The X-Formats Family: Linking Geometry and Semantics for Product-Centric Engineering Data Management

Alessio Pacini[1,*], Francesco Lupi[1], Michele Lanzetta[1]
[1] *Department of Civil and Industrial Engineering, University of Pisa, 56122 Pisa, Italy*
*Corresponding author: alessio.pacini@phd.unipi.it

ABSTRACT

Machine-readable semantic information, such as simulation settings, inspection configurations, machine parameters, process outcomes/logs, is central to digital manufacturing, enabling automation, interoperability, data traceability, and adaptive reconfiguration across the product lifecycle. When anchored to specific geometric entities, these semantic structures form the backbone of Model-Based Definition (MBD), a data engineering architecture in which the CAD model acts as the single source of truth to realize the digital thread. Within this architecture, both rule-based and machine-learning pipelines can orchestrate simulation, optimization, and control tasks using shared, semantically enriched models. Despite this potential, most engineering file formats either lack expressive, machine-readable semantics or encode them in ways that are format-specific, difficult to adapt to different needs, and hard to reuse. This letter presents a manifesto for the eXtended Formats (X-Formats) family, a lightweight stand-off markup approach that links structured annotations to engineering files without modifying the original source. Semantics can be defined independently of the underlying syntax (e.g., STL, B-rep, STEP, and DXF for CAD; PNG, JPEG, GIF, and SVG for images; MP4 and MOV for videos; GeoTIFF for geospatial data) and are anchored to file-specific entities that can be detected by external feature-extraction mechanisms. X-Formats structure semantics in a three-level hierarchy composed of property, layer, and schema, which is fundamental for supporting modular dataset design and allowing rapid customization across domains by replacing only the geometry-descriptor block. This letter introduces the core architectural principles, anchoring strategy, supporting tooling, and broader implications of this novel, product-centric, approach to engineering data management.

***Keywords**: Model-Based Definition, Semantics, Interoperability, CAD, Digital Thread, Data Management.*

1. Introduction

Semantics provide the connective tissue of modern engineering data [1–3]. They convey intent, context, and constraints across the whole product lifecycle, clarifying design assumptions and supporting computer-based process parametrization. Multiple communities have attempted to encode semantics to meet their own needs.

In manufacturing, for example, Model-Based Definition (MBD) integrates geometry and semantics into a single authoritative artifact [1], capturing information relevant to inspection [4–6], assembly [7–9], simulation [10–12], and maintenance and end-of-life activities [13], promoting the development of exchange standards that support richer, human- and machine-readable content [2].

Despite substantial progress, most solutions remain format-bound and rely on in-line markup, where semantics are embedded directly within the native file syntax [1,14,15]. This tight coupling raises the barrier to adoption, complicates user customization (e.g., rigid file syntactic structures constrain the types of information that can be represented and hinder the reuse of domain knowledge across heterogeneous sources), and increases the risk of corrupting the original file when data are extended or edited.

In contrast, stand-off markup approaches store semantics in a sidecar file and link them to source artifacts via stable identifiers, preserving the original data, enabling more flexible enrichment, and allowing semantics and file content to evolve independently [1,14,15].

This letter introduces the eXtended Formats (X-Formats) family, a format-agnostic framework that generalizes stand-off markup across domains and file types. The key innovation is an anchoring strategy that binds semantics to externally detectable geometric features, such as geometric characteristics, rather than to internal file structures. As a result, the same semantic definitions can operate consistently across diverse engineering representations, including tessellated meshes, boundary-representation (B-rep) models, 2D drawings, and images.

Section 2 presents the design principles. Section 3 details the architecture, anchoring modes, and practical variants of the X-Formats framework. Section 4 discusses implications, limitations, and future work.

2. Design principles

The X-Formats family is guided by eight principles that emphasize neutrality, robustness, and practical adoption:

1. **Stand-off markup**. Semantics reside in a sidecar file and link to source artifacts, reducing corruption risks, avoiding vendor lock-in, and allowing users to revise annotations without regenerating geometry.
2. **Format-agnostic anchoring.** Entity references are resolved through functional or geometric descriptors (e.g., centroids, normals, axes, keypoints, polylines, Regions of Interest (ROIs), and timestamps) that can be computed externally and independently of the native syntax.

