mathsf{fsmsize}(c)$ and the model-checking effort $w(l, c) = \mathsf{effort}(c \models \varphi)$ to determine the local and assurance hardness in terms of the effort of verifying $\varphi$ of $c$.

Note that a combination of estimates using different metrics will either require an alignment (e.g., normalisation) into "complexity units" or the use of complexity vectors (Section 3.1) keeping incompatible measures separate.

### 5.3 Comparing and Ordering Systems by Hardness

The need for comparing configurations can arise from several situations.

**Make-or-buy decisions [18].** One may want to compare two systems by their hardness by systematically breaking down the available hardness information according to the considered configuration tree, roles, lifecycle stages, effect types, encodings, and complexity types (Figure 3).

**Question:** Should we make or buy this component?

**Example:** One question in the unmanned aerial vehicle (UAV) case in [8] is whether a component should be outsourced or developed in-house. Outsourcing reduces the production effort for that component but can increase the integration hardness because of a new interface. Similarly, for assurance, the supplier might not be able to provide guarantees, while assuring an outsourced component in-house will be harder if it is to be treated as a black box.

**Product line engineering [29].** We can order the members of a *family* of systems by hardness. Such a family is typically characterised by similar *variants* that can be derived from a product-line model of the family (e.g., a feature tree with variation points).

**Question:** Which of the configurations minimise hardness while maximising utility?

**Example:** Given a UAV and five variants of a perception system (e.g., camera-based vs. LIDAR-based localisation), we can order the resulting five UAV variants according to the influence of the perception system hardness on the overall hardness of each variant.

**Technical debt assessment [20].** Looking at a system evolution, one can also view hardness as an indicator of accumulated technical debt.

**Question:** Where and how does hardness increase during the accumulation of technical debt?

**Example:** Based on a configuration, we can observe changes both in functionality and hardness. A large increase in hardness in the context of a small increase in utility might give rise to the creation of technical debt.

The vector functional $h$ allows one to provide a notion of *hardness relative to some other quantity*, such as *utility*. Lexicographical orders (Section 4.2) can then be used to Pareto-optimise for multiple objectives.

**System evolution [19].** One can view the *evolution* of a particular system by re-ordering its (partially temporally ordered) *versions* by hardness.

**Questions:** How does hardness evolve throughout the system evolution? To which extent does it decrease or increase?

**Example:** For example, consider version $c^0 = (C^0, E^0, w^0)$ where $E^0$ contains a definition of "TRL2" and version $c^1 = (C^0, E^1, w^1)$ defines "TRL5". Assume that $E^0 \subset E^1$ and, thus, $c^0 \subset c^1$ (Definition 2). Then, one would further expect $c^0 \preceq c^1$ with respect to some order, say $\preceq_{\mathtt{bb}}$, because TRL5 will normally be harder to achieve than TRL2.

**Combinatorial design or workflow optimisation [16].** Hardness information gathered in form of a configuration can serve as a feedback on the choice of a design structure matrix [11].

**Question:** To which extent is assurance hardness influenced by the component or task dependencies established in a system or workflow?

**Example:** A densely coupled architecture can be much harder to verify because the lack of modularity hampers decomposition for isolated verification of components. Consequently, one may minimise such coupling during the selection of an optimal design structure matrix.

# 6 Discussion

**From Diagrams to Hypergraphs to Linear Algebra.** Corollary 4 enables a separation of concerns, allowing one to decompose configurations in an appropriate language (e.g., UML class diagrams) with a focus on visual decomposition and complexity attribution (e.g., hardness stemming from design, assurance, and environmental embedding, from a single component or all elements together, or from integrating a particular group of elements). For complexity assessment, the mapping into hypergraphs and their matrix representation enables scalable automation based on linear algebra.

**Well-formedness of Hyperedges.** In a UML class diagram, hyperedges can easily be defined to span several levels of a *configuration tree*. This situation is not defined and could lead to redundant complexity accumulation. Hyperedges spanning several tree levels are, hence, prohibited by Definition 1.

