Computer-Aided Design 200 (2026) 104153

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Cross-solver conversion and canonicalized validation of 3D solid finite-element mesh data using a neutral hierarchical intermediate representation ☆

Peter Robert Grohmann *, Stephan Tremmel

Engineering Design and CAD, University of Bayreuth, Universitätsstr. 30, Bayreuth, 95447, Bavaria, Germany

ARTICLE INFO

Keywords:
Finite-element mesh conversion
Mesh-level structural interoperability
Intermediate representation
Canonicalization
Solver-deck conversion

ABSTRACT

Finite-element mesh transfer between solver decks is difficult to validate because formats differ in syntax, capabilities, hierarchy/placement, identifiers, and local node ordering. Existing routes rarely combine capability-aware hierarchy handling, deterministic identifier normalization, node-order mapping, and documented fidelity reporting. We evaluate 3D solid mesh exchange across Abaqus, ANSYS Mechanical coded database (CDB), MSC NASTRAN bulk data file (BDF), and Z88. We claim mesh-level structural interoperability within a declared subset, not full analysis-model, solver-semantic, or solver-result equivalence. We present a neutral hierarchical intermediate representation, format-specific readers/writers, solver-specific Tet4/Tet10 and Hex8/Hex20 node-order permutations to guard against orientation regressions, deterministic identifier normalization, and JSON artifacts. The subset covers nodal coordinates, solid-element connectivity, node- and element-set membership, selected representable hierarchy/placement metadata, and structurally tracked reference-point candidates; shell/beam elements, other element families, solver-level analysis entities and semantics, solver controls, and solver results are outside scope. We assess fidelity through a Z88-pivot-canonicalized route comparison — not a literal source-to-target-to-source deck cycle — using tolerance-quantized coordinate signatures, order-invariant element-incidence signatures, and capability-aware, channel-specific reporting. Across 23 models, 92 evaluated conversions completed without processing failures. Connected-node geometry and element-incidence signatures achieved $F_1 = 1.0$ under declared policies; set-membership signatures agreed in non-empty reference channels, and reference-point candidates were tracked or re-identified where present. Observed reductions in Name-F_1 reflected target naming and namespace constraints rather than loss of evaluated set membership. Three-stage validation-workflow runtimes, comprising reference canonicalization, source-to-target conversion, and converted-deck canonicalization, ranged from 0.05 to 408 s (median 1.42 s). Within this subset, structural evidence supports deterministic, auditable mesh exchange and mesh-level structural interoperability.

1. Introduction

Finite element analysis (FEA) is widely used in industrial simulation workflows. In collaborative and multidisciplinary settings, finite-element (FE) data often have to cross organizational and software boundaries because project partners rely on different solver environments or because multi-stage workflows intentionally combine tools with different domain- and task-specific strengths [1]. In practice, such exchange frequently requires FE meshes to be transferred between solver ecosystems. However, solver input decks differ in syntax, supported concepts, representational constraints, and identifier rules [2–4], which complicates reliable transfer, reuse, and auditability. Existing import/export paths and translator utilities typically implement documented subsets; unsupported entities may be skipped, ignored, or otherwise not translated depending on the chosen route [5–7]. Within the mesh-level scope of this paper, two challenges dominate: (i) mismatches in representational capabilities and target-specific constraints, most notably hierarchy/instancing, rigid placement where available, and naming/namespace rules; and (ii) differing element-type encodings and local node-order conventions, which can induce systematic orientation regressions if not treated explicitly [2–4,8–10]. We address (i) through deterministic, capability-aware normalization in

☆ This article is part of a Special issue entitled: 'JCAD_CAD-CAE Interoperability' published in Computer-Aided Design.

* Corresponding author.
E-mail addresses: peter.grohmann@uni-bayreuth.de (P.R. Grohmann), stephan.tremmel@uni-bayreuth.de (S. Tremmel).

https://doi.org/10.1016/j.cad.2026.104153
Received 8 February 2026; Received in revised form 29 July 2026; Accepted 6 August 2026
Available online 19 August 2026
0010-4485/© 2026 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

a neutral hierarchical intermediate representation (IR), and (ii) through explicit element-family mappings with solver-specific node-order permutations for common 3D solid volume elements. The resulting modular reader/writer converter supports the declared mesh-level subset across Abaqus, ANSYS Mechanical (CDB), MSC NASTRAN (BDF), and Z88. To quantify conversion fidelity without relying on native solver identifiers, deck-level equality, or solver results, we compare canonicalized structural artifacts using the evaluation procedure defined in Section 5. The primary evaluation uses a Z88-pivot-canonicalized route comparison and capability-aware reporting; complementary direct FE-IR/JSON structural checks are reported in the Supplementary Information, Section S7. Across 23 models and 92 source-to-target conversions, the evaluation shows agreement under the defined invariants for geometry and element-incidence connectivity, while making explicit any capability-limited metadata transformations.

Scope and claim boundary. We convert and evaluate mesh-level structural information, including nodal coordinates and supported solid element definitions (Tet4/Tet10 and Hex8/Hex20 types and connectivities). We transfer node and element sets under the declared membership signatures (and set names where representable under the target identifier model) and include structurally tracked reference-point candidates (Section 2.1). Reference-point candidates are treated as structural point-like candidates, not as native solver remote-point, MPC, RBE, or coupling objects. Where supported by the target format, we represent hierarchy metadata and rigid placement information. Shell and beam elements, other element families, and analysis semantics — including materials, sections, surfaces, loads, boundary conditions, contacts, MPC/RBE/coupling semantics, solver controls, and solver results — are intentionally out of scope; the reported fidelity metrics therefore quantify mesh-level structural interoperability only and do not imply native deck identity, full analysis-model equivalence, solver-semantic equivalence, or solver-result equivalence.

Claim summary. Evaluated for agreement: connected-node geometry, with placement effects reflected in canonicalized global coordinates; supported Tet4/Tet10 and Hex8/Hex20 element-incidence connectivity; set-membership signatures; and exact `Part/PartInstance` counts for nontrivial reference hierarchies on hierarchy-capable targets. *Normalized and audited:* solver-facing IDs, set/group identifiers, namespaces, local node ordering, resolved coordinate/placement encodings, and target-constrained naming. *Structurally tracked:* reference-point candidates; where a target has no dedicated evaluated representation, proxy encoding and re-identification follow the documented writer/parser policy. *Out of scope:* entities outside the boundary above and any claim of full analysis-model, solver-semantic, or solver-result equivalence.

Research questions.

RQ1: To what extent can mesh-level structural FE data be transferred between Abaqus, ANSYS Mechanical, MSC NASTRAN, and Z88 without losing evaluated structural information within the supported scope?

RQ2: What total validation-workflow runtimes are observed across the evaluated model sizes and target formats, and are they practical for scripted validation and occasional interactive checks within this benchmark range?

Contributions. The main contributions are:

1. **Neutral hierarchical intermediate representation**: a format-independent model for mesh-level structural entities (nodes, elements, sets, and reference-point candidates) and selected hierarchy/placement metadata where representable, with auditable target-constrained normalization.
2. **Conversion and evaluation**: format-specific components for reading and writing the declared mesh-level subset in the four considered solver-deck formats, together with a deterministic, capability-aware evaluation framework centered on the Z88-pivot-canonicalized route comparison (Section 5.6.2).
3. **Element mapping and orientation handling**: systematic type mappings and explicit node-order permutations for common solid-element families (Tet4/Tet10 and Hex8/Hex20) to preserve incidence connectivity and guard against systematic orientation regressions.

Paper structure. Section 2 reviews solver-deck capabilities and existing exchange approaches and derives the data-model requirements; Sections 3 and 4 present the data model and converter architecture. Section 5 defines the evaluation methodology, Section 6 reports the results, Section 7 discusses implications and limitations, and Section 8 concludes the paper.

2. State of the art: structural data formats and FE exchange approaches

This section establishes the mesh-level structural interoperability target used throughout the paper. We first define the structural entities considered in the evaluated mesh-level subset, summarize the relevant capabilities and constraints of the considered solver-deck formats, derive requirements for the IR, and then position the resulting target against existing FE exchange approaches.

2.1. General requirements on FE structural data

FE structural data comprise the entities required to represent a discrete mesh topology. At a minimum, a transferable structural description must encode (i) nodes with unique identifiers and spatial coordinates and (ii) elements with a type identifier and an incidence list referencing node identifiers. Beyond nodes (IDs, coordinates) and elements (type, incidence), practical workflows often include groupings (node/element sets), point-like structural entities that may be referenced by downstream solver constructs, and — where available — hierarchical organization (parts/instances/assemblies) including rigid placement. Because such point-like entities are typically meaningful only together with solver-specific constraint semantics (e.g., couplings or multi-point constraints (MPCs)), we treat them strictly as structural point-like entities and keep solver-level MPC/RBE/coupling semantics outside the mesh-only scope.

Operational definition used in this work. We use the term *reference-point candidate* for point-like entities that may remain unconnected to the continuum mesh and may be referenced by out-of-scope downstream solver constructs (e.g., remote loading or coupling). Within the mesh-only scope, candidates are represented as node-like structural entities with coordinates and deterministic structural tracking information; they may exist without incident elements. When a target lacks a dedicated candidate representation, candidates are exported via deterministic proxy encoding and, where supported by the documented writer/parser policy, re-identified on re-import (details in Section 3.5 and Supplementary Information, Section S2).

Thus, mesh nodes and reference-point candidates differ in structural role rather than coordinate data type. Mesh nodes are evaluated through connected-node and element-incidence signatures when they participate in the supported solid-element connectivity. Reference-point candidates are point-like, possibly unconnected structural entities kept in a separate IR and evaluation channel. In formats without such a representation, deterministic proxy tracking keeps candidates distinguishable for re-import and evaluation under the evaluated writer/parser policy. This structural proxy representation supports re-import and evaluation only; it does not imply native reference-point object identity, proxy-ID preservation, MPC/RBE/coupling semantics, or solver-level remote-point behavior.

Table 1
Overview of the deck-format capabilities considered in this work within the evaluated mesh-level subset. Check marks indicate support in the evaluated subset; crosses indicate lack of support in the evaluated subset; N/A indicates concepts that are not applicable within the declared scope.

Capability	Abaqus	ANSYS CDB	NASTRAN BDF	Z88
Mesh core				
Nodes/elements	✓	✓	✓	✓
Sets (membership)	✓	✓	✓	✓
Human-readable set names	✓	✓	✗[a]	✓
Hierarchy and placement				
Hierarchy/part-membership metadata in evaluated scope	✓	✗	✗	✓[b]
Rigid instance placement (explicit transform)	✓	✗	✗	✗
Coordinate handling				
Node coordinate-system references on input (resolved)	✓	N/A[c]	✓	✗
Point-like entities				
Dedicated reference-point-candidate representation	✓	N/A[d]	N/A[e]	✓[f]

[a] NASTRAN set cards (e.g., SET3) are addressed by numeric set IDs; human-readable set names are not supported (see [3]).
[b] For Z88, this denotes part-membership metadata carried by evaluated sidecar artifacts, not Abaqus-like native instancing (Supplementary Information, Section S4.6 and Table S6).
[c] Within the mesh-only scope considered here, coordinate-system definitions are not imported/exported for ANSYS CDB; node coordinates are treated as global Cartesian coordinates (consistent with the Workbench External Data workflow; see [11]).
[d] ANSYS Mechanical Remote Points are implemented via MPC equations (outside the scope); thus reference-point candidates are not natively distinguishable in this work, and only structural candidate tracking is evaluated (see [11,12]).
[e] In NASTRAN, reference-point behavior is typically realized by an independent GRID point together with constraint entities (e.g., RBE2/RBE3); since constraint semantics are out of scope, reference-point candidates are not natively distinguishable here, and only structural candidate tracking is evaluated (see [3]).
[f] For Z88, this denotes dedicated candidate records in the toolchain-specific sidecar artifact `z88iRP.txt` used by the evaluated writer/parser, not a native Z88 solver entity or general remote-point semantic.

2.2. Structural data formats of the considered FE solvers

We consider text-based structural deck representations for Abaqus (.inp), ANSYS Mechanical coded database (CDB) exports, MSC NASTRAN Bulk Data Files (BDF), and Z88. Throughout, the supported subset is limited to mesh-level entities and selected structural metadata within the scope defined in Section 1.

Abaqus. Keyword-based input decks (`.inp`) define nodes, elements, and sets explicitly and can be organized in an assembly/part/instance hierarchy [2]. The format supports instance placement through explicit transformations and coordinate-system references. In our workflow, rigid placement is represented explicitly in the intermediate representation (IR). We resolve placement to globally positioned coordinates when exporting to targets that have no explicit instance placement (see Table 1). Abaqus input decks can define sets at part, part-instance, or assembly scope, including assembly-level sets that may reference members from one or more instances. Where a target format cannot represent these Abaqus scoping levels explicitly, the corresponding set information is flattened deterministically under the declared policy while preserving the evaluated set membership. In the evaluated Abaqus parser path, nodes from `*NODE` blocks parsed in `*ASSEMBLY` scope are stored directly in the separate reference-point-candidate channel. The parser additionally promotes nodes in its flat/model-level scope when they are not referenced by imported supported solid-element connectivity and do not originate from a `*PART` definition; any associated solver-level constraint/control-point semantics remain outside the evaluated scope [13].

ANSYS Mechanical (CDB). ANSYS Parametric Design Language (APDL) coded database files (`.cdb`) encode meshes primarily through node and element blocks. Groupings can be represented as APDL components and are mapped to node/element sets [14]. The evaluated representation is treated as flat, with no explicit part/instance hierarchy. When importing a CDB file via the Workbench *External Data* workflow, only `NBLOCK` and `EBLOCK` are read [11]. In contrast, our converter parses CDB decks directly and additionally supports `CMBLOCK` component definitions to transfer node/element sets as MAPDL components. Remote Points in Mechanical are implemented via MPC equations [12] and are therefore outside the CDB subset supported in this work. Any point-like candidates are handled only as structurally tracked candidates as defined in Sections 2.1 and 3.5.

MSC NASTRAN. Bulk Data Files (BDF) represent structural models using GRID cards for nodes and element cards such as CTETRA/CHEXA for 3D solid elements, with groups represented via SET cards (e.g., SET3). Within the evaluated BDF subset, the representation is treated as flat and provides no explicit part/instance hierarchy comparable to Abaqus [3]. Coordinate-system references in GRID are resolved during import into a global Cartesian representation. NASTRAN-style reference behavior is commonly realized by independent GRID points combined with constraint or rigid-element constructs, such as RBE2/RBE3 [3]. Because these constraint semantics are outside scope, the evaluated mesh-level subset treats GRID points that are not referenced by imported supported solid-element connectivity only as structural reference-point candidates, as defined in Section 2.1.

Z88. Structural information is distributed across multiple plain-text files, separating mesh topology (nodes/elements) from sets and optional part membership [4,15]. For Z88, set membership is stored in an auxiliary set artifact; to preserve set attachment levels (assembly vs. part/instance) across hierarchy-capable sources, we use a toolchain-compatible extension that records an optional owner token per set and is supported symmetrically by the evaluated writer/parser. Details are provided in Supplementary Information, Section S4.6.1. Dedicated reference-point-candidate records provided through the toolchain-specific Z88 sidecar artifact `z88iRP.txt` are mapped to the IR reference-point-candidate channel and treated as dedicated structural candidate records within the evaluated Z88-sidecar subset.

For the Z88-pivot-canonicalized route comparison, the evaluated Z88 export policy imposes practical identifier constraints (Section 5.3), notably strictly increasing, gap-free node-identifier and element-identifier sequences; export may therefore require deterministic renumbering. Additional structural sidecar artifacts used for auditable evaluation are documented in the Supplementary Information, Section S4.6, including Section S4.6.1 and Table S6.