IP Notice. The X-Formats framework and related automation-oriented implementations are patent pending. © 2026 The Authors. This preprint is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

3. **JSON as the data container**. JavaScript Object Notation (JSON) [16] is both machine- and human-readable, machine-enforceable, and integrates well with modern pipelines and services.
4. **Unified semantics via a three-level hierarchy.** A single hierarchical structure is shared across the family: (i) *property*, the atomic semantic concept with type, input mode, constraints, and visualization hints; (ii) *layer*, a coherent group of *properties* for a specific process (e.g., manufacturing or inspection); and (iii) *schema*, a dataset profile that bundles *layers* and *properties* for a given use case, domain, or organization.
5. **Layered semantics for scalability and filtering**. Systems selectively load the subset of properties required for a specific task (e.g., inspection), reducing cognitive and computational load while maintaining a consistent global structure.
6. **Low-friction customization.** *Properties* declare applicability (entity types), per-type cardinalities, input modes (e.g., single value or selection), and data types (e.g., real, integer, text, Boolean, vector). These constraints support tailored adoption while reusing the same format.
7. **Tool neutrality and interoperability.** The X-Formats family minimizes dependencies on specific proprietary software or ecosystems.
8. **Security and data minimization.** Sidecar files store only semantic values and minimal anchoring descriptors. Sensitive geometry remains in the original, unmodified source file, while semantic interpretation and decoding require access to the corresponding layer-definition files.

3. The X-Formats family architecture

3.1. Overview

Each X-file comprises three JSON blocks:

a) **Metadata.** Captures provenance (authors, versions, timestamps), coordinate conventions, and references to the *layers* and *schemas* that serve as the "reading keys" required for interpretation.

b) **Geometry.** Exposes externally detectable entities and their descriptors to enable robust anchoring, independent of the source format's native syntax. It includes (i) a reference to the source file or an embedded payload and (ii) entity definitions. Each entity is assigned a unique, stable identifier and one or more descriptors used for matching. Entities are classified as automatically recognized or user-declared. Automatically recognized entities are derivable from the source file (e.g., faces, edges, curves, 2D primitives, and image regions of interest). The parameters and algorithms used for automatic recognition are also recorded in the X-file to support reproducible matching across runtimes. User-declared entities are specified interactively during annotation (e.g., inspection points, custom ROIs, and reference datums).

c) **Semantics**. Stores machine-readable annotations validated against the *layer* definitions referenced in the metadata. X-Formats support three anchoring modes: (i) entity-bound, which maps semantic annotations to specific entity identifiers; (ii) global, which applies to the entire file; and (iii) inter-annotation references, which link semantics to previously defined *properties*.

3.2. The concept of anchors across domains and variants

The proposed structure applies consistently across domains, enabling uniform enrichment of heterogeneous files. Effective anchoring requires domain-aware semantics and format-aware descriptor extraction.

In 3D computer-aided design (CAD), anchors derive from volumes, surfaces, edges, and points in tessellated meshes (e.g., STL [17]) or parametric models (e.g., STEP B-rep [18]). In 2D CAD (e.g., DXF [19]), anchors are extracted from the analytical description of primitives such as polylines. In imaging, anchors correspond to pixels aggregated into ROIs, while georeferenced imagery uses coordinates expressed in a spatial reference system. Point clouds extend tessellated geometry with sampled points and fitted primitives; video extends image-based anchoring with temporal identifiers such as timestamps.

Files within the same domain (e.g., 3D CAD models) share a common semantic *layer*; however, the computation of a given anchor (e.g., a Cartesian point in STL or STEP) remains file specific. Application fields span manufacturing, healthcare imaging, agriculture, infrastructure, and logistics, all benefiting from a unified semantic approach that reduces cognitive overhead and improves interoperability.

Table 1 summarizes typical anchors and X-Formats variants, and Figure 1 illustrates examples of semantic anchoring across domains.

