An Automated Approach for the Discovery of Interoperability

Duygu Sap

*International Computer Science Institute, Berkeley, CA, USA.*

Daniel P. Szabo

*University of Wisconsin-Madison, WI, USA*

Abstract

In this article, we present an automated approach that would test for and discover the interoperability of CAD systems based on the approximately-invariant shape properties of their models. We further show that exchanging models in standard format does not guarantee the preservation of shape properties.

Our analysis is based on utilizing queries in deriving the shape properties and constructing the proxy models of the given CAD models\[1\]. We generate template files to accommodate the information necessary for the property computations and proxy model constructions, and implement an interoperability discovery program called DTest to execute the interoperability testing.

We posit that our method could be extended to interoperability testing on CAD-to-CAE and/or CAD-to-CAM interactions by modifying the set of property checks and providing the additional requirements that may emerge in CAE or CAM applications.

**Keywords:** automated system, interoperability, model interchangeability, STEP, proxy model

1. Introduction

1.1. Motivation

Interoperability has been a challenging unsolved problem that relies on manual, error-prone solutions and costs billions of dollars annually \[2, 3\]. Semi-automated verification of interoperability can be achieved by a set of limited tools. However, there does not exist any automated tools for the verification and the validation of interoperability solutions. This work may enable the next generation of automatically composable and reconfigurable systems, and support formal verification of the currently used standards. In this article, we focus on the theoretical framework we built in \[1\], and construct an algorithmic framework that can be used to apply the theory presented in \[1\]. We also provide practical applications using the automated system we built based on the algorithmic framework we present here.

To our knowledge, there does not exist any work in the literature which has developed an algorithmic framework or an automated system that is capable of testing for the interoperability of CAD systems based on the interchangeability of their models with respect to their shape properties. By constructing such a framework and a system, we aim to show that it is possible to discover the interoperability between CAD systems with a pre-determined tolerance without translating formats or converting representations. We note that the interoperability we test for hinges on the interchangeability of a pair of CAD models that may be constructed by different CAD systems. Therefore, it may be identified as conditional interoperability. We further test if exchanging models in standard formats guarantees the preservation of the shape properties. We show that model transfers may result in unpredictable changes in the shape properties even when the standard formats are used.

The diagram below illustrates the logical structure of the automated system with its components.

The interoperability diagnosis focuses on the interchangeability of CAD models with respect to a specified shape property with a given accuracy. Within the scope of this paper, we consider the following shape properties: geometric (convexity, centroid, Hausdorff distance), topological (homotopy types, manifoldness), integral (volume, surface area), combinatorial (Euler characteristic).

We utilize the queries: PMQ, distance, integral, and we derive a result of the following form: “CAD systems \(C_1\) and \(C_2\) that provide the respective models \(M_1\) and \(M_2\) can interoperate in carrying out a task that allows using \(M_1\) and \(M_2\) interchangeably with the given accuracy \(\epsilon\) for the specified property”.

We note that the differential properties could easily be added to
the list of properties if the CAD software is capable of providing differential information through queries. Differential information could also be approximated via the lower level queries [2].

1.2. Challenges
Here is the list of challenges we address in this article:

1. Investigating the automated verification of interchangeability of CAD models, thus, the automated verification of the conditional interoperability of CAD systems.
2. Validating the interoperability of CAD systems based on model transfer via direct data translation.
3. Investigating the invariant properties under data transfers and translations through Round-Robin testing.

Challenge (3) may be utilized in solving the problem defined as the characterization of interoperability in [4] in an automated manner.

1.3. Related Work
Interoperability is supported by semi-automated heuristic tools, requiring expertise and significant manual labor. There are various software companies that offer interoperability, translation verification and validation solutions. [5], [6], [7], [8] are some of the popular software tools provided by these companies. These software tools run checks on the models transferred and then utilize some automated or manual healing to compensate for the data and model quality loss. Thus, the models are modified to establish interoperability without any quantifiable measure. Geometry validation testing based on a standard format, namely, STEP, has been carried out by CAx Implementor Forum since 1998[9]. Their methods are based on validating the shapes with respect to the properties such as centroid, volume and surface area for solid models, centroid and surface area for surface models, and centroid and total length for independent curves [9]. They also list a cloud of points method as an additional validation technique, but they do not provide a fully-developed and a practical methodology for this method. Their general testing procedure is based on the comparison of the results they derive from experimenting over various sets of models every six months. However, it is unclear why these sets of models are considered to constitute sufficient testing domains for validation since they do not seem to form bases for any geometric model space.

