

An AI-Driven Graph-Based Architecture for Variant-Intensive BOM and Configuration Management in Passenger Seating Systems

Shuvdeep Bhattacharya

IT Director - Seating Engineering, Lear Corporation, 21577 Telegraph Road, Southfield MI 48033

Abstract Seat programs carry more configuration variability than almost any other vehicle subsystem, yet much of it never reaches the product structure. Color and trim content stays in illustration callouts, customer color-and-trim reports, and engineering-change records, so engineers read those documents, create or reuse an internal trim-color code, and hand the plant a spreadsheet, because product lifecycle management systems do not structure seat assemblies by color and plants cannot consume the engineering view directly. This paper proposes a graph architecture for that gap. One authoritative 150% engineering structure is held per seat family as a typed graph in which part usages, surface zones, color codes, features, plants, and revisions are separate nodes carrying bitemporal effectivity and provenance. Deriving a 100% engineering and manufacturing bill of material then becomes a resolution problem: deterministic predicates handle inclusion, exclusion, and cardinality; semantic closure carries implications a hierarchy cannot express; and a learned model scores the closed result and routes implausible configurations to engineering review with feature-level explanations. A versioned, plant-parameterized operator produces the manufacturing view. The paper contributes a seat-domain ontology, a resolution algorithm, and a measurable evaluation framework; validation is defined as a future instrumented pilot.

Keywords Passenger seating systems, Bill of materials, Configuration management, EBOM-MBOM transformation, Knowledge graph, Variant management

1. Introduction

A vehicle seat has to satisfy packaging, comfort, safety, feature, regional-market, and manufacturing requirements at once. Within one family, trim style, surface material, stitching, piping, hard- plastic color, comfort hardware, electronics, market content, and plant build strategy interact instead of varying independently, so a single family yields many buildable variants of which only a small subset is valid for a given vehicle line, model year, and plant. The theoretical end-product count is too large to hold as separate bills of material (BOMs), which is why mass-customization environments keep a generic superset structure [1], [5], and why variety is itself a cost driver [2]. Seat engineering has grown more quantitative over three decades [3], [4], but product- structure management has not kept pace.

In production programs observed by the author at a Tier 1 seating supplier, assigning color to a part means moving between interior callout documents, engineering-change tabs, and PLM to establish which material applies where, then

recording the answer in a form PLM will accept. A second workflow consolidates several customer color definitions from the color-and-trim report (CTR) into one internal trim-color code and rebuilds the result as a plant-facing spreadsheet, because the plant cannot read the engineering representation. Both depend on a few trim engineers who know each customer's conventions. The architecture proposed here holds one authoritative 150% engineering structure per seat family as a graph, with deterministic rules carrying documented constraints and machine learning recovering constraints never written down.

A. Contributions

The primitives used here are established: generic or 150% superset structures, predicate-based derivation, and ontology-linked configuration. No novelty is claimed for them. Three contributions are claimed. First, a seat-domain ontology in which SurfaceZone, ColorCode, MaterialFamily, and TrimStyle are constraining entities rather than part attributes, which makes color and trim logic computable. Second, treatment of document-sourced configuration semantics as a problem class of its own: configurator and product- line traditions assume configuration knowledge is authored inside the enterprise, whereas in seating much of it arrives from customers as documents that are late and

* Corresponding author:

shuvdeep@umich.edu (Shuvdeep Bhattacharya)

Received: Sep. 4, 2026; Accepted: Sep. 25, 2026; Published: Sep. 29, 2026

Published online at <http://journal.sapub.org/jmea>

repeatedly revised. What is claimed is the handling of extracted content as governing logic, carrying a confidence value and a provenance path. Third, a bitemporal view of that knowledge, with a resolution algorithm that keeps deterministic validation separate from learned assessment.

B. Scope and Assumptions

Failure modes were identified from direct engineering and information-technology experience with production seating programs at a Tier 1 supplier, covering color-part configuration, EBOM-to-MBOM transformation, configurator variant-rule behavior, and change-impact analysis. Program, customer, supplier, and system identities are withheld, and no proprietary data or part numbers are included. No deployment results are reported. Five assumptions bound the work. A1: one authoritative 150% structure can be maintained per seat family. A2: enough released configurations exist to train a plausibility model. A3: source systems emit change events or can be polled. A4: CTR documents can be parsed to a recorded confidence level. A5: engineers retain release authority.

A2 is the weakest. A front-seat family may hold several thousand theoretically buildable variants but only a few hundred released over program life, and the feature vector described in Section 5 is high-dimensional against that sample, so an autoencoder would memorize it rather than learn the manifold. Early deployment should use distance-based and rule-consistency detection and defer reconstruction models until a multi-program training set exists; the governance posture in Table 4 does not depend on the detector.