**Multiple Occurrence versus Shared Use of Configurations.** *Multiple occurrence* involves the use of $c$ in multiple, say $n$, *independent* locations in a configuration tree. Importantly, Definition 3 then takes into account an overall local complexity of $n \cdot h(c)$. It needs to be ensured that such a complexity multiplication is

justified. *Shared use* involves the interaction of a central $c$ with several elements $c_1, \ldots, c_n$. This should be done by keeping the relevant elements in a single hypergraph, that is, $\{c, c_1, \ldots, c_n\} \subseteq C$.

**Avoiding Complexity Blow-ups.** To avoid a rapid increase of complexity when using the scaling facet, one can (i) limit $\lambda_c \in (0, \lambda_{\max}]$, (ii) use the additive facet instead (e.g., $w_+(e_{\mathrm{TRL1}}) = 2$, $w_+(e_{\mathrm{TRL2}}) = 1$, and $w_+(e_{\mathrm{TRL3}}) = 2$ leading to $\|\mathbf{E}_+\mathbf{w}_+\|_1 = 5$), or (iii) use $\lambda_c$ only at the top level of the configuration tree. Option (iii) seems rather easy to use and option (ii) might bring the necessary flexibility depending on the chosen norm.

**Complexity Dampening and Calibration.** One can reduce the influence of a facet or hyperedge via counter-incidence (-1 in $\mathbf{E}$) or negative weights (in $\mathbf{w}$). While the former mechanism can be adjusted per incident element of a hyperedge, the latter affects all elements of a hyperedge.

Definitions 3 and 4 imply a reoccurrence of $h(c_i)$ proportional to the incidence or adjacency of $c_i$, which might lead to a quick blow-up of $h$. Hence, the choice of weights will have to take place under guidance and calibration.

Note that the use of counter-incidence and negative weights weakens the results in the Sections 3.3 (compositionality) and 4.1 (monotonicity) and, thus, limits the discussed range of property-preserving hyperedge modifications.

# 7 Conclusion

This work represents a companion paper to our triple-case analysis in [8]. Here, we propose the notion of a configuration to model a systems architecture (i.e., its components, connections, and properties) in a lean yet versatile and compositional way. Moreover, we encode configurations as weighted and hierarchical hypergraphs, their quantitative assessment as a multilinear operator over the flattened and positive incidence matrices of the hypergraphs, and relations over configurations via linear, partial, and lexicographic orderings. We provide proofs of basic properties as well as a result on compositionality allowing complexity-preserving modifications of a configuration. Future work includes the implementation of the discussed concepts in a software tool.

**Funding and Acknowledgements.** Mario Gleirscher is partially supported by the Chair «Architecture des Systémes Complexes» of the Institut Polytechnique de Paris. Clare Dixon is partially supported by the Centre for Robotic Autonomy in Demanding and Long-Lasting Environments (CRADLE) under EPSRC grant EP/X02489X/1.

**No-GenAI/LLM Clause.** This report was crafted entirely by humans.

# References


[1] Benjamin S. Blanchard and Wolter J. Fabrycky. *Systems Engineering and Analysis*. 4th ed. Upper Saddle River, NJ: Prentice Hall, 2005. ISBN: 0131869779.

[2] Farouk Bonilla, Thomas Holzer, and Shahryar Sarkani. "Complexity Measure for Engineering Systems Incorporating System States and Behavior". In: *IEEE Syst. J.* 15.4 (2021), pp. 4792–4803. ISSN: 2373-7816. DOI: 10.1109/jsyst.2020.3033792.

[3] Frederick Phillips Brooks Jr. *The Mythical Man-Month. Essays on Software Engineering*. Reading, MA: Addison-Wesley, 1975. ISBN: 9780201006506. DOI: 10.1109/mahc.1996.539925.

[4] Nanette Brown et al. "Managing Technical Debt in Software-reliant Systems". In: *Future of Software Engineering Research, FSE/SDP Workshop*. Sante Fe, NM, USA: ACM, 2010, pp. 47–52. ISBN: 978-1-4503-0427-6. DOI: 10.1145/1882362.1882373.

[5] Manfred Broy. "A logical approach to systems engineering artifacts. Semantic relationships and dependencies beyond traceability – From requirements to functional and architectural views". In: *Software Syst Model* 17.2 (2017), pp. 365–393. DOI: 10.1007/s10270-017-0619-4.