3D model validation techniques are also developed as military standards in [10]. Department of Defense (DoD) requires approved validation processes to show that 3D models are suitable for reference data. Since models may contain subtle defects that can prevent them from being used by downstream applications such as numerically-controlled manufacturing, finite element analysis, and inspection with coordinate-measuring devices, formal algorithmic validation processes are needed. The validation in [10] is done through identifying and classifying the defects in 3D digital models along with their effects on various applications that use these models, and providing recommended tolerances and acceptance criteria for these models. However, the tolerances listed for the validation of different geometric components have some inconsistencies, and there is redundancy in the testing procedures. A thorough analysis of these military standards is provided in [11].

1.4. Contributions
The main contributions of this article can be listed as follows:

• The first known theoretically supported automated system for the interoperability of CAD systems with respect to shape properties, namely, geometric, topological and integral properties. The automated system presented here trivializes the construction of a system with an extended set of properties such as the physical and material properties.
• The first known algorithmic framework that could be used for verifying CAD model interchangeability, and validating CAD model transfers or format translations via queries.
• A method for investigating the properties preserved under model transfers via standard formats, which in turn would indicate the ability to predict and maintain the model quality for long-term archival and retrieval (LOTAR) [12].
• Demonstrated applicability of the theory developed in [1] to current practices that are being tested by NIST [9].

2. Research Methodology
Our methodology is based on building proxy models that would substitute for the CAD models in property-based comparisons and investigating shape equivalences with predictable accuracies to determine the interchangeability of given CAD models, or validate a CAD model translation [1]. Proxy model and shape equivalence constructions are based on a parameter $\epsilon$ which is determined by the accuracy of the point membership classification query (PMQ), tolerances and algorithm precisions of systems and the minimum feature sizes of the CAD models [1]. In the following subsections, we present an overview of our theoretical framework and provide details on the structures and inner mechanisms of our algorithmic framework and automated approach.

2.1. Problem Definitions
We mainly consider two problems:

• Automated Verification of Model Interchangeability
• Automated (Round-robin) Testing

2.1.1. Automated Verification of Interchangeability
Given two CAD models, determine if the models are interchangeable based on a shape property with the stated accuracy
2.1.2. Automated (Round-robin) Testing
Given a CAD model, read and write the model in STEP format multiple times in the same system or circulate it over different systems and observe the variations in the properties to determine if or when any shape property value deviation converges to zero.

2.2. Theoretical Framework
Our theoretical framework for the verification of interoperability (detailed description of which can be found in [1]) enables determining the interchangeable usability of the CAD models created by distinct CAD systems through a query-based data analysis. The query-based approach allows us to test and discover the interoperability of CAD systems and model representations via a set of queries instead of translators and/or transferring files. Queries are computable functions with semantics specified with respect to a standard reference. In query-based interoperability scenarios, systems are allowed to have different representations and algorithms, and they retain their copies of the models separately[2, 4].

In [1], we provided the sufficient conditions for establishing a correspondence based on a topological equivalence and a geometric similarity between model instances authored by distinct CAD systems. We referred to the proxy model notion defined by Hoffmann et al in [2] and described how to construct and utilize proxy models of given CAD models.

A proxy model
- substitutes for the CAD model in property calculations,
- may take different forms (e.g., a point cloud, union of balls, a graph, an algebraic complex such as a Cech complex),
- may be built in one of the systems or exist as a separate (abstract) reference model,
- could be set as one of the CAD models in a model comparison scenario,
- provides estimates for the models’ properties that depend on model-specific data, which is implicitly related to systems’ attributes.

We note that the proxy model constructions and the property computations could be completely carried out through a query-based approach as a result of which the model instances are liberated from their system-dependent representations [1].

2.3. Algorithmic Framework
In this section, we present the algorithmic framework that enables building proxy models of the given CAD models and computing the properties of these proxy models to allow a property-based model comparison. We run the tests over the template files that substitute for the models. Thus, the testing procedure does not require model transfers or translations, and the property information can be derived directly through standardized sources. Figure ?? illustrates the structure of the algorithmic framework.