2. Background and Related Work

Five streams of published work bear on seat BOM and configuration management, and each leaves a different gap. High-variety BOM structure supplies the generic bill of material, a product-family model held once as a parameterized superset from which specific bills are generated [5], later extended to routing so that engineering, ordering, and operations planning share one representation [6]; carmaker practice shows how constraints reduce a large alternative space to a buildable subset [1], [2]. This is the foundation of the 150% BOM used here, but it addresses neither color and trim semantics nor many-to-one customer-color mapping. Configuration research establishes how predicates derive a specific product from a generic model [7]-[10], and product-line engineering treats the same problem through feature

models [11], [12].

Enterprise- and plant-level work connects configuration to BOMs and downstream plans [13], supplies the commonality-versus-variety framing [14], recovers generic BOM structure from what was built [15], derives engineering and manufacturing views from one source [16], formalizes transformation for production planning [18], and retargets an EBOM to a maintenance BOM [17]. Graph-based lifecycle integration connects design, manufacturing, and quality artifacts through persistent identifiers [19], [21], where what is promoted to an addressable entity determines what can later be reasoned about [20], [24]; knowledge graphs have been applied to configuration [22], [23], and a graph BOM model couples product development with production planning [25]. No published work known to the author applies these primitives to a commodity in which trim and color drive variety.

A. Capability Gap Versus Deployment Gap

In the reference environment, saved variant rules can express a 100% BOM derived from a released 150% BOM, but when the 150% structure is revised there is no reliable way to identify every saved rule affected. That is an observation about one deployment, not evidence against the technology class. The narrower claim made here is that three capabilities were not available together: color and trim semantics as first-class constraints, configuration knowledge extracted from external documents with recorded confidence, and bitemporal state separating when a change applied from when the enterprise learned about it.

3. Problem Definition

A seat variant combines several option spaces, each with its own structure and constraints. Table 1 multiplies the main dimensions of a representative front-seat family before left/right position, model year, or plant destination are considered.

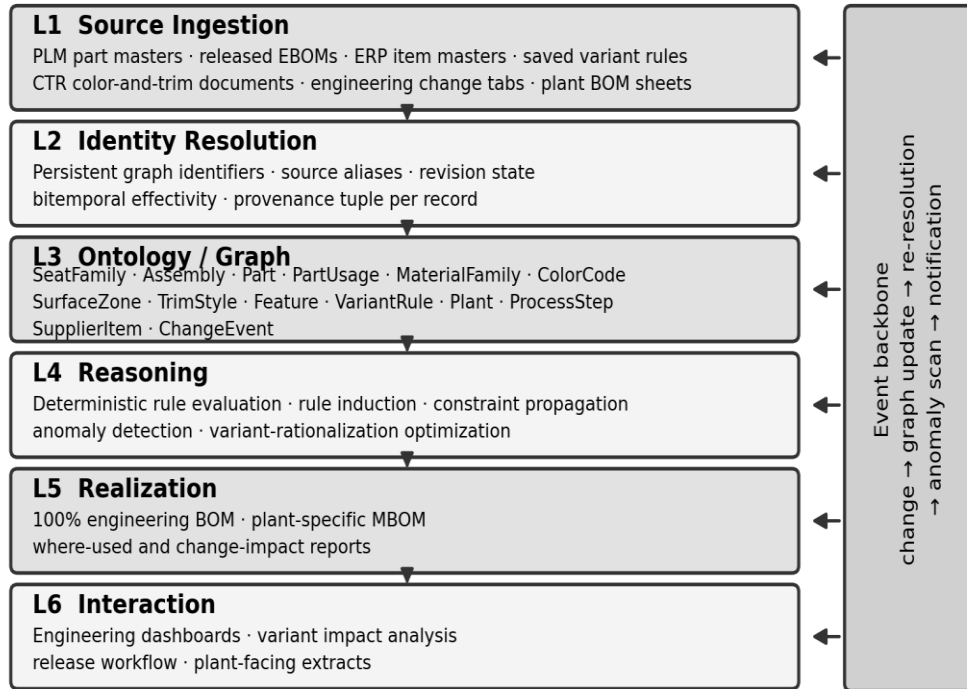
Customer-visible choices are weak predictors of the engineering content beneath them. One internal trim-color code may stand for several customer color values across six seat areas, and whether two zones share a code depends on piping, stitching, and whether the theme is single- or multi-tone. The real problem is the constrained mapping from commercial language and supplier documents to engineering parts, and from there to plant-consumable content. Table 2 lists the recurring failure modes observed and the architectural response to each.