Table 1. Illustrative application domains and anchoring mechanisms (all entries are non-exhaustive examples).

Domain	Base format(s)	Entity types	Anchor descriptors	X-variant(s)
3D CAD (tessellated)	STL, OBJ	Volume, surface, edge, point	Type, centroid, keypoints	STLX (Figure 1a), OBJX
3D CAD (B-rep)	STEP, IGES	Volume, face, edge, vertex	Type, centroid, keypoints	STEPX (Figure 1b), IGESX
2D CAD	DXF	Line, arc, polyline	Type, endpoints, vertex set	DXFX (Figure 1c)
Image	PNG, JPEG, DCM	ROI, contour, keypoints	ROI centroid, vertex set, contour perimeter and area	PNGX, JPEGX, DCMX (Figure 1e)
Video	MP4	Image descriptors for each frame	Timecode and image descriptors	MP4X (Figure 1d)
Georeferenced imagery and maps	GeoTIFF, ESRI Shapefile	ROI, polyline, polygon	Latitude, longitude, altitude; bounding box; path nodes	TIFX (Figure 1f)

© 2026 The Authors. **Preprint**. Not peer-reviewed. Submitted for publication. Licensed under CC BY-SA 4.0.

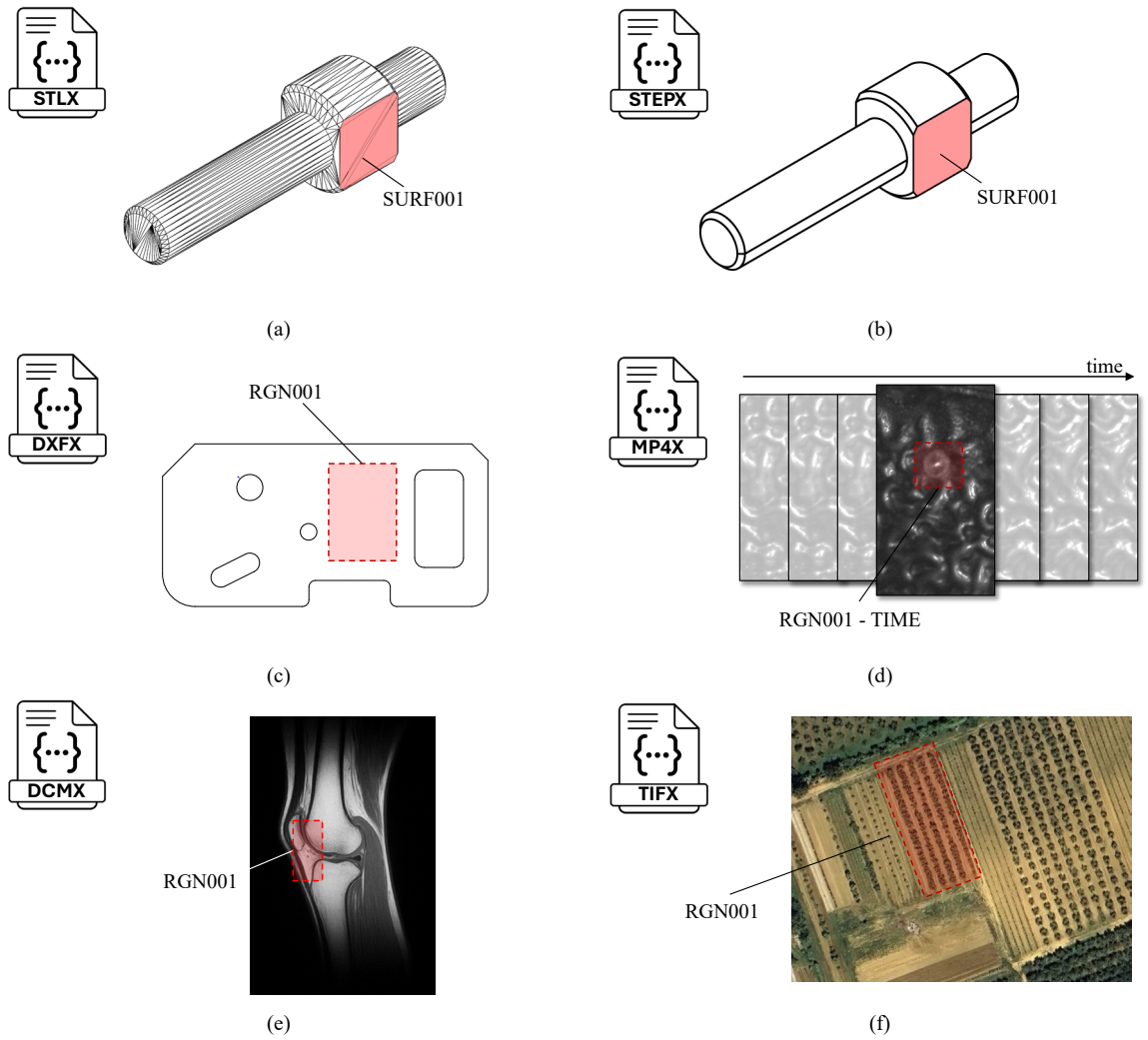

Figure 1. Cross-domain applications of the X-Formats family: (a, b) 3D manufacturing illustrated across different file types (tessellated versus B-rep); (c) 2D manufacturing; (d) image-based machine learning from a video sequence; (e) medical imaging; (f) agricultural imaging.

3.3. Supporting framework and software tools

The X-Formats ecosystem requires an underlying information infrastructure to support a comprehensive workflow that includes authoring, extraction, validation, exchange, re-anchoring, and governance. Figure 2 outlines the main workflow, while Table 2 lists the supporting functions (F1-F9).

Dataset designers author *properties* and *layers* (F1a) by declaring anchor scope, per-type cardinalities, input modes, data types, and optional visualization hints. The result is a semantic catalog (i.e., the *schema*) referenced by X-files (F1b).