[6] Manfred Broy and Ketil Stølen. *Specification and Development of Interactive Systems*. New York, NY: Springer, 2001. ISBN: 9781461300915. DOI: 10.1007/978-1-4613-0091-5.

[7] Michele Colledanchise and Petter Ögren. *Behavior Trees in Robotics and AI. An Introduction*. Florida, US: Taylor & Francis, 2018, p. 192. ISBN: 9781138593732. DOI: 10.1201/9780429489105. arXiv: 1709.00084 [cs.RO].

[8] Christian Colombo et al. "Hardness in Autonomous Systems. Towards a Compositional Framework across Design, Integration, Assurance, and Operation". In: *Phil Trans A* (2026): *Engineering Reliable Autonomous Systems*, pp. 1–19. To appear.

[9] Clare Dallat, Paul M. Salmon, and Natassia Goode. "Identifying risks and emergent risks across sociotechnical systems: the NETworked hazard analysis and risk management system (NET-HARMS)". In: *Theor. Issues Ergon. Sci.* 19.4 (2017), pp. 456–482. DOI: 10.1080/1463922x.2017.1381197.

[10] Klaus Ehrlenspiel and Harald Meerkamm. *Integrierte Produktentwicklung. Denkabläufe, Methodeneinsatz, Zusammenarbeit*. 5th. München: Hanser, 2013. ISBN: 9783446435483. DOI: 10.3139/9783446436275.

[11] Steven D. Eppinger and Tyson R. Browning. *Design Structure Matrix Methods and Applications*. The MIT Press, 2012. ISBN: 9780262301428. DOI: 10.7551/mitpress/8896.001.0001.

[12] Gisela A. Garza Morales, Kostas Nizamis, and G. Maarten Bonnema. "Engineering complexity beyond the surface: discerning the viewpoints, the drivers, and the challenges". In: *Res. Eng. Des.* 34.3 (2023), pp. 367–400. ISSN: 1435-6066. DOI: 10.1007/s00163-023-00411-9.

[13] Mario Gleirscher et al. "Introduction of Static Quality Analysis in Small and Medium-Sized Software Enterprises: Experiences from Technology Transfer". In: *Software Qual. J.* 22.3 (Sept. 2014), pp. 499–542. DOI: 10.1007/s11219-013-9217-z.

[14] Anthony Hennig, Taylan G. Topcu, and Zoe Szajnfarber. "So You Think Your System Is Complex?: Why and How Existing Complexity Measures Rarely Agree". In: *J. Mech. Design* 144.4 (2021). ISSN: 1528-9001. DOI: 10.1115/1.4052701.

[15] George E. Heyliger. "Coupling". In: *Encyclopedia of Software Engineering*. Wiley, 2002. ISBN: 9780471028956. DOI: 10.1002/0471028959.sof068.

[16] Shuo Jiang, Min Xie, and Jianxi Luo. "Large language models for design structure matrix sequencing as combinatorial optimization". In: *Res. Eng. Des.* 37.4 (2026). ISSN: 1435-6066. DOI: 10.1007/s00163-026-00494-0.

[17] Rick Kazman. "Assessing architectural complexity". In: *Software Maintenance and Reengineering, 2nd Euromicro Conf.* IEEE CS, 1998, pp. 104–112. DOI: 10.1109/csmr.1998.665762.

[18] Alexander Kossiakoff et al. *Systems Engineering. Principles and Practice*. 2nd. Hoboken, NJ: Wiley, 2011. ISBN: 9781118001011. DOI: 10.1002/9781118001028.

[19] M. M. Lehman. "Software Process Technology". In: Berlin Heidelberg: Springer, 1996. Chap. Laws of software evolution revisited, pp. 108–124. ISBN: 9783540706762. DOI: 10.1007/bfb0017737.

[20] Valentina Lenarduzzi et al. "A systematic literature review on Technical Debt prioritization: Strategies, processes, factors, and tools". In: *J. Syst. Software* 171 (2021), p. 110827. ISSN: 0164-1212. DOI: 10.1016/j.jss.2020.110827.

