Software Usability: A Comparison Between Two Tree-Structured Data Transformation Languages

Nikita Schmidt
Department of Computer Science
University College Dublin
Belfield, Dublin 4, Ireland
cetus@cnds.ucd.ie

Corina Sas
Computing Department
Lancaster University
Lancaster, LA1 4YR, UK
c.sas@lancaster.ac.uk

ABSTRACT
This paper presents the results of a software usability study, involving both subjective and objective evaluation. It compares a popular XML data transformation language (XSLT) and a general purpose rule-based tree manipulation language which addresses some of the XML and XSLT limitations. The benefits of the evaluation study are discussed.

Author Keywords
Data transformation language, metadata, tree-structured data, software metrics, software usability.

ACM Classification Keywords
H1.2. User/Machine Systems: Software psychology.

INTRODUCTION
Hierarchical (tree-structured) formats have long been used for data and metadata representation. The explosive growth of the Internet, and then of the World Wide Web, has emphasised the need to exchange heterogeneous data structures between diverse networked systems, and hierarchical formats came to the rescue. Tasks such as extraction of relevant fragments from tree-structured records, or conversion from one structure to another, became very common. Specifying such tasks is labour-intensive, as it requires human understanding of the semantics of data formats involved. The usability of systems that do this conversion and, in particular, of their specification languages has effect on human productivity and quality of data and metadata processing. Unfortunately, despite the large amount of theoretical work in this field, fewer studies focused on implementing these theoretical results and even fewer on evaluating them [5,6].

The Extensible Stylesheet Language Transformations (XSLT) [14] language is commonly used for transforming tree-structured data. It owes its popularity to the widespread adoption of XML (Extensible Markup Language), which is an explicitly hierarchical format. By design, XML is a *document markup* language. Its use for other types of data is beyond the domain for which is was originally intended. Similarly, XSLT's design goal is to aid in *presentation* of XML *documents*. Despite that, and primarily because of the existing gap between theory and practice of tree-structured data manipulation, XSLT is often used for general purpose manipulation of XML-encoded data.

In order to cover this gap, an abstract architecture for tree-structured data manipulation has been developed. Its aim is to provide a common theoretical foundation for a variety of practical tasks that involve processing of tree-structured documents. The architecture was implemented through the Tree Processing Machine (TPM) which is a general purpose tree manipulation tool that has at its core a Turing-complete computational model based on pointed string trees [10,11].

This paper presents an experimental evaluation which has been carried out by comparing the TPM language with XSLT. Both subjective and objective studies have been conducted in order to evaluate the usability of the TPM system and its language.

Subjective methods consist of users' attitude measurement regarding their interaction with the system, focusing primarily on user's satisfaction. A common approach to objectively measuring software complexity is through computing *software metrics*. These metrics arose in a research attempt to find relationships between the characteristics of programs and difficulty of performing programming tasks [2]. Different types of complexity metrics exist, such as [1,7]: number of lines of code; number of lexical entities (token count); functions of the number of operators and operands in the program; number of linearly independent execution paths through the program; logical complexity metrics; amount of information that flows in and out of a procedure.

Of these metrics, the last one is not usable due to the very low number (1–3) of procedures (i.e., functions in TPM and templates in XSLT) and their inputs and outputs used in the solutions to the sample tasks. The previous three metrics are oriented towards conventional procedural languages and are hard to apply to the specialised languages of TPM

NordiCHI '04, October 23-27, 2004 Tampere, Finland

and XSLT. Thus, the lines of code and token count metrics have been employed for this evaluation. The lines of code metric has long been noted for its stability across different programming languages and thus often used for programming language productivity comparisons. This makes it especially suitable for this study. In addition, to accommodate the similar "finger typing" metric [12], the "raw" program size (the number of characters without indentation) was also included in this evaluation.

EXPERIMENTAL DESIGN

Procedure

This usability study is organised as a *within subjects* experiment where the independent variable is the type of the system: TPM or XSLT. Subjects were randomly assigned to two groups where the order of exposure to the two conditions of the independent variable varied. Before solving the tasks, participants are invited to read the documentation for the system under evaluation. Then, they go through six programming tasks: three examples, and three exercises designed to be done by the subjects themselves. These tasks are programmed in the language of the first system evaluated by each subject. They were chosen to cover different classes of problems related to data transformation:

- tag extraction – extraction of all elements names that are children of <link> elements directly under the root of the input document;
- 'identity' transformation – passing any input document to the output without changes;
- name splitting 1 – splitting of people's names represented in the form of '[Given-Name] Family-name' into individual elements for each Given-Name and Family-Name;
- name splitting 2 – splitting of people's names represented as 'Family-Name [, Given-Name]' into individual elements for each Given-Name and Family-Name;
- character mapping – conversion of strings while substituting characters according to a mapping table;
- country code mapping – conversion of text elements according to a string mapping table.

A printed copy of the evaluation scenario with task descriptions was given to each participant at the beginning of the experiment. After the completion of the tasks, participants are asked to fill out a questionnaire regarding the perceived usability of the system they used. After a break, the users follow the same route for the second system. To assist the subjects in performing evaluation tasks, a web interface to both systems (TPM and XSLT) is provided. The interface allows the subjects to type in and edit their solutions (transformation programs) and input data, and to compute and see the processing results. This interface is organised as a set of six pages, one page per task. Figure 1 presents the web interface for the character mapping task.