We note that this algorithmic framework could be used in executing tests on CAD models that are saved in different formats in distinct CAD systems as well as on CAD models given in standard formats.

Figure 1: Automated testing and discovery for interoperability diagram

Every CAD model either comes with a template file or in standard format, namely, STEP.

2.3.1. DTest
DTest is the key component of the algorithmic framework and the executable component of the automated system. Here is the list of tasks DTest carries out:

- Running on model template files and picking up the property with its tolerance entered by the user
- Determining the set of query responses that should be picked from the template files
- Collecting the necessary information from the template files to determine parameters such as the ball radius \( \epsilon : \epsilon_i + \alpha_i < \epsilon < \delta_i \), where \( \alpha_i \) and \( \epsilon_i \) are the respective algorithm precision and absolute tolerance of the CAD system \( C_i \), and \( \delta_i \) is the minimum feature size of the model \( M_i \) created by \( C_i \).
- Creating the configure for constructing a proxy and computing properties
- Creating the evaluation that would evaluate the results of the tests with respect to the allowable tolerance level specified by the user for the specified property

We note that if the CAD systems the CAD models are authored by use different scripting languages, wrapper functions [13] need to be used to build a standardized coding environment based on a single programming language.

2.3.2. Configure
Configure is responsible for the following tasks:

- Constructing the proxy model that can substitute for \( M_i \) in the property comparison.
• Computing the relevant property of the proxy models with the parameters it receives from the template files via DT est

Note that the choice of the proxy depends on what is sufficient for computing the property. Moreover, in some cases, configure does not need to define functions for computing model properties. For example, if we are investigating the similarity of models in the Hausdorff metric, then it computes the Hausdorff distance between the proxy models, which would be point clouds in such case.

2.3.3. Evaluation

Evaluation carries out the following tasks:

• Compares the properties of the models

• Derives and outputs an interoperability report with respect to the standards that DT est sets for the model interchangeability based on the information provided by the user

2.3.4. Template File

Template files are model-specific, and they would ideally accommodate the system information that was active in the design environment where the model was constructed (See Appendix A). However, in this research we include the system information active in the environment where the model was read in STEP format in our experiments. It is important to note that the template files that do not provide the authoring system specifications are likely to suffer from the drawbacks of the existing standard formats. The current automated system requires the users to create the template files.

A template file for a model denoted by \( M_i \) has the following content:

• Authoring CAD system \( C_i \)

• API options

• Scripting languages

• System tolerances: absolute tolerance \( \epsilon_i \), angular tolerance \( \epsilon_i^\alpha \)

• Algorithm precisions: reading precision \( \alpha_i^r \), writing precision \( \alpha_i^w \), PMQ accuracy

• List of queries supported by \( C_i \)

• Measurement units

• Topological class of \( M_i \)

• Minimum feature size \( \delta_i \)

Note that we could split a template file into two sections concerning the information content: system-specific section and model-specific section.

Figure 2: A CAD model provided as a STEP file by [14]. For this model, the minimum feature size, \( \delta_i = \text{min} \{ r: r \text{ is the radius of a hole on the sphere} \} \).

3. Automated System Design

In this section, we provide the automated testing procedure for the interoperability of the CAD systems. The testing is done over model template files by focusing on the invariance of a given shape property within a specified interval. For example, when we run DT est on the model template files \( \text{Temp}(M_1) \) and \( \text{Temp}(M_2) \) by stating a shape property \( P \) along with an accuracy \( \epsilon \), we test for the interchangeability of \( M_1 \) and \( M_2 \) with respect to \( P \) with \( \epsilon \) accuracy. The test result would be positive if \( |P_{M_1} - P_{M_2}| \leq \epsilon \), thus, the models would be deemed interchangeable with \( \epsilon \) accuracy for the applications that only use the property \( P \). This further yields a conditional interoperability of the CAD systems \( C_1 \) and \( C_2 \) since for any application that needs to use only the property \( P \) of the shape represented by \( M_i \) in the CAD systems \( C_i \), the systems \( C_1 \) and \( C_2 \) can interoperate with accuracy \( \epsilon \) in using their models \( M_1 \) and \( M_2 \) interchangeably.