[21] Udo Lindemann, Maik Maurer, and Thomas Braun. *Structural Complexity Management. An Approach for the Field of Product Design*. Berlin, Heidelberg: Springer, 2009. ISBN: 9783540878896. DOI: 10.1007/978-3-540-87889-6.

[22] Manal Linjawi. "Classifying the Capabilities of Robotic Systems. What is a robot?" PhD thesis. U Sheffield, 2020. URL: https://etheses.whiterose.ac.uk/27542/1/MLinjawi_thesis.pdf.

[23] J. E. Matson, B. E. Barrett, and J. M. Mellichamp. "Software development cost estimation using function points". In: *IEEE Trans. Softw. Eng.* 20.4 (1994), pp. 275–287. ISSN: 0098-5589. DOI: 10.1109/32.277575.

[24] Steve McConnell. *Software Estimation. Demystifying the Black Art*. 2nd ed. Sebastopol, CA: Microsoft / O'Reilly, 2009, p. 212. ISBN: 9780735635135.

[25] Luigi Nardi, David Koeplinger, and Kunle Olukotun. "Practical Design Space Exploration". In: *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), IEEE 27th Int Symp*. IEEE, 2019, pp. 347–358. DOI: 10.1109/mascots.2019.00045.

[26] Jason M. O'Kane and Steven M. LaValle. "Comparing the Power of Robots". In: *Int. J. Rob. Res.* 27.1 (2008), pp. 5–23. ISSN: 1741-3176. DOI: 10.1177/0278364907082096.

[27] Xavier Ouvrard. "Hypergraphs: an introduction and review". In: *CoRR* (2021). DOI: 10.48550/arXiv.2002.05014. arXiv: 2002.05014.

[28] David Lorge Parnas. "On the Criteria To Be Used in Decomposing Systems into Modules". In: *Commun. ACM* 15.12 (1972), pp. 1053–1058. DOI: 10.1007/978-3-642-48354-7_20.

[29] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. *Software Product Line Engineering. Foundations, Principles and Techniques*. Springer, 2010. ISBN: 9783642063640. DOI: 10.1007/3-540-28901-1.

[30] On-Road Automated Driving (ORAD) committee. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. Surface Vehicle Recommended Practice*. Standard J3016s-2018-06. SAE International, 2018. URL: https://www.sae.org/standards/content/j3016_201806/preview/.

[31] V. Rupanov et al. "Early safety evaluation of design decisions in E/E architecture according to ISO 26262". In: *ISARCS*. 2012, pp. 1–10. DOI: 10.1145/2304656.2304658.

[32] Kaushik Sinha and Eun Suk Suh. "Pareto-optimization of complex system architecture for structural complexity and modularity". In: *Res. Eng. Des.* 29.1 (2017), pp. 123–141. ISSN: 1435-6066. DOI: 10.1007/s00163-017-0260-9.

[33] Donald V. Steward. "The design structure system: A method for managing the design of complex systems". In: *IEEE Trans. Eng. Manage.* EM-28.3 (1981), pp. 71–74. ISSN: 1558-0040. DOI: 10.1109/tem.1981.6448589.

[34] C.R. Symons. "Function point analysis: difficulties and improvements". In: *IEEE Trans. Softw. Eng.* 14.1 (1988), pp. 2–11. ISSN: 2326-3881. DOI: 10.1109/32.4618.

[35] David D. Walden et al. *Systems Engineering Handbook. A Guide for System Life Cycle Processes and Activities*. 5th ed. Hoboken, NJ: Wiley, 2023. ISBN: 978-1-119-81429-0.

[36] Jules White et al. "Evolving feature model configurations in software product lines". In: *J. Syst. Software* 87 (2014), pp. 119–136. ISSN: 0164-1212. DOI: 10.1016/j.jss.2013.10.010.

[37] Saizheng Zhang et al. "Architectural Complexity Measures of Recurrent Neural Networks". In: *Advances in NeurIPS*. Ed. by D. Lee et al. Vol. 29. Curran, 2016. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/860320be12a1c050cd7731794e231bd3-Paper.pdf.