Title:
A Neutral XML Design Framework For Generating Parametric Parts In Multiple CAD Systems

Authors:
Jonathan E. Sadler, jonathansdlr@gmail.com, Brigham Young University
Ryan D. Day, ryan.david.day@gmail.com, Brigham Young University
Parker G. Bronson, parbro3@gmail.com, Brigham Young University
Jacob L. Tovar, jacobltovar@gmail.com, Brigham Young University
John L. Salmon, johnsalmon@byu.edu, Brigham Young University

Keywords:
XML, Parametric Design, Design Intent, Neutral Format

DOI: 10.14733/cadconfP.2018.412-416

Introduction:
In today's global environment, a company may require the interaction and interoperability of many different CAD systems in order to model a complex mechanical assembly. Through the history of CAD there have been many attempts to create a neutral standard that transfers and preserves design intent across these systems. Current standards have limitations – they mainly preserve geometric data and not design intent. Errors in transferring these types of neutral CAD formats across systems and interpreting their design intent cost the automotive industry an estimated one billion dollars each year [1]. Finding a neutral format for CAD systems that captures design intent is therefore imperative in order to solve this problem. Most solutions in literature focus on a neutral format treated as a file that is difficult to edit and understand [3, 2, 5].

Our solution focuses on treating the neutral format as a script with features such as *if* statements and *for* loops. Moreno and Baz discussed a similar solution using scripting, but their scripts require an expert user to generate them and are not easily understood. In addition, their model is focused on non-parametric CAD programs [4]. We focus in this effort to display design intent from parametric CAD systems in a human readable format.

Methodology:
The neutral framework seeks to accomplish several tasks. Primarily the system must be able to recreate CAD models that reflect the design intent defined in the neutral framework. A second task that needs to be accomplished is that the generated models in different CAD systems must be accurate to each other for any useful engineering to occur. The overall process for how this was implemented is presented in Fig. 1.

The design framework begins with a neutral script that expresses the operations to be performed. XML was utilized for this task as it can be modified quickly and is human readable. Satisfying these primary objectives enables users to updated information and maintain recording and communicating design intent to the engineer. To achieve this, use of complex features associated with XML such as XML

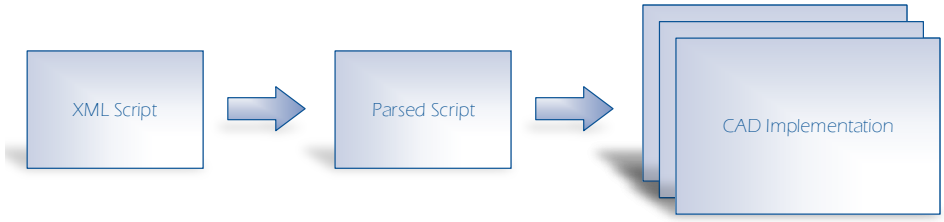

Fig. 1: Pipeline for script processing

namespaces were avoided and the neutral format was designed with terminology that is commonly used by designers and engineers. The XML files are currently stored locally; however, with few additional steps they can be stored on a central server. An example XML script can be seen in the Listing of Fig. 2.

The primary commands found in the script are used to make CAD features (i.e., extrudes, revolves, etc.). Included in the commands are the data required to make the features. For example, extrudes generally require a sketch and a value defining how far the sketch will be extruded. These features can either be referenced directly or by referencing previously created objects by their provided name.

One difficulty that arises with scripting is the ability to reference existing geometry. This was solved by use of "Finders" (See code Listing of Fig. 3). The finder code allows geometry to be searched for based on different criteria. The current method for searching existing features is based on near/far geometry. Likewise, input parameters allow finding different objects based on existing geometry (e.g., retrieving the end points from a line).

Additional functionality can be added by means of adding additional scripts. By referencing other scripts, more complex features can be created by reducing the need to have repeated code. Scripts also allow feature references and values to be passed into the script. An example can be seen in the Listing of Fig. 4.

The scripts also allow for conditional statements. This gives it the flexibility to make a variety of different parts. Currently supported is the ability to have *if* statements as well as *for* and *while* loops. Finally, input commands allow the program to prompt the user for information. This information may be to identify or select already existing features or defined values or quantities.

When the program is executed, users are presented with the scripts that are available. Upon selection, the XML is loaded into memory and interpreted for easier access to the data. During this process, it is also checked for accuracy. Following this step, the data can be translated by the respective CAD systems and versions for later use. This generally involves finding referenced features within the part as well as making the necessary API calls.

Implementation and Case Study:

The above described framework was implemented and tested on a variety of simple shapes and features to evaluate consistency and in particular to assess the framework on accuracy and flexibility. Accuracy is the frameworks ability to make identical parts consistently across multiple CAD systems. Flexibility is the ability to make parametric changes to make new parts while maintaining accuracy.