In the following subsections, we show how the automated system works for the two problems stated in Section 2

3.1. Testing for Model Interchangeability

Here is the list of problems we addressed using the automated system to determine model interchangeability:

(i) **Verification of Model Interchangeability**: Suppose \( M_1 \) and \( M_2 \) are CAD models created in the respective CAD systems \( C_1 \) and \( C_2 \). To decide on the interchangeability of \( M_1 \) and \( M_2 \), we construct their proxies \( M'_1 \) and \( M'_2 \) with the query responses and compute the properties of these proxies via configure. Then, evaluation would return a quantitative comparison between the models of \( M_1 \) and \( M_2 \) by using the accuracy of \( M_i \) in computing the properties of \( M_i \) and the computed values passed on from configure.

(ii) **Translation Validation**: Suppose \( M_1 \) is a CAD model authored by a CAD system \( C_1 \) and let \( M'_1 \) be its translated version in another CAD system \( C_2 \). To validate the translation, we follow the same strategy in Step-i by setting \( M_2 = M'_1 \).

Automated testing for the interoperability of CAD systems based on the interchangeability of their models can be carried out by the following steps:
• Fix a model instance $M_i$ in a CAD system $C_i$
• Identify if the system $C_j$ would validate the shape properties of $M_i$ (interoperability of $C_j$ with $C_i$)
• Repeat the same procedure by switching the roles of $C_i$ and $C_j$

3.2. Round-Robin Testing

CAD users may not know when and in which CAD system the STEP files they are working with were created and how many times they were transferred. To verify the reusability of CAD models after iterative transfers in a standard format, we investigate if any of the properties we check stabilizes at a particular iteration during Round-Robin tests[15].

Suppose $M_i$ is a CAD model created by a CAD system $C_i$ and saved as a STEP file. Let $\{M_{ij}\}_{j=0}^k$ be the sequence of models in the Round-Robin testing process for $k$ rounds of tests. We mainly consider two types of round-robin tests:

(i) Read and write $M_i$ in $C_i$ $k$ times and compute the properties of $\{M_{ij}\}$ to observe the changes and determine if there exists $l \in \{1, \ldots, k\}$ such that $P(M_{ij}) = P(M_{il})$ where $P$ denotes a property function.

(ii) Let $C_m$ be another CAD system and run the round tests between the systems $C_i$ and $C_m$, that is, $M_{ij}$ is a STEP file generated by $C_i$ and $C_m$ for $j$ even and odd, respectively. The goal is the same as in the case (i).

We note that in these experiments $C_i$ and $C_m$ denote OpenCASCADE and Rhinoceros, respectively. The diagram along with the list of actions below shows how the invariance of shape properties is investigated.

1. Construct a solid model in system $C_i$
2. Transfer the model to another system $C_j$, then transfer the translated model back to $C_i$
3. Investigate the shape properties that would change less than a specified value under the recursive format translations or model transfers
   i. Determine the property with a negligible invariance at the end of a round test
   ii. Repeat (2) and (i)

4. Experiments

In this section, we illustrate some of the experiments we carried out following our automated approach. In Section 4.1 the models are represented by template files. In Section 4.2, the models are saved in STEP format, and they either maintained their format throughout the experimentation or a format translation took place before they are saved as STEP again.

4.1. Testing for Model Interchangeability

Here is how we execute DT es t:

```
./DTest <TemplateFile1> <TemplateFile2><TestName><Tolerance>
```

<TestName> is chosen from the list {volume, surface area, Hausdorff distance, centroid).

In the following experiment, we examine the interchangeability of a CAD model read by two distinct systems.

