In [6] [7] [8] [9] [10] [11], all these data visualization and transformation tools represents the data in the form of a tree and the relationship/ mappings in the form of lines, the only difference between them is the equivalent code which is generated. It varies from C++, C##, J#, XSLT, JAVA etc.. All these tools were developed to work as a part of another system, like in [7], the TIBCO business event mapper was developed to work in conjunction with TIBCO studio, hence these tools cannot work as a standalone application. As a result, these tools cannot be plugged into the other system where such feature is required [1].

The tools discussed in [6] [9] [10], were developed to work only for windows based operating system, hence these tools was of no use if mapping feature is required on other platforms such as Linux.

## 3. Mathematical Model

The mapper can be represented in the mathematical set form as follows:-

S = {s, e, I, O, Fs, DD, NDD, Φs , Su, Fa}
Where S represents proposed system.

s = start state = load and import the required files.
e = end state = generation of xslt and updated source and target xsds.

I = input set = {sXSD, tXSD, eS, fP, pXSLT}
Where,
sXSD = source xml schema definition.
tXSD = target xml schema definition.
eS = external schema.
fP = function provider.
pXSLT = previously generated xslt with the same set of sXSD tXSD and eS.

O = Output set = {xsltEquivalent, updatedSXSD, updatedTXSD}.

Where,
xsltEquivalent = XSLT code generated from the relationships / mapping created by the user.
updatedSXSD = updated source xml schema definition.
updatedTXSD = updated target xml schema definition.

Fs = set of supporting functions of the mapper, exposed to the user = {createMapper(), displayMapper(), generateXSLT(), destroyMapper(), fetchUpdatedXsd()}

DD = Deterministic Data = {sXSD, tXSD, eS, fP, pXSLT}
NDD = Non – Deterministic Data = {connections, operations}.

Where,
connections = mapping / relationships generated by user between source tree node and target tree node.
operations = various operations applied by the user.

Φs = basic functions of the system
= {
parseInputXSD(),
generateJsTreeObject(),
generateJSTreeObject(),
traverseTree(),
parseJSTreeObjectToGenerateXSLT(),
searchNode(),
generateXpath() ,
createConnections(),
applyFunction()
}.

Su = Success state / condition of the mapper.
- Generated syntactically correct XSLT code.
- Generated logically correct XSLT code.
- Generation of update source XSD and external schemas as per the modifications performed by the user.

Fa = Failure state / condition of the mapper.
- Generation of Invalid XSLT.
- Improper visualization of data.
- Improper visualization of connections while expanding or collapsing tree nodes.
- Generation of invalid source or target XSD.

## 4. Proposed System

The proposed system is responsible for generating a data visualization and transformation tool which generates visualization of the input data in the tree format and XSLT code which would be used for transformation of one xml into another xml. It has following three stages:

1) Pre – processing of the data.
2) Generation of mapping/relationship between tree nodes.
3) Validation.
4) Generation of transformation code.
5) Generation of new source and target XML schema definitions.