Summary and capability gaps. All considered formats represent nodes, supported solid elements, and set membership within the evaluated

mesh-level subset. Hierarchy or part-membership metadata is available in Abaqus and, via evaluated sidecar artifacts, in Z88. Rigid instance placement through explicit transforms is available only in Abaqus. Abaqus and NASTRAN BDF accept node coordinate-system references, but these are resolved to global Cartesian coordinates on import and are not preserved or re-exported within the present scope. Within the evaluated mesh-only CDB handling, coordinate-system definitions are not imported or re-exported; node coordinates are treated as global Cartesian coordinates. We therefore (i) flatten hierarchy and placement deterministically where required by a target and (ii) report metrics in a capability-aware way, using N/A where a concept is not representable in the evaluated target subset rather than penalizing representational absence. Even where a concept is nominally supported, such as human-readable set names, practical constraints including allowed character sets, truncation, case handling, and uniqueness enforcement can affect identifier portability and motivate deterministic, auditable normalization.

2.3. Requirements for a neutral data model

From the format analysis above, we derive the following requirements for a neutral data model:

1. **Stable internal representation of nodes and elements**: internal identifiers independent of source-deck IDs with deterministic mappings to solver-facing IDs for import and export.
2. **Neutral element-family abstraction with orientation safety**: explicit type mapping for common solid families and a canonical local node ordering with solver-specific permutations.
3. **First-class sets**: node/element sets with membership signatures stable under renumbering; human-readable identifiers normalized and audited where applicable.
4. **Support for flat and hierarchical structures**: parts/instances/assemblies and rigid placement where representable, with deterministic flattening for targets that do not support these constructs.
5. **Deterministic, auditable normalization**: collision-resolving remapping, renaming/sanitization, and stable serialization, with optional change reports that attribute target-enforced transformations to the responsible target constraints.

These requirements motivate the hierarchical data model introduced in Section 3.

2.4. Exchange objectives and positioning of this work

Building on these requirements, existing FE mesh exchange approaches can be distinguished by their primary exchange objective. Vendor and computer-aided engineering (CAE) tool translators primarily target practical, route-specific solver-deck migration; neutral mesh formats and mesh I/O libraries primarily target geometry, topology, and attached data exchange; scientific mesh and results data formats primarily target persistent storage of meshes and simulation results for preprocessing, postprocessing, and data transfer between simulation programs; and STEP AP209 embeds analysis information in a broader product/analysis information model. These objectives overlap, but they do not imply the same preservation claim. The approaches discussed below are therefore treated according to their primary exchange objective rather than as interchangeable solutions to a single, uniform conversion problem. The comparison is about scope and evaluated transfer targets, not about replacing established converter routes, neutral formats, libraries, or standards. In the mesh-level scope of this paper (Section 1), the relevant claim is narrower: capability-aware transfer and agreement of declared mesh-level structural channels across heterogeneous solver representations. The evaluated evidence includes set-membership signatures, exact `Part`/`PartInstance` count diagnostics for nontrivial reference hierarchies, placement effects reflected in canonicalized global coordinates, and topology-specific local-node-order diagnostics. Target-specific addressing constraints, such as identifier namespaces and naming rules, are handled through deterministic normalization (Section 2.3). We summarize the main classes of existing exchange strategies and position our approach with respect to these requirements.

Deck-to-deck translation in solver environments and computer-aided engineering (CAE) toolchains. Industrial exchange often relies on built-in import/export functions or translator pipelines in solver graphical user interfaces and pre-/post-processors. A common pragmatic strategy is to use NASTRAN Bulk Data as an interchange hub because many tools offer direct import/export support (e.g., Abaqus supports import from and export to NASTRAN BDF) [5,16]. However, documented translator routes implement supported-entry lists or route-specific mapping schemes and may skip unsupported entities or require route-specific substitutions; concepts without a target counterpart may also be resolved or flattened depending on source and target capabilities [6,7,17]. As a consequence, fidelity is bounded by representational overlap and the entities supported for the chosen route. This motivates explicit scoping, deterministic normalization under target constraints, and post-import verification when translated decks serve as workflow artifacts rather than purely as mesh carriers.

Legacy neutral mesh formats (geometry-centric exchange). Legacy neutral mesh formats such as the I-DEAS Universal File format (UNV/UFF) were introduced for CAD/CAE data transfer and remain in practical use due to their open specification [18]. For mesh exchange, toolchains commonly rely on the node and element datasets (e.g., 2411/2412) to transport coordinates and element incidence, and may additionally carry simple group/set and results datasets depending on the selected datasets and reader support [19–21]. Reader support is therefore dataset- and implementation-dependent. As a consequence, UNV/UFF is effective when geometry and incidence connectivity transfer is the primary goal (Requirements 1–2 in Section 2.3), but additional mapping and validation rules are needed for hierarchical instancing, rigid placement, attachment-level set scoping, or target-specific naming and namespace constraints (Requirements 3–5).

General-purpose mesh I/O and processing toolchains. Tools such as `meshio` and the Visualization Toolkit (VTK) provide general-purpose mesh I/O and visualization workflows. They focus on format conversion and dataset-centric representations of points, cells (elements), connectivity, and per-point/per-cell data arrays for downstream processing and visualization [22–26]. VTK further provides composite dataset types to organize multiple datasets in a hierarchical tree [27]; this is useful for dataset organization, but distinct from solver-deck assembly, instance, and placement semantics. These abstractions primarily cover geometry, topology, and attached attributes. They are therefore complementary for conversion and inspection, but do not by themselves define the solver-deck structural-information target evaluated here, in particular capability-aware hierarchy and placement normalization, solver-specific identifier models, naming and namespace constraints, or deck-level set scoping rules (Requirements 3–5 in Section 2.3).

Scientific mesh and results data formats. Scientific mesh and results data formats such as Exodus II focus on persistent storage and exchange of finite-element mesh topology and associated simulation variables [28–30]. They are well suited for large meshes, simulation variables, and results-oriented workflows. However, their primary exchange objective differs from solver input-deck migration; hence, deck-specific constructs such as keyword syntax, solver addressing constraints, and solver-level hierarchy or placement semantics are outside the structural-information target considered here. Exodus-like formats and the present IR therefore address related but distinct interoperability layers.

Product/analysis information standards. ISO 10303 AP209 defines a comprehensive product/analysis information model for multidisciplinary analysis and design [31]. This breadth is valuable for end-to-end product/analysis data management. At the same time, repeatable use in a specific mesh-level exchange workflow still requires a clearly stated scope, declared subsets, implementation guidance, tailored mappings, and validation targets for the analysis constructs of interest [32,33]. AP209 therefore provides the broader standards context for this work rather than the compact, solver-deck-oriented IR and validation target evaluated here.

Research prototypes and conversion-focused neutral representations. Prior research highlights that robust conversion requires consistent element-type mapping and careful handling of local node ordering to prevent systematic orientation regressions [9,10]. These observations motivate treating element families and ordering conventions as explicit, testable conversion concerns (Requirement 2 in Section 2.3). Our work complements this line by additionally addressing (i) hierarchy and rigid placement where representable, (ii) capability-aware treatment of sets and identifiers under target constraints, and (iii) a deterministic evaluation pipeline that is insensitive to permitted representational differences (renumbering, serialization ordering, and deterministic capability-driven flattening), enabling cross-route structural reporting.

Positioning of this work. Mesh import/export, solver-to-solver translation, and neutral mesh exchange are established practices. The gap addressed here is therefore not the absence of conversion routes, but the need for an explicit, compact mesh-level structural interoperability target that combines (i) deck-oriented structural information beyond geometry and incidence, (ii) capability-aware hierarchy and placement handling where representable, (iii) set membership under target-specific naming and namespace constraints, (iv) explicit local node-order mappings for supported solid families, and (v) auditable fidelity reporting. The proposed IR is positioned between these levels: it is narrower in scope than STEP AP209, but more explicit than plain geometry/topology-oriented mesh containers with respect to the evaluated mesh-level solver-deck information. It is not intended to replace vendor translators, mesh I/O libraries, scientific mesh and results data formats, or neutral standards. Rather, it provides an auditable intermediate representation and validation basis for the mesh-level structural subset evaluated in this paper.

3. Neutral hierarchical intermediate representation

The intermediate representation (IR) is an object-oriented, neutral data model for mesh-level structural entities. It provides a common representation for the supported subset and decouples solver-specific deck syntax from the structural information used for conversion and validation. Within this paper, the IR covers nodes, supported solid elements, node and element sets, selected hierarchy/placement metadata where representable, and structurally tracked reference-point candidates. The IR follows the declared mesh-level subset of Section 1 and therefore does not encode solver-level analysis semantics. It enables exchange between Abaqus, ANSYS Mechanical (CDB), MSC NASTRAN (BDF), and Z88 within the supported scope. Fig. 1 summarizes the core classes and relationships. The implementation follows a data-model-centered architecture that builds on prior work [34].

Representation invariants (used throughout the converter and evaluation). Unless stated otherwise, the IR satisfies the following invariants:

- **Identifier integrity and deterministic remapping**: Node and element identifiers are valid within their respective scopes. Target-specific constraints (e.g., single namespace, gap-free IDs) are enforced deterministically on export and applied consistently to all dependent references (Section 3.5).
- **Explicit ownership**: Each `Part` owns exactly one `Mesh` containing its nodes, elements, and set information. `PartInstance` objects reference `Part` definitions.
- **Set attachment levels**: NodeSets and ElementSets may be attached at model, assembly, part, or instance level. If a target format cannot represent a specific attachment level, the writer applies deterministic flattening while retaining the evaluated membership.
- **Coordinate convention**: Node coordinates are stored in Cartesian coordinates. For sources capable of hierarchies, part meshes are stored in *part-local* coordinates and rigid instance placement is represented explicitly by the `PartInstance` transformation. For flat targets and in the Z88-pivot-canonicalized route comparison (Section 5.3), placement and referenced coordinate systems are resolved to produce globally positioned coordinates.
- **Element topology and local ordering**: Supported solid elements are represented by neutral element-family identifiers and canonical local node ordering. Solver-specific local node-order conventions are handled through explicit import/export permutations (Section 3.3.2).
- **Reference-point-candidate separation**: Structurally tracked reference-point candidates are stored in a separate IR channel. This channel supports structural tracking and evaluation only; it does not imply native solver remote-point objects, MPC/RBE/coupling semantics, or solver-level remote-point behavior.
- **Deterministic serialization**: Entities are serialized in a stable order (e.g., by internal IDs) to support repeatable deck and evaluation-artifact generation.

3.1. Object-oriented core structure

FE structures are represented using a small set of central entities (Fig. 1):

- **Model:** root object containing one Assembly, a collection of Parts, and model-level nodes, elements, sets, and reference-point candidates for flat sources.
- **Part:** part definition with an associated Mesh and optional part-level NodeSet/ElementSet definitions.
- **Assembly:** container for the instantiated structure. Holds PartInstance objects and optional assembly-level sets and reference-point-candidate entities.
- **PartInstance:** instantiates a referenced Part in the Assembly; may carry a rigid placement described by a Transformation (Section 3.2).
- **Mesh:** stores the `Node`, `Element`, `NodeSet`, and `ElementSet` objects associated with a `Part`.
- **Node/Element:** a Node has an integer ID and Cartesian coordinates; an element has an integer ID, a neutral element-type identifier, and an ordered list of incident node IDs (local node ordering).
- **ReferencePoint**: separate node-like candidate entity with an internal ID and Cartesian coordinates. In this paper, `ReferencePoint` denotes a structural reference-point candidate only.
- **NodeSet/ElementSet:** named collections of node or element identifiers. NodeSet membership may include reference-point candidates where such candidates are represented in the corresponding IR channel.
- **Transformation**: rigid placement represented by rotation and translation parameters (Section 3.2).

This design accommodates hierarchy-capable sources and artifacts, such as Abaqus and the evaluated Z88 sidecar-based hierarchy metadata, as well as predominantly flat sources, such as ANSYS Mechanical

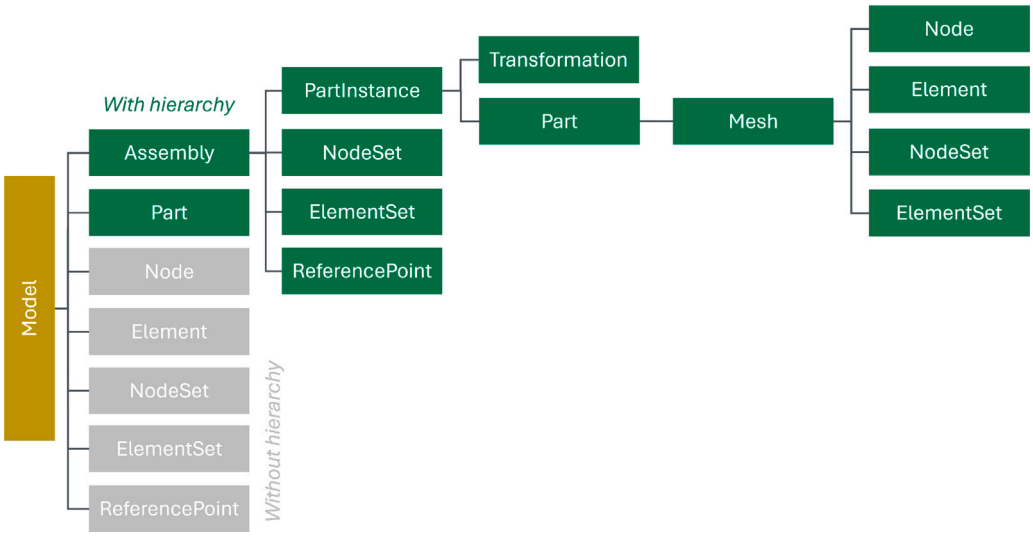


Fig. 1. Unified Modeling Language (UML) overview of the IR (`Model`, `Assembly`, `Part`, `PartInstance`, `Mesh`, `Node`, `Element`, `NodeSet`, `ElementSet`, `Transformation`, and `ReferencePoint`). Here, `ReferencePoint` denotes the IR class used for structurally tracked reference-point candidates; it does not imply native solver remote-point, MPC, RBE, or coupling semantics. Green boxes indicate hierarchy-aware representation; gray boxes indicate the flat/non-hierarchical representation path, not unsupported entities. The architecture extends the data-model-centered approach introduced in [34]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(CDB) and MSC NASTRAN (BDF), under a uniform internal structure. Flat sources are handled through the flat/non-hierarchical representation path and may be materialized as a single `Part` instantiated once where this keeps downstream handling uniform. An illustrative hierarchy-exchange example is provided in the Supplementary Information, Section S3 (Figure S1).

Reference-point candidates. Reference-point candidates (Section 2.1) are represented as node-like structural entities that may remain unconnected to supported solid-element connectivity. Where an evaluated target format provides no dedicated representation for these candidates within the declared subset, they are carried through deterministic proxy tracking and, where supported by the documented writer/parser policy, re-identified on re-import (Section 3.5; Supplementary Information, Section S2). This proxy representation supports structural comparison only under the reference-point-candidate boundary defined in Section 2.1.

Set membership and names. `NodeSet` and `ElementSet` entities store evaluated structural membership independently from target-specific naming and namespace constraints. Each set has an internal identifier, an optional human-readable name, an attachment level where representable, and an explicit membership list. This separation allows the converter to preserve and evaluate set membership even when a target format normalizes, truncates, renames, or omits human-readable set identifiers. Accordingly, set-membership agreement is evaluated through membership signatures, whereas name portability is treated as a separate name-sensitive diagnostic where applicable.