Table 1. Representative option dimensions for a front-seat family

Option dimension	Cardinality	Cumulative combinations	Primary visibility
Trim style	4	4	Customer-visible
Visible color theme	8	32	Customer-visible
Substrate / foam variant	3	96	Hidden (engineering)
Feature package	6	576	Customer-visible
Regional package	3	1,728	Regulatory / market
Manufacturing alternative	2	3,456	Hidden (build method)

Table 2. Current-state failure modes and their architectural response

#	Failure mode	Observed manifestation	Architectural response
F1	Data fragmentation	CTR documents, change tabs, illustrations, part masters, and plant BOMs sit in separate tools and file systems.	Layer L1-L2 ingestion with persistent identity and provenance
F2	Semantic inconsistency	Customer color label, internal trim-color code, and plant-consumable color label overlap in meaning without being identical.	Ontology with distinct ColorCode, SurfaceZone, MaterialFamily, and TrimStyle nodes
F3	Weak propagation	A change to one component in the 150% BOM does not identify the 100% variants it affects.	Graph traversal over USES / USED_IN and REVISED_TO edges
F4	Representational mismatch	PLM does not structure seat assemblies by color, which blocks extraction of a plant-usable color BOM.	Color modeled as a first-class semantic node, not an attribute
F5	Manual EBOM-MBOM transformation	Spreadsheet-mediated part and master transformation, with ERP maintained by hand.	Versioned transformation operator T with plant templates
F6	Revision latency and document churn	Externally supplied CTR documents accumulate hundreds of revisions, each requiring manual request, validation, storage, and redistribution.	Bitemporal versioning and event-driven re-resolution
F7	Portfolio complexity	Trim, color, and plant combinations accumulate across program life without a mechanism for retiring those that no longer earn their verification cost.	Constrained rationalization over retained bundles and unique parts

**Figure 1.** Six-layer reference architecture. Sources are ingested and resolved to persistent identities, held in the seat-domain ontology of Table 3, reasoned over by rules and AI together, and materialized as engineering and manufacturing output

These are data-engineering problems, and stricter process discipline alone has not produced durable improvement. The data model has to reconcile conflicting identifiers and states, analytics have to catch invalid combinations before material reaches the plant, and changes have to propagate through variant space at bounded cost. Saved variant rules do not propagate.

4. Proposed Architecture

A. Design Principles and Layering

The architecture keeps one authoritative 150% seat

structure per product family and represents entities, constraints, and events directly as a graph, following the generic BOM and configuration literature [1], [5], [7], [13] and graph-based digital-thread work [19], [21], [25]. Layer L1 in Figure 1 ingests structured sources such as PLM part masters, released EBOMs, and saved variant rules, together with the semi-structured sources carrying most of the color logic. Layer L2 assigns each record a persistent graph identifier and a provenance tuple holding source system, source object identifier, source revision, ingestion timestamp, confidence, and validation flags. Versioning is bitemporal [27]–[29]: valid time records when content applies, transaction time when the platform learned it. External documents often

arrive after the change they describe, so valid time alone cannot answer what the platform believed on a given date.

B. Ontology and Graph Schema

A seat BOM ontology has to carry more than parent-child assembly. PartUsage is modeled as a node rather than an edge property because seat variants need instance-level attributes: quantity, side, stitch pattern, color treatment, effectivity, and plant- specific substitution. Making PartUsage addressable keeps variant filtering manageable, following the principle that only entities made explicit can later be reasoned over [20], [24]. Table 3 gives the canonical schema from which Figure 1 is drawn. PartUsage is the pivot at

which structural, configuration-semantic, and operational relations meet, and the lifecycle edges force re- resolution when a change event commits.

Consider a trim-cover assembly whose substrate is shared across several visible color variants while the foam adhesive changes in one region and the sewing thread in another. In the graph the trim-color code, regional package, substrate material, and thread choice are separate nodes joined by typed constraints, and a valid 100% BOM is the subgraph produced by a feature assignment and an effectivity window. In a hierarchy the same dependencies have to be duplicated across every affected branch or moved into spreadsheets, where nothing validates them.

Table 3. Canonical node and edge specification of the seat configuration graph

Element	Class	Role	Element	Class	Role
Node classes	Node classes	Node classes	PART_OF	Structural	Assembly to seat family membership
SeatFamily	Structure	Product family root; owns one authoritative 150% structure	ALTERNATIVE_TO	Structural	Interchangeable part usages within one context
Assembly	Structure	Structural assembly within a seat family	REALIZED_AS	Structural	Part usage to the released part that realizes it
Part	Structure	Released engineering part, independent of usage context	HAS_COLOR_CODE	Semantic	Part usage to its governed internal color code
PartUsage	Structure	Instance of a part in an assembly context; the pivot at which structural, semantic, and operational relations meet	APPLIES_TO_ZONE	Semantic	Part usage to the surface zone it covers
MaterialFamily	Semantic	Grouping of surface materials sharing processing and approval characteristics	HAS_MATERIAL_FAMILY	Semantic	Part to its material family
ColorCode	Semantic	Governed internal trim-color code; target of many-to-one customer color mapping	REPRESENTS_FEATURE	Semantic	Feature or trim style to the variant rule expressing it
SurfaceZone	Semantic	Named addressable seat region to which a color or material assignment applies	CONSTRAINED_BY	Semantic	Part usage to the variant rule governing its inclusion
TrimStyle	Semantic	Commercial trim level expressed as a constraining entity	REVISED_TO	Lifecycle	Revision chain between successive states of one entity
Feature	Semantic	Customer- or market-selectable option	SUPERSEDED_BY	Lifecycle	Replacement of an entity by a different one
VariantRule	Semantic	Predicate binding features and trim styles to admissible part usages	EFFECTIVE_IN	Lifecycle	Binding of an entity to a valid-time effectivity window
Plant	Operational	Manufacturing location with its own templates and process approvals	TRIGGERED_BY_CHANGE	Lifecycle	Affected entity to the change event requiring its re-resolution
ProcessStep	Operational	Plant process operation consuming or producing part content	BUILT_AT	Operational	Part usage to the plant that builds it
SupplierItem	Operational	Sourced item mapped to an engineering part	SOURCED_FROM	Operational	Part to its supplier item
ChangeEvent	Lifecycle	Engineering or document change requiring re-resolution	REQUIRED_BY_PROCESS	Operational	Part usage to the process step that consumes it
Edge types	Edge types	Edge types	CONSUMED_IN_PLANT_BOM	Operational	Part usage to its realization in a plant manufacturing BOM
USES, USED_IN	Structural	Assembly decomposition and where-used traversal; USED_IN is the declared inverse of USES			

