Migration Method for Parametric Features in Heterogeneous CAD Systems

Gang WANG, Guangfa ZHANG
College of Mechanical Engineering, Shanghai Dianji University, Shanghai, China

***Abstract*—Parametric feature migration across different Computer-Aided Design (CAD) systems is a core technology that supports cross-platform collaborative design and product data reuse. Addressing the fundamental limitation of traditional geometry-based data exchange methods—their inability to transfer high-level design intent information such as construction history, parametric constraints, and feature dependency relationships—this study proposes a three-stage parametric feature migration method based on JavaScript Object Notation (JSON) neutral files. Following the methodology of "exchanging the modeling process rather than the model itself", the complete migration workflow is decomposed into three stages: Feature parameter extraction, which leverages the source CAD system's application programming interface to traverse the model feature tree in depth and serialize the modeling history into a structured JSON exchange file; Topology identification, which employs a multi-dimensional geometric descriptor-based topological element identification method and writes identification results into the JavaScript Object Notation file; and Data validation, which validates feature parameters to be transferred and retrieves transient data required for modeling to ultimately achieve model reconstruction. Taking Siemens NX and ZW3D (formerly VX CAD) as representative source and target systems, a complete feature extraction and regeneration prototype plugin was developed, verifying the feasibility of the proposed method between heterogeneous CAD systems.**

Keywords*—Heterogeneous CAD; feature migration; JSON neutral file; information extraction; topology identification

I. Introduction

In the context of globalization and deep integration of digital technologies in manufacturing, product collaborative design has become a critical means for enterprises to enhance their core competitiveness. Hoffmann et al. [1] pointed out in their foundational work on user-defined features that parametric features are not merely collections of geometric elements, but rather encapsulations of design knowledge—feature dependency relationships, constraint networks, and modeling sequences collectively constitute the design intent of a product. In collaborative design practice, different enterprises and departments select different computer-aided design (CAD) systems for product development based on functional requirements, operational habits, and economic considerations, leading to heterogeneity in model data formats and feature semantic systems [2]. Concurrently, with the widespread adoption of Model-Based Definition (MBD) technology in high-end manufacturing sectors such as aerospace and aviation, product 3D models have evolved from purely geometric carriers into comprehensive data containers that embody design, manufacturing, and inspection information [3]. This places higher demands on information exchange between heterogeneous CAD systems—requiring not only the transfer of geometric shapes but also the complete preservation of construction history, parametric constraints, feature associations, and engineering annotations as high-level design intent. Therefore, achieving reliable migration of parametric feature models between heterogeneous CAD systems has become a critical technology for building cross-platform product collaborative development frameworks.

Existing data exchange methods between heterogeneous CAD systems primarily fall into two categories: geometry-based exchange and feature-based exchange [4, 5]. The first category employs the Initial Graphics Exchange Specification (IGES), Standard for the Exchange of Product Model data (STEP), and other neutral file formats as bridges, realizing cross-system geometric data transfer through the exchange of Boundary Representation (B-Rep) models. Rappoport [6] proposed an early architecture for universal CAD data exchange, establishing a theoretical framework for subsequent research. The STEP AP242 application protocol can express certain feature semantic information, but mainstream commercial CAD systems have only implemented a limited functional subset of the standard; in practical engineering scenarios, high-level information such as modeling history, parametric constraints, and feature associations cannot be effectively transferred [7]. After conversion, the model essentially degrades into a "static geometric entity", losing its parametric editing capability.

To overcome these deficiencies, academia has proposed feature model-based data exchange methods. Choi et al. [8] were among the first to explore CAD part model exchange based on the macro-parametric approach. Mun et al. [9] subsequently proposed a set of standard neutral modeling commands, achieving procedural data exchange between heterogeneous CAD systems. Li et al. [10] further proposed a two-stage modeling procedure recovery method—whose core concept is to exchange the complete modeling operation process rather than the model itself, "simulating a human designer" in the target system to reconstruct parametric feature models step by step. This method decomposes the feature migration workflow into first-order information extraction and second-order information reconstruction stages: the first-order information extraction stage recovers explicit modeling operation sequences from the source CAD system's macro recording file; the second-order information reconstruction stage reconstructs interactive selection information (such as reference elements, datum elements, and direction elements of features) and implicit information automatically generated by the CAD system (such as constraint relationships, implicit parameters, and feature

geometric attributes) from operation sequences, thereby supplementing the key semantic information missing from macro files. This two-stage framework has been validated between typical CAD systems such as SolidWorks, UG, Pro/E, and CATIA. Zhang et al. [11] further investigated interoperability issues during feature data exchange from a quantitative optimization perspective, proposing a computational model for evaluating conversion quality. Shao et al. [12] systematically established a feature operation-centric model conversion interface scheme, defining feature operation primitives independent of specific CAD systems and accessing and reconstructing CAD model information at both the feature layer and geometric layer simultaneously, ensuring consistency of feature trees, dimensions, and constraint information before and after conversion. Zheng et al. [13] built upon Shao's work to propose four migration principles for singular features and the (F, T, P) topological element identification method, providing important methodological guidance for the complete preservation of design intent between heterogeneous CAD systems. At the semantic level of feature migration, Qin et al. [14] constructed an ontology-based feature mapping framework, while Tessier and Wang [15] proposed a feature mapping and verification method between CAD systems from an ontological perspective; these two contributions established a theoretical foundation for semantic interoperability of heterogeneous CAD models. In the domain of engineering annotation exchange, Camba et al. [16] proposed an extended 3D annotation mechanism to explicitly convey geometric design intent, offering new possibilities for the preservation of non-geometric information during feature migration.