The primary case study utilized to test these aspects was an airplane rib model. This example shows the ability of the XML framework to utilize more complicated operations on a surface. Each airplane rib was generated using the developed script within the XML framework with Fig.5 showing a comparison between the CAD systems. Based on the generated visuals and the measured values, the system remains

```
<InstructionSet>
    <Instructions>
        <Revolve>
            <Sketch>
                <Plane FromDefault="XY"/>
                <Line>
                    <Name>Axis</Name>
                    <StartPoint>
                        <X>0</X>
                        <Y>0</Y>
                        <Z>0</Z>
                    </StartPoint>
                    <EndPoint>
                        <X>0</X>
                        <Y>10</Y>
                        <Z>0</Z>
                    </EndPoint>
                </Line>
                ...
            </Sketch>
            <Axis FromName="Axis" />
            <StartAngle>0</StartAngle>
            <EndAngle>360</EndAngle>
        </Extrude>
    </Instructions>
</InstructionSet>
```

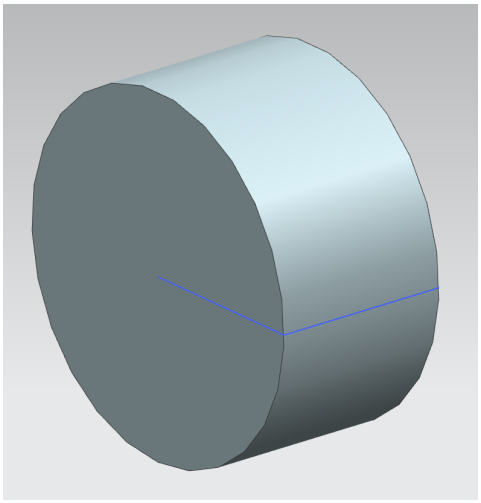

Fig. 2: Basic XML example script (left) with associated CAD output (right)

accurate for complex geometry. For example, the surface area for the 4th pocket of the model was calculated to be 19734.4707 mm^2 in NX while that in the CATIA system was 19734.992 mm^2, for a percent difference of less than 0.003%. For the 7th pocket, comparing the surface areas were 18892.6888 mm^2 and 18893.463 mm^2 for NX and CATIA respectively, for a 0.004% difference.

Conclusions:

The use of an XML neutral script allows design intent to be carried not only across CAD systems, but versions as well. By incorporating features such as conditionals and the ability to call other scripts, additional flexibility and adaptability can be added. Care must be taken to ensure files can be generated consistently across CAD systems and versions.

In the future when CAD systems adopt AP203 version 2, an alternative implementation is to make a translator that changes our simple neutral format to a AP203 version 2 implemented STEP file. Through this method one could maintain the human readability of our neutral language with the robustness and already implemented translators for AP203.

Acknowledgement:

This research was funded in part by Spirit AeroSystems, Inc.

```
<Finder>
    <Name>Point</Name>
    <References>
        <Reference Type="Near">
            <Object>
                <Plane FromDefault="XY"/>
            </Object>
        </Reference>
        <Reference Type="Near">
            <Object>
                <Plane FromDefault="YZ"/>
            </Object>
        </Reference>
        <Reference Type="Near">
            <Object>
                <Plane FromDefault="XZ"/>
            </Object>
        </Reference>
    </References>
    <Output>Point</Output>
</Finder>
```

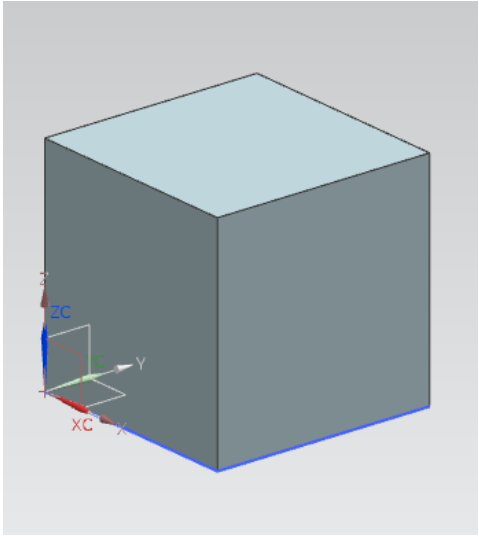

Fig. 3: Finder example script (left). If the script is used with the cube (right), the origin would be the point that is found.

References:

[1] Brunnermeier, S.B.; Martin, S.A.: Interoperability costs in the us automotive supply chain. Supply Chain Management: An International Journal, 7(2), 71–82, 2002. https://doi.org/10.1108/13598540210425821.