Table 4. Analytics methods, target failure modes, and governance posture

Method	Failure modes	Technique	Governance posture
Deterministic rule evaluation	F3, F4	Predicate resolution over features, plant, and effectivity (Equation 1)	Authoritative; human-authored
Rule induction	F2	Weakly supervised mining of co-occurrence between callouts, trim codes, and CTR entries, after [15]	Proposes candidate rules; engineer validates and promotes
Constraint propagation	F3	Fixpoint closure over semantic implication edges (Equation 2)	Deterministic given the rule set
Anomaly detection	F2	Distance-based or one-class detection on graph and tabular features; reconstruction models once data volume permits	Advisory; has to emit a feature-level explanation
Event-driven re-resolution	F1, F6	Bitemporal diff on commit; resolution re-run for impacted variants only	Deterministic; triggered, not discretionary
Variant rationalization	F7	Constrained optimization over retained bundles and unique parts (Equations 3, 4)	Decision support for product planning

5. AI and Analytics Methods

Letting a model generate configurations directly is not defensible for a product structure carrying safety and quality obligations. Deterministic rules stay authoritative for mandatory inclusions, exclusions, and cardinality, and AI supports them. Table 4 identifies the analytics applied above the structural responses of Table 2, the failure modes they address, and the authority each is allowed.

A. Hybrid Derivation and Constraint Propagation

Let F be the selected features and options, C the hard constraints, P the part usages in the 150% BOM, t the effectivity instant, and $plant$ the target location. Deterministic evaluation produces the candidate set

$$B^*0 = \{ p \in P : \text{predicate}(p, F, C, t, \text{plant}) \} \quad (1)$$

Equation 1 filters P and cannot introduce part usages outside that set, but seat configuration often depends on implications running the other way: a premium perforated substrate in one zone may require a thread family and a side-shield color mapping that no predicate over F names. Those relationships are the semantic edges of Table 3. With I the active implication set, each implication written q implies p , the resolved set is the least fixpoint

$$B^* = \mu X. B^*0 \cup \{ p : \exists q \in X, (q \Rightarrow p) \in I \} \quad (2)$$

so B^* holds B^*0 plus every part usage reachable through active implications, and B^* is what gets scored and materialized. Equation 2 is monotone and the closure step does not check C , so if q implies p and q is in B^*0 , then p enters B^* even where the predicate would have excluded it for exclusivity, cardinality, or effectivity. The risk is not incompleteness but that B^* may violate the constraints that defined B^*0 .

The architecture therefore re-validates B^* against C after closure and treats any post-closure violation as a hard stop rather than a low score. Let $V = \text{violations}(B^*, C)$. If V is non-empty the configuration is not materialized under any plausibility score; it returns as a review package carrying the implication paths responsible. A hard-constraint violation and a low plausibility score are different findings: one

breaches authored logic whatever the history contains, the other is a statistical judgment calibrated against what has been built. Algorithm 1 keeps the two separate, and assumption A5 depends on it.

The AI layer then scores B^* for plausibility and completeness against released configurations, manufacturing output, and source- document implications. Let $\sigma(B^*)$ be that score on $[0,1]$, where 1 means indistinguishable from accepted practice; where the detector produces a reconstruction error or outlier distance e , σ is a fixed monotone decreasing normalization of e . Below threshold τ the system raises an explainable alert instead of emitting an unchecked BOM. τ is governed by review capacity rather than model tuning. Graph reasoning adds transitive impact analysis, propagating a color-code change to trim-cover part usages, plant MBOM rows, and supplier package references, and cross-domain conflict detection, which catches inconsistencies no single system sees.

B. Anomaly Detection and Variant Rationalization

Anomaly detection is applied here to product-structure and configuration data rather than to machine telemetry; the literature reviewed above showed no prior work applying outlier detection directly to released BOM or configuration structures, which is recorded as a gap rather than an exhaustive claim. Let xv be a feature vector for variant v built from selected features, resolved part families, commodity-level count distributions, color-zone signatures, and plant assignment. A distance-based or one-class detector trained on accepted variants flags configurations outside the accepted region, with specific alert classes: missing mandatory usages, duplicated mutually exclusive parts, impossible color-zone pairings, and cost jumps absent from neighboring variants. Training variants came from the manual process this architecture is meant to replace, a circularity that restricting training to never-corrected configurations reduces but does not remove.