4.1.1. Experiment

In this experiment, the coil model in Figure 3 is read as a STEP file by OpenCASCADE and converted to a 3DM format in Rhinoceros. We query the systems to build the template files, then run DT es t. The template file is provided in Appendix-A. Table 1 illustrates the key components of the template files.

| CAD System | OpenCASCADE | Rhinoceros |
|------------|-------------|------------|
| System Tolerance | $10^{-3}$ | $10^{-5}$ |
| Ball Radius | $\epsilon$ | $\epsilon$ |
| PMQ accuracy | $2\epsilon - 1$ | $2\epsilon - 1$ |
| Model | Non-Convex | Non-Convex |

Table 1: Main specifications for the electric coil model in the template file

Running DTest on (RhinoModel) Rcoil.xml and (OCCModel) Ocoil.xml Property $\epsilon = 0.0001$

Output-1:
Volume: Systems Rhino and OpenCASCADE have incompatible volumes with a difference of 11.06102413
Volume of first proxy model: 476.73668518,
Volume of second proxy model: 487.79770932
Surface Area: Systems Rhino and OpenCASCADE have incompatible areas with a difference of 132.73228961
Surface area of first proxy model: 5720.84022219,
Surface area of second proxy model: 5853.57251180
Hausdorff Distance: Systems Rhino and OpenCASCADE have an incompatible Hausdorff Distance of 1.18016199

Report:
Rhinoceros and OpenCASCADE that provide the respective models, Rcoil and Ocoil, cannot interoperate in carrying out a task that allows using Rcoil and Ocoil interchangeably with the given accuracy $\epsilon$ for the specified property.

The following shows an output of the same experiment done with ball radius equal to $1e-1$.

Running DTest on (RhinoModel) Rcoil.xml and (OCCModel) Ocoil.xml Property $\epsilon = 0.001$
Output-2:
Volume: Systems Rhino and OpenCASCADE have incompatible volumes with a difference of 1058.03869381
Volume of first proxy model: 4907.00478882,
Volume of second proxy model: 3848.96609501
Surface Area: Systems Rhino and OpenCASCADE have incompatible areas with a difference of 3907.53289433
Surface area of first proxy model: 5823.00495112,
Surface area of second proxy model: 1915.47205679
Hausdorff Distance: Systems Rhino and OpenCASCADE have an incompatible Hausdorff Distance of 4.44160210.
Report:
Rhinoceros and OpenCASCADE that provide the respective models, Rcoil and Ocoil, cannot interoperate in carrying out a task that allows using Rcoil and Ocoil interchangeably with the given accuracy $\epsilon$ for the specified property.

The results emphasize the significance of the ball radius $\epsilon$ in the property comparison that is based on proxy models. For more reliable results and better approximations, the ball radius should be chosen closer to the system tolerance and non-convex shapes should be avoided in order to capture small features and get $\epsilon$ dependent bounds on the integral properties as the theory suggests.\footnote{11}

4.2. Round-robin Testing
The experiments we present in this section show that the models could change slightly in an almost random way during transfers in STEP format.

4.2.1. Experiment 1
This experiment exemplifies the case (i) of Section 3.2. Our experiments on the STEP files provided by NIST\footnote{9} show that after a couple of round tests, the variations in the properties become hard to detect within available precision.

Figure 4: Test model labelled as NIST 904. This model is chosen from the list of models provided for a testing-round carried out by CAX Implementor forum \footnote{9}.

The Round-Robin tests of STEP files in OpenCASCADE show that the shape properties, which are considered as the basic properties to be checked for the interchangeable use of CAD models in engineering applications, exhibit a noticeable amount of change in the first few rounds (See Table\footnote{2} & \footnote{3} and then they stabilize in the third or fourth transfer in general. We note that this deduction is based on the tests we ran over a large set of STEP files provided by \footnote{9} for a round test. Additionally, the graph in Figure ?? shows how reading and writing the same model as STEP in the same system multiple times can result in relatively random changes in properties such as volume and surface area.

| Model | NISTModel.STEP Integral Property Computations | Volume | Area |
|-------|-----------------------------------------------|--------|------|
| $M_i$ | 22869.801015681573                            | 20900.779662695128 |
| $M_i$ | 22869.801015681736                            | 20900.779662695266 |
| $M_i$ | 22869.801015681785                            | 22869.80101568175 |
| $M_i$ | 22869.801015681785                            | 22869.80101568175 |

Table 2: Round-robin Test Integral Property Check Results. The model NIST 904 is read by OpenCASCADE.