3.2. Coordinate systems and positioning

Because solvers differ in coordinate-system conventions and placement mechanisms, the IR stores node coordinates in Cartesian form and represents rigid placement explicitly when available. We represent a rigid transformation T by a rotation matrix $R \in \mathbb{R}^{3\times3}$ and a translation vector $t \in \mathbb{R}^3$, mapping a local point x_{local} to a global point x_{global} via

$$x_{\text{global}} = R\,x_{\text{local}} + t. \tag{1}$$

During import from Abaqus, supported instance transformations are parsed and stored as `Transformation` objects. These transformations are applied when exporting to flat targets or when generating globally positioned pivot artifacts for canonicalization.

Table 2
Solver-specific element designations for the supported hexahedral and tetrahedral solid-element families in the evaluated mesh-level structural subset. MSC NASTRAN CHEXA and CTETRA variants are distinguished by node count.

Software	Hexahedral		Tetrahedral	
	Linear	Quadratic	Linear	Quadratic
Abaqus	C3D8	C3D20	C3D4	C3D10
MSC NASTRAN	CHEXA	CHEXA	CTETRA	CTETRA
ANSYS	SOLID185	SOLID186	SOLID285	SOLID187
Z88	1	10	17	16

Supported coordinate-system references are resolved to Cartesian coordinates on import. Native coordinate-system definitions are not re-exported; only their resolved geometric effect is preserved within the present mesh-level scope.

3.3. Element families, type mappings, and canonical local node ordering

3.3.1. Element families and type mapping

The evaluated element subset comprises linear and quadratic tetrahedral and hexahedral 3D solid elements, represented internally as Tet4, Tet10, Hex8, and Hex20 families. Solver decks differ in element designations and in local node-order conventions even when they represent the same geometric element family. To support route-level structural comparison, the IR separates two concerns: (i) mapping solver-specific element designations to neutral internal element families and (ii) mapping solver-specific local node orderings to a canonical IR ordering.

Table 2 maps the solver-specific element designations used in the evaluated subset to the neutral internal element families. The mapping is based on geometric family and node count. A detailed mapping table, including the internal family tokens and parser-side mapping rules, is provided in Appendix A, Table A.1.

Element formulation qualifiers. Solver-specific formulation qualifiers, where present in accepted element designations, are not represented as separate analysis semantics in the IR. Within the present mesh-level structural scope, elements are mapped by geometric family and node count. Consequently, the reported metrics quantify mesh-level structural interoperability for the declared element families under Section 1 claim boundary.

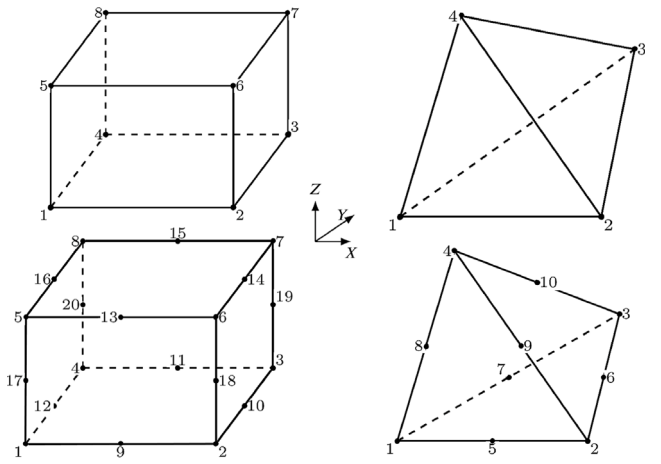


Fig. 2. Canonical local node numbering used in the IR for Hex8/Hex20 and Tet4/Tet10 elements. Top row: Hex8 and Tet4; bottom row: Hex20 and Tet10. Node numbering follows the Abaqus solid-element convention [2].

3.3.2. Canonical local node numbering and permutations

Connectivity-preserving conversion and route-level comparison require explicit handling of local node ordering because solvers use different conventions for otherwise equivalent element families [2–4,8]. The IR therefore uses an Abaqus-based canonical local node ordering for the supported Tet4/Tet10 and Hex8/Hex20 families, following the corresponding Abaqus solid-element node-numbering convention [2]. Fig. 2 illustrates this canonical ordering.

For each supported format μ and element family $\eta \in \{\mathrm{Tet4}, \mathrm{Tet10}, \mathrm{Hex8}, \mathrm{Hex20}\}$ with n_η local nodes, let

$$P^{\mu,\eta} = \left(p_1^{\mu,\eta}, \ldots, p_{n_\eta}^{\mu,\eta}\right)$$

denote the format-specific solver-to-canonical local node-order permutation, using one-based positional indices. Here, $p_i^{\mu,\eta}$ is the position in the format-local node list that corresponds to canonical IR position i. For import from a source format μ_{src}, the format-local node list

$$s = (s_1, \ldots, s_{n_\eta})$$

is mapped to the canonical IR ordering

$$a = (a_1, \ldots, a_{n_\eta})$$

by

$$a_i = s_{p_i^{\mu_{\mathrm{src}},\eta}}, \qquad i = 1, \ldots, n_\eta.$$

For export to a target format μ_{tgt}, the same target-specific permutation is used in the opposite direction: each canonical entry a_i is written to the target-local position $p_i^{\mu_{\mathrm{tgt}},\eta}$,

$$t_{p_i^{\mu_{\mathrm{tgt}},\eta}} = a_i, \qquad i = 1, \ldots, n_\eta.$$

Because $P^{\mu_{\mathrm{tgt}},\eta}$ is a permutation, each target-side local position is filled exactly once. Equivalently, this export step applies the inverse of the target permutation. Thus, node-order normalization is symmetric with respect to import and export: source-specific orderings are mapped into the canonical IR representation, and target-specific orderings are reconstructed from it.

The complete solver-specific permutation tables used in the benchmark are provided in the Supplementary Information, Tables S1 and S2. These tables define the authoritative permutation configuration for the evaluated conversions.

Because the primary element-incidence signature used in the evaluation is intentionally order-invariant, element F_1 alone should not be interpreted as evidence of preserved local node ordering or element orientation. Ordering regressions are monitored separately using an order-sensitive coordinate-key connectivity sentinel. For tetrahedral elements, a corner-node signed-volume diagnostic provides an additional orientation check; hard-integrity diagnostics are reported in the Supplementary Information. This separation keeps the primary element metric robust to solver-specific local-order conventions while retaining a separate local-order sentinel across the evaluated element families and an additional corner-node orientation diagnostic for tetrahedral elements.

3.4. Mapping overview

This section summarizes how source-format constructs are mapped into the IR. Target-driven normalization is addressed separately in Section 3.5. Detailed per-format parser and writer rules are provided in the Supplementary Information, Section S4; writer-side compliance interventions and their metric interpretation are documented in the Supplementary Information, Section S5.

We group the mapping into four structural channels. First, nodes and supported solid elements are imported as IR `Node` and `Element` objects. Solver-specific element designations are mapped to the neutral Tet4/Tet10 and Hex8/Hex20 element families defined in Section 3.3.1, and solver-specific local node orderings are converted to the canonical IR ordering using the permutations described in Section 3.3.2.

Second, structural groupings are imported as `NodeSet` and `ElementSet` objects. Where representable, the IR retains the corresponding attachment level, such as model, assembly, part, or instance level. Human-readable set names are retained as metadata where supported, but set membership is represented independently from target-specific name, namespace, and identifier constraints. This separation allows the evaluation to distinguish set-membership agreement from name-sensitive portability diagnostics.

Third, hierarchy and placement metadata are imported where the source format exposes them within the evaluated subset. For hierarchy-capable inputs, including Abaqus decks and the evaluated Z88 sidecar metadata, part/instance or part-membership information is represented explicitly in the IR where available. When exporting to flat targets, such as ANSYS Mechanical (CDB) or MSC NASTRAN (BDF), hierarchy and placement information are flattened or resolved deterministically according to the target policy.

Fourth, point-like entities are represented as structurally tracked reference-point candidates. On import, candidates are identified according to the format-specific structural rules defined for the evaluated subset. Where an evaluated target format lacks a dedicated structural representation for such candidates, writers use deterministic proxy tracking so that candidates can be re-identified for structural comparison where possible. This mapping follows the reference-point-candidate interpretation defined in Section 2.1; proxy materialization is specified in Section 3.5.

Together, these mappings define the structural information transferred through the IR. The corresponding target-driven normalization steps, including identifier remapping, name normalization, gap-free identifier requirements, hierarchy flattening, and proxy materialization, are defined in Section 3.5.

3.5. Deterministic normalization under target constraints

The IR separates internal structural identity from solver-facing identifiers. This separation is necessary because target formats impose different namespace, naming, and serialization constraints. During export, the writer therefore applies deterministic target-driven normalization where required. These actions are part of the IR export contract: they make the generated target artifact compliant with the evaluated target subset and update all dependent references consistently. They are not interpreted as preservation of native source identifiers, native set names, native namespaces, or solver-level semantics.

Single-namespace targets and deterministic identifier remapping. Some targets operate in a single global node or element namespace, whereas hierarchy-capable sources may reuse identifiers across parts or instances. When exporting to such targets, node and element identifiers are remapped deterministically to avoid collisions. Element connectivity, set membership, hierarchy-related ranges where applicable, and other dependent references are updated consistently. The resulting target-facing identifiers are therefore serialization identifiers, not an ID-preservation metric.

Set and group names. Human-readable set or group names are retained where representable. However, target formats may impose allowed-character rules, length limits, reserved-name constraints, case or whitespace semantics, and namespace restrictions. When required, writers apply deterministic name normalization, including sanitization, quoting or truncation where applicable, and collision disambiguation. These transformations are treated as identifier-level metadata and, where change reports are enabled, recorded there. Consequently, name-sensitive diagnostics such as Name-F_1 measure identifier portability under target constraints, whereas membership-based metrics measure agreement of the evaluated set members (Section 5.6.6).

Reference-point proxies. Reference-point candidates are tracked as structural point-like candidates, not as native solver remote-point or coupling objects. If a target format lacks a dedicated reference-point-candidate representation in the evaluated subset, the writer serializes the candidate through deterministic proxy encoding while maintaining the IR/evaluation distinction through the writer/parser policy. Proxy identifiers are assigned collision-free under the evaluated writer/parser policy, and candidates remain separable during re-import where the documented policy supports re-identification. This mechanism supports structural re-identification and evaluation only; it does not imply native reference-point object identity, proxy-ID preservation, MPC/RBE/coupling semantics, or solver-level remote-point behavior.

Z88-specific gap-free identifier constraint. For Z88 export, including its use in the Z88-pivot-canonicalized route comparison, node and element identifiers are normalized to strictly increasing, gap-free sequences. Element definitions, set membership, reference-point-candidate records, and supported auxiliary artifacts are updated consistently. This normalization is part of the evaluated Z88 export and canonicalization policy and supports deterministic route-level comparison.

Traceability. Where enabled, the implementation emits a structured change report that records applied normalization actions and their target-driven rationale, including identifier remapping, name sanitization, reserved-name handling, collision resolution, proxy materialization, and other target-compliance actions. The report serves as an audit trail and, where sufficiently detailed, can also serve as an alias table between source-facing and exported identifiers. The enforced identifier and naming constraints are derived from the target-format rules summarized in the Supplementary Information, Table S10; representative writer-side interventions and their metric interpretation are documented in the Supplementary Information, Section S5. The software architecture implementing these rules is described in Section 4.3.

4. Implementation of the FE converter

4.1. Architecture

Based on the neutral hierarchical IR introduced in Section 3, we implemented a modular FE mesh converter for the declared mesh-level structural subset. The converter follows a two-step translation pattern:

solver input deck → IR → target deck or exchange artifact.

This architecture decouples solver-specific deck syntax from the neutral structural representation used for conversion and validation. It comprises (i) an application layer for input validation and run orchestration (Section 4.4), (ii) a converter core that hosts the IR, enforces representation invariants, and applies deterministic normalization required by target constraints (Section 3), and (iii) format-specific reader/writer components for Abaqus, ANSYS Mechanical (CDB), MSC NASTRAN (BDF), and Z88, together with IR-JSON serialization for evaluation and exchange artifacts (Fig. 3). A conversion run first identifies and validates the input format, then parses the supported structural subset into the IR. The converter core applies representation checks and target-specific post-processing, including deterministic identifier remapping where required. The corresponding writer then serializes the IR to the target deck or exchange artifact. Where enabled, writers also emit a structured change report that records target-driven normalization actions for later audit and interpretation in the evaluation workflow (Sections 4.2–4.3).

4.2. Parser components, dialect support, and failure policy

For each supported deck format, a dedicated parser implements the declared structural subset and maps recognized constructs into the IR. The parsed information includes nodes, supported solid elements, node and element sets, selected hierarchy/placement metadata where representable, and structurally tracked reference-point candidates according to the rules defined in Sections 2.1 and 3. The parsers also handle common deck dialect variants within the supported subset, such as MSC NASTRAN field variants and continuation rules or optional and incomplete metadata in ANSYS CDB blocks. Unsupported solver-level analysis entities remain outside the evaluated scope and are not included in structural-fidelity claims. Malformed or unsupported non-structural entries are reported as warnings where possible. Imports are aborted when structural integrity would be compromised, for example by non-parseable core entities or invalid element-node references. This failure policy distinguishes three cases: supported structural information is imported into the IR; unsupported or out-of-scope non-structural information is reported, where possible, without being included in the evaluated claim; and structurally invalid input that would undermine the IR invariants causes the import to fail.

4.3. Writer components, target constraints, and traceability

Writer components implement the inverse direction by generating target-format decks or exchange artifacts from the IR. They emit nodes, supported solid elements, sets, and, where representable, hierarchy or auxiliary structural metadata. During export, writers apply target-specific element-type mappings and inverse local node-order permutations as defined in Section 3.3.2 and in the Supplementary Information, Tables S1 and S2. Writers also enforce target-format constraints that cannot be represented as a direct one-to-one serialization of the source deck. These constraints include, where applicable, single-namespace identifier models, gap-free identifier requirements, allowed character sets, name truncation, reserved-name handling, collision resolution, hierarchy flattening, and proxy materialization for reference-point candidates. Such transformations are deterministic target-compliance actions under the documented writer policy used in the evaluation. To support traceability, writers can emit a structured change report. This report records target-enforced compatibility actions, such as identifier remapping, name normalization, collision handling, and other serialization changes required for target-compliant import under the documented identifier rules. The report serves as an audit trail for interpreting route-dependent identifier changes and name-related deviations in the capability-aware evaluation (Section 5).