Portfolio complexity is a separate question from single-variant validation. The optimization layer treats variant rationalization as a constrained choice over retained feature bundles and unique parts. Let $y_i = 1$ when bundle i is retained

and $z_j = 1$ when part j is retained; let c_j be the unit cost attributable to part j , b_i the change burden of bundle i , and v_i its verification load. The objective is

$$\min \alpha \sum_j c_j z_j + \sum_i (\beta b_i + \gamma v_i) y_i \quad (3)$$

subject to constraints that give it meaning, with M the market segments to serve, $I(m)$ the bundles covering segment m , $U(i)$ the parts consumed by bundle i , X the incompatible bundle pairs, w_i the customer value of bundle i , and W the minimum portfolio value accepted by product planning:

$$\begin{aligned} \sum_{i \in I(m)} y_i &\geq 1 \quad \forall m \in M; \quad z_j \geq y_i \quad \forall i, \forall j \in U(i) \\ y_i + y_k &\leq 1 \quad \forall (i, k) \in X; \quad \sum_i w_i y_i \geq W \\ y_i, z_j &\in \{0, 1\} \end{aligned} \quad (4)$$

Coverage prevents the trivial solution, since all coefficients are non-negative; the linking constraint keeps a part whenever a retained bundle uses it. The formulation is NP-hard in general, so it belongs in offline decision support rather than inside Algorithm 1. Where coverage and incompatibility conflict, returning the minimal infeasible subsystem to product planning exposes the binding segment instead of quietly relaxing W .

6. Deriving 100% BOMs from the 150% Structure

Figure 2 shows the seven-stage resolution workflow, and Algorithm 1 states it.

Algorithm 1 extends the saved-variant-rule mechanism of the reference configurator with post-closure re-validation, lifecycle-aware scoring, explainable validation, and synchronized engineering and manufacturing output. Stage S4 dominates cost. The fixpoint in Equation 2 uses a worklist that marks each part usage on first entry, so each element of B^* expands at most once and each outgoing implication edge is examined at most once. With d the maximum out-degree over semantic edges,

$$\text{Cost}(S4) = O(|B^*| \cdot d) \quad (5)$$

A naive path enumeration would be $O(|B^*| \cdot d^k)$ for traversal depth k because it revisits elements already reached. Marking removes depth from the bound: the traversal is linear in the reachable implication subgraph, so the size of the reachable set, not the depth needed to reach it, decides whether S4 is inexpensive. A pilot should instrument the distribution of $|B^*|$ relative to $|B^*0|$ and report k separately.

A. Producing the Manufacturing Graph

Producing the manufacturing graph is more than a flattening step. Plants need top-level assemblies, process phantoms, packaging, labels, line-side sequence kits, color-consumable rows, and supplier mappings that do not exist in the engineering structure [16]-[18]. The operator T applies plant templates, phantom-assembly rules, procurement views, routing links, and color interpretation to the resolved engineering subgraph. Where no engineering-side color part exists, current practice substitutes a customer part number in a spreadsheet and the decision becomes hard to trace; here it is a governed node with provenance. Recipes are versioned templates linked to plants, so an engineering revision recomputes only the affected MBOM fragments.

```
ResolveVariant(F, plant, t):
S1 G <- active 150% graph at time t
S2 X <- configuration context from F,
    plant, model year, region
S3 B*0 <- hard rules, effectivity,
    exclusivity, cardinality over X
S4 B* <- fixpoint closure of B*0 under
    active semantic implications I
S5 V <- violations(B*, C) # hard-constraint
    re-validation after closure
s <- sigma(B*)
if V is not empty: # authored-rule breach
S6 return ReviewPackage(B*, V,
    implication paths that
    introduced each violation)
elif s < tau: # low plausibility
S6 return ReviewPackage(B*,
    explanation paths,
    affected variants)
else:
S7 return Materialize(
    EBOM_100 = B*,
    MBOM_100 = T(B*, plant))
```

Algorithm 1. ResolveVariant

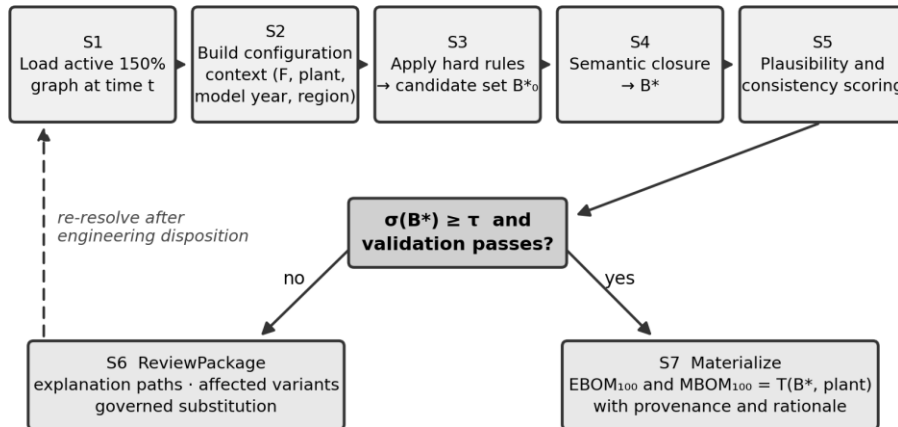


Figure 2. Seven-stage resolution workflow. B^* is scored before anything is materialized