| Model | NISTModel.STEP Geometric Property Computation | Centroid coordinates |
|-------|-----------------------------------------------|----------------------|
| $M_i$ | (51.67985064942907, 34.72604686416809, 2.0243846113346495) | \footnote{12} |
| $M_i$ | (51.6798506494292625, 34.7260468641684, 2.0243846113346287) | \footnote{12} |
| $M_i$ | (51.6798506494296225, 34.72604686416839, 2.0243846113346287) | \footnote{12} |
| $M_i$ | (51.6798506494296225, 34.72604686416839, 2.0243846113346287) | \footnote{12} |

Table 3: Round-robin Test Geometric Property Check Results. The model NIST 904 is read by OpenCASCADE.
4.2.2. Experiment 2

In this experiment, we aim to observe the changes due to format translations, namely, translating a model from a standard format to a native format and vice versa. In this respect, we read and rewrite the model illustrated in Figure 5. We read the model as a STEP file in Rhinoceros. Then, we translate it to the native format, 3DM, of Rhinoceros. After that, we translate it back to STEP, we observe that Rhinoceros was unable to fully close all of the solid objects in the STEP file. Some objects have naked edges (See Figure 7).

4.2.3. Experiment 3

This experiment exemplifies the case (ii) stated in Section 3.2. Here, we mainly test the STEP conversion capabilities of the systems and search for the data loss in terms of the properties. We run Round-Robin tests between a pair of distinct CAD systems, namely, OpenCASCADE and Rhinoceros. During the Round-robin testing of STEP files in the following sequence of CAD systems:

\[ \text{OCC} \rightarrow (\text{STEP}) \rightarrow \text{Rhinoceros} \rightarrow (\text{STEP}) \rightarrow \text{OCC} \]

where OCC denotes OpenCASCADE, and Rhino stands for Rhinoceros, we observe the following:

- **Change in topology:** Rhinoceros is unable to fully close all of the solid objects in this STEP file. Some objects have naked edges.

- **Change in geometry:** Shifts occur in the centroid positions

These experiments show that there is a need for a mechanism that would provide measures on the quality of the standardization. The "standard" formats such as STEP can only offer...
a weak standardization as a result of which additional healing or repair software accompanies the model transfers in standard formats. We also note that the limitation on the number of round testing for STEP file transfers poses a limitation on predicting the convergence properties of the data loss. As a result, the data loss during transfers has an unpredictable nature.

5. Technical Challenges

Here is a list of technical challenges we faced during this research:

(i) **Limited access to commercial systems**: This hinders the development of the testbed for Round-Robin experiments.

(ii) **Theoretical computational complexity**: Complete verification with respect to an external proxy model may become intractable in the presence of small features or high precision. However, this challenge can be alleviated by using selective testing, localization and statistical measures.

(iii) **Operating System & CAD software incompatibility**: Some CAD software do not work on every operating system. For example, there is no downloadable version of Rhinoceros for Linux.

(iv) **Programming language incompatibility**: Different scripting languages may be used by different CAD software, which was the case for the CAD software used in this project. Rhinoceros uses IronPython, which is firmly integrated with the .NET Framework\[17\], whereas the OpenCASCADE community uses OCE, which is a C++ 3D modeling library[18].

6. Conclusion and Future Direction

In this article, we presented an algorithmic framework and an automated approach that would test for and discover conditional CAD-to-CAD interoperability which is based on the approximately-invariant shape properties between two given CAD models. We further showed that exchanging models in standard format does not guarantee the preservation of the shape properties.

We posit that our method could be extended to interoperability testing on CAD-to-CAE and/or CAD-to-CAM systems by modifying the set of property checks and testing the additional conditions that would be required by the CAE or CAM applications.

Our system gives results with guarantees, therefore, our interoperability testing surpasses the other interoperability techniques that have been offered for CAD model verification and validation in literature. Ideally, a fully-automated system should include a single build environment to query the systems and run DTest, and allow investigating the tolerance levels that would allow the systems to interoperate with respect to the given properties if a negative interoperability result is reached within the specified tolerance. However, the technical challenges and the short project duration hindered the progress towards building this ideal system.

In future research, we plan to update the template files with additional query responses and properties, and experiment with different CAD systems and file formats. We would also like to integrate the existing commercial interoperability checks for the verification of interoperability and translation validation into our automated system approach.