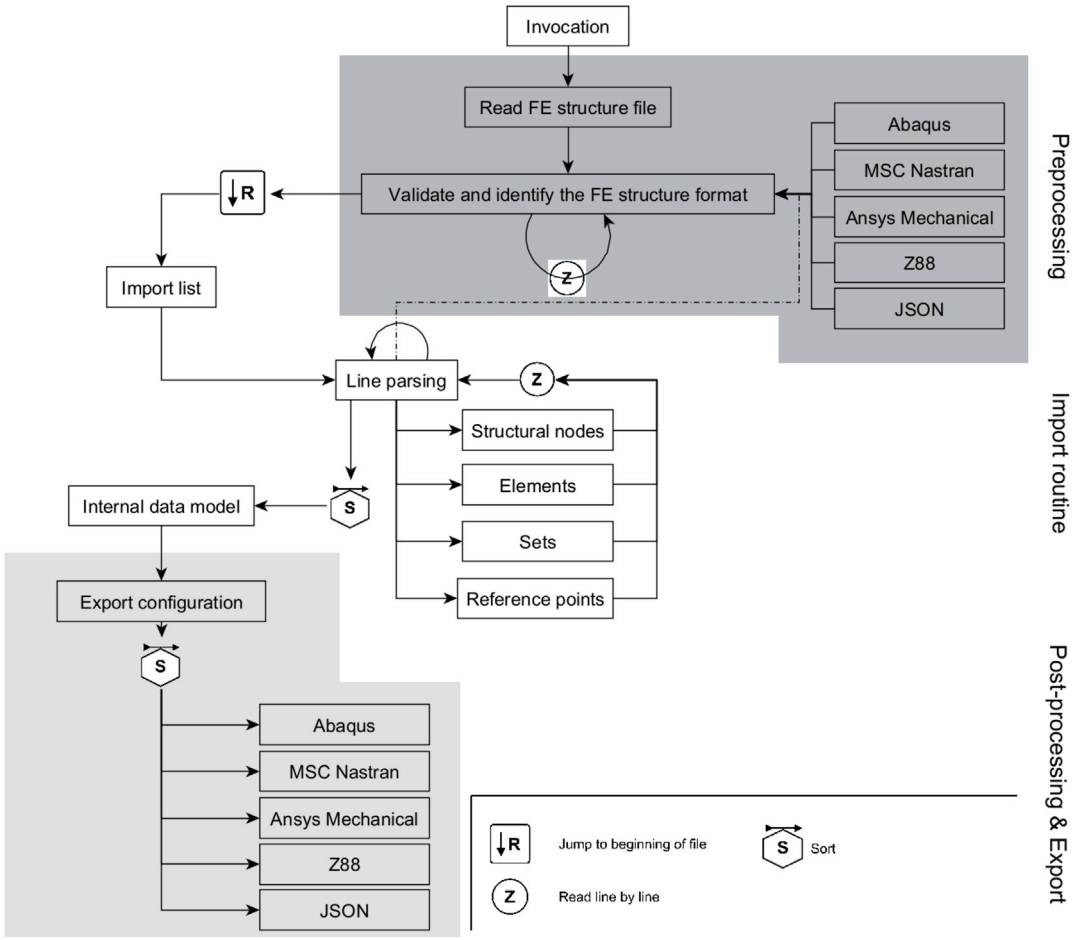


Fig. 3. Conceptual architecture of the FE mesh converter with a central IR and format-specific reader/writer components, expanded from [34]. Labels refer to the evaluated components for Abaqus, ANSYS Mechanical (CDB), MSC NASTRAN (BDF), Z88, and JSON-based IR artifacts. The source-derived "Reference points" label is retained in the schematic; throughout this article, it denotes structurally tracked reference-point candidates within the mesh-level subset, not native solver remote-point, MPC, RBE, or coupling semantics.

4.4. Interfaces and JSON evaluation artifacts

The implementation provides a command-line interface and JSON evaluation/exchange artifacts used in the workflows reported in this paper:

- **Command-line interface (CLI).** The CLI supports interactive use and script-based automation. It accepts an input path and an output path and then triggers the conversion pipeline described above.
- **JSON evaluation/exchange artifacts.** JSON artifacts mirror the IR and serve as neutral, deterministic exchange and validation representations for the declared mesh-level subset. In this paper, they are used as inputs to the structural evaluation pipeline (Sections 5.2–5.3).

The JSON representation is therefore an evaluation and exchange artifact for the declared mesh-level subset used by the route-level structural-fidelity evaluation in Section 5. Together, the modular component architecture, deterministic normalization rules, warning/failure policy, and optional change reports provide the implementation basis for that evaluation.

5. Evaluation objectives and methodology

Evaluation framework and claim boundary. The evaluation framework is centered on the Z88-pivot-canonicalized route comparison and is designed to assess the converter workflow within the mesh-level structural scope defined in Section 1. It addresses two questions: whether the evaluated model-level conversions retain the declared structural information under the canonicalized comparison (RQ1), and what total validation-workflow runtimes are observed across the evaluated model sizes and target formats (RQ2). All results are interpreted under Section 1 claim boundary.

Comparison principle. Direct equality of native solver decks is not a meaningful fidelity criterion because the considered formats differ in syntax, numbering, representable concepts, coordinate-system and placement encodings, hierarchy representations, and identifier rules. We therefore compare canonicalized structural artifacts rather than native deck text. In the primary evaluation, the source deck and each converted target deck are processed through the same Z88 writer/parser path and exported as JSON artifacts for structural comparison. We refer to this procedure as a Z88-pivot-canonicalized route comparison. It is not a literal source-to-target-to-source deck cycle. A formal definition is provided in Section 5.6.2.

Interpretation of the Z88 pivot. Within the declared scope, Z88 is used as a deterministic and auditable canonicalization instrument, not as an external reference standard. Agreement under the Z88 pivot is therefore interpreted as agreement under the declared structural invariants, tolerance settings, and canonicalization rules. It does not establish native deck identity, solver-semantic equivalence, solver-result equivalence, or independence from the shared Z88 writer/parser path.

Evaluation endpoints. For RQ1, the primary endpoint is agreement of declared mesh-level structural signatures after canonicalization. The evaluated channels include connected-node geometry, supported Tet4/Tet10 and Hex8/Hex20 element connectivity, set membership, reference-point candidates, and the hierarchy-count diagnostic defined below. Placement effects are reflected in canonicalized global node coordinates rather than evaluated as native placement syntax. These channels are interpreted in a capability-aware way: concepts not representable in the evaluated target subset are reported as not applicable rather than as failed conversions.

For RQ2, the endpoint is the operationally defined three-stage validation-workflow runtime. This runtime comprises reference canonicalization, source-to-target conversion, and converted-deck canonicalization. The reference-canonicalization stage can be cached per source model, but it is included in each reported total so that the same three-stage cost is reported for every evaluated model-level conversion.

Complementary evidence. The primary benchmark uses the Z88-pivot-canonicalized route comparison. A complementary direct FE-IR/JSON structural cross-check is reported in the Supplementary Information, Section S7, to reduce, but not eliminate, dependence on the shared Z88 writer/parser path for selected structural checks. This complementary path is interpreted as dependency-reducing structural evidence. It is not a replacement for the primary route-level benchmark, not a complete second benchmark covering all 92 evaluated model-level conversions, and not solver-semantic verification.

The following subsections define the benchmark model set and conversion routes, the deterministic comparison artifacts, the structural signatures and diagnostic channels, and the validation-workflow runtime measurement protocol.

5.1. Test models and conversion routes

The benchmark comprises 23 finite-element models originating from four source-format classes: Abaqus, ANSYS Mechanical (CDB), MSC NASTRAN (BDF), and Z88. The model set combines publicly redistributable, vendor-accessible but non-redistributable, and restricted industrial examples to exercise the supported 3D solid-element subset across heterogeneous solver-deck representations. The benchmark focuses on the Tet4/Tet10 and Hex8/Hex20 element families and includes, where present in the respective source models, node sets, element sets, hierarchy metadata where representable, and structurally tracked reference-point candidates. The full benchmark inventory, including source format, model size, element families, structural metadata channels, and redistribution status, is provided in the Supplementary Information, Table S12.

For reproducibility, the benchmark inputs are separated into three categories. Raw benchmark inputs for ten publicly redistributable models are included in the reproducibility package. Three additional Abaqus example decks are available to licensed Abaqus users through the Abaqus Fetch utility but are not redistributed because of vendor licensing restrictions. The remaining ten models are proprietary or otherwise non-redistributable. Accordingly, the public archive supports inspection of the redistributed inputs and publicly released derived evaluation artifacts, together with checksum verification of released artifacts, but it does not enable a complete public rerun of the full 23-model benchmark from public raw input decks alone.

Each source model was converted to each of the four target formats: Abaqus, ANSYS Mechanical (CDB), MSC NASTRAN (BDF), and Z88. We refer to the 16 combinations of source-format class and target format as source-to-target route classes, and to each individual model-route execution as an evaluated model-level conversion. Across the 23 source models, this design yields 92 evaluated model-level conversions. This conversion set provides the basis for the structural-fidelity and total validation-workflow runtime evaluations reported in Section 6.

5.2. Metrics and signatures

This section defines the structural signatures and agreement metrics used for RQ1. The metrics are computed on canonicalized structural artifacts, not on native solver decks. They are therefore interpreted as channel-specific agreement measures under the declared structural signatures, tolerance settings, and canonicalization policies.

For each test case, the evaluation compares two canonicalized JSON artifacts generated through the Z88-pivot-canonicalized route comparison defined in Section 5.6.2: the canonical reference artifact M_{ref} and the canonicalized converted-deck artifact M_{conv}^{τ}. The Z88 pivot is used as a deterministic canonicalization instrument, not as an external reference standard. For metric computation, the evaluation uses the following declared channels derived from these JSON artifacts: connected-node coordinates, supported solid elements represented by neutral element-family tokens and connectivity, node- and element-set membership, comparable human-readable set identifiers where applicable, and — where applicable — reference-point candidates and exact `Part/PartInstance` count diagnostics for nontrivial reference hierarchies on hierarchy-capable targets. The metric definitions are restricted to these channels; entities outside the declared mesh-level subset are neither preservation evidence nor counted as structural mismatches. The evaluation pipeline exports per-conversion structural-fidelity metrics, diagnostic metrics, timing logs, and run metadata for auditability.

5.2.1. Fidelity definition via structural invariants

We define route-level structural fidelity as agreement between M_{ref} and M_{conv}^{τ} under the following declared invariants:

- *Geometry, tolerance-aware*: agreement of connected-node coordinate keys at quantization-grid spacing $\varepsilon_{\mathrm{coord}}$ (Sections 5.6.4 and 5.6.5).
- *Element-incidence connectivity*: agreement of order-invariant incidence signatures derived from neutral element-family tokens and incident node keys (Section 5.6.5).
- *Complementary diagnostics*: an order-sensitive connectivity sentinel based on coordinate keys monitors ordering regressions omitted from the order-invariant primary signature. For tetrahedral elements, a corner-node signed-volume diagnostic provides an additional orientation check; hard-integrity checks are reported separately.
- *Sets*: node- and element-set membership is evaluated using name-agnostic membership signatures. Human-readable set identifiers are evaluated separately using the name-sensitive Name-F_1 diagnostic where comparable names are represented on both sides (Section 5.6.6).
- *Hierarchy counts and reference-point candidates*: exact `Part/PartInstance` counts are compared for nontrivial reference hierarchies on hierarchy-capable targets; candidates use a separate channel where present in the canonical reference artifact (Section 5.6.8).

These invariants define the reported mesh-level structural-fidelity claim for the evaluated channels.

5.2.2. Agreement metrics, determinism, and audit trail

The principal evaluation parameters comprise the grid spacing $\varepsilon_{\mathrm{coord}}$, deterministic tokenization settings, enabled diagnostics, and fixed toolchain versions. Selected effective settings are summarized in Appendix Table A.2; detailed settings, hardware and software versions, and related diagnostic policies and results are documented in the Supplementary Information, Tables S15–S20. The evaluation pipeline records run metadata in the generated configuration and output artifacts for auditability.

To distinguish target-enforced identifier changes from structural information loss, writers can emit structured change reports. These reports document deterministic target-compliance actions such as identifier remapping, name sanitization, truncation, reserved-name handling, and collision resolution. They support the interpretation of name-related deviations without conflating identifier portability with multiset-based signature agreement (Section 4.3).

The precision–recall and balanced F-measure terms follow standard information-retrieval definitions [35]. Here, we apply these measures to finite multisets of deterministic structural signature tokens rather than to unordered retrieved-document sets; the multiset notation follows the usual multiplicity-based interpretation of multisets [36]. For an applicable signature channel X, let X_{ref} and X^{τ}_{conv} denote the reference and converted finite multisets of signature tokens derived from M_{ref} and M^{τ}_{conv}, respectively. Here, τ denotes the evaluated target format and its declared target-specific normalization and tokenization policy. For a finite multiset A, let $c_A(t)$ denote the multiplicity of token t. We use

$$|A|_{\mathrm{m}} = \sum_{t \in \mathrm{supp}(A)} c_A(t), \qquad c_{A \cap_{\mathrm{m}} B}(t) = \min\{c_A(t), c_B(t)\}.$$

We define

$$\mathrm{TP}_X = \left|X_{\mathrm{ref}} \cap_{\mathrm{m}} X^{\tau}_{\mathrm{conv}}\right|_{\mathrm{m}},$$

$$\mathrm{FP}_X = \left|X^{\tau}_{\mathrm{conv}}\right|_{\mathrm{m}} - \mathrm{TP}_X, \qquad \mathrm{FN}_X = \left|X_{\mathrm{ref}}\right|_{\mathrm{m}} - \mathrm{TP}_X.$$

where the corresponding denominators are positive, precision and recall are

$$\mathrm{precision}_X = \frac{\mathrm{TP}_X}{\mathrm{TP}_X + \mathrm{FP}_X},$$

$$\mathrm{recall}_X = \frac{\mathrm{TP}_X}{\mathrm{TP}_X + \mathrm{FN}_X}.$$

For applicable channels with $\mathrm{TP}_X + \mathrm{FP}_X + \mathrm{FN}_X > 0$, the F_1 score is computed as

$$F_{1,X} = \frac{2\,\mathrm{TP}_X}{2\,\mathrm{TP}_X + \mathrm{FP}_X + \mathrm{FN}_X}.$$

Equivalently,

$$F_{1,X} = \frac{2\left|X_{\mathrm{ref}} \cap_{\mathrm{m}} X^{\tau}_{\mathrm{conv}}\right|_{\mathrm{m}}}{\left|X_{\mathrm{ref}}\right|_{\mathrm{m}} + \left|X^{\tau}_{\mathrm{conv}}\right|_{\mathrm{m}}},$$

which is the multiset analogue of the Dice coefficient for the declared signature channel [35–37]. The expression is equivalent to the harmonic mean of precision and recall when both are defined and their sum is positive, but it also makes the no-overlap case explicit: if two non-empty multisets have no matching signatures, $F_{1,X} = 0$.

Within each applicable signature channel of a model-level conversion, TP, FP, and FN are accumulated over token instances; the resulting per-conversion channel score is therefore a micro-averaged token-instance agreement measure. Metrics are reported only after channel applicability and empty-channel cases have been resolved according to the channel-specific policies documented in the Supplementary Information, Table S17. In the machine-readable evaluation artifacts, jointly empty applicable channels are recorded with a vacuous-agreement convention for completeness, whereas the paper tables and aggregate claims distinguish such cases from non-empty evidence-bearing channels using applicability and count fields. If exactly one applicable multiset is empty, $F_{1,X} = 0$ under the above definition. Whichever of precision or recall has a zero denominator follows the declared zero-division convention in the evaluation artifact, while the other equals 0. N/A denotes non-applicability or non-representability in the evaluated target subset, not failed conversion; failed conversions are recorded separately as run-status failures. The resulting F_1 values are therefore channel-specific agreement measures under the declared signatures and policies. They do not, by themselves, imply native deck identity, preservation of native identifiers, solver-semantic equivalence, or solver-result equivalence.

5.3. Z88-pivot-canonicalized route comparison

Direct comparison of native or intermediate artifacts across solver routes would conflate conversion errors with benign representational differences, including renumbering, serialization order, placement encoding, target naming rules, and hierarchy flattening. We therefore use a Z88-pivot-canonicalized route comparison that handles these differences through deterministic normalization and capability-aware diagnostics within the supported mesh-level structural scope. Changes that alter the declared signatures for geometry, element incidence, or set membership propagate into the canonicalized artifacts and are detected by the corresponding metrics.