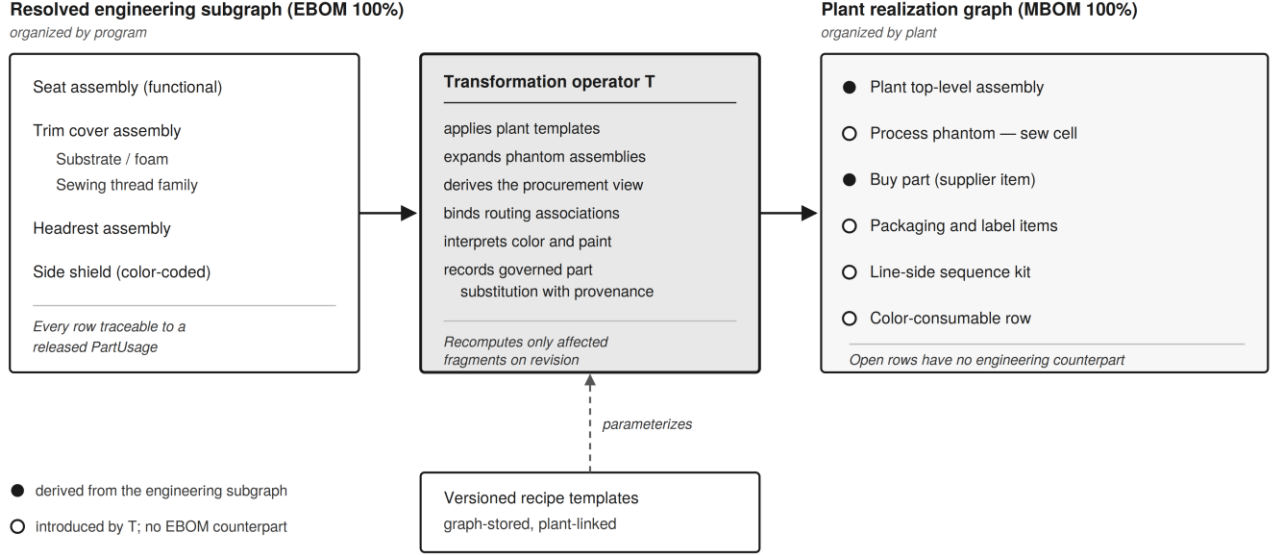


Figure 3. EBOM-to-MBOM transformation. T is parameterized by versioned, plant-linked recipe templates, which makes every structural difference between the program view and the plant view explicit and attributable

Table 5. Baseline versus proposed handling of representative seat configuration tasks

Task	Baseline (current state)	Proposed architecture
Derive a plant BOM for trim style T2, theme cocoa / light linen, substrate variant S3 (premium perforated), feature package CP3, regional package NA, manufacturing alternative M1, plant P2	Callouts are looked up by hand in customer documents. Trim style and material are interpreted. An internal color code is created or reused. The result is inserted into PLM where PLM will take it, and plant-consumable spreadsheet rows are built by hand.	The resolver applies deterministic rules, closes the result over semantic implications to recover the hidden trim and plastic selections, validates against variants that have actually been built, and materializes a synchronized 100% EBOM and MBOM with provenance intact.
Assess the impact of an engineering change revising a perforated trim substrate and introducing a new thread family for one theme	Analysts search spreadsheets and change tabs. Configurator behavior in the reference environment offers no automated way to identify which saved variant rules a 150% BOM revision affected.	REVISED_TO and USED_IN traversal finds every variant context in which the substrate is reachable under current predicates. Anomaly detection checks thread and plant sewing compatibility and the missing side-shield color mapping. Passing variants re-materialize on their own; failing ones produce a review package with explanation paths.