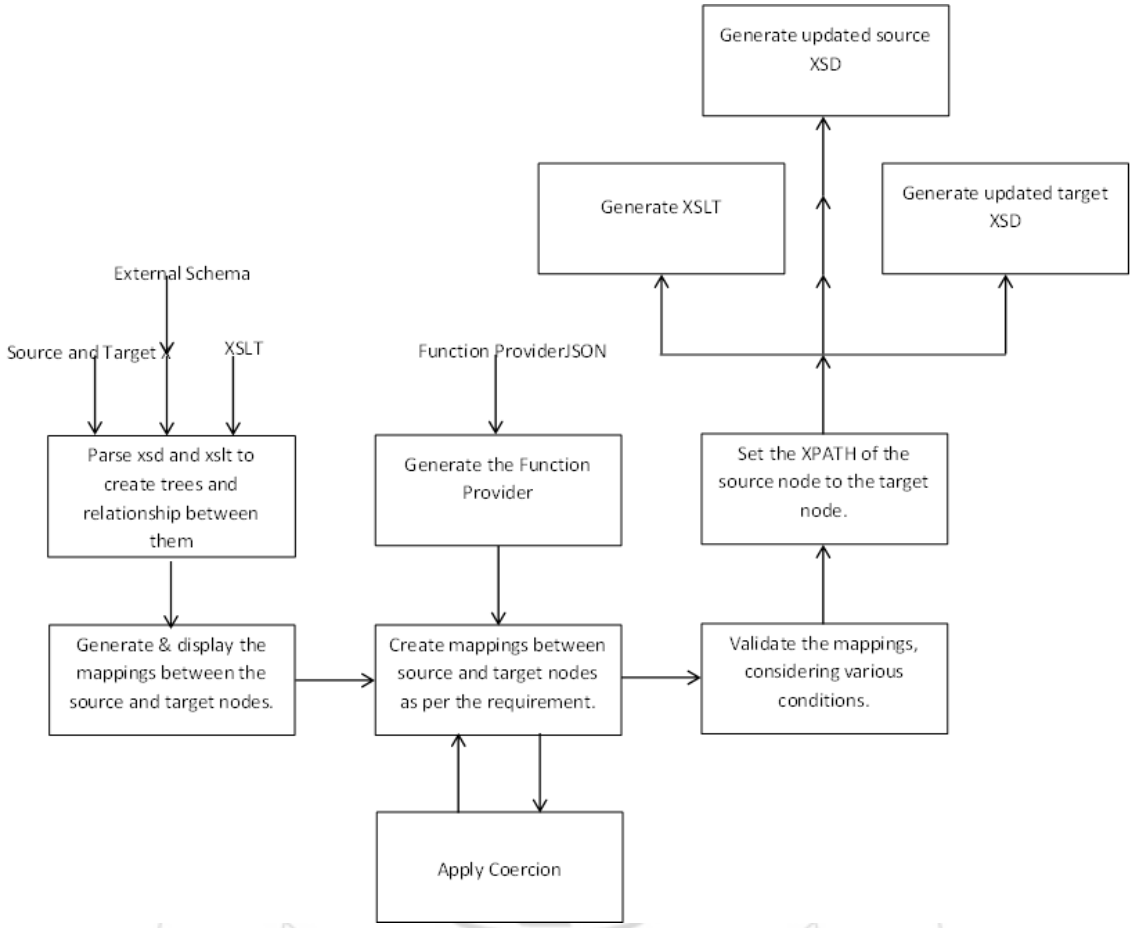


**Figure 1:** Block diagram of proposed system

1) Pre – processing of the data: - This system accepts the input in the form of a Java script object, which has following key – value pairs.
   - Source tree: - Xml Schema Definition (XSD) of source tree. The generated XSLT will be applied on xml which satisfies this XSD.
   - Target tree: - Xml Schema Definition (XSD) of target tree. The xml generated after applying generated XSLT on an xml which satisfies source XSD, would require to satisfy target XSD.
   - External Schema: - The source and target XSDs could contain external reference schema. Hence, these are required to be resolved, this key will contain the locations from where external schemas can be resolved.
   - The source and target XSDs are resolved using external schemas and parsed to generate the equivalent Java script object. Using this Java script object, equivalent trees in the form of HTML is generated that is XSD's are represented in the form of HTML trees.
   - There is a possibility that there is an XSLT code previously generated using this tool with the same set of source and target XSDs. So it would be required to generate Java script object on the basis of this XSLT and use it to generate mapping / relationship between the source and target tree nodes.

2) Generation of the mapping / relationship between the nodes: - The elements and attributes of the XSD will be represented in the form of nodes of the tree. For example the first element node will be acting as root node. The mapping will be represented by connecting lines between the source tree nodes to the target tree node. For generating the relationship, it may be the requirement to apply some operations over the source node and the results of those operations given to the target node. Hence, in the input one of the key – value pair would be function provider XML which would be containing the definitions of the function which can be applied. The example of such operations is the concatenation of string, finding maximum of two numbers, etc. The function provider may contain other user define functions. The user would be able to create relations between source and the target node as per his requirements. There could be a possibility of the presence of an 'any element' in source XSD. Hence, this 'any element' can be replaced by element of another complex type or by another XSD. So as shown in the diagram, there is an option to apply the complex type element or new XSD in the form of 'apply coercion'.
3) Validation: After applying generation of the relationship between the nodes, it is required to validate those. The first case of validation is in the case, if the coercion is applied, it would have modified the source XSD. Hence it is required to check whether updated XSD is valid or not. Another case is when XSLT statements such as if, for – each or choice is applied to the target node, in such case validation is required to be done. The validation system would not only be checking for errors and warnings, but also giving possible reasons and the steps to be taken to

eradicate those errors and warnings.

4) Generation of transformation code: - If the generated relationships are valid, then user can fetch the XSLT code. This XSLT code should be applied to an XML which satisfies the source XSD to generate an XML which satisfies the target XSD.
5) Generation of new source and target XML schema definitions: - As discussed earlier, the source and target trees can be modified by the user at run time. Hence, to generate the same state of the system, it may be required to get updated source and target XSD.

## 5. Algorithm

The algorithm to populate a web based system with this mapper tool is as follows:

1) Create a div element tag with a unique id on the web page where the mapper has to be populated.
2) Include jQuery, jsTree and jsPlumb source files & corresponding stylesheets.
3) Import the js file of the tool and its style sheet.
4) Select the DOM of the div created in the first step using jQuery.
5) On selected Dom, create a new instance of the mapper by calling the widget. During this step, pass the parameters required for initialization of the system such as input XSDs, XSLT (if existing with same set of XSDs), required size of the parameter, locations of external schemas if available.
6) Call display function of the widget on the instance created in step 5 to populate the system with this tool.