Within the declared workflow, the Z88 pivot is practical for three reasons. First, the Z88 writer/parser path materializes the evaluated mesh in global coordinates, including cases where the IR retains placement explicitly. Second, it enforces deterministic gap-free renumbering, yielding stable node- and element-identifier sequences for canonical comparison. Third, the Z88 pivot artifact bundle used in this workflow carries the supported sidecar information needed for the evaluated set, hierarchy-count, and reference-point-candidate channels.

Z88 is used here as a deterministic and auditable canonicalization instrument, not as an external reference standard. Agreement under the Z88 pivot operator C_{Z88} is therefore interpreted as agreement under the declared structural invariants and canonicalization rules. It is not evidence of native deck identity, solver-result equivalence, solver-semantic equivalence, or independence from the shared Z88 writer/parser path.

Examples of native-deck channels that are normalized, resolved, re-encoded, or reported diagnostically rather than treated as mesh-structural mismatches include native node and element IDs, raw human-readable set/group names, native set scope and namespace structure, `InstanceName`, coordinate-system declarations, and explicit instance placement syntax. Depending on target capabilities, these channels may be deterministically transformed for target compliance and later interpreted through capability-aware diagnostics.

The primary 92-conversion benchmark uses this Z88-pivot-canonicalized route comparison. A complementary direct FE-IR/JSON structural cross-check is reported in the Supplementary Information, Section S7, to reduce dependence on the shared Z88 writer/parser path for selected structural checks. This complementary path does not replace the primary benchmark and is not interpreted as solver-semantic verification, native deck-identity validation, or a complete second benchmark over all routes.

5.4. Qualitative solver-side sanity checks

As an additional qualitative check, all evaluated converted models were opened in the respective target environments to record import completion and visual plausibility within the declared mesh-level scope. This check supports workflow sanity and importability but is not part of the primary quantitative structural-fidelity evaluation. The checklist is reported in the Supplementary Information, Section S6.2, Table S13.

5.5. Downstream simulation-level evidence (contextual)

A solver-level displacement comparison under controlled load cases was reported previously in [38]. We cite it here as contextual evidence only; it is not part of the present benchmark and does not establish solver-result equivalence. Its scope and relationship to the present study are summarized in the Supplementary Information, Table S14.

5.6. Experimental setup and metric definitions

5.6.1. Hardware and software environment

The reported validation-workflow runtimes were measured on a single workstation with fixed toolchain versions. Hardware and toolchain details are provided in the Supplementary Information, Section S6.4, Tables S15 and S16.

5.6.2. Z88 canonicalization operator and evaluation artifacts

Let $\mathcal{R}$ be the format-dispatching reader that parses a supported solver-deck artifact or artifact bundle f into the IR, and let $\mathcal{W}_\tau$ be the writer that serializes an IR model to a target-format artifact or artifact bundle for τ. For the Z88-pivot-canonicalized route comparison, we use Z88 as the canonicalization pivot and define the corresponding operator as the composition

$$C_{\mathrm{Z88}} := \mathcal{W}_{\mathrm{JSON}} \circ \mathcal{R}_{\mathrm{Z88}} \circ \mathcal{W}_{\mathrm{Z88}} \circ \mathcal{R}, \tag{2}$$

where function composition is read from right to left, $\mathcal{R}_{\mathrm{Z88}}$ denotes the Z88 reader plugin, and $\mathcal{W}_{\mathrm{JSON}}$ emits the neutral JSON evaluation artifact used for comparison. For a supported solver-deck artifact or artifact bundle f, the intermediate Z88 pivot artifact bundle is

$$f^{\mathrm{Z88}} := \mathcal{W}_{\mathrm{Z88}}\big(\mathcal{R}(f)\big), \tag{3}$$

and the canonicalized JSON artifact is

$$C_{\mathrm{Z88}}(f) = \mathcal{W}_{\mathrm{JSON}}\big(\mathcal{R}_{\mathrm{Z88}}(f^{\mathrm{Z88}})\big). \tag{4}$$

Here, f^{Z88} denotes the Z88 pivot artifact bundle produced by the writer (entry point `z88i1.txt` plus any sidecar files within the supported subset). Under fixed toolchain versions and settings, the readers and writers in this composition apply deterministic parsing, normalization, and serialization policies, including the Z88 writer's gap-free renumbering and global positioning of nodal coordinates. Thus, C_{Z88} is deterministic for a fixed input artifact or artifact bundle. For each test case with source artifact or artifact bundle f_{src} and each evaluated target deck format τ, we compute the canonical reference artifact

$$M_{\mathrm{ref}} := C_{\mathrm{Z88}}(f_{\mathrm{src}}), \tag{5}$$

and the converted target artifact or artifact bundle together with its canonicalized representation:

$$\begin{aligned} f_\tau &:= \mathcal{W}_\tau\big(\mathcal{R}(f_{\mathrm{src}})\big), \\ M^\tau_{\mathrm{conv}} &:= C_{\mathrm{Z88}}(f_\tau). \end{aligned} \tag{6}$$

For $\tau = \mathrm{Z88}$, the converted Z88 artifact bundle is still reprocessed through the same canonicalization map rather than compared as native Z88 text. All primary route-comparison metrics in the 92-conversion benchmark are computed between M_{ref} and M^τ_{conv}. The resulting JSON artifacts are evaluated under the declared claim boundary, metric definitions, and pivot interpretation in Sections 1 and 5.2–5.3. Metrics are generated by the evaluation implementation archived on Zenodo [39]. This implementation consumes the canonicalized JSON artifacts, produces per-conversion metrics and aggregated summaries, and records effective settings and versions for auditability (Supplementary Information, Tables S15–S20).

5.6.3. Validation-workflow runtime measurement protocol

For one evaluated model-level conversion with target format τ, the operationally defined three-stage validation-workflow runtime comprises the measured wall-clock times of: (i) reference canonicalization $C_{\mathrm{Z88}}(f_{\mathrm{src}})$, (ii) source-to-target conversion $f_\tau := \mathcal{W}_\tau(\mathcal{R}(f_{\mathrm{src}}))$, and (iii) canonicalization of the converted target artifact or artifact bundle $C_{\mathrm{Z88}}(f_\tau)$. The reference-stage measurement is obtained once per source model and reused for its target conversions; although it can be cached operationally, it is included in each reported total so that the same three-stage cost is reported for every evaluated model-level conversion. Each converter invocation contributing to these stage measurements is executed $r = 5$ times; the first run is treated as a warm-up run, and the median of the remaining $r - 1$ repetitions is used. The reference- and converted-artifact canonicalization stage times are each computed as the sum of the post-warm-up medians for their input-to-Z88 and Z88-to-JSON invocations; the reported total is the sum of these two stage times and the post-warm-up median for the source-to-target invocation. Subsequent loading of the canonical JSON artifacts, computation of structural signatures and diagnostics, and aggregation or writing of evaluation results are not included in this timing endpoint. For the descriptive variability shown in Fig. 4, the four post-warm-up measurements of the five constituent invocations are paired by run index and summed, yielding four run-index-paired totals per evaluated model-level conversion. Their median and sample standard deviation are plotted; because the constituent invocations were timed in separate loops, these paired totals are descriptive summaries rather than separately measured end-to-end executions. Full runtime settings and the corresponding aggregation protocol are provided in the Supplementary Information, Sections S6.4–S6.5.

5.6.4. Connected-node basis

Node fidelity is evaluated on the **connected-node** basis, i.e., for nodes incident to at least one supported solid element. This basis aligns the node metric with the supported element-incidence invariants and avoids route-dependent interpretation of unconnected auxiliary nodes. Unconnected node-like entities are outside the main connected-node geometry metric; where they are classified as reference-point candidates or covered by related diagnostics, they are handled only in those separate channels.

5.6.5. Tolerance-aware, ID- and ordering-invariant geometry and incidence signatures

Tolerance-aware node keys. Nodes are compared using deterministic coordinate quantization with grid spacing $\varepsilon_{\mathrm{coord}}$. For a node n with Cartesian coordinates (x_n, y_n, z_n), the coordinate token is

$$k(n) = H_{128}\big(q_\varepsilon(x_n), q_\varepsilon(y_n), q_\varepsilon(z_n)\big), \tag{7}$$

where q_ε denotes the configured coordinate-quantization operator and H_{128} denotes the deterministic fixed-width tokenization function used by the evaluation pipeline. The configured grid spacing is defined in Section 5.6.7; the exact quantization rule, collision sentinels, and multiscale robustness checks are documented in the Supplementary Information. Because the coordinate key is quantization-based, the geometry score is not interpreted as preserving distinctions at scales below or near the fixed grid spacing.

Element signatures and complementary diagnostics. Elements are compared using order-invariant signatures that combine a neutral element-family token with the sorted multiset of incident connected-node coordinate keys,

$$\sigma(e) = H_{128}\big(t(e),\ \mathrm{sort}_{\mathrm{m}}\big([k(n_1), \ldots, k(n_m)]\big)\big),$$

where $\mathrm{sort}_{\mathrm{m}}$ denotes deterministic sorting with multiplicities retained. The signature is invariant to global renumbering and local node ordering and tests incidence and connectivity preservation under solver-specific identifier and local-order conventions. Because it intentionally removes local node-order differences, element F_1 alone should not be interpreted as evidence of preserved local ordering or element orientation. Ordering regressions are monitored separately using an order-sensitive coordinate-key connectivity sentinel. For tetrahedral elements, a corner-node signed-volume diagnostic provides an additional orientation check; hard-integrity diagnostics are reported in the Supplementary Information. The complete local node-order permutation configuration used for the benchmark is provided in the Supplementary Information, Tables S1 and S2.

5.6.6. Sets and set-name agreement: membership versus names

Set membership is evaluated name-agnostically using node or element member tokens. This avoids false negatives caused solely by deterministic renaming, renumbering, ordering, or capability-driven flattening. For set-membership evaluation, each set is represented by a set-level signature token computed from sorted unique member tokens and, where applicable, the representable instance/scope component. Human-readable set names are excluded from this membership signature and enter only the separate Name-F_1 diagnostic. These membership signatures test agreement of the evaluated set members under the declared capability-aware canonicalization policy. They do not by

themselves imply preservation of native set names, native set scope, `InstanceName`, or solver-specific namespace structure.

Where scope information is representable on both compared sides, it is handled by the capability-aware comparison. Where it is not representable, scope, name, namespace, and `InstanceName` effects are treated as representation diagnostics rather than evaluated membership loss.

Separately, we report agreement of human-readable set identifiers using the name-sensitive diagnostic Name-F_1 only for routes where both source and target representations expose comparable human-readable set names under the declared policy. Name-F_1 is computed from named set-signature tokens that combine the normalized human-readable set identifier with the corresponding membership signature. It is therefore sensitive to both identifier and membership agreement and must be interpreted alongside membership-F_1. Where the separate membership-F_1 score is 1.0, reductions in Name-F_1 localize to identifier non-portability under the declared target policy. Name-F_1 is not a raw native-name preservation metric or a replacement for the separate membership metric. Otherwise, Name-F_1 is reported as not applicable (N/A). Names are normalized to avoid spurious mismatches caused by superficial representation differences. Normalization rules, writer-side interventions, and representative change-report entries are documented in the Supplementary Information, Section S5. Compact range syntax, such as `GENERATE` or `THRU`, is expanded before signature computation, and empty-set signatures are explicit. Empty-channel and non-applicability cases follow the policy in the Supplementary Information, Section S6.6 and Table S17.

Hierarchy-count diagnostics. For hierarchy-capable targets, Parts and Instances are assessed using per-conversion exact-count indicators. A conversion contributes when its canonical reference artifact contains more than one `Part` or more than one `PartInstance`; the aggregate is the percentage with matching reference and converted-artifact counts. This count-only diagnostic does not assess names, relationships, placement semantics, or native hierarchy identity. Flat targets are reported as N/A rather than as failed conversions.

5.6.7. Quantization tolerance and robustness

We use an absolute coordinate quantization-grid spacing $\varepsilon_{\mathrm{coord}} = 0.005$ model units to accommodate ASCII rounding and print-precision differences across formats. This spacing is not a pairwise distance threshold. For interpretation across model scales, normalized spacing-to-scale statistics are reported in the Supplementary Information. Robustness is assessed by recomputing connected-node and element-incidence multiset scores at $\varepsilon_{\mathrm{coord}}/2$ and $2\varepsilon_{\mathrm{coord}}$ (Supplementary Information, Section S6.6, Tables S18 and S19).

5.6.8. Reference-point candidates: capability-aware structural tracking

Some targets do not provide a dedicated reference-point-candidate representation in the evaluated subset, including ANSYS CDB and MSC NASTRAN BDF. In such cases, reference-point candidates are structurally tracked through deterministic writer-derived proxy encoding so that they can be re-identified during re-import under the evaluated writer/parser policy and materialized in the canonical evaluation artifacts.

For compact table labels, RPC denotes reference-point candidates. The evaluation computes RPC F_1 per model-level conversion under the empty-channel policy in Section 5.2.2. Only conversions with a non-empty canonical reference candidate channel contribute to the evidence-bearing summaries in Table 3. The metric compares multisets of tolerance-aware coordinate tokens from the reference and converted channels, retaining multiplicity but excluding native and proxy identifiers. A one-sided empty channel yields $F_1 = 0$, not N/A. For targets without a dedicated candidate representation, deterministic proxies materialize and re-identify candidates under the evaluated writer/parser policy. RPC F_1 therefore measures structural candidate-signature agreement, not native reference-point intent or object identity, proxy-ID preservation, solver-level remote-point behavior, or MPC/RBE/coupling semantics. Section 2.1 and Supplementary Information Section S2 detail the interpretation.

6. Results

The following subsection reports the structural-fidelity results for RQ1. Section 6.2 reports the validation-workflow runtime results for RQ2.

Unless stated otherwise, metrics compare the canonical reference artifact M_{ref} to the canonicalized converted-deck artifact M^{τ}_{conv} produced through the Z88-pivot-canonicalized route comparison defined in Section 5.6.2; node fidelity is reported on the connected-node basis defined in Section 5.6.4. Hierarchy-count diagnostics are interpreted only for nontrivial reference hierarchies on hierarchy-capable targets. Reference-point-candidate metrics are interpreted only where candidates are present in the canonical reference artifact and structurally trackable under the evaluated target writer/parser policy. These results are not universal coverage claims across all benchmark models.

6.1. Route-level structural fidelity within the supported structural scope (RQ1)

This subsection reports route-level structural fidelity for RQ1. The scores are interpreted under the claim boundary in Section 1 and the metric definitions in Section 5.2.

Table 3 summarizes channel-specific results for set, hierarchy-count, and reference-point-candidate channels under the declared target-capability and evidence-selection rules.

Pipeline reliability and integrity diagnostics. All 92 evaluated model-level source-to-target conversions across the 23 benchmark models and 16 source-to-target route classes completed without processing failures (conversion status `ok`; Table 4). Across the 44 conversions contributing at least one nondegenerate tetrahedral-element check, the corner-node signed-volume diagnostic recorded no negative signs in the canonicalized converted-deck artifacts. All reported hard-integrity counters were zero. Across the 92 conversions, no mesh-node coordinate-key hash collision events and no non-convertible/out-of-range or zero-token element–node reference events were recorded in the canonical reference and canonicalized converted-deck artifacts. No out-of-range NodeSet member IDs were recorded in the canonical reference-side evaluations.