Table 6. Evaluation framework for a future deployment study

Metric	Definition	Target and comparison
Engineering effort per release	Engineering labor hours per variant release, current versus proposed state	At least 40% reduction in median hours per release; two-sided paired test at the 5% level over $N \geq 30$ releases
Defect containment rate	BOM defects detected before release versus after release; containment rate is $\text{pre} / (\text{pre} + \text{post})$	Containment rate above 0.90, from a baseline measured over the preceding 12 months
Change propagation time	Mean elapsed time from approved engineering change to all impacted plant BOM outputs	From days to under one shift for changes with no review escalation
Configuration artifacts maintained	Number of maintained distinct configuration artifacts per product family	Reduced by at least 50% with market coverage preserved, verified against the coverage constraint of Equation 4
Provenance completeness	Share of released BOM rows carrying a complete provenance path from plant content back to a source document	Above 0.95 on the pilot family; no comparable baseline exists, so this is measured on the platform only
Validation-set compression	Unique validation-relevant configurations identified by the system versus the naive combinatorial count	At least an order of magnitude below the naive bound, with no validated configuration missed
Semantic closure cost	Size of the resolved set B^* relative to the deterministic candidate set B^*0 , with observed traversal depth k reported alongside it (Equations 2 and 5)	Both reported as distributions, not targets; $ B^* / B^*0 $ is the primary cost measure and k is descriptive of how deeply the populated ontology chains implications
CTR parse confidence	Parse confidence achieved on CTR documents at L1 ingestion	Reported as a distribution; the share below the human-review threshold sets the resolution-queue load

7. Application Scenario

Take the front-seat family of Table 1, built across 3 plant destinations. Trim covers, headrests, side shields, armrests, and cushions each carry color and material semantics the base PLM structure does not fully capture. Table 5 compares current and proposed handling of two representative tasks. In the proposed architecture, customer CTR rows, trim engineering outputs, PLM parts, and plant MBOM templates are all graph nodes; surface zones are named explicitly; and each customer color entry maps first to a semantic color entity and then to a governed internal color-code node.

8. Discussion

None of the following has been measured. Anomalies and validation failures would surface before material reaches the plant, which is where BOM defects are usually caught today. Cycle time is the most visible candidate for improvement: a color program that currently takes weeks to process manually would resolve against the graph in a single pass. One graph-derived structure also replaces numerous parallel spreadsheets and saved-variant artifacts. Table 6 defines the pilot metrics: a paired comparison on one seat family of N variant releases under current practice against N releases after cutover, evaluated by a two-sided paired test at the 5% level.

A. Limitations and Implementation Risks

The architecture is conceptual, so its benefit claims are architectural arguments rather than measured outcomes, and the pilot of Table 6 is what would settle them. Training data volume is small relative to the feature space, which is why rule- consistency detection comes first. Critical color and trim semantics still arrive in semi-structured external documents, so ingestion must be staged with confidence flags, a human-in-the- loop resolution queue, and reported parse confidence. Induced rules that go unreviewed merely relocate the ambiguity, so they need a promotion workflow of candidate, engineer validation, then controlled logic. Closure size grows with the reachable implication subgraph, so a depth cap is enforced at query time. Alignment across PLM, manufacturing, trim engineering, release, and the plants requires a canonical identity layer.

9. Conclusions

Seating is a demanding and underexamined domain for configuration and BOM research. Product structures have to connect customer- facing variety with hidden engineering and manufacturing dependencies, and failures concentrate in trim, color, and plant- specific realization. In the programs observed, color-structured BOM extraction, propagation of affected variants after a 150% revision, and automated EBOM-to-MBOM transformation all remain unresolved.

The proposed architecture combines a graph-native ontology, one authoritative 150% seat structure, deterministic reasoning,

AI- supported review, anomaly detection, and event-driven synchronization to derive explainable 100% engineering and manufacturing BOMs. Promoting PartUsage, SurfaceZone, and ColorCode to first-class nodes makes seat color and trim logic computable. Bitemporal versioning is standard [27]-[29] but becomes necessary when configuration semantics arrive through late external documents, because valid time alone cannot show what the enterprise believed when a release decision was made. AI should advise and explain rather than decide. The next step is an instrumented pilot on one seat family evaluated against Table 6.

ACKNOWLEDGEMENTS

The author thanks colleagues in seat engineering, trim engineering, manufacturing engineering, and product lifecycle management whose day-to-day practice framed the problem addressed here.

REFERENCES

- [1] C. Chatras, V. Giard, and M. Sali, "High variety impacts on bill of materials structure: carmakers case study," IFAC-PapersOnLine, vol. 48, no. 3, pp. 1067–1072, 2015.
- [2] H. ElMaraghy, G. Schuh, W. ElMaraghy, F. Piller, et al., "Product variety management," CIRP Annals, vol. 62, no. 2, pp. 629–652, 2013.
- [3] S. Park, Y. Lee, Y. Nahm, and J. Lee, "Seating physical characteristics and subjective comfort: design considerations," SAE Technical Paper 980653, 1998.
- [4] S. Kim, "Perceived comfort prediction by occupant package layout and vehicle seat engineering factors," in Advances in Human Factors of Transportation, AHFE International, 2024.
- [5] H. M. H. Hegge and J. C. Wortmann, "Generic bill-of-material: a new product model," International Journal of Production Economics, vol. 23, no. 1–3, pp. 117–128, 1991.
- [6] J. Jiao, M. Tseng, Q. Ma, and Y. Zou, "Generic bill-of-materials -and-operations for high-variety production management," Concurrent Engineering, vol. 8, no. 4, pp. 297–321, 2000.
- [7] L. L. Zhang, "Product configuration: a review of the state-of-the-art and future research," International Journal of Production Research, vol. 52, no. 21, pp. 6381–6398, 2014.
- [8] A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, Knowledge-Based Configuration: From Research to Business Cases, Waltham, MA: Morgan Kaufmann, 2014.
- [9] L. Hvam, N. H. Mortensen, and J. Riis, Product Customization, Berlin: Springer, 2008.
- [10] A. Trentin, E. Perin, and C. Forza, "Overcoming the customization-responsiveness squeeze by using product configurators: beyond anecdotal evidence," Computers in Industry, vol. 62, no. 3, pp. 260–268, 2011.
- [11] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, et al., "Feature-oriented domain analysis (FODA) feasibility study,"