The above research has laid an important theoretical foundation for heterogeneous CAD feature migration, but the following common limitations exist: in the process of heterogeneous CAD feature migration, to maintain the transfer of parameters, feature information, and assembly relationships while avoiding data loss, the majority of transfer content is stored and compiled in Extensible Markup Language (XML) file format. Sun [17] also adopted XML as the intermediate file format in his research on feature-based data exchange between heterogeneous CAD systems, further corroborating the prevalence of this technical approach. However, XML, while being a markup language with strong semantics, strict validation, and extensibility suitable for complex documents, metadata, configuration files, and scenarios requiring rigorous data validity, exhibits high redundancy and slow parsing when handling simple models composed of multiple small components.

To improve the compatibility and generality of neutral files in heterogeneous CAD data transfer, and to facilitate efficient conversion of simple model feature data, this study employs JavaScript Object Notation (JSON) neutral files as the intermediate data storage container for research on feature-based heterogeneous CAD model conversion technology. The proposed method follows a three-stage experimental framework of "feature parameter extraction + topology identification + data validation".

The overall technical approach is illustrated in Fig. 1. This framework divides the process of heterogeneous CAD parametric feature migration into three sequentially connected stages: the first stage is feature parameter extraction, which accesses the source CAD system's internal feature tree through its Application Programming Interface (API), serializing feature parameters and modeling history into a structured JSON exchange file; the second stage is topology identification, which constructs a multi-dimensional geometric descriptor-based topological element identification method and writes results into the JSON file; the third stage is data validation, which takes into account the fact that operating methods and kernels may differ between distinct CAD systems in order to maximally preserve the model's design intent during CAD system migration and ensure successful regeneration of feature models, thus verifying that feature parameter data can be accurately imported. Taking NX and ZW3D as typical cases, a corresponding prototype system was developed to verify the feasibility and effectiveness of the method.

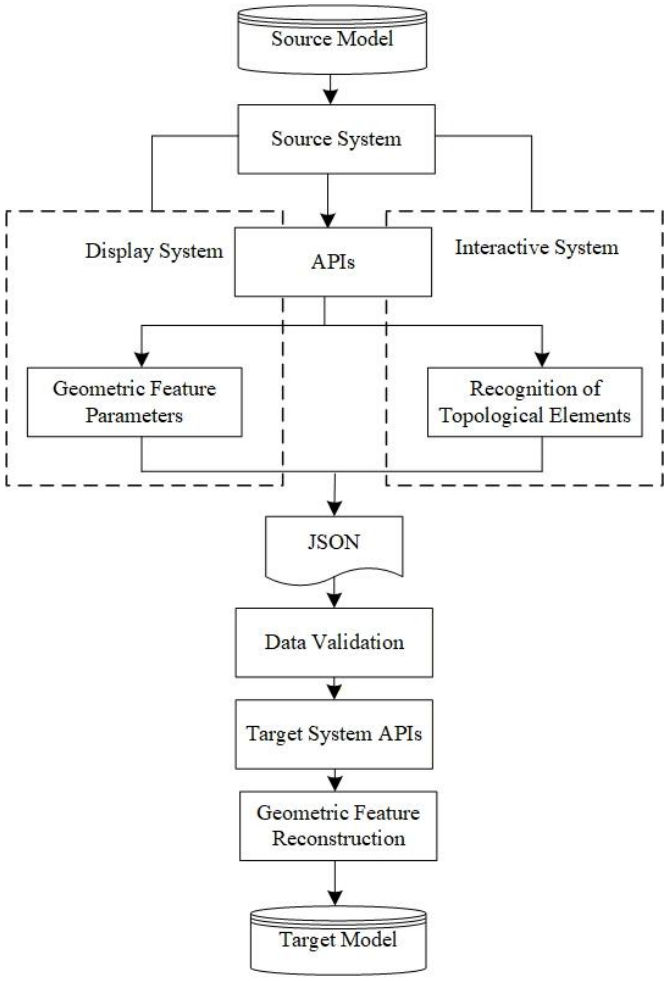


Fig. 1. Overall framework of heterogeneous model conversion.

II. Three-Stage Method for Heterogeneous CAD Parametric Feature Migration

As shown in Fig. 2, for the entire process of creating a source model, each operation step performed by the user generates a corresponding feature model based on the user's input command parameters, including constraints, implicit parameters, feature geometry, and physical attributes. The resulting feature model can be observed by the user in the display system, and the

driving feature parameter data is stored in the corresponding feature modules within the system. In addition to the model's feature parameters, there exist interactive information generated during the user's creation process, typically referring to the feature reference elements, base elements, and direction elements at the time of feature creation. This information is crucial for model reconstruction and serves to preserve design intent, particularly the correspondence of topological structures before and after model modifications.