Robustness to grid-spacing choice was assessed by recomputing the connected-node and element-incidence multiset scores at $\varepsilon_{\mathrm{coord}}/2$, $\varepsilon_{\mathrm{coord}}$, and $2\varepsilon_{\mathrm{coord}}$. At both the baseline spacing $\varepsilon_{\mathrm{coord}}$ and $\varepsilon_{\mathrm{coord}}/2$, the worst-case scores over the 92 successful conversions were 1.000 for nodes and 1.000 for elements. At $2\varepsilon_{\mathrm{coord}}$, the corresponding worst-case values were 0.998 and 0.994. For the three tested spacings, the bin partitions induced by the declared round-to-nearest rule are not nested: changing the spacing shifts the bin boundaries and can place close reference and converted coordinates on opposite sides of a boundary. Worst-case multiset agreement therefore need not vary monotonically with grid spacing. The small decreases at $2\varepsilon_{\mathrm{coord}}$ are consistent with quantization-grid boundary sensitivity; considered alone, they do not establish a systematic conversion error. Detailed spacing-to-scale statistics and multiscale robustness results are reported in the Supplementary Information, Section S6.6.

Connected-node geometry and element-incidence connectivity. Across all evaluated models and target formats, connected-node geometry achieved $F_1 = 1.0$ under the declared signatures and tolerance-aware comparison policies. The same held for order-invariant element-incidence signatures. Per-model minima are reported in the Supplementary Information, Table S21. Figure S2 shows connected-node geometry, the ordered-connectivity diagnostic, set-membership metrics, and Name-F_1. Route-level worst cases are reported in Supplementary Information, Table S22.

Table 3
Channel-specific set, hierarchy-count, and RPC results by target format for conversions with status `ok`. Set and RPC cells report worst-case F_1; hierarchy cells report exact-count agreement (%). Parentheses report contributing cases over the successful row total: $n_{\text{nonempty}}/n_{\text{conv}}$ for non-empty canonical reference set and RPC channels and $n_{\text{nontrivial}}/n_{\text{conv}}$ for nontrivial canonical reference hierarchies on hierarchy-capable targets. Counts do not denote failures; N/A denotes target-policy non-representability.

Target	n_{conv}	Set-membership F_1		Name-F_1[a]		Hierarchy counts[b]		RPC[c]
		NodeSets (min)	ElemSets (min)	NodeSets (min)	ElemSets (min)	Parts (%)	Instances (%)	RPC F_1 (min)
Abaqus	23	1.000 (12/23)	1.000 (11/23)	1.000 (12/23)	1.000 (11/23)	100.0 (5/23)	100.0 (5/23)	1.000 (4/23)
ANSYS CDB	23	1.000 (12/23)	1.000 (11/23)	0.618 (12/23)	0.071 (11/23)	N/A	N/A	1.000 (4/23)
MSC NASTRAN BDF	23	1.000 (12/23)	1.000 (11/23)	N/A	N/A	N/A	N/A	1.000 (4/23)
Z88	23	1.000 (12/23)	1.000 (11/23)	1.000 (12/23)	1.000 (11/23)	100.0 (5/23)	100.0 (5/23)	1.000 (4/23)

[a] Name-F_1 combines identifier and membership agreement. Because membership-F_1 is 1.000 in this table, its reductions localize to identifier non-portability; neither metric establishes native-name or native-scope preservation.

[b] Parts and Instances report exact-count agreement only for reference artifacts with more than one `Part` or `PartInstance`; single-Part/single-Instance flat wrappers do not contribute. The diagnostic does not test names, relationships, placement semantics, or native hierarchy identity; targets without evaluated hierarchy support are N/A.

[c] RPC denotes reference-point candidates; RPC F_1 is the entity-level structural candidate-signature metric. For ANSYS CDB and MSC NASTRAN BDF targets, reference-point candidates are structurally tracked through writer-derived proxy encoding in the evaluation artifacts. The score does not test native remote-point objects, proxy-ID preservation, MPC/RBE/coupling semantics, or solver-level remote-point behavior.

Table 4
Aggregate validation-pipeline reliability, tetrahedral orientation, and hard-integrity diagnostics over all 92 evaluated source-to-target conversions. "Checked" denotes the number of conversions for which the corresponding diagnostic was applicable and evaluated. The tetrahedral counter is the corner-node signed-volume orientation diagnostic described in Section 5.6.5; its implementation and counting policy are detailed in Supplementary Information, Section S6.6. The counters for mesh-node coordinate-key hash collisions, non-convertible/out-of-range or zero-token element–node references, and out-of-range NodeSet member IDs are hard-integrity checks within the declared structural scope.

Metric	Value
Conversions (ok/total)	92/92 (100%)
Mesh-node coordinate-key hash collision events (events/checked)	0/92
Non-convertible/out-of-range or zero-token element–node reference events (events/checked)	0/92
Conversions with negative tetrahedral corner-node signed volumes (affected/checked)	0/44
Out-of-range NodeSet member IDs in canonical reference-side evaluations (total; maximum per evaluation)	0; 0

Note. Mesh-node coordinate-key collision and element-reference checks were available for all 92 conversions. The tetrahedral corner-node signed-volume diagnostic contributed at least one nondegenerate element check in the canonicalized converted-deck artifacts of 44 conversions; only those nondegenerate checks enter the sign counts. Collision events denote detected hash collisions of the fixed-width mesh-node coordinate-key tokenization. Element-reference totals combine non-convertible/out-of-range IDs and in-range references without a nonzero coordinate token across canonical reference and canonicalized converted-deck artifacts. The counter for out-of-range NodeSet member IDs covers canonical reference-side evaluations only.

The order-sensitive connectivity sentinel showed no deviations in the ordered coordinate-key connectivity signatures for the supported Tet4/Tet10 and Hex8/Hex20 element families. These signatures retain the local sequence of quantized coordinate keys but do not compare native node identifiers or deck element order; coincident nodes, or nodes assigned the same coordinate key, can remain indistinguishable. The tetrahedral corner-node signed-volume diagnostic reported no negative signs, and the hard-integrity diagnostics reported no events (Supplementary Information, Table S20). Because the primary element-incidence signature is intentionally order-invariant, these complementary diagnostics are interpreted separately from element F_1.

Set membership, names, hierarchy, and reference-point candidates. Across all reported evaluations with a non-empty canonical reference set channel, NodeSet and ElementSet membership signatures agreed exactly (membership $F_1 = 1.0$; Table 3; Supplementary Information, Table S23). These membership metrics are name-agnostic and test agreement of the evaluated set members under the declared capability-aware canonicalization policy. They do not imply preservation of native set names, native set scope, `InstanceName`, or solver-specific namespace structure.

The name-sensitive set-signature diagnostic, Name-F_1, reached 1.0 for Abaqus and Z88 targets. For ANSYS CDB targets, lower Name-F_1 values occurred for Abaqus-derived sets, with worst-case values of 0.618 for NodeSets and 0.071 for ElementSets. These reductions are attributable to deterministic MAPDL component-name normalization, target naming restrictions, shared namespace constraints, and collision-driven disambiguation. The evaluated set-membership signatures still agreed. Reduced Name-F_1 therefore indicates identifier non-portability under target constraints, not loss of evaluated set members. For MSC NASTRAN BDF targets, Name-F_1 is not applicable because the evaluated BDF subset does not provide a comparable human-readable set-name channel. Representative deterministic writer-side identifier interventions are documented, where available, in separate or embedded change reports; examples are provided in the Supplementary Information, Section S5.

For the five contributing reference models, part and instance counts matched exactly for Abaqus and Z88 targets. This count-only diagnostic does not test hierarchy names, relationships, placement semantics, or native hierarchy identity. It is N/A for ANSYS CDB and MSC NASTRAN BDF because these targets are treated as flat within the evaluated subset.

The reference-point-candidate coordinate-token multisets agreed exactly ($F_1 = 1.0$) wherever the canonical reference candidate channel was non-empty. For targets without a dedicated candidate representation, this score reflects structural candidate-signature agreement and re-identification through deterministic proxy encoding under the interpretation defined in Section 2.1 and Table 3.

Table 3 parentheses report contributing cases, not failures. In each target row, 12/23, 11/23, and 4/23 conversions had non-empty canonical reference NodeSet, ElementSet, and RPC channels, respectively. The same five reference models had multiple `Parts` and multiple `PartInstances`; only hierarchy-capable targets receive corresponding count-agreement results. N/A denotes non-representability under the evaluated target policy. For each contributing comparison with $F_1 = 1.0$, $\text{FP}_X = \text{FN}_X = 0$; the score denotes exact signature-multiset equality under the declared tokenization policy, not a rounded aggregate.

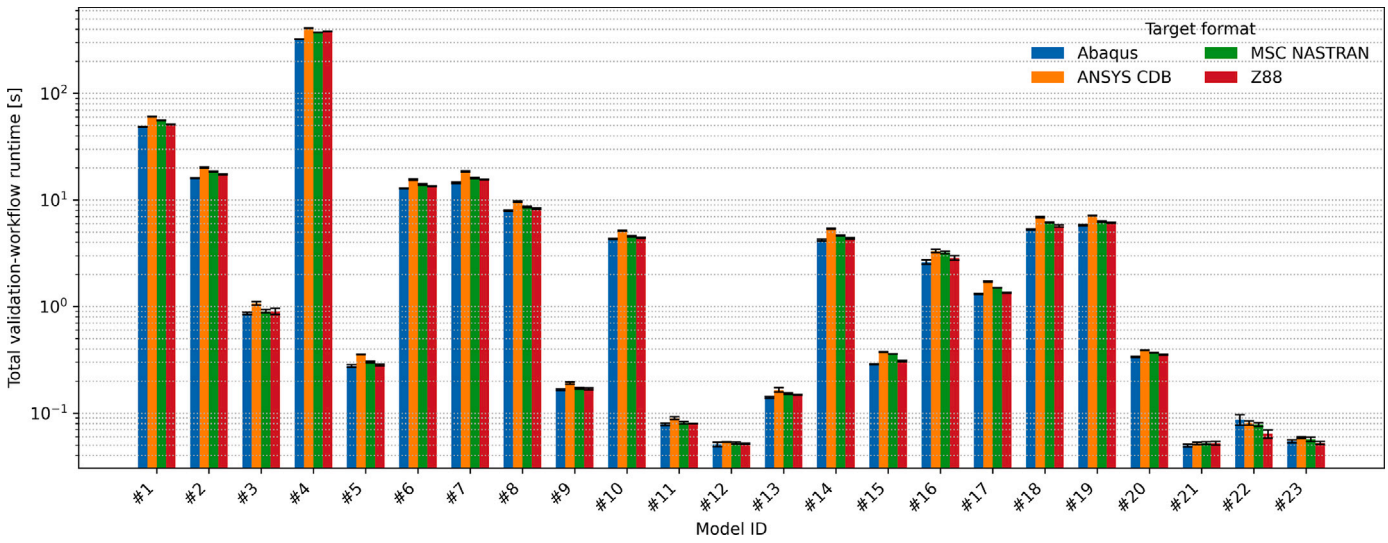


Fig. 4. Descriptive total validation-workflow runtime per test model (Model ID) and target format. Each run-index-paired total includes source-to-target conversion, reference canonicalization, and converted-target canonicalization. Bars show the median of four post-warm-up paired totals; error bars denote their sample standard deviation. Constituent invocations were timed in separate loops, so the paired totals summarize run-to-run variability rather than separately measured end-to-end executions. Subsequent JSON loading, metric computation, and report generation are not timed. The y-axis is shown on a logarithmic scale.

6.2. Validation-workflow runtime (RQ2)

Fig. 4 reports descriptive validation-workflow runtime summaries per model and target format using the run-index-paired totals defined in Section 5.6.3. These totals include reference canonicalization, source-to-target conversion, and canonicalization of the converted target artifact or artifact bundle. They therefore represent the operationally defined combined conversion-and-canonicalization cost per evaluated model-level conversion, not either the source-to-target pass alone or a separately timed end-to-end validation execution. Hardware, software, and toolchain details are provided in the Supplementary Information, Section S6.4, Tables S15 and S16.

Using the reported-total definition in Section 5.6.3, total validation-workflow runtime across all 92 evaluated model-level conversions ranged from 0.05 s to 408 s, with a median of 1.42 s. On the test workstation, the maximum reported three-stage total, observed for the largest benchmark model, remained below 7 min. These observations indicate that the measured conversion-and-canonicalization stages are compatible with scripted validation and occasional interactive checks within the evaluated model-size range.

The reference-canonicalization stage can be cached per source model in repeated model-level conversions. It is nevertheless included in each reported total, so the reported measure is conservative with respect to reference-stage caching. As descriptive stage-level summaries, the median source-to-target conversion time was 0.309 s, the median reference-canonicalization time was 0.517 s, and the median converted-deck-canonicalization time was 0.616 s. These stage-level summaries separate the cost of the actual conversion pass from the additional cost of canonicalization-based validation. Raw invocation-level stage-timing outputs are provided with the archived evaluation artifacts [40].

7. Discussion

The methodological core is a neutral hierarchical IR and an evaluation framework centered on the Z88-pivot-canonicalized route comparison. Within the declared 3D solid-element subset, the evaluation covers connected-node geometry, element-incidence connectivity, set membership, hierarchy-count diagnostics for nontrivial reference hierarchies on hierarchy-capable targets, and structurally tracked reference-point candidates under the declared signatures and policies.

The runtime results further indicate that the measured conversion-and-canonicalization stages remain compatible with scripted validation and occasional interactive checks within the evaluated model-size range. Because the reported totals include reference canonicalization and converted-deck canonicalization, they should be interpreted as the defined three-stage validation-workflow measure rather than as isolated source-to-target converter throughput.

7.1. Methodological role of the evaluation framework

The evaluation framework centered on the Z88-pivot-canonicalized route comparison is part of the methodological contribution, not merely a post-processing step. It defines what structural agreement means for the declared mesh-level subset, which representation differences are intentionally normalized, which channels are evaluated quantitatively, and which target-dependent concepts are reported as not applicable or diagnostic rather than as conversion failures.

This distinction is necessary because the considered formats differ in syntax, identifier models, hierarchy and placement encodings, coordinate-system conventions, set-name support, and local node-order conventions. Native deck equality would therefore be neither portable nor meaningful. The primary comparison instead uses declared structural signatures and metrics: tolerance-aware connected-node coordinate signatures, order-invariant element-incidence signatures, name-agnostic set-membership signatures, the separate name-sensitive Name-F_1 diagnostic, hierarchy-count diagnostics, and capability-aware reference-point-candidate channels. An order-sensitive coordinate-key connectivity sentinel, a tetrahedral corner-node signed-volume diagnostic, and hard-integrity checks are reported separately as complementary diagnostics.

Z88 is used as a deterministic and auditable canonicalization instrument, not as an external reference standard. Agreement after the Z88 pivot means agreement under the declared structural invariants, tolerance settings, and canonicalization rules; it does not establish native deck identity, solver-semantic equivalence, solver-result equivalence, or independence from the shared pivot path. The complementary direct FE-IR/JSON structural cross-check reported in the Supplementary Information reduces dependence on the shared Z88 writer/parser path for selected structural checks, but remains dependency-reducing structural evidence rather than solver-semantic verification or a complete second benchmark over all routes.

What $F_1 = 1.0$ does and does not establish. An $F_1 = 1.0$ score in a reported, evidence-bearing pivot-canonicalized metric channel indicates agreement between the canonicalized reference artifact and the canonicalized converted-deck artifact under that channel's declared

structural signature. This applies, for example, to connected-node geometry and element-incidence signatures under their declared tolerance and ordering policies. Claims for set-membership and reference-point-candidate signatures refer only to reported non-empty reference channels. Such agreement provides structural evidence for mesh-level structural fidelity within the declared comparison policy.