## 6. Experimental Setup

The system was developed using Java script and Java script based frameworks. To use this system, the browsers should support Java script and Java script frame – works. The framework and technology specifications are as follows:

- jQuery 1.11.1
- jstree 3.0.8
- jsPlumb 1.4.1

Languages: HTML, CSS, Java Script.
Browsers: IE11, FireFox 34.0.5, Chrome 39 and higher versions.

## 7. Results

- Generic: All the existing browsers running on any of the operating systems, supports HTML, CSS and Java script. Hence, these browsers will support frameworks mentioned in section 5. Hence, this system can be utilized without taking into consideration the browser or the operating system on which the browser is running.
- Pluggable: As discussed in the algorithm section, the only requirement for populating a web based system with this tool is to create a unique html DIV element and a java script function call to create and display the mapper. Hence, from this it can be said that, this tool is pluggable in any web based system where such visualization and transformation feature is required.

- Light weight:

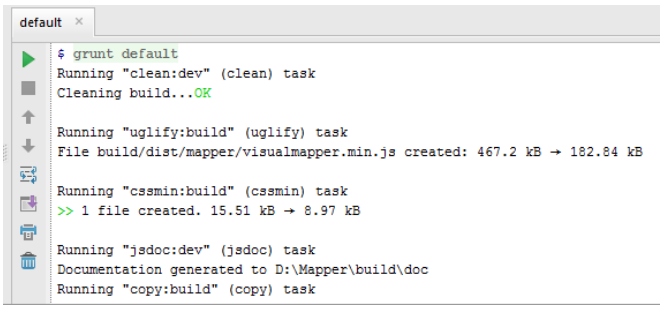


**Figure 2:** Size of minified java script file and .css file.

As shown in figure 2, the size of the minified version of the developed system is 182.84 kB and that of the cascading style sheet (.css) is 8.97 kB. Hence the overall, size of the system is 191.81 kB i.e. less 200 kB, which means that the overhead, which a system will incur by plugging this tool, will be only 191.81 kB while the size of mapper tool from the stylus – studio of version 15r2 is about 87,163 kB i.e. 8.7 MB. The other data visualization and transformation tools are available as part of development platforms and not as standalone plugins, hence it is ought to have large size compare to this tool.

## 8. Conclusion

In this paper, a system is discussed which is capable of generating an XSLT code that is used to transform an XML to another XML. The data formats were represented in the form of trees while the relations between them, i.e. the transformation rules were represented using lines and on the basis of this graphical representation, transformation (XSLT) code was generated.

## References

[1] Rahil A. Khera, P.S.Game, "A Survey of Data Visualization and Transformation Tools," International Journal of Science and Research, Volume 4 Issue 4, pp. 2315 – 2317, Jan 2015.
[2] Alessandro Raffio, Daniele Braga, "Clip: a Visual Language for Explicit Schema Mappings," in IEEE' ICDE 2008, 978-1-4244-1837-4
[3] Tae-Jin Ha, Hye-Ja Bang, "Implementation of XSLT-based Schema Mapper using RCP", in IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.5, May 2009
[4] Sebastian Bossung, Hermann Stoeckle "Automated Data Mapping Specification via Schema Heuristics and User Interaction " in Proceedings of the 19th International Conference on Automated Software Engineering 2004.
[5] Lucian Popa, Yannis Velegrakis, "Translating Web Data" in Proc. 2002 Very Large Databases Conference (VLDB'02), pp. 598-609.

[6] BizTalk Mapper, "Using BizTalk Mapper" https://msdn.microsoft.com/en-us/library/aa547076.aspx., Dec 25, 2014.
[7] ALTOVA Map Force, "MapForce – Graphical Data Mapping, Conversion, and Integration Tool" Available: http://www.altova.com/mapforce.html, Jan, 05,2015
[8] TIBCOMapper,https://docs.tibco.com/pub/businesseventsts_process_orchestration/1.1.2/doc/pdf/tib_be_process_developers_guide.pdf, Oct 18,2014
[9] David Handlos, "BizTalk Mapper 2004 V/s Map Force 2004", http://www.osnews.com/story/6809, Jan, 10,2015.
[10] "XML Editor, XML Tools and XQuery – Stylus Studio " http://www.stylusstudio.com, Jan, 30, 2015.
[11] "Transforming Data Using XQuery Mapper", http://docs.oracle.com/cd/E13171_01/alsb/docs21/dtguide/dtguideMapper.html, Jan, 31, 2015.

## Author Profile

**Rahil Khera** is pursuing a Master of Engineering in Computer Engineering from Computer Dept. of Pune Institute of Computer Technology, under the affiliation of Savitribai Phule, Pune University. He received his B.E degree in Computer Engineering in 2010.

**Prof. P.S. Game** is an Assistant Professor in the computer department of Pune Institute of Computer Technology.