A feature-extraction module (F2) identifies entities, computes their descriptors, and assigns stable identifiers. These automatically recognized entities are then consumed by the X-file generator (F3), which composes the geometry block, organizes it together with the metadata and semantics blocks, and validates their structure. Packaging and exchange modules (F4) handle filtering, encryption, and licensing.

Once created, interactive tools (F5) provide *schema*-aware editing; assisted definition of new user-declared entities or selection of entities already present in the geometry block; live validation against cardinalities and value domains; and consistent overlays with *layer*-based filtering. Assisted annotation mechanisms, powered by rules or machine-learning models, can further support the definition of suitable anchors and values.

X-files are consumed by downstream visualization tools (F6) and by software-specific interfaces (F7) that integrate X-Formats data with CAD, Computer-Aided Manufacturing (CAM), Geographic Information Systems (GIS), Product Lifecycle Management (PLM), Manufacturing Execution Systems (MES), Quality Management Systems (QMS), and analytics environments, enabling these systems to read or write X-Formats data. Certain automation-oriented implementation aspects, specifically those related to semantic-driven execution and orchestration, are currently covered by a pending patent application [20].

For change management, re-anchoring and semantic migration (F8) maintains stable bindings after geometry or semantic structure (e.g., *layers* or *schema*) updates, while semantic diff and merge (F9) computes semantic differences across multiple versions and reconciles concurrent edits into a consolidated master version.

© 2026 The Authors. **Preprint**. *Not peer-reviewed. Submitted for publication. Licensed under CC BY-SA 4.0.*

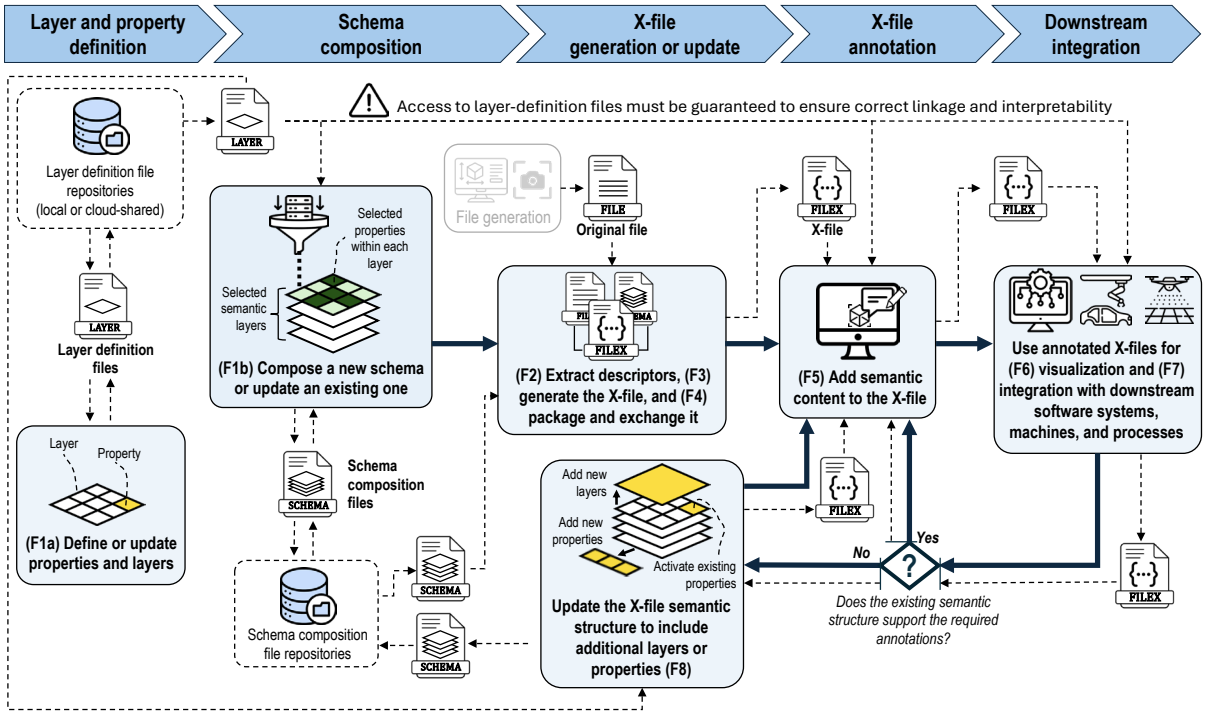

Figure 2. General workflow of the X-Formats family, showing (F1a) *layer* and *property* definition and (F1b) *schema* composition; (F2-F4) descriptor extraction, X-file generation, and exchange; (F5) X-file annotation; (F6-F7) downstream integration; and semantic update (F8). Solid arrows indicate activity flows, while dashed arrows indicate information flows. The original file-generation process (light gray box, e.g., 3D modeling, image acquisition) lies outside the X-Formats framework. For the sake of readability, only the "semantic update" branch of F8 is represented; original file updates and the diff-and-merge function F9 are omitted.

Table 2. Ecosystem functionalities (F1-F9) for X-Formats tooling.

Functionality	Purpose	Primary inputs	Primary outputs
F1: (a) *Property* and *layer* definition, (b) *schema* composition	Design datasets by defining *properties*, *layers*, and *schemas* (anchor scopes, per-type cardinalities, input modes, data types)	Domain requirements; controlled vocabulary; governance rules	*Property*, *layer*, and *schema* catalogs (JSON files)
F2: Descriptor extraction	Extract descriptors for automatically recognized entities and assign stable identifiers (IDs) and external anchors	Original file; entity detector algorithm settings	Stable entity IDs; descriptor sets
F3: X-file generation	Generate unified X-files by composing metadata, geometry, and semantic blocks, associating *schemas* and *layers*, instantiating entities, and validating structure	Original file; selected *schemas*/*layers*; descriptor extractors; registry references	Validated X-file linked to source artifact
F4: Packaging and exchange	Package and exchange X-files with interoperable distribution, encryption, access control, licensing, and role policies	X-files; security policies; license terms	Protected or encrypted X-files
F5: X-file annotation	Annotate X-files through *schema*-aware semantic editing with visual support, live conformance checks, and user-declared entities definition	X-file; *schema*; source artifact	Updated X-file
F6: X-file visualization	Visualize and review datasets with overlays and *layer*-based filtering	X-file; *schema*; source artifact	Interactive views
F7: API/SDK connectivity	Connect to downstream processes, software systems, and machines via Application Programming Interfaces (APIs) and Software Development Kits (SDKs)	X-files; application requirements	Machine-readable data; adapters/APIs
F8: Re-anchoring and semantics migration	Maintain stable semantics after changes in the original file or in the semantic structure (e.g., updated *layers* or *schemas*) via semantic remapping, geometry-block substitution, and *schema* upgrades	Original file (old and new version); X-file; semantic structure update details	Migrated X-file; remapping report
F9: Semantic diff and merge	Compute semantic differences between X-file versions and merge concurrent edits into a consolidated master X-file, including conflict detection and resolution.	Master X-file; derived X-file versions; merge rules	Semantic patches/diffs; master X-file; merge report

© 2026 The Authors. **Preprint**. *Not peer-reviewed. Submitted for publication. Licensed under CC BY-SA 4.0.*

4. Discussion and conclusion

The stand-off strategy at the core of the X-Formats family ensures persistent mappings between semantic information and file entities while avoiding rigid coupling to native syntax. This reduces the risk of corruption from inline edits, facilitates interoperability, and increases the reuse of domain knowledge across tools and representations. By externalizing semantics and anchoring them to externally detectable features, the same semantic definitions can operate across domains and formats. This decoupling supports vendor-neutral workflows and reduces integration effort in PLM, MES, QMS, and Artificial Intelligence (AI) training pipelines.

Layered semantics further streamline security, authoring and visualization. Selective loading reduces cognitive load; *schema*-aware editing and live validation prevent inconsistencies; and graphical overlays enable task-focused, consistent views. Cross-format reuse is achieved by reconfiguring only the applicability to the relevant entity types (e.g., “surface” in 3D versus “planar region” in 2D).

Two considerations are essential for robust deployment. First, anchoring robustness requires descriptor baselines that are both domain-specific and format-appropriate. Second, to enable reuse at scale, *properties* and *layers* should be curated in versioned registries enriched with examples and conformance tests, and governed by clear backward-compatibility policies.

The X-Formats family provides a practical foundation for interoperable, human- and machine-readable datasets across engineering and related domains. Ongoing work will strengthen adoption through standardized descriptor baselines, domain-specific benchmarks, and unified APIs, consolidating X-Formats as a neutral language for product-centric digital manufacturing.

Declaration of interest

The authors declare a pending patent application filed with the University of Pisa related to this work. This preprint is licensed under the CC BY-SA 4.0.

References

[1] A. Pacini, F. Lupi, M. Lanzetta, Semantically Enriched CAD Models for Digital Manufacturing: A Systematic Review of Model-Based Definition, *J. Intell. Manuf.* (2026). https://doi.org/10.1007/s10845-026-02794-7.

[2] K. Goher, E. Shehab, A. Al-Ashaab, Model-Based Definition and Enterprise: State-of-the-art and future trends, *Proc Inst Mech Eng B J Eng Manuf 235* (2021) 2288-2299. https://doi.org/10.1177/0954405420971087.

[3] A. Corallo, V. Del Vecchio, M. Lezzi, A. Luperto, Model-Based Enterprise Approach in the Product Lifecycle Management: State-of-the-Art and Future Research Directions, *Sustainability 14* (2022) 1370. https://doi.org/10.3390/SU14031370.

[4] F. Lupi, N. Freitas, M. Arvana, A.D. Rocha, A. Maffei, J. Barata, M. Lanzetta, Next-generation Vision Inspection Systems: a pipeline from 3D model to ReCo file, *J Intell Manuf* (2024). https://doi.org/10.1007/s10845-024-02456-6.