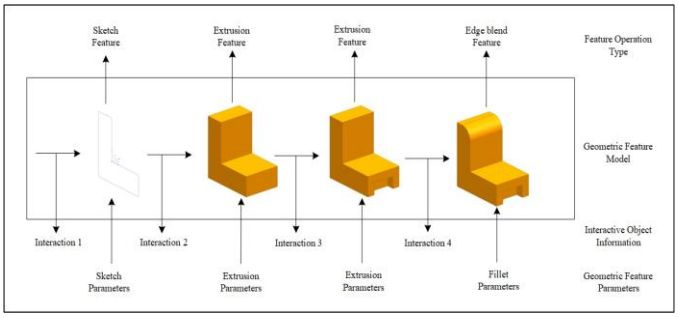


Fig. 2. Information generated during the creation of a parametric feature model.

A. Geometric Feature Parameter Extraction and Storage

The geometric feature parameter extraction stage requires the complete and accurate acquisition of all explicit modeling information of the parametric feature model from the source CAD system, organized in a structured and self-describing format to provide a data foundation for subsequent topology identification and target CAD feature model reconstruction.

Feature-based CAD systems accessed directly through their APIs generate a corresponding feature tree concurrently with user-interactive modeling, within which geometric feature information is recorded in internal data structures [12]. The traversal of the feature tree uses depth-first traversal (DFT), which was chosen over breadth-first traversal (BFT) for a deliberate reason: in parametric CAD modeling, feature creation follows a strict temporal sequence in which each feature depends on the geometric state produced by previously created features. DFT naturally reproduces this chronological ordering because it follows each branch of the feature tree to its deepest leaf before backtracking, thereby preserving the parent-to-child dependency chain. BFT, by contrast, would group features by their depth level and disrupt the temporal dependency order, potentially causing reconstruction failures in the target system when a child feature is reconstructed before its prerequisite parent feature has been fully resolved.

In the domain of heterogeneous CAD data exchange, the selection of a neutral file format directly affects the completeness of information, expressive capability, and engineering practicality. The fundamental deficiency of traditional STEP and IGES standards as neutral files for geometry-level exchange lies in the fact that their exchange target is the B-Rep of the parametric feature model rather than the modeling process [4]. In terms of information storage format selection, this method adopts JSON as the neutral exchange file format.

Compared with XML, JSON offers four principal advantages for storing CAD feature data:

1) Stronger adaptability: Through key-value pairs and nested arrays matching the CAD feature tree hierarchy, it can flexibly accommodate heterogeneous parameters of various features without strict format constraints.
2) Lower system resource consumption: It can intuitively express entity association relationships; one-to-many array relationships can be directly displayed through arrays, and topological elements carry their own tags and geometric data, eliminating the need for system reverse queries.
3) Higher parsing efficiency: As a native data structure of programming languages, it can be directly deserialized into C++ in-memory objects through lightweight libraries such as nlohmann/json, significantly reducing parsing overhead.
4) Better cross-platform generality: With concise syntax and a compact type system (only six data types: object, array, string, number, Boolean, and null), it can adapt to various CAD systems with API capabilities, effectively reducing the data interfacing costs between heterogeneous systems.

The JSON storage structure is illustrated in Fig. 3. Using the sketch feature as an example: this structure is capable of storing the basic feature parameters, sketch member parameters, constraint member parameters, and work coordinate system parameters of a sketch in a CAD system. By indexing the feature tree sequence, it reflects the temporal order of CAD feature creation, the structure, and the relationships between features created at each structural level.

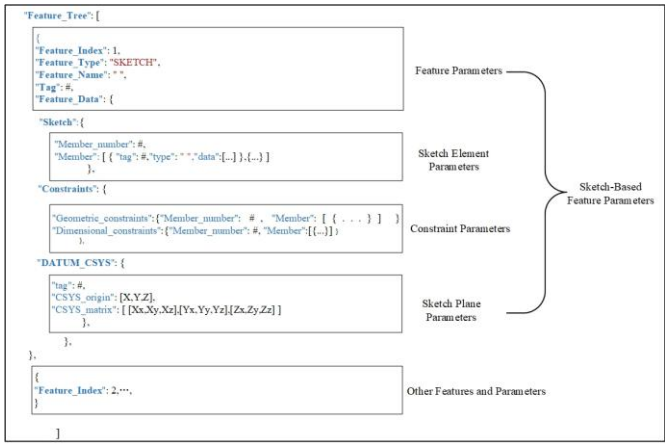


Fig. 3. JSON storage structure of feature data.

B. Basic Topological Element Identification

Basic topological elements refer to vertices, edges, and faces in a model. In the heterogeneous CAD system feature migration process, the creation of sketch-based features, dress-up features, and transformation features all depend on the precise positioning of topological elements.

When users create features in a CAD system, interaction with topological elements of the target geometric body is unavoidable.