Pages with examples are pre-filled with solutions, whereas pages with exercises contain just program templates which the subjects can use as a starting point. Navigation links to

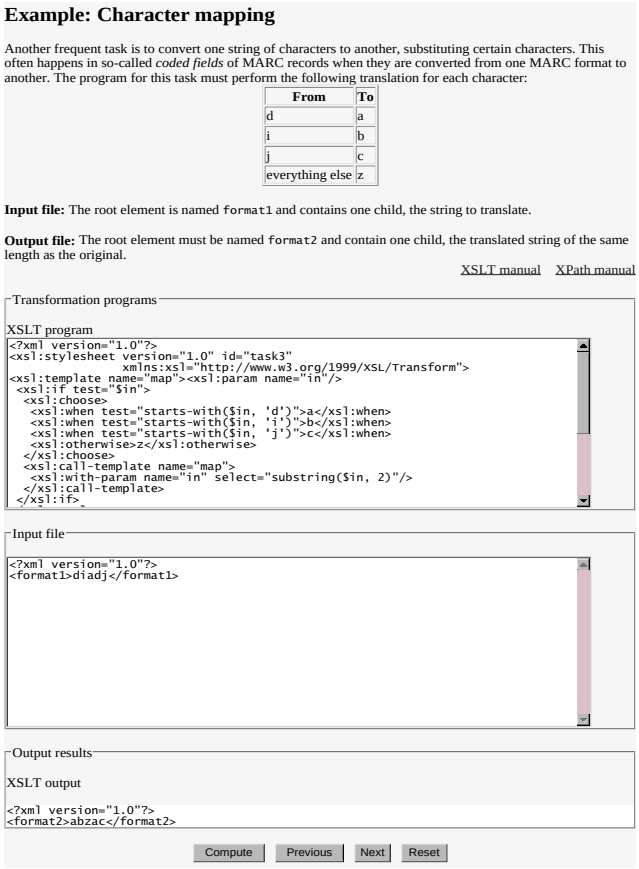

Figure 1. Example of web interface for evaluation.

the next and previous tasks are provided. The web interface is implemented as a 167-line Python [3] script, running under Apache web server[1] as a CGI (Common Gateway Interface)[2] application.

Documentation for the TPM system was given to the participants in the form of a printed copy of the TPM User's Manual [10]. For XSLT, links to the W3C XSLT and XPath Recommendations [13,14] (serving as user manuals) were provided on each page of the Web interface.

Participants

The sample consisted of 13 students and staff from University College Dublin, with different levels of education and computing experience, and mixed gender. From this sample, only 6 returned completed questionnaires for both systems, and 2 more returned one questionnaire each (one on TPM, one on XSLT). The other 5 subjects did not consider themselves familiar enough with the area. Study participants (those who have successfully evaluated at least one system) are all men, within the age range 21–36. None of them have been previously exposed to the TPM system and a few have had some experience of working with XSLT.

[1] URL: http://www.apache.org/
[2] URL: http://hoohoo.ncsa.uiuc.edu/cgi/

RESULTS

Subjective Evaluation: Perceived Usability

The perceived usability questionnaire contains 21 items measured on a 5-point Likert scale from 1 (disagree) to 5 (agree). These items are loosely based on published instruments for measuring usability [4,9]. The Cronbach's alpha coefficient of 0.89 indicates its high reliability.

Table 1 presents the mean, median, and standard deviation of scores along four usability dimensions, and a summary across all questionnaire items (overall usability). The questionnaire and its mapping onto these dimensions is presented in [10]. For each dimension, the scores of all its constituent items for the same participant were averaged, and these average values were analysed across the respondents. As Table 1 shows, all dimensions imply a medium to good level of satisfaction (rank 5 means completely satisfied, and 1 means completely unsatisfied) for the TPM system, and a rather poor level of satisfaction for XSLT system.

Dimensions	TPM		XSLT	
	Median	Mean (SD)	Median	Mean (SD)
Learnability	3.14	3.04 (0.96)	2.29	2.37 (0.81)
Effectiveness	4.14	4.04 (0.70)	2.86	2.94 (0.64)
Efficiency	3.33	3.00 (0.58)	2.67	2.71 (0.76)
Satisfaction	3.25	3.28 (0.76)	2.00	2.57 (0.83)
Usability	3.38	3.41 (0.62)	2.43	2.65 (0.48)

Table 1. Subjective evaluation of language usability.

The effectiveness of TPM is significantly higher than that of XSLT ($t(12) = 3.07$, $p < 0.05$). Marginally significant, this relationship maintains for the other dimensions of usability. In other words, the perceived usability of the TPM is better in terms of learning, efficiency, and satisfaction, and significantly better with respect to system effectiveness, as compared with the usability of XSLT. The overall perceived usability is significantly higher for the TPM system in comparison with the XSLT system ($t(12) = 2.60$, $p < 0.05$).

Objective Evaluation: Software Metrics

An objective evaluation of TPM in comparison with XSLT was done using software complexity measurement techniques. Below are the solutions for the 'Identity' transformation task (passing any input document to the output without changes).

XSLT code:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" id="null"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<xsl:copy-of select="/"/>
</xsl:template>
</xsl:stylesheet>
```

TPM code:

```
main(in)in{}
```

This is an example of the programs whose metrics were taken for this objective evaluation study and will be discussed below. The lines of code and character count metