USE OF XML FILE FOR MOTION ANALYSIS DATA MIGRATION BETWEEN CAD TOOLS

[1]Amit C. Kamble, [2]Prof. A. S. Rao
[1]M Tech, CAD CAM & Automation, VJTI Mumbai,
[2]Assistant Professor Mechanical Engg, VJTI Mumbai
Email:[1]amitkamble035@gmail.com, [2]asrao@vjti.org.in

Abstract- There are many analysis tools used in industries. Most of the times, it is required to exchange the data between these analysis tools. XML is light weight, platform and software independent file format. Being a software independent format, XML provide good option for information exchange. Data required for motion analysis like mass, inertia, transformations etc. can be easily stored in the XML file. Proposed import algorithm reads the XML file and creates the entities in the analysis tool. To validate the import code, simple motion analysis problem is solved in commercial analysis tool and same problem is imported in another motion solver through XML. In the implementation of the algorithm, XML file is imported with the help of import code in the motion solver and results are compared with the commercial analysis tool. Code is developed considering particular XML schema. The schema can be different of different analysis tool but the approach will be the same.

Index Terms— Import, Motion Analysis, XML.

I. INTRODUCTION

Motion analysis is required at many levels in the industries. There are many tools used for analysis so exchange of data among different analysis tool is important to industries. Every analysis tool has its own file formats. We cannot open these files in other software directly.
Industries need some standard neutral file format. This format should be platform and machine independent. Also it should be lightweight so that it can be easily transferred.
Currently some standard formats like IGES which are used to transfer CAD data. In a motion analysis there are many entities like bodies, joints, forces, constraints present. All these entities have many parameters and we have to take care of all these. XML is file format which is used to store the data. All the entities required for motion analysis can be stored in human readable format in XML. Also it is lightweight file as compared to other neutral file formats so it is suitable for data exchange.
In this project algorithm is developed and also implementation is done. Some free tools are used in this project. To validate the algorithm and the code, one simple motion analysis is done in the commercial analysis tool and results are recorded. Same motion model is imported in another solver through XML and results are compared.

There are many parsers available which can parse the XML file. But parsers will create run time data so it is required to convert that run time data in to entities and form the complete analysis model. This analysis model can be different for different solvers or it can be exported to the solver input file.

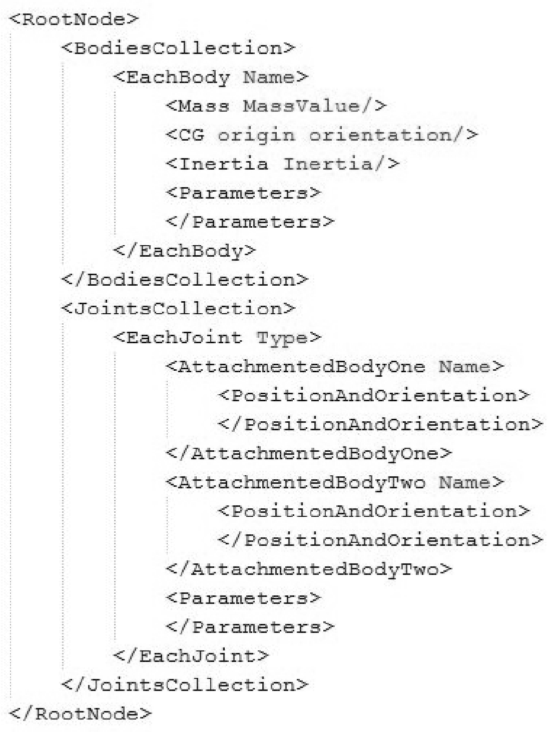
Code is developed to import the XML file in the GEAR (Geometric Engine for Articulated Rigid Body) solver. This is free motion solver available on internet. Also free XML parser is used for parsing. There are two main steps, XML parsing and entities creation. In the first phase, only parsing of the XML is done and data is stored run time in the memory. XML parsing can be generic; it is not dependent on the solver. In second phase this run time data is converted in to the motion entities. But entities creation is dependent on solver because each solver has its own formats. Also some visualization libraries are used to plot the results and graphs.

II. Procedure for Paper Submission

In XML file all data is stored in the nodes. There is one root node and this root node has many child nodes. Entities of similar type are grouped together to form a collection of entities. All these collection nodes are the child of the root node. Collection node has number of entities. All the information related to the particular entity is present in the entity node. Some time there is reference to other entity, in such cases reference is mention in the node. There is no duplication of the entities.

To read the XML file, parser is used. There are many parsers available in the market. But if you use such parser from third party then the import code is directly depends on the third party. To avoid this dependency, one interface is developed. This interface internally calls the methods of the third party parsers.

The importance of this interface is that we can easily change the third party parser. Even if we change the parser, there will not be any effect on the import code. Only implementation of the interface will be changed. This parser reads the XML file and run time data is maintained. While creating the individual entities, it is inconvenient to read the XML file every time to take the required data. Instead of that retrieve the required data from this run time data. This approach improves the performance of the program and transfers very large analysis model in short time.



```
<RootNode>
    <BodiesCollection>
        <EachBody Name>
            <Mass MassValue/>
            <CG origin orientation/>
            <Inertia Inertia/>
            <Parameters>
            </Parameters>
        </EachBody>
    </BodiesCollection>
    <JointsCollection>
        <EachJoint Type>
            <AttachmentedBodyOne Name>
                <PositionAndOrientation>
                </PositionAndOrientation>
            </AttachmentedBodyOne>
            <AttachmentedBodyTwo Name>
                <PositionAndOrientation>
                </PositionAndOrientation>
            </AttachmentedBodyTwo>
            <Parameters>
            </Parameters>
        </EachJoint>
    </JointsCollection>
</RootNode>
```

Fig.1 Sample XML schema

This module stores the data and provides methods for traversing through it. Traversing is very useful in entity creation. XML file contains separate node for each entity and in this node all properties and parameters required for motion analysis are mentioned. All data required to create the entity is present in the single node, however some time particular entity can be referred from other entity.

For the above file, first get the root node. This root node contains all information. If you take the child node on the root node, you will get the body’s collection node. There can be multiple children. In this sample file two children are present. This body collection has many bodies. We can get required body node by referring its number. Similarly any property of the body can be extracted from each body node. The run time data formed during parsing is automatically deleted by the parser when we release the parser.

III. Analysis entity creation

After parsing of the XML file, we get the run time data. We have to convert that run time data in to the entities and construct the analysis model. Bodies, joints and forces are the basic entities. The implementation of the entity creation code is depends on the solver. For each solver or analysis tool, there is particular format of the entities. For this project we have selected GEAR (Geometric Engine for Articulated Rigid

Body) motion solver. It is free motion solver available on internet. This solver can solve the articulated rigid body motion. It supports bodies, some types of the joints, forces. For the motion analysis, geometric data is not required. So the XML file doesn't contains any geometrical data like shape of the body.

First we parse the XML file and get the root node and store it in data member. Then scan the root node and check for the entity collection node. If collection node is present then get the each entity node from that collection which will have all the information required to create the entity. Sometime other entities are required to create the entity. For example to create the joint, two bodies are required. If the reference entity is required then search it in the local entity collection. Get that reference entity and use it when required in the entity creation. If the reference entity is not present in the entity collection then get the next each child node and create that entity. After creating the entity, store this entity in the local entity collection for the reference. Check for the next entities in the same entity collection node and create them in the same way. When we finish with creation of all entities in the entity collection node then scan in the root node for any other type of entity collection node. If such collection is present then create all entities in similar steps.

For creating the body with this algorithm, we scan the run type data and get the body collection node. This node contains all bodies in the model. Count the number of individual bodies and get the each body node and send it to the body creation method. The data required to create the body is present in the each body node. For motion analysis, mainly mass, inertia, transformations are required. Along with these data there are some parameters which define some additional properties. In the code, the special class is inherited from the solver class to handle the rendering of the bodies. This rendering of the body will not be exact. It is representative to visualize only. While implementing the rendering code, it is assumed that the body is in cuboid shape and dimensions are calculated on the basis of the inertia values. The length value is calculated using following formula

$$Lx = ((6/mass) * (Izz + Iyy - Ixx))^{(1/2)} \quad (1)$$

Also, Ly and Lz are calculated using similar formulas. Mass value and inertia values are present in the XML. So for visualization we can calculate the dimensions in a code. After creation of the body, store it in the local vector so that while creating the joints, we can easily refer this body when required. If any other body is present in the body collection node then create that in similar way. After creation of all the bodies present in the body collection, scan the root node and check if any other entity collection like joints and forces are present and create all entities present in that collection. While creating the joint, we required reference bodies. All bodies will be present in the local vector so find the required body in this entity collection. Use these found bodies to create the joints. The position and orientation of the bodies is present in the each joint node. Set these transformations on the joints and bodies. This transformation is in the local reference frame of the body. That means it defines position and orientation and the joint in the body.

Local entity collection stores all entities according to their type. In the GEAR solver, motion model is built at the end of the process. So when reference entity is required, we cannot search the entity in the motion model. To overcome this problem we have to store all entities in the local collection temporary.

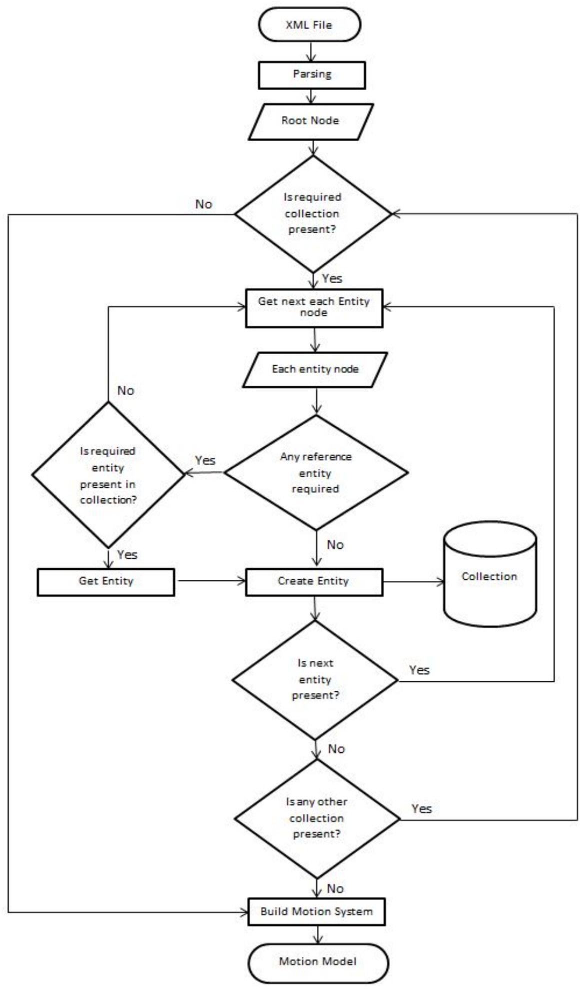

XML File
Parsing
Root Node
Is required collection present?
No
Yes
Get next each Entity node
Each entity node
Any reference entity required
Yes
No
Is required entity present in collection?
No
Yes
Get Entity
Create Entity
Collection
Is next entity present?
Yes
No
Is any other collection present?
Yes
No
Build Motion System
Motion Model

Fig. 2 Flow chart of algorithm

Many other solvers build the system step by step. In such case there no need of the local collection. We can directly search the reference entity in the motion model itself. After creation of the all entities present in the collection nodes, call the solver methods to build the system. This is solver specific step. GEAR solver starts scanning from the fixed ground body and detects all bodies and joints present in the system. Once the whole motion system is imported in the solver, we can call the solver methods to solve the problem and can get the results. The results of solver are depended on the analysis model created by the code. So there should not be any data loss during the transfer of the model from one solver to another. The data loss can be possible at two places, one is when model is stored in to the XML file and other is when XML file is parsed an entities are created. In this project it is assumed that there is no any data loss during saving the XML file.

IV. Validation

The purpose of this project is to exchange the motion analysis data between different analysis tools. So to validate the migration, we have compared the analysis results of two solvers. As we have to validate the migration of data only, no need to validate the solution mathematically. It is expected that the results obtained from the first solver should be matched with the results obtained from second solver after data migration. If these results match, then we can say the data migration is done successfully and there is no any data loss. First we solve the simple motion problem in commercial analysis solver and record some results. Then we export the analysis model in the XML file. This XML file is imported in the GEAR solver using the import code. You can use any language to build the libraries. In this project C++ is used and two libraries are built for parsing and for entity creation. Pares library is not directly used. The entity creation library calls the function from the parser library.

The problem of compound pendulum is solved in the commercial software. The dimensions of the pendulum are as shown in the figure and the mass is 5 kg. We iteratively solve this problem by changing the pivot point so that we can get many sets of results to validate the data.

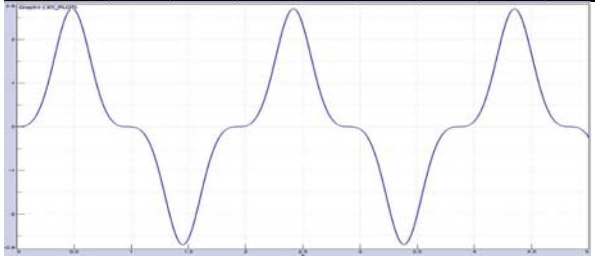
Pivot point is kept at 1 mm, 100 mm, 200 mm and 300 mm from top. The maximum displacement, velocity, acceleration and time period of center of mass is obtained from the solver. The pendulum is released from horizontal position. In this problem there is one fixed body to which the pendulum is pivoted with revolute joint. After getting results from commercial analysis tool, we export it in the XML format. The exported XML file contains fixed body node and pendulum body node. It also contains joint node. After importing it in GEAR solver, we can see the position and orientation of the pendulum is similar. After solving the problem, we get the displacement, velocity and acceleration values. Also we can plot the graph of these values.

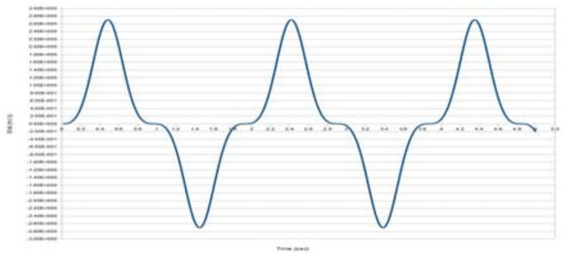
From the table below, we can see that the results obtained from the GEAR solver are matching within the limit with the commercial analysis tool. Also we can get the values of all these quantities for particular time step and can plot the graph of all these values. The graph profile shows the nature of the motion.

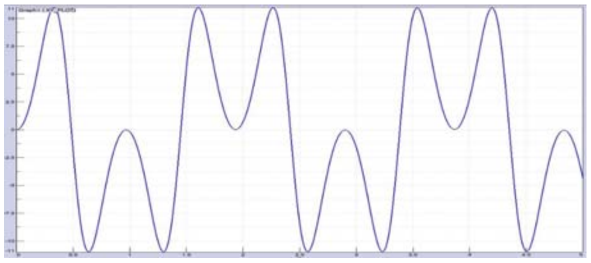
From these graphs we can see that the profile of graphs is matching. Also we can get the value of displacement, velocity and acceleration at any particular time. Also from graph we can easily find the time period of the pendulum. All these values are matching within the limit.

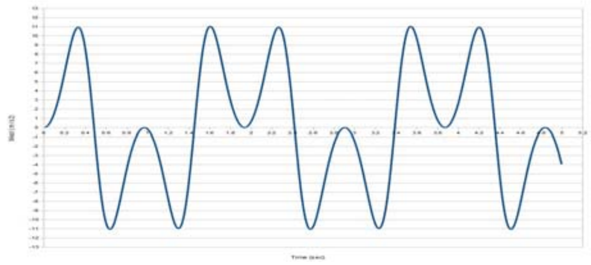
Table 1. Results from solvers

Sr. No	Pivot from top (m)	Results from commercial analysis tool				Results from GEAR solver after import			
		Disp. (m)	Vel (m/s)	Accl. (m/s^2)	T (sec)	Disp. (m)	Vel (m/s)	Accl. (m/s^2)	T (sec)
1	0.001	0.499	2.7	10.99	1.93	0.499	2.7	11.04	1.94
2	0.1	0.399	2.27	9.64	1.85	0.4	2.27	9.69	1.85
3	0.2	0.3	1.74	7.6	1.8	0.3	1.74	7.65	1.8
4	0.3	0.199	1.124	4.74	1.86	0.2	1.12	4.79	1.86
5	0.4	0.1	0.456	1.56	2.29	0.1	0.45	1.6	2.29


Graph 1. Velocity Vs Time from commercial tool


Graph 2. Velocity vs Time from GEAR after import


Graph 3. Acceleration vs Time from Commercial tool


Graph 4. Acceleration vs Time from GEAR after import

V. Limitations

In this project only some entities are supported. There are many types of joints, forces, constraints are not supported. To exchange complete data, we have to support all such entities. Also the geometric data is not present in the XML file so we cannot import it. As geometrical data is not needed for motion analysis, it can be ignored. This geometrical data is for visualization only. Motion analysis results are not dependent on the geometry of the body. But for simulation purpose, geometry may be required. To support the geometrical data exchange we have to define the schema. Also each solver has their own limitations so we can get some deviation in the results.

VI. Conclusion

We can transfer the motion analysis data through XML file to other analysis tool. We need a XML parser to parse the XML file. But by using the wrapper we can avoid the direct dependency of the import code. Implementation of import code is depended on the solver format. As there no any difference in the results from both solvers, the import is successfully done. There are some limitations but that can be overcome.

REFERENCES

[1] Guk-Heon Choi, Duhwan Mun and Soonhung 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).

[2] Lubomir Dimitrov and Fani Valchkova, “Problems With 3d Data Exchange Between Cad Systems Using Neutral Formats” Proceedings in Manufacturing Systems, Volume 6, Issue 3, 2011.

[3] Nathan W. Hartman, Ed.D., Adrian Lim, “Examining Neutral Formats for Visualization and Data Exchange” Proceedings of The 2008 IAJC-IJME International Conference.

[4] Nathan W. Hartman, “Evaluating lightweight 3D graphics formats for product visualization and data exchange” journal of applied science & engineering technology 2009.