For example, an extrude operation requires a closed profile or face within a sketch to drive it, as shown in Fig. 4(a). The termination condition "extrude to face" requires referencing a target termination face, as shown in Fig. 4(c). Chamfer feature creation requires selecting a target edge, as shown in Fig. 4(b). Draft features require specifying the neutral face and faces to be drafted, mirror features require a mirror plane, and so on. Therefore, the topology identification stage is a core component of heterogeneous CAD system data exchange: to ensure topological relationship stability and that the target model possesses the same design intent as the source model, geometric elements must be identified.

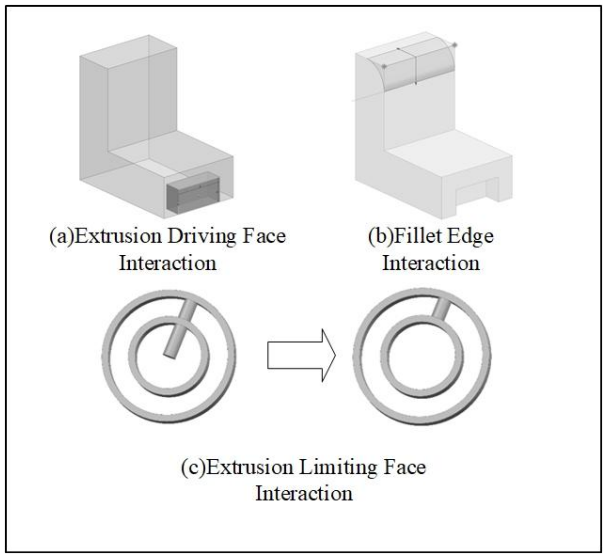
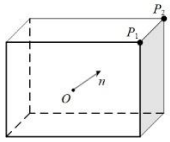
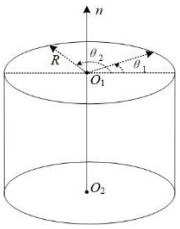


Fig. 4. User-interactive object selection.

To precisely identify referenced topological elements, the parametric definitions of such elements must be clearly established. Referenced geometric elements are classified into two categories: edges and faces. Edges include straight lines, circles, circular arcs, etc.; faces include planar faces, cylindrical faces, conical faces, spherical faces, toroidal faces, etc. Ensuring accurate and error-free referencing of topological elements is critical for the subsequent geometric feature reconstruction process.

As shown in Table I, the format for obtaining various parameters of referenced topological elements from a rectangular block and a cylinder is provided. The approach is fundamentally similar: select the target topological element type, invoke the corresponding API functions to obtain its individual parameters. For straight edges, the primary parameters are the 3D coordinates of the start point and end point. For circular and arc edges, the primary parameters are the radius, center point, and start/end angles; it is worth noting that all arcs in the UGNX system rotate in the counterclockwise direction. For faces, the primary parameters are the center point and normal vector. Because face types differ, the reference for center point and normal also differs: for example, the center point of a planar face is located at the 3D coordinate center of the midpoint between its tight bounding box minimum and maximum corner points, and its normal is perpendicular to the face. The center point of a cylindrical face is located at the center of its bottom circular edge, and its normal is aligned with its axial vector.

TABLE I. PARAMETRIC DEFINITION OF TOPOLOGICAL GEOMETRY AND ITS JSON REPRESENTATION.

Geometry	Parameter	JSON Representation
	Edge type	"edge_type": "Linear",
	Parent feature handle	"parent_feat_tag": #,
	Start point	"start_point": [X, Y, Z],
	End point	"end_point": [X, Y, Z]
	Face type	"face_type": "Plane",
	Center point	"center_point": [X, Y, Z],
	Normal	"normal": [X, Y, Z],
	Box min	"box_min": [X, Y, Z],
	Box max	"box_max": [X, Y, Z],
	Parent feature handle	"parent_feat_tag": #
	Edge type	"edge_type": "Arc",
	Parent feature handle	"parent_feat_tag": #,
	Radius	"radius": #,
	Arc center	"arc_center": [X, Y, Z],
	Start angle	"start_angle": #,
	End angle	"end_angle": #,
	Sweep angle	"sweep_angle": #
	Face type	"face_type": "Cylinder",
	Center point	"center_point": [X, Y, Z],
	Normal	"normal": [X, Y, Z],
	Box min	"box_min": [X, Y, Z],
	Box max	"box_max": [X, Y, Z],
	Radius	"radius": #,
	Parent feature handle	"parent_feat_tag": #

The topology identification workflow is shown in Fig. 5.

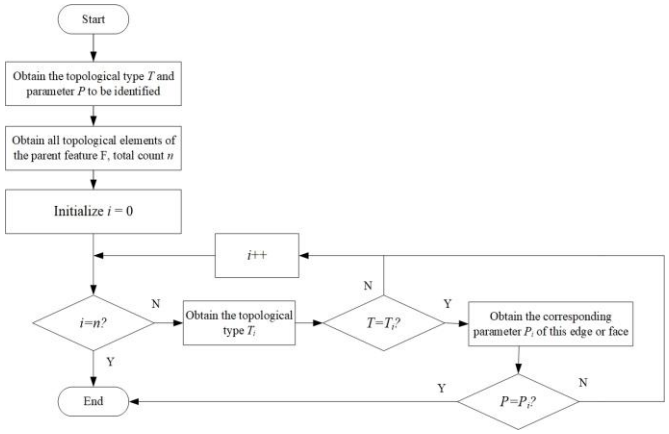


Fig. 5. Topology element identification process.