However, these scores do not establish native deck identity, full analysis-model equivalence, solver-semantic equivalence, solver-result equivalence, or independence from the shared canonicalization path. Analysis outcomes further depend on materials, sections, surfaces, loads, boundary conditions, contacts, MPC/RBE/coupling definitions, solver controls, and other solver-level settings outside the converted subset; solver results themselves are not converted or compared. The canonicalization procedure also intentionally resolves, normalizes, or re-encodes solver-facing identifiers, native set-name and namespace representations, placement encodings, and coordinate-system declarations where required for structural comparison. Workflows with such requirements therefore need requirement-specific validation criteria and, as applicable, additional mapping artifacts or an extended converted subset.

The reported $F_1 = 1.0$ scores should also not be read as preservation of every node-like entity or every geometric distinction between coordinates assigned the same quantized key. The node metric is evaluated on the connected-node basis. Unconnected auxiliary nodes are outside that metric; where they are classified as reference-point candidates or covered by a related diagnostic, they are handled only in those separate channels. Similarly, the main element-incidence signature is intentionally order-invariant, so ordering and tetrahedral orientation are assessed using the separate order-sensitive element-connectivity sentinel and tetrahedral corner-node signed-volume diagnostic rather than inferred from element F_1 alone; hard-integrity diagnostics are also reported separately.

Practical interpretation of capability-aware channel reporting. Capability-aware channel reporting separates structural-signature agreement, identifier portability, and hierarchy-count diagnostics. This distinction is important because solver-deck formats differ not only in syntax but also in which channels they can represent. Where a channel is reported as N/A because the evaluated target subset lacks a comparable representation, this denotes capability-aware non-applicability, not a conversion failure.

Reference-point candidates are interpreted only as structurally tracked point-like candidates. Where the evaluated target subset lacks a dedicated reference-point-candidate representation, candidates can be carried through deterministic proxy encoding and re-identified for structural comparison under the evaluated writer/parser policy. This supports structural candidate tracking, but does not imply native reference-point object identity, proxy-ID preservation, MPC/RBE/coupling semantics, or solver-level remote-point behavior.

7.2. Writer-side interventions: identifiers versus evaluated membership

The results illustrate why evaluated set membership and human-readable identifier portability should be interpreted separately in automated exchange workflows. Across all reported evaluations with a non-empty corresponding NodeSet or ElementSet channel in the canonical reference artifact, the membership signatures agreed exactly ($F_1 = 1.0$) under the declared policy. This provides benchmark evidence that the evaluated set memberships were retained within the declared mesh-level scope, independently of human-readable identifier portability.

In contrast, human-readable set identifiers are not uniformly portable across target formats. Because the corresponding membership-F_1 values were 1.0, the reduced Name-F_1 values observed for Abaqus-derived sets in ANSYS CDB targets reflect identifier non-portability rather than loss of evaluated set members. The reductions arise from deterministic normalization required by MAPDL constraints on component names and the shared component namespace, including collision-driven disambiguation. For MSC NASTRAN BDF targets, Name-F_1 is reported as N/A because the evaluated BDF subset does not expose a comparable human-readable set-name channel.

Where generated and sufficiently detailed, writer change reports can serve as alias tables between source-facing and target-facing identifiers for workflows that depend on human-readable identifiers. Such reports make recorded target-driven renaming, truncation, sanitization, reserved-name handling, collision resolution, and other deterministic target-compliance actions traceable. The membership-F_1 metric assesses agreement of the evaluated set-membership signatures under the declared policy, whereas Name-F_1 serves as an identifier-portability diagnostic under target constraints.

Thus, writer-side interventions are not hidden post-processing steps but part of the deterministic export contract. They make target artifacts compliant with the constraints of the evaluated target subset; agreement in the claimed structural channels is assessed separately through the corresponding reported signatures and capability-aware diagnostics. This distinction is central to interpreting the reported channel results: identifier-level deviations are expected when target formats impose stricter naming or namespace rules, whereas the connected-node geometry, element-incidence, and set-membership signatures serve to detect structural deviations within their declared scope and sensitivity.

7.3. Limitations and threats to validity

The results should be interpreted within the declared mesh-level structural subset and the validation policy defined in Section 5.2. The limitations below delimit the conditions under which the reported claim of mesh-level structural interoperability is valid.

Scope of the converted subset. The reported metrics quantify mesh-level structural interoperability only. They do not evaluate full analysis-model equivalence, solver-semantic equivalence, solver-result equivalence, or preservation of solver-level analysis entities. Shell and beam elements, other element families, materials, sections, surfaces, loads, boundary conditions, contacts, MPC/RBE/coupling semantics, solver controls, solver results, and full solver-level remote-point behavior remain outside the evaluated subset. The reported hierarchy result is limited to exact `Part` and `PartInstance` count agreement for five reference models with nontrivial canonical hierarchies on hierarchy-capable Abaqus targets and Z88 targets using the evaluated hierarchy sidecar; it does not test hierarchy names, part–instance relationships, placement semantics, or native hierarchy identity. Extending the converted and evaluated subset to additional model entities or hierarchy properties would require, as applicable, corresponding IR support, parser/writer mappings, target-capability policies, and validation criteria.

Canonicalization versus native deck identity. The Z88-pivot-canonicalized route comparison intentionally normalizes or re-encodes solver-specific identifiers, placement representations, coordinate-system references, set namespaces, and other target-dependent serialization details to obtain comparable structural artifacts. Agreement after canonicalization supports agreement under the declared invariants and rules, not strict native deck identity, source-ID preservation, native coordinate-system declaration preservation, or stability of human-readable identifiers. Workflows that depend on these properties therefore need dedicated validation criteria and, where appropriate, mapping artifacts or a broader converted subset.

Non-unique coordinate keys. The geometry signatures use quantized coordinate keys. If a model contains intentionally coincident nodes or distinct nodes whose coordinates map to the same quantized coordinate triple, coordinate-only keys are non-unique. Topology changes that reassign connectivity among nodes sharing the same coordinate key could then be masked. For such cases, the comparison should be complemented by neighborhood-aware or graph-based matching, explicit ID mapping, or additional geometry/topology checks.

Fixed quantization-grid spacing and scaling regimes. The reported coordinate signatures use a fixed absolute quantization-grid spacing $\epsilon_{\text{coord}} = 0.005$ model units, selected for the evaluated benchmark to accommodate ASCII-format precision effects. This setting may mask very small deviations or over-penalize benign rounding noise in extreme scaling regimes. Adaptive or scale-aware grid spacing, or additional geometric checks, may be required for very small features, large coordinate ranges, or precision-sensitive near-contact geometries.

Element-family and orientation coverage. The evaluated element-family mappings and local-node-order safeguards cover Tet4/Tet10 and Hex8/Hex20 solid elements only. The main element-incidence signature is intentionally order-invariant and does not by itself establish orientation preservation. Ordering regressions are monitored by an order-sensitive coordinate-key connectivity sentinel. For tetrahedral elements, a corner-node signed-volume diagnostic provides an additional orientation check. It uses only the four corner nodes for Tet4 and Tet10 and is not a complete higher-order Tet10 validity analysis. Shell, beam, mixed-dimensional, or other element families would require additional canonical element definitions, solver-specific local node-order permutations, and element-specific validity diagnostics.

Pivot-path dependence. The primary benchmark uses a shared Z88 pivot operator to generate both M_{ref} and M^{τ}_{conv}. This design makes route-level structural comparison deterministic and auditable, but it cannot fully exclude pivot-specific common-mode errors. The primary comparison is supplemented by an order-sensitive coordinate-key connectivity sentinel, a tetrahedral corner-node signed-volume diagnostic, solver-side import sanity checks, and hard-integrity diagnostics. For selected structural checks, the complementary direct FE-IR/JSON structural cross-check reported in the Supplementary Information reduces dependence on the shared Z88 writer/parser path. The direct path bypasses that shared path but still depends on the native parsers, shared FE-IR, JSON export, IR/JSON canonicalization, and metric definitions; it remains bounded by the declared claim scope. It provides dependency-reducing structural evidence but is neither a complete second benchmark nor a replacement for the primary Z88-pivot-canonicalized route comparison; it does not provide solver-semantic verification.

Identifier-policy sensitivity. Name-F_1 is sensitive to both membership agreement and the deterministic target-specific naming and namespace policy. Where the corresponding membership-F_1 value is 1.0, a lower Name-F_1 value reflects identifier non-portability under that policy rather than loss of evaluated set members. Alternative naming policies may therefore change Name-F_1 without changing membership-F_1. Neither metric establishes preservation of native set names, native set scope, `InstanceName`, or solver-specific namespace structure. For workflows that depend on human-readable identifiers, writer change reports, where generated, provide an audit trail of recorded interventions and, where sufficiently detailed, can also serve as alias tables between source-facing and target-facing identifiers.

Reference-point-candidate interpretation. Reference-point candidates are evaluated as structural point-like candidates in a separate metric channel. Candidate detection is structural only and may also classify generic unconnected auxiliary nodes as candidates. Where the evaluated target subset lacks a dedicated reference-point-candidate representation, candidates may be carried through deterministic proxy encoding and re-identified for structural comparison under the evaluated writer/parser policy. The reference-point-candidate F_1 metric therefore reflects structural candidate-signature agreement only; it does not test native reference-point intent, native object identity, proxy-ID preservation, solver-level remote-point behavior, MPC semantics, RBE semantics, or coupling semantics.

Benchmark and reproducibility boundary. The benchmark covers 23 models and 92 evaluated model-level source-to-target conversions across the declared 3D solid-element subset. The model set comprises ten publicly redistributable models, three licensed-user fetch-only Abaqus examples, and ten proprietary or otherwise non-redistributable models, but it is not exhaustive over solver-deck dialects, modeling practices, element combinations, identifier patterns, or coordinate scales. A complete public rerun of the full benchmark is not possible from the released materials alone because raw inputs for 13 models are not included and the converter implementation is not public. The public archive supports inspection and file-integrity verification against the published checksums for the released inputs and derived artifacts, while the full reported benchmark remains dependent on the 13 non-redistributed inputs and the non-public converter implementation.

Runtime generalization. Reported validation-workflow runtimes are specific to the stated workstation, toolchain, model set, and target formats. They include reference canonicalization, source-to-target conversion, and converted-deck canonicalization. They exclude subsequent JSON loading, structural-signature and diagnostic computation, and result aggregation or report generation, and should not be interpreted as pure converter throughput or as a scaling guarantee for substantially larger models, different storage systems, different parser/writer implementations, or different hardware.

7.4. Outlook: toward broader and standards-based interoperability

The present work therefore provides a deterministic and auditable basis for exchange within the declared mesh-level structural scope. A natural next step is to broaden the converted and evaluated subset toward analysis-preparation workflows by adding structural and analysis-adjacent entities, such as surfaces, sections, and selected boundary-condition constructs. This would entail IR extensions, parser and writer mappings, target-capability policies, and validation criteria tailored to the intended use. Once that subset includes the analysis definitions required for a given test case, representative solver-output regression tests could provide additional, solver- and configuration-dependent validation evidence; they would not substitute for structural-fidelity reporting.

The IR, mapping rules, writer change reports, and canonicalized structural artifacts can also serve as a lightweight foundation for defining workflow-specific exchange profiles and regression test suites. Such profiles would make explicit which entities, target constraints, normalization rules, and comparison signatures are covered by a given exchange workflow. This would support incremental extension of the converter while keeping each interoperability claim tied to a declared subset and an auditable validation policy.

A further direction is to define and validate a mapping from the pragmatic IR to relevant STEP AP209 information structures for multi-disciplinary analysis and design. This could support embedding transferred mesh and structural information in a richer product and analysis context. However, standards-based integration does not remove the need to declare the implemented subset, follow applicable implementation guidance, and establish explicit validation targets. Without these safeguards, interoperability issues may simply shift from solver-deck mappings to the standards layer. Future work should therefore combine broader data-model alignment with the same explicit scoping, deterministic normalization, and route-level validation principles used in the present study.

8. Conclusion

Across 23 finite-element models, ranging up to 5.02 million nodes and 3.22 million elements, and 92 evaluated source-to-target conversions, the proposed neutral hierarchical IR supported mesh-level structural interoperability among Abaqus, ANSYS Mechanical (CDB), MSC NASTRAN (BDF), and Z88 within the declared 3D solid-element subset. All evaluated conversions completed without processing failures. In the declared Z88-pivot-canonicalized route comparison, tolerance-aware connected-node geometry and order-invariant element-incidence signatures reached $F_1 = 1.0$ throughout; the complementary order-sensitive

connectivity sentinel showed no deviations, the tetrahedral corner-node signed-volume diagnostic recorded no negative signs, and the hard-integrity diagnostics reported no events. Evaluated set-membership signatures agreed in all reported non-empty canonical reference set channels, and reference-point candidates were structurally tracked or re-identified where present. Observed reductions in Name-F_1 reflected target naming and namespace constraints rather than loss of evaluated set membership. Three-stage validation-workflow runtimes ranged from 0.05 s to 408 s, with a median of 1.42 s; the measured conversion-and-canonicalization stages remained compatible with scripted validation and occasional interactive checks within the evaluated model-size range.

These results are bounded by the declared mesh-level scope. Unsupported element families, solver-level analysis entities and semantics, and full remote-point/coupling behavior remain outside the converted and evaluated subset. The evaluation does not establish native deck identity, full analysis-model equivalence, deck-level identifier stability, native coordinate-system declaration preservation, solver-semantic equivalence, or solver-result equivalence. The reported F_1 values therefore indicate channel-specific agreement under declared structural signatures, tolerances, and canonicalization policies.

Within this explicitly bounded scope, the IR and validation workflow provide a deterministic and auditable basis for cross-solver mesh-level structural interoperability. The same principles — explicit scoping, target-aware normalization, structural signatures, and capability-aware reporting — can support future extensions through clearly defined workflow-specific exchange profiles and validation targets.

CRediT authorship contribution statement

Peter Robert Grohmann: Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Data curation, Conceptualization. **Stephan Tremmel:** Writing – review & editing, Supervision.

Code availability

The converter implementation is not publicly available due to contractual restrictions and third-party intellectual property. The evaluation implementation used to compute the primary metrics and the reported three-stage validation-workflow runtime summaries for the Z88-pivot-canonicalized route comparison is archived on Zenodo [39]; it is not the converter, and end-to-end execution of the primary benchmark workflow requires the non-public converter implementation and a populated benchmark input tree. Together with the input restrictions described above, the public materials support inspection and checksum verification of released inputs and derived artifacts, but they do not enable a complete rerun of the full 23-model benchmark from public raw input decks alone. The accompanying Zenodo dataset record includes a structured index of the released canonical JSON outputs, parser/writer change reports, diagnostic files, raw invocation-level stage-timing data, SHA-256 checksums and manifests, and the public evidence addendum; the index states the archive location and public coverage of each artifact class. The separately versioned Zenodo software record provides the evaluation implementation described above.

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process

During the preparation of this work, the authors used ChatGPT and Codex (OpenAI) to improve language, clarity, and readability and to support spelling, grammar, terminology, cross-document consistency, and submission-guideline checks. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication. No generative AI tools were used to create or alter any figures or artwork. The scientific content, interpretations, and conclusions of this work are based on the authors' original work and judgment. AI-assisted tools for code completion and generation (JetBrains AI Assistant in CLion) were used to support software development tasks; all code was reviewed and validated by the authors.

Funding

A foundational prototype was developed within research project FVA 972 I of the Forschungsvereinigung Antriebstechnik e.V. (FVA). The extensions and evaluation methodology reported here were developed after project completion as part of the first author's doctoral research; no additional dedicated third-party funding was received.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank the Forschungsvereinigung Antriebstechnik e.V. (FVA) and the FVA working group "Berechnung und Simulation (BUS)" for their support within the cooperative research project FVA 972 I, in which a foundational FE converter prototype was developed. The extensions and evaluation reported in this manuscript were developed after completion of FVA 972 I and do not constitute an official project deliverable.