[2] Lee, H.C.: Evaluation of shape representation methods in step for korean industry. Evaluation, 4(02), 2015.

[3] Manhua, C.; Yanjun, S.; Guojiang, S.: Building a midcad platform for cad model integration and data exchange. In Electrical and Control Engineering (ICECE), 2011 International Conference on, 2046–2049. IEEE, 2011. https://doi.org/10.1109/ICECENG.2011.6057999.

[4] Moreno, R.; Bazán, A.: Design automation using script languages. high-level cad templates in non-parametric programs. In IOP Conference Series: Materials Science and Engineering, vol. 245, 1–9. IOP Publishing, 2017. https://doi.org/10.1088/1757-899X/245/6/062039.

[5] Staves, D.R.; Salmon, J.L.; Red, W.E.: Associative cad references in the neutral parametric canonical form. Computer-Aided Design and Applications, 14(4), 408–421, 2017. https://doi.org/10.1080/16864360.2016.1257184.

```
<Script>
    <ScriptName>Revolve</ScriptName>
    <Input>
        <Expression>
            <Name>Angle</Name>
            <Value>125</Value>
        </Expression>
        <Instruction>
            <Name>BaseSketch</Name>
            <Value FromName="Sketch" />
        </Instruction>
    </Input>
</Script>
```

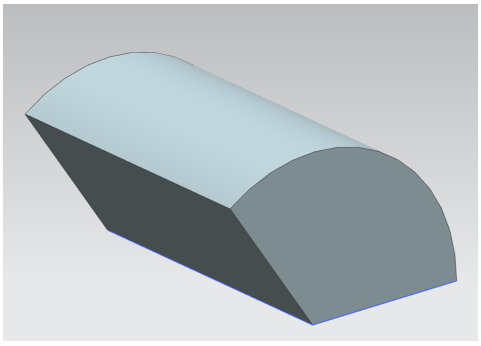

Fig. 4: XML script with called scripts (left) with associated CAD example output (right). The angle parameter and sketch are used withing the called script.

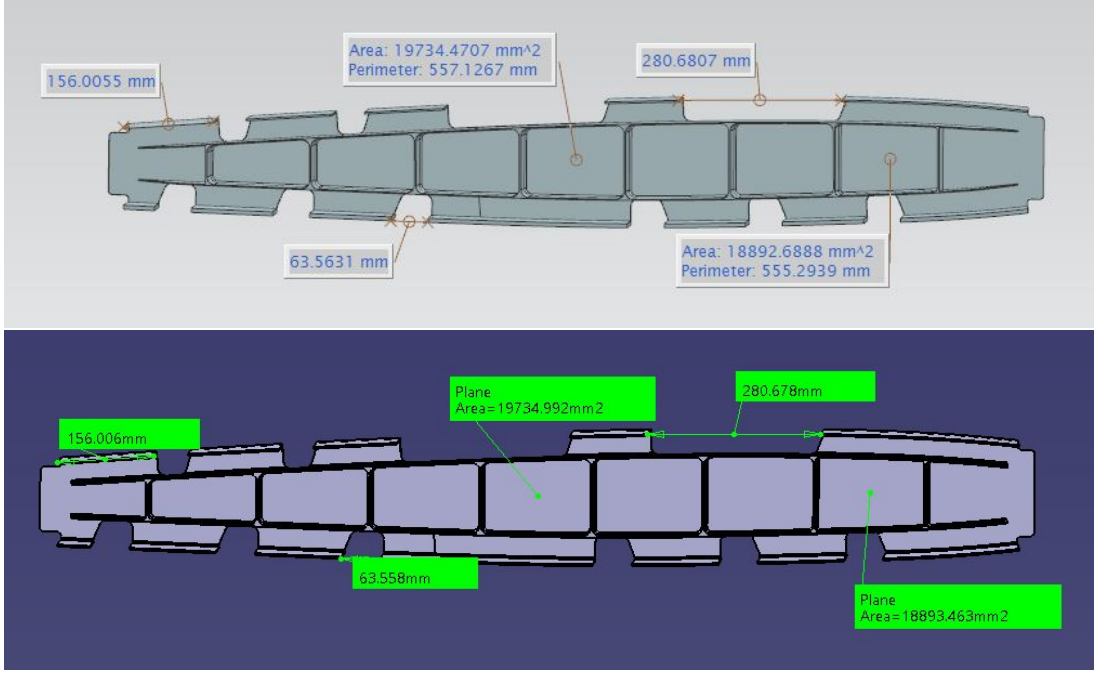

Fig. 5: NX (top) vs. CATIA (bottom) Comparison for the Airplane Rib Case Study