This approach references the (*F, T, P*) triple identification method proposed by Xie et al. [18]—where the parent feature *F* is used to narrow the search scope, the topology type *T* is used for filtering, and geometric parameters *P* are used to complete identification and matching. During the feature reconstruction process, the parameters of topological elements to be identified are obtained from the JSON file; using the feature name *F* from the parameters, all edges or faces in that feature are selected; a determination is made as to whether the selected edge or face is of the required type; if so, the geometric parameters P_i of the

selected topological elements are sequentially compared with the corresponding parameters in *P* to determine whether they are equal; if they are equal, the topological element is identified as the target topological element. When the model is relatively complex, selecting edges or faces dependent on features can effectively narrow the search scope and improve the search efficiency for topological elements.

C. Data Validation

Because the operating methods and kernels of different CAD systems may differ, it is necessary to extract and verify the feature data before regeneration to maximally preserve the model's design intent during CAD system migration and ensure the successful reconstruction of feature models. Prior to this, essential data validation and supplementation of missing information must be performed.

1) Tracing disappearing transient information. During feature operations in a CAD system, certain topological structures may undergo deformation or even disappear as a result of the operation. These disappearing transient topologies are often an indispensable part of maintaining operational intent. For transient information that disappears after modeling, the fillet feature is used as an example: when the user performs a fillet operation and selects a target edge, that edge becomes a cylindrical face, and the target edge is suppressed by the system. By obtaining the edge's handle, temporarily suppressing its feature, and querying its parent feature, the original parameter information of the edge can be retrieved. This method also applies to draft features and other features that alter geometric shape parameters.

A related but more challenging scenario involves features that are completely consumed by subsequent Boolean operations (union, difference, or intersection). For example, a cylindrical extrude feature may be entirely absorbed into a larger body through a Boolean union, leaving no standalone geometric entity that can be traversed in the final model. To handle this, the feature parameter extraction stage captures all feature information before the Boolean operation is applied. Specifically, the traversal algorithm records a snapshot of each feature's geometric parameters and topological references at the moment of its creation within the feature tree, rather than from the final consolidated model. This is achieved by iterating through the feature tree's creation history in temporal order: for each feature, its parameters are extracted immediately after the feature node is visited, regardless of whether the feature remains visible in the final model. Boolean operations themselves are recorded as separate feature entries in the JSON file with a tag and an array listing the consumed features' handles. During reconstruction, the target system first reconstructs all constituent features and then applies the Boolean operation, faithfully reproducing the original modeling sequence.

2) Validating cross-system incompatibility information. For incompatible information arising from differences in feature construction methods between heterogeneous CAD systems, alternative solutions must be adopted when unique feature constraint types in the source CAD system have no direct correspondence in the target system. One situation occurs when the source CAD system possesses a certain design intent during feature interaction operations that the target CAD system does not possess. For example, the extrude feature in ZW3D includes a design intent of "extrude to a point", whereas the UGNX system lacks this design intent. According to the third principle proposed by Zheng et al. [13]—"converting using different features and auxiliary formulas"—an auxiliary plane is created at the target point, and by leveraging extrusion-to-plane along with its extended surface features and constructing an auxiliary plane, the conversion of extrusion-to-vertex features can be achieved. This ensures that when the vertex position changes, the constructed auxiliary plane changes accordingly, thereby causing the extrusion length to adaptively vary and thus preserving the design intent.

Another situation occurs when both the source and target CAD systems contain the same construction method, but due to differences in their underlying kernels, data extracted from the source CAD system cannot be directly used between the two systems and requires certain processing. In practice, to maintain the stability and consistency of internally similar data, CAD systems establish prescribed operational procedures. For example, in the UGNX system, when creating an arc in a sketch feature, to ensure the arc's rotation direction remains consistent, the system relies on the arc orientation matrix and uses the absolute coordinate system origin (0, 0, 0) as the reference for orienting the arc, thereby ensuring the arc's rotation direction is counterclockwise. If APIs are called to obtain the coordinates of the arc center, these coordinates correspond to the arc center in the local coordinate system of the arc. Generally, when creating sketch curves, the input sketch is in the local coordinate system or the absolute coordinate system. Therefore, data conversion is required: the arc center coordinates in the local coordinate system of the arc must be transformed to the arc center coordinates in the sketch coordinate system.

Using API functions, the arc center coordinates and arc rotation matrix based on the absolute coordinate system can be obtained. The coordinate transformation is accomplished through the following transformation equations:

Let P_{local} be the arc center in the local coordinate system of the arc, and let R_{acr} be the corresponding arc rotation matrix. The arc center P_{abs} in the absolute coordinate system is obtained from:

$$P_{abs} = R_{acr} P_{local} \tag{1}$$

The sketch coordinate system is described by its origin O_{sk} and rotation matrix R_{sk} (where R_{sk} transforms sketch local coordinates to absolute coordinates, i.e.,$v_{abs} = R_{sk} v_{sk} + O_{sk}$). Hence, the sketch local coordinate P_{sk} is computed as:

$$P_{sk} = R_{sk}^{T}(P_{abs} - O_{sk}) \tag{2}$$

where, R_{sk}^{T} is the transpose of R_{sk} and, due to the orthogonality of a rotation matrix, $R_{sk}^{T} = R_{sk}^{-1}$. In practical implementation, taking the three column vectors C_0, C_1, C_2 of R_{sk}, the dot product operation is employed:

$$P_{sk,i} = C_i(P_{abs} - O_{sk}), \quad i = 0,1,2 \tag{3}$$

Thus, the parameter information that can be directly utilized by the target CAD system in sketch features is obtained.

3) Complex Sketch Constraint Serialization. The complex sketch geometric constraints addressed herein are primarily constraint outcomes generated by patterned curves (mirrored curves represent a special subtype of patterned curves).

Sketch constraints fall into two categories: dimensional constraints and geometric constraints. Whether they are geometric constraints such as perpendicularity or parallelism, or dimensional constraints, all serve as mandatory restrictions imposed on pre-existing curves.

Taking UG NX as an example, the constraint relationships of existing curves within a sketch feature can be retrieved by invoking two Application Programming Interface (API) functions: “ UF_SKET_ask_geo_cons_of_sketch() ” and “UF_SKET_ask_dimensions_of_sketch()”.

Child curves generated via the patterned curve operation maintain a patterning association with one another and carry identical geometric information. Counterintuitively, when the information of these curves is extracted and reconstructed in the target CAD system, the curves are found to be completely coincident in appearance, rather than arranged sequentially in accordance with the patterning order.

To address this issue, the following strategy is proposed:

- Aggregate all curves generated by the corresponding patterned curve operation using the aforementioned geometric constraint acquisition function.
- Identify the parent curve through the "smart container" APIs in UG Open, and label the remaining curves as child curves.
- Document the patterning mode, the parent curve and the labeled child curves in the JavaScript Object Notation (JSON) file, and mark the child curves as disabled during reconstruction in the target CAD system.
- Execute the patterned curve operation on the parent curve in the target CAD system.

This approach can maximally ensure the successful conversion of patterning constraints.

III. Experimental Validation and Analysis

A. Experimental Case Study

To verify the feasibility and effectiveness of the proposed method, this study selects Siemens NX 12 as the source system and ZW3D 2024 as the target system, and establishes a complete feature migration prototype. First, the NX Open C++ API is used to traverse the model feature tree in depth, recursively obtaining the construction type, parameter set, sketch data and constraint information, topological face references, and geometric attributes of each feature. All information is organized in a structured hierarchical manner and output as a JSON exchange file. Fig. 6 shows a code excerpt for parameter and topology extraction from a UGNX extrude feature.

Next, JSON parsing and system reconstruction are implemented using ZW3D C++ API. Fig. 7 shows a code excerpt for reconstructing an extrude feature in ZW3D.

```
void Get_Extrude_Info(tag_t extrude_feat, nlohmann::ordered_json& j_Info)
{
    TaggedObject* taggedObj = NXObjectManager::Get(extrude_feat);
    Features::Extrude* extrude = dynamic_cast<Features::Extrude*>(taggedObj);

    Session* ses = Session::GetSession();
    Part* workPart = ses->Parts()->Work();
    Features::ExtrudeBuilder* builder = workPart->Features()->CreateExtrudeBuilder(extrude);

    Direction* dirObj = builder->Direction();
    Vector3d vec = dirObj->Vector();
    j_Info["direction"] = { vec.X, vec.Y, vec.Z };

   auto* offset = builder->Offset();
   PrintFeatureOffset(offset, &j_Info);  //Function to obtain extrude offset information

    auto* startExt = builder->Limits()->StartExtend();
    auto* endExt = builder->Limits()->EndExtend();

    auto printExtend = [](const char* which, NXOpen::GeometricUtilities::Extend* ext,
        nlohmann::ordered_json& j){ ... }

        printExtend("Start", startExt, j_Info["limits"]["start"]);
        printExtend("End", endExt, j_Info["limits"]["end"]);

    builder->Destroy();
}
```

Fig. 6. Code excerpt for parameter and topology extraction from UGNX extrude feature.

```
int ZWClass::ProcessExtrude(const json& featData, int featureTag)
{
...

    extrude.profileHandle = sketchHandle;// 5. Populate data
    extrude.startS = startDist;extrude.endE = endDist;
    extrude.startType = startType;
    extrude.endType = endType;
    extrude.boolean.combine = combine;
    extrude.settings.capType = ZW_END_CAP_BOTH;
    extrude.draft.isUseExtrusionDirection = 1;
    extrude.direction.x = dirX;
    extrude.direction.y = dirY;
    extrude.direction.z = dirZ;

    if (startFaceHandle.innerData){
        extrude.startEntityHandle = startFaceHandle;
        extrude.startToPoint = &startPoint;
    }
    if (endFaceHandle.innerData){
        extrude.endEntityHandle = endFaceHandle;
        extrude.endToPoint = &endPoint;
    }

    szwEntityHandle shapeHandle = { 0 };
    evxErrors err = ZwFeatureExtrudeCreate(extrude, &shapeHandle);// 6.Execute
...
    ZwEntityHandleFree(&shapeHandle);
    ZwEntityHandleFree(&sketchHandle);
    Set_View();
    return extrudeFeatureId;
}
```

Fig. 7. Code excerpt for reconstructing an extrude feature in ZW3D.