Appendix A. Template Files for the Coil Model

![Figure A.9: Template file for the Rhino model](image)

![Figure A.10: Template file for the OpenCASCADE model](image)

Appendix B. Software Components

We used the following software and libraries in this research:

- **OpenCASCADE [13]**: An object-oriented C++ class library designed for fast production of advanced domain-specific CAD/CAM/CAE applications.

- **PythonOCC [13]**: A 3D CAD/CAE/PLM development framework for Python. It includes features such as advanced topological and geometrical operations, data exchange (STEP, IGES, STL import/export), GUI based visualization (wx, Qt), jupyter notebook rendering.

- **OpenCASCADE Community Edition (OCE) [18]**: A C++ 3D modeling library. It can be utilized to develop CAD/CAM softwares such as FreeCad or IfcOpenShell. It aims to gather patches/changes/improvements from the OCC community.

- **Rhinoceros 5 [19]** : A 3D computer graphics and CAD application software developed by Robert McNeel & Associates. Rhinoceros geometry is based on the NURBS model, which focuses on building mathematically precise representation of curves and freeform surfaces in computer graphics (as opposed to polygon mesh-based applications).
• Structural Bioinformatics Library (SBL) [20]: A template C++/Python library for solving structural biology problems. It provides programs (executables) for end-users and a rich framework to develop new applications.

The code for the automated system presented here is not publicly available due to the ongoing improvements but could be provided on reasonable request.

Acknowledgments

This research is funded by the DARPA contracts HR00111620042 and HR0011623402. We would like to thank Vadim Shapiro from the University of Wisconsin-Madison and ICSI, Berkeley, USA, for his helpful comments and support.

References

[1] D. Sap, V. Shapiro, On verification of interoperability of CAD systems with a focus on invariant properties, Computer-Aided Design 115 (2019) 256–266.
[2] C. Hoffmann, V. Shapiro, V. Srinivasan, Geometric interoperability via queries, Computer-Aided Design 46 (2014) 148–159.
[3] C. Hoffmann, V. Shapiro, V. Srinivasan, Geometric interoperability for resilient manufacturing, Technical report, Purdue e-Pubs, Department of Computer Science (2011).
[4] M. Behandish, V. Shapiro, A guide to property-based interoperability, Technical report, International Computer Science Institute (ICSI) (2017).
[5] CADIQ, CAD validation software, ITI: International TechnetGroup, Available at https://www.iti-global.com/cadiq (2019).
[6] CADfix, CAD translation, repair and simplification software, ITI: International TechnetGroup, Available at https://www.iti-global.com/cadfix (2019).
[7] CADdoctor, CAD healing, optimization and simplification software, Elysiun, Available at https://www.elysium-global.com/en/product/caddoctor (2019).
[8] CADinterop, Computer software, Available at https://www.cadinterop.com/en/ (2019).
[9] J. Boy, P. Kosch, D. Cheney, Recommended practices for geometric and assembly validation properties, Technical report, CAX Implementor Forum (2012).
[10] Technical Data Package 31000-A, Military standard practice, Available at http://everyspec.com/MIL-STD/MIL-STD-10000-and-Up/ (2013).
[11] D. Sap, A Review of Geometric Integrity Criteria for Military Standards - 31000A, Technical report, ICSI (2019).
[12] Lotar International: Long Term Archiving and Retrieval - LOTAR, Tech. rep., Available at http://lotar-international.org/home.html (2019).
[13] T. Paviot, PythonOCC, Computer software, Available at https://buildmedia.readthedocs.org/media/pdf/pythonocc-core (2019).
[14] Game controller, GrabCAD community, CAD files, Available at https://grabcad.com/library/xbox-one-controller-5 (2019).
[15] Round-robin, Wikipedia contributors, Online encyclopedia, Available at https://en.wikipedia.org/wiki/Round-robin (2019).
[16] PCB 3D STEP models and footprints of electronic components, Available at https://www.pcb-3d.com/models/ (2019).
[17] IronPython, Computer software, Available at https://ironpython.net (2019).
[18] T. Paviot, OpenCASCADE Community Edition, Computer software, Available at https://github.com/tpaviot/ocae (2019).
[19] Rhinoceros, Computer software, Available at https://www.rhino3d.com (2019).
[20] Structural Bioinformatics Library, Computer software, Available at https://sbl.inria.fr (2019).