- Tech. Rep. CMU/SEI-90-TR-021, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1990.
- [12] D. Benavides, S. Segura, and A. Ruiz-Cortés, “Automated analysis of feature models 20 years later: a literature review,” *Information Systems*, vol. 35, no. 6, pp. 615–636, 2010.
 - [13] P. T. Helo, Q. L. Xu, S. J. Kyllönen, and R. J. Jiao, “Integrated vehicle configuration system—connecting the domains of mass customization,” *Computers in Industry*, vol. 61, no. 1, pp. 44–52, 2010.
 - [14] J. Jiao, T. W. Simpson, and Z. Siddique, “Product family design and platform-based product development: a state-of-the-art review,” *Journal of Intelligent Manufacturing*, vol. 18, no. 1, pp. 5–29, 2007.
 - [15] C. J. Romanowski and R. Nagi, “A data mining approach to forming generic bills of materials in support of variant design activities,” *Journal of Computing and Information Science in Engineering*, vol. 4, no. 4, pp. 316–328, 2004.
 - [16] L. He, Y. Ni, X. Ming, M. Li, et al., “Integration of bill of materials with unified bill of materials model for commercial aircraft design to manufacturing,” *Concurrent Engineering*, vol. 22, no. 3, pp. 206–217, 2014.
 - [17] M. Liu, J. Lai, and W. Shen, “A method for transformation of engineering bill of materials to maintenance bill of materials,” *Robotics and Computer-Integrated Manufacturing*, vol. 30, no. 2, pp. 142–149, 2014.
 - [18] S. Wang, X. Liu, Z. Bai, and J. Xiao, “A BOM model transformation method for hierarchical production planning management process of complex products,” *Advanced Engineering Informatics*, vol. 58, 102138, 2023.
 - [19] T. D. Hedberg, Jr., M. Bajaj, and J. A. Camelio, “Using graphs to link data across the product lifecycle for enabling smart manufacturing digital threads,” *Journal of Computing and Information Science in Engineering*, vol. 20, no. 1, 011011, 2020.
 - [20] S. K. Chandrasegaran, K. Ramani, R. D. Sriram, I. Horváth, et al., “The evolution, challenges, and future of knowledge representation in product design systems,” *Computer-Aided Design*, vol. 45, no. 2, pp. 204–228, 2013.
 - [21] G. Buchgeher, D. Gabauer, J. Martinez-Gil, and L. Ehrlinger, “Knowledge graphs in manufacturing and production: a systematic literature review,” *IEEE Access*, vol. 9, pp. 55537–55554, 2021.
 - [22] K. Zhang, Z. Tu, D. Chu, and X. Lu, “AIC: an industrial knowledge graph with abstraction-instance-capability reasoning abilities for personalized customization,” *Journal of Intelligent Manufacturing*, vol. 35, no. 7, pp. 3419–3440, 2024.
 - [23] Y. Yang, M. Yang, S. Shangguan, and Y. Cao, “A novel method to build knowledge graph models for the configuration and operation design of smart and connected industrial products,” *Journal of Computational Design and Engineering*, vol. 11, no. 2, pp. 327–344, 2024.
 - [24] E. K. Abbasi, T. Leclercq, and P. Heymans, “A meta-model for product configuration ontologies,” in *Proc. 26th ACM International Systems and Software Product Line Conference (SPLC ’22)*, vol. B, pp. 166–173, 2022.
 - [25] X. Hu, A. Barwasser, A. Werner, F. Schuseil, et al., “Graph model based bill of material structure for coupling product development and production planning,” in *Intelligent and Transformative Production in Pandemic Times*, pp. 593–605, 2023.
 - [26] S. Franceschetto, C. Amico, M. Brambilla, and R. Cigolini, “Improving supply chain in the automotive industry with the right bill of material configuration,” *IEEE Engineering Management Review*, vol. 51, no. 1, pp. 214–237, 2023.
 - [27] R. T. Snodgrass, *Developing Time-Oriented Database Applications in SQL*, San Francisco, CA: Morgan Kaufmann, 2000.
 - [28] C. S. Jensen and R. T. Snodgrass, “Temporal data management,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 1, pp. 36–44, 1999.
 - [29] K. Kulkarni and J.-E. Michels, “Temporal features in SQL: 2011,” *ACM SIGMOD Record*, vol. 41, no. 3, pp. 34–43, 2012.