B. Results Analysis and Discussion

The experimental results are shown in Fig. 8. The results indicate that the proposed method successfully achieved cross-system migration of parametric features from NX to ZW3D. By comparing the feature trees and feature models of the source and target CAD systems, it can be observed that the target CAD system is able to find feature trees corresponding to those in the source system, and the feature model displays are similar to the source models. This validates the effectiveness of the three stages—feature parameter extraction, topology identification, and data validation. It also demonstrates that JSON, as a neutral file format, is capable of storing feature parameter information and maintaining the potential for interactive design. The method effectively preserves high-level semantics such as design intent, parameter information, and feature information, enabling the migration and sharing of heterogeneous CAD models with significant practical potential and application value.

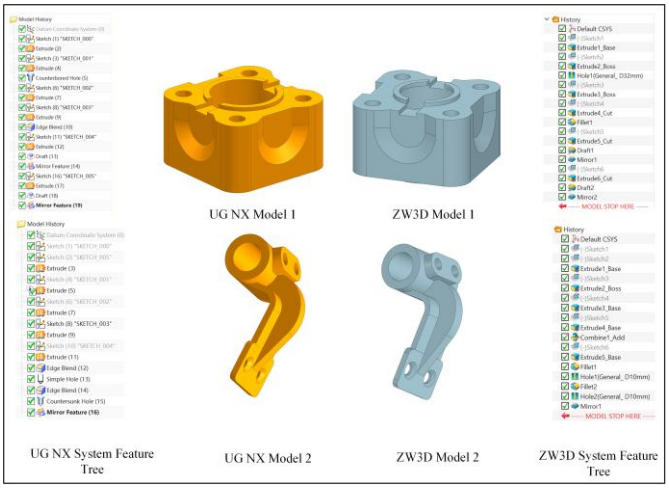


Fig. 8. Model migration cases between UGNX and ZW3D.

IV. Discussion

A. Parametric Editability Verification

A key criterion for successful feature migration is whether the migrated model remains fully parametric and editable in the target CAD system. To evaluate this, parameter modification tests were conducted on the migrated models. For each feature in the target model, its critical dimension parameters were modified and the model was regenerated. The test results showed that for features reconstructed at the feature-level (those with direct API mapping), all parameter modifications triggered correct model updates, preserving the intended update behavior—the model geometry changed exactly as it would in the source system when the same parameter was modified.

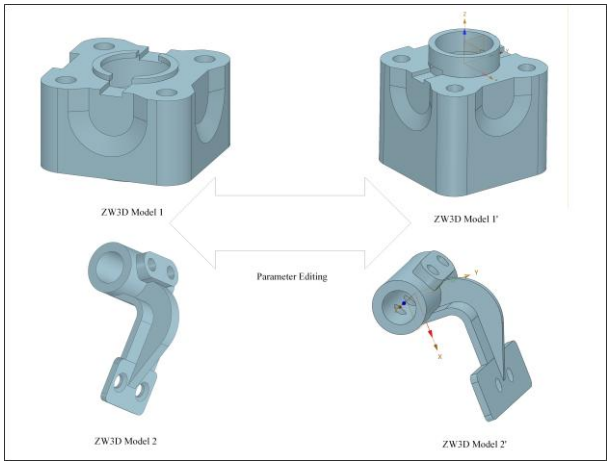


Fig. 9. Editable target CAD model.

B. Generalizability Across CAD Kernels

The proposed method's generalizability to additional CAD systems with different geometric kernels relies on the API abstraction layer implemented in the extraction and reconstruction modules. The method has been designed with kernel independence as a core architectural principle. The feature parameter extraction module communicates with the source CAD system solely through its published API, without depending on kernel-specific data structures. Similarly, the reconstruction module invokes only the target system's documented API functions.

C. API Version Compatibility

The software used for development is generally lower-version releases, so API invalidation in CAD systems is extremely rare. Should such a scenario arise, alternative APIs of the same category may be invoked in adherence to the principle of ensuring successful conversion.

For instance, two primary API function systems are available in UG NX, namely UFun and NX Open. Both sets of APIs shall be invoked simultaneously during the implementation of the aforementioned patterning constraint guarantee strategy.

V. Conclusion

Addressing the common challenge of feature migration between heterogeneous CAD systems, this study proposes a tiered regeneration feature migration method based on JSON neutral files, founded on a three-stage framework of feature parameter extraction, topology identification, and data validation. In the feature parameter extraction stage, modeling history is serialized into JSON structured data, improving the completeness of information extraction and the generality of the method. In the topology identification stage, a multi-dimensional geometric descriptor-based topological element identification method is constructed and JSON data is populated and written. In the data validation stage, to maximally preserve the model's design intent during CAD system migration, system transient information is traced and retrieved, and cross-system incompatibility information is refined and converted.