Table A.1
Mapping of solver-specific element designations to neutral internal element families used by the converter.

FE software	Designation	Details in solver	Internal type
Abaqus	C3D8	Hexahedron (8 nodes)	Hex8
	C3D20	Hexahedron (20 nodes)	Hex20
	C3D4	Tetrahedron (4 nodes)	Tet4
	C3D10	Tetrahedron (10 nodes)	Tet10
NASTRAN	CHEXA	Hexahedron (8/20 nodes)[a]	Hex8/Hex20
	CTETRA	Tetrahedron (4/10 nodes)[a]	Tet4/Tet10
ANSYS	SOLID185	Hexahedron (8 nodes)	Hex8
	SOLID186	Hexahedron (20 nodes)	Hex20
	SOLID285	Tetrahedron (4 nodes)	Tet4
	SOLID187	Tetrahedron (10 nodes)	Tet10
Z88	Type 1	Hexahedron, linear (8 nodes)	Hex8
	Type 10	Hexahedron, quadratic (20 nodes)	Hex20
	Type 17	Tetrahedron, linear (4 nodes)	Tet4
	Type 16	Tetrahedron, quadratic (10 nodes)	Tet10

Note: Abaqus element names may carry formulation qualifiers (e.g., C3D8R, C3D8RH, C3D10M). In the supported subset of this work, such qualifiers are stripped and mapped to the corresponding base family by node count.

[a] In MSC NASTRAN, CHEXA/CTETRA cover linear and quadratic variants; the node count disambiguates the internal type.

Appendix A. Element type mappings

Solver designations are mapped to neutral internal element families (Table A.1). Local node ordering is handled via an Abaqus-based canonical ordering in the IR (Section 3.3.2). The complete solver-specific node-order permutation tables used in the experiments are provided in the Supplementary Information (Tables S1 and S2).

A.1. Key evaluation settings

Table A.2 summarizes settings for the route comparison and the reported validation-workflow runtime. Detailed parser subsets, writer-side intervention examples, and robustness and diagnostic tables (spacing-to-scale statistics, multiscale robustness, and tetrahedral corner-node signed-volume diagnostics) are provided in the Supplementary Information.

Table A.2
Key settings for route-comparison metrics and validation-workflow runtime.

Setting	Value
Coordinate quantization spacing ϵ (EPS_COORD)	0.005
Hash width (HASH_BITS)	128-bit
Node-fidelity basis (NODE_BASIS)	`connected`
Timing repetitions (REPEATS)	5
Timing warmup runs (WARMUP)	1
NodeSet member-token policy for shared mesh-node/RPC IDs	RPC coordinate token (`collision`)
Multiscale eps factors	0.5, 1.0, 2.0
Multiscale eps values	$\epsilon/2 = 0.0025$, $\epsilon = 0.005$, $2\epsilon = 0.01$

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cad.2026.104153.

Data availability

Of the 23 benchmark models, the reproducibility package [40] includes raw benchmark inputs for all 10 redistributable models and the publicly released derived evaluation artifacts. Three additional Abaqus example decks (model IDs 6, 7, and 8) are available to licensed Abaqus users through the Abaqus Fetch utility but are not included in the archive because of vendor licensing restrictions. The raw input decks for the remaining 10 models are proprietary or otherwise non-redistributable because of third-party intellectual property or confidentiality restrictions; access requests should be directed to the relevant rights holders in accordance with their terms.

References

[1] Afazov SM, Becker AA, Hyde TH. Development of a Finite Element Data Exchange System for chain simulation of manufacturing processes. Adv Eng Softw 2012;47(1):104–13. http://dx.doi.org/10.1016/j.advengsoft.2011.12.011, Available from: https://doi.org/10.1016/j.advengsoft.2011.12.011. [Accessed 19 January 2026].

[2] Dassault Systèmes (SIMULIA). Abaqus 2024 User Assistance (documentation set referenced in this paper). 2023, Available from (institutional mirror): https://docs.software.vt.edu/abaqusv2024/English/. [Accessed 17 January 2026].

[3] MSC Software Corporation (Hexagon). MSC Nastran 2022.4 reference guide (bulk data entries incl. GRID, CTETRA/CHEXA, RBE2, RBE3). Hexagon Nexus Documentation Center; 2022, Available from: https://nexus.hexagon.com/documentationcenter/en-US/bundle/MSC_Nastran_2022.4_Reference_Guide/resource/MSC_Nastran_2022.4_Reference_Guide.pdf. [Accessed 17 January 2026].

[4] Rieg F, Hackenschmidt R, Alber-Laukant B. Finite Elemente Analyse für Ingenieure: Grundlagen und praktische Anwendungen mit Z88Aurora. 6th ed.. Munich, Germany: Carl Hanser Verlag; 2019.

[5] Dassault Systèmes (SIMULIA). Abaqus/CAE User Assistance: Importing a model from a Nastran input file. 2023, Release 2024. Available from (institutional mirror): https://docs.software.vt.edu/abaqusv2024/English/SIMACAECAERefMap/simacae-c-impnastran.htm. [Accessed 17 January 2026].

[6] Dassault Systèmes (SIMULIA). Abaqus User Assistance: Translating Nastran Data to Abaqus Files (supported entries; others skipped). 2023, Release 2024. Available from (institutional mirror): https://docs.software.vt.edu/abaqusv2024/English/SIMACAEEXCRefMap/simaexc-c-nasabaproc.htm. [Accessed 17 January 2026].

[7] Dassault Systèmes (SIMULIA). Abaqus user assistance: Translating abaqus files to nastran bulk data files (supported entities). 2023, Release 2024. Available from (institutional mirror): https://docs.software.vt.edu/abaqusv2024/English/SIMACAEEXCRefMap/simaexc-c-tonasabaproc.htm. [Accessed 17 January 2026].

[8] ANSYS, Inc. ANSYS mechanical APDL element reference (node numbering/ordering for SOLID185/186/187/285). 2024, Release 2024 R2 (v242). Available from: https://ansyshelp.ansys.com/public/Views/Secured/corp/v242/en/ans_elem/Hlp_E_SOLID185.html, https://ansyshelp.ansys.com/public/Views/Secured/corp/v242/en/ans_elem/Hlp_E_SOLID186.html, https://ansyshelp.ansys.com/public/Views/Secured/corp/v242/en/ans_elem/Hlp_E_SOLID187.html, https://ansyshelp.ansys.com/public/Views/Secured/corp/v242/en/ans_elem/Hlp_E_SOLID285.html. [Accessed 09 June 2026].

[9] Iványi P. Finite element mesh conversion based on regular expressions. Adv Eng Softw 2012;51:20–39. http://dx.doi.org/10.1016/j.advengsoft.2012.05.002.

[10] Moita GF, Almeida PEM, Oliveira DMLM, Pinheiro MC. A data exchange neutral format for finite element codes using an intelligent element classification. Rev CIATEC-UPF 2011;3(1):48–64. http://dx.doi.org/10.5335/ciatec.v3i1.2190.

[11] ANSYS, Inc. Ansys workbench user's guide: External data — Importing a CDB file as input (NBLOCK/EBLOCK import). 2024, Release 2024 R2 (v242). Available from: https://ansyshelp.ansys.com/public/Views/Secured/corp/v242/en/wb2_help/wb2h_MechLoadapp.html. [Accessed 09 June 2026].

[12] ANSYS, Inc. Ansys mechanical user's guide: Remote points. 2024, Release 2024 R2 (v242). Available from: https://ansyshelp.ansys.com/public/Views/Secured/corp/v242/en/wb_sim/ds_remote_point.html. [Accessed 09 June 2026].

[13] Dassault Systèmes (SIMULIA). Abaqus keywords reference: *COUPLING. In: Abaqus user assistance. 2023, Release 2024. Available from (institutional mirror): https://docs.software.vt.edu/abaqusv2024/English/SIMACAEKEYRefMap/simakey-r-coupling.htm. [Accessed 17 January 2026].

[14] ANSYS, Inc. ANSYS mechanical user's guide: Converting named selection groups to mechanical APDL application components. 2024, Release 2024 R2 (v242). Available from: https://ansyshelp.ansys.com/public/Views/Secured/corp/v242/en/wb_sim/ds_named_selection_to_ansys_component.html. [Accessed 09 June 2026].

[15] Chair of Engineering Design and CAD. Z88Aurora® version 6 [software]. Bayreuth, Germany: University of Bayreuth; 2023, Available from: https://en.z88.de/download-z88aurora/. [Accessed 08 August 2026].

[16] Dassault Systèmes (SIMULIA). Abaqus/CAE user assistance: Exporting model data to nastran bulk data file format. 2023, Release 2024. Available from (institutional mirror): https://docs.software.vt.edu/abaqusv2024/English/SIMACAECAERefMap/simacae-c-dbsexportnastran.htm. [Accessed 07 January 2026].

[17] Altair Engineering, Inc. Abaqus to Nastran conversion mapping. 2025, HyperWorks Help 2026. Available from: https://help.altair.com/hwdesktop/hwx/topics/conversion_between_solvers/abaqus_nastran_conversion_mapping.htm. [Accessed 05 June 2026].

[18] Kortelainen J, Klinge P, Katajamäki K. Managing mesh-based data in a semantic database. VTT research report no. VTT-R-03154-11, VTT Technical Research Centre of Finland; 2011, Available from: https://cris.vtt.fi/ws/portalfiles/portal/41830646/VTT_R_03154_11.pdf. [Accessed 06 February 2026].

[19] Structural Dynamics Research Laboratory (SDRL). Universal file formats (UFF/UNV) — complete UFF library, index (ASCII files). University of Cincinnati, Available from: https://www.ceas3.uc.edu/sdrluff/all_files.php?list=asc. [Accessed 17 January 2026].

[20] Autodesk. Moldflow insight help: Importing IDEAS universal model files — Supported universal datasets (incl. Data set 2412 "Elements"). 2018, Available from: https://help.autodesk.com/cloudhelp/2018/ENU/MoldflowInsight/files/GUID-03956F5F-D1C7-4E75-99ED-73F1E2ECF225.htm. [Accessed 17 January 2026].

[21] Altair Engineering, Inc. HyperView interfacing: UNV files (supported blocks incl. nodes, elements, results, and set information). 2025, HyperWorks Help 2026. Available from: https://help.altair.com/hwdesktop/hwx/topics/pre_processing/interfacing/hv_interfacing_unv_files_r.htm. [Accessed 06 February 2026].

[22] Schlömer N. Meshio: tools for mesh files [software]. Zenodo; 2024, http://dx.doi.org/10.5281/zenodo.1288334, Version v5.3.5. Available from: https://doi.org/10.5281/zenodo.1288334. [Accessed 17 January 2026].

[23] Schlömer N. Meshio 5.3.5 (PyPI project page): I/O for many mesh formats. 2024, Python Package Index (PyPI). Available from: https://pypi.org/project/meshio/5.3.5/. [Accessed 17 January 2026].

[24] Schroeder W, Martin K, Lorensen B. The visualization toolkit: an object-oriented approach to 3D graphics. 4th ed.. Kitware; 2006.

[25] VTK Developers. VTK 9.5.2 documentation: VTK file formats. 2025, Available from: https://docs.vtk.org/en/v9.5.2/design_documents/VTKFileFormats.html. [Accessed 17 January 2026].
[26] Kitware, Inc. ParaView 5.8.1 documentation: Understanding data (VTK data model). 2020, Available from: https://docs.paraview.org/en/v5.8/UsersGuide/understandingData.html. [Accessed 17 January 2026].
[27] VTK Developers. VTK 9.5.2 doxygen documentation: vtkMultiBlockDataSet class reference. 2025, Available from: https://vtk.org/doc/nightly/release/9.5/html/classvtkMultiBlockDataSet.html. [Accessed 17 January 2026].
[28] Schoof LA, Yarberry VR. Exodus II: A finite element data model. Report SAND92-2137, Sandia National Laboratories; 1994, http://dx.doi.org/10.2172/10102115, Available from: https://www.osti.gov/biblio/10102115. [Accessed 17 January 2026].
[29] Sjaardema GD, Schoof LA, Yarberry VR. EXODUS: A Finite Element Data Model [draft]. Sandia National Laboratories; 2023, Available from: https://sandialabs.github.io/seacas-docs/exodusII-new.pdf. [Accessed 17 January 2026].
[30] Rew RK, Davis GP. NetCDF: an interface for scientific data access. IEEE Comput Graph Appl 1990;10(4):76–82. http://dx.doi.org/10.1109/38.56302.
[31] International Organization for Standardization (ISO). ISO 10303–209: 2014 industrial automation systems and integration—product data representation and exchange—part 209: application protocol: multidisciplinary analysis and design (STEP AP209, edition 2). Geneva, Switzerland: ISO; 2014, Available from: https://www.iso.org/standard/59780.html. [Accessed 17 January 2026].
[32] Lanza R, Haenisch J, Bengtsson K, Rølvåg T. ISO 10303 AP209—Why and how to embed nonlinear FEA. Adv Eng Softw 2021;154:102976. http://dx.doi.org/10.1016/j.advengsoft.2021.102976, Available from: https://doi.org/10.1016/j.advengsoft.2021.102976. [Accessed 19 January 2026].
[33] MBx Interoperability Forum (MBx-IF). CAE recommended practices: AP209 edition 2. listed current documents include *rec. practices for AP209 ed.2*, version 2.0 (2016-03-30), and *linear static structural FEA handbook*, vol. 1 version 2.2 (2018-05-16), vol. 2 version 1.1 (2019-03-25). Available from: https://www.mbx-if.org/home/cae/recpractices/. [Accessed 05 June 2026].
[34] Grohmann P, Hüter F, Tremmel S. FE-Konverter: Konvertierung von FE-Netzen zwischen den Programmsystemen Z88, Abaqus, ANSYS und NASTRAN [in German]. FVA research project no. 972 I, final report, FVA report No. 1594, Frankfurt am Main: Forschungsvereinigung Antriebstechnik e. V. (FVA); 2023.
[35] Manning CD, Raghavan P, Schütze H. Introduction to information retrieval. Cambridge: Cambridge University Press; 2008, http://dx.doi.org/10.1017/CBO9780511809071.
[36] Blizard WD. Multiset theory. Notre Dame J Form Log 1989;30(1):36–66. http://dx.doi.org/10.1305/ndjfl/1093634995.
[37] Dice LR. Measures of the amount of ecologic association between species. Ecology 1945;26(3):297–302. http://dx.doi.org/10.2307/1932409.
[38] Berger JF, Grohmann P, Otto M, Hüter F, Stahl K, Tremmel S. FE-Strukturen in RIKOR: Berücksichtigung des elastischen Einflusses komplexer Radkörpergeometrien in der Getriebeauslegung am Beispiel von RIKOR [in German]. FVA research project No. 30 XII, final report, FVA report No. 1661, Frankfurt am Main: Forschungsvereinigung Antriebstechnik e. V. (FVA); 2025.
[39] Grohmann PR. Evaluation implementation for cross-solver finite-element mesh conversion and canonicalized validation [software]. Zenodo; 2026, http://dx.doi.org/10.5281/zenodo.21848108, Version 1.0.1. Available from: https://doi.org/10.5281/zenodo.21848108. [Accessed 08 August 2026].
[40] Grohmann PR. Reproducibility package and public evidence for cross-solver finite-element mesh conversion and canonicalized validation [dataset]. Zenodo; 2026, http://dx.doi.org/10.5281/zenodo.21843668, Version 1.0.1. Available from: https://doi.org/10.5281/zenodo.21843668. [Accessed 08 August 2026].