[5] T.D. Hedberg, M.E. Sharp, T.M.M. Maw, M.M. Helu, M.M. Rahman, S. Jadhav, J.J. Whicker, A. Barnard Feeney, Defining requirements for integrating information between design, manufacturing, and inspection, *Int J Prod Res 60* (2022) 3339-3359. https://doi.org/10.1080/00207543.2021.1920057.

[6] A. Pacini, F. Lupi, M. Lanzetta, Design of Reconfigurable Handling Systems for Visual Inspection, *J. Manuf. Mater. Process. 9* (2025) 257. https://doi.org/10.3390/JMMP9080257.

[7] S.K. Mohammed, M.H. Arbo, L. Tingelstad, Using semantic Geometric Dimensioning and Tolerancing (GD&T) information from STEP AP242 neutral exchange files for robotic applications, *Int. J. Interact. Des. Manuf. 18* (2024) 6587-6603. https://doi.org/10.1007/s12008-023-01242-7.

[8] S.K. Mohammed, M.H. Arbo, L. Tingelstad, Leveraging model based definition and STEP AP242 in task specification for robotic assembly, *Procedia CIRP 97* (2021) 92-97. https://doi.org/10.1016/j.procir.2020.05.209.

[9] N. Rea Minango, M. Hedlind, A. Maffei, Handling features in assembly: Integrating manufacturing considerations early in design discussions, *J Manuf Syst 77* (2024) 1077-1100. https://doi.org/10.1016/J.JMSY.2024.11.012.

[10] A.R. Aderiani, K. Wärmefjord, R. Söderberg, Model-based definition in computer aided tolerance analyses, *Procedia CIRP 114* (2022) 112-116. https://doi.org/10.1016/J.PROCIR.2022.10.016.

[11] S. Bala Murugan, R. Manu, K. D. Lawrence, STEP AP 242 file-based automatic tolerance analysis of mechanical assembly using unified Jacobian Torsor Model and direct linearization method, *Int J Comput Integr Manuf 36* (2023) 756-788. https://doi.org/10.1080/0951192X.2022.2145017.

[12] A. Pacini, N.R. Minango, F. Lupi, M. Lanzetta, A. Maffei, Digital thread in fixture design: leveraging model-based definition for seamless information flow, *Int J Comput Integr Manuf* (2025) 1-30. https://doi.org/10.1080/0951192X.2025.2558831.

[13] J. Geng, X. Tian, M. Bai, X. Jia, X. Liu, A design method for three-dimensional maintenance, repair and overhaul job card of complex products, *Comput Ind 65* (2014) 200-209. https://doi.org/10.1016/J.COMPIND.2013.08.008.

[14] J. Camba, M. Contero, M. Johnson, P. Company, Extended 3D annotations as a new mechanism to explicitly communicate geometric design intent and increase CAD model reusability, *Comput.-Aided Des. 57* (2014) 61-73. https://doi.org/10.1016/j.cad.2014.07.001.

[15] L. Ding, S. Liu, Markup in Engineering Design: A Discourse, Future Internet 2 (2010) 74-95. https://doi.org/10.3390/fi2010074.

[16] ECMA, ECMA-404, the JSON data interchange syntax, 2nd edition, Geneva, Switzerland, 2017. https://ecma-international.org/publications-and-standards/standards/ecma-404/ (accessed November 26, 2025).

[17] Adobe, STL files explained, Adobe Inc. (n.d.). https://www.adobe.com/creativecloud/file-types/image/vector/stl-file.html (accessed May 28, 2025).

[18] ISO, ISO 10303-242:2014, Industrial automation systems and integration - Product data representation and exchange, Part 242: Application protocol: Managed model-based 3D engineering, 2014.

[19] Autodesk Inc., AutoCAD DXF Archive, (n.d.). https://aps.autodesk.com/developer/overview/autocad-dxf-archive (accessed November 26, 2024).

[20] A. Pacini, F. Lupi, M. Lanzetta, Method for interacting with a physical object based on a digital CAD reproduction of the object, Italian Patent Application No. 102026000003784, filed Feb. 2026.

© 2026 The Authors. ***Preprint***. *Not peer-reviewed. Submitted for publication. Licensed under CC BY-SA 4.0.*