Taking NX and ZW3D as typical cases, prototype development and verification were completed. Experimental results demonstrate that models migrated using the proposed method are capable of inheriting the feature information and modeling history of the source CAD system, preserving the model's design intent.

Future research on this method will advance along two directions: first, tracking the iterative expansion of target CAD system APIs to broaden the coverage of mappable features; second, addressing the limited expressive capability of JSON for complex curves or features—further investigation into more flexible data organization approaches is needed to handle mixed hierarchical relationships that the tree structure of JSON may struggle to represent.

References

[1] M. Hoffmann and R. Joan-Arinyo, "on User-defined Features," Computer-Aided Design, Vol. 30, No. 5, Pp. 321–332, 1998.

[2] M. Gao and F. Z. He, "Survey of Distributed and Collaborative Design," Journal of Computer-Aided Design & Computer Graphics, Vol. 16, No. 2, Pp. 149–157, 2004.

[3] Alemanni, F. Destefanis, and E. Vezzetti, "Model-based Definition Design in the Product Lifecycle Management Scenario," the International Journal of Advanced Manufacturing Technology, Vol. 52, No. 1/2/3/4, Pp. 1–14, 2011.

[4] J. Pratt, B. D. Anderson, and T. Ranger, "Towards the Standardized Exchange of Parameterized Feature-based CAD Models," Computer-Aided Design, Vol. 37, No. 12, Pp. 1251–1265, 2005.

[5] Kim, M. J. Pratt, R. G. Iyer, and R. D. Sriram, "Standardized Data Exchange of CAD Models with Design Intent," Computer-Aided Design, Vol. 40, No. 7, Pp. 760–777, 2008.

[6] Rappoport, "an Architecture for Universal CAD Data Exchange," in Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications. Seattle, WA: ACM, 2003, Pp. 266–269.

[7] Z. Zhang and X. F. Luo, "Design Intent Information Exchange of Feature-based CAD Models," in Proceedings of the World Congress on Computer Science and Information Engineering. Los Angeles, CA: IEEE, 2009, Pp. 11–15.

[8] H. Choi, D. Mun, and S. Han, "Exchange of CAD Part Models Based on the Macro-parametric Approach," International Journal of CAD/CAM, Vol. 2, No. 1, Pp. 13–21, 2002.

[9] Mun, S. Han, J. Kim, and Y. Oh, "a Set of Standard Modeling Commands for the History-based Parametric Approach," Computer-Aided Design, Vol. 35, No. 13, Pp. 1171–1179, 2003.

[10] Li, F. Z. He, X. T. Cai, and D. Zhang, "Using Procedure Recovery Approach to Exchange Feature-based Data Among Heterogeneous CAD Systems," in Proceedings of the 2009 13th International Conference on Computer Supported Cooperative Work in Design. Santiago, Chile: IEEE, 2009, Pp. 716–721.

[11] J. Zhang, F. Z. He, S. H. Han, and X. X. Li, "Quantitative Optimization of Interoperability during Feature-based Data Exchange," Integrated Computer Aided Engineering, Vol. 23, No. 1, Pp. 31–50, 2015.

[12] D. Shao, F. Chen, H. L. Liu, and L. Liu, "Research on Feature-based Heterogeneous CAD Model Transformation Technology," China Mechanical Engineering, Vol. 18, No. 1, Pp. 60–64, 2007.

[13] L. Zheng, J. R. Li, L. Wu, and X. D. Shao, "Research on Singular Feature Migration Method for Heterogeneous CAD Systems," Machinery Design & Manufacture, No. 3, Pp. 204–208, 2025.

[14] W. Qin, L. Y. Li, and S. M. Gao, "an Ontology Mapping Method for Heterogeneous Parametric Feature Model Retrieval," Computer Integrated Manufacturing Systems, Vol. 19, No. 7, Pp. 1472–1483, 2013.

[15] Tessier and Y. Wang, "Ontology-based Feature Mapping and Verification between CAD Systems," Advanced Engineering Informatics, Vol. 27, No. 1, Pp. 76–92, 2013.

[16] Camba, M. Contero, M. Johnson, and P. Company, "Extended 3D Annotations as a New Mechanism to Explicitly Communicate Geometric Design Intent and Increase CAD Model Reusability," Computer-Aided Design, Vol. 57, Pp. 61–73, 2014.

[17] Sun, "Research on Feature-based Technology of Data Exchange between Heterogeneous CAD Systems," M.S. Thesis, Dalian University of Technology, Dalian, China, 2010.

[18] K. Xie, G. Zhao, Y. Yu, and Y. D. Wang, "Research and Implementation of Feature-based Data Exchange Method of Heterogeneous 3D Digital Definition Model," Computer Integrated Manufacturing Systems, Vol. 23, No. 9, Pp. 1833–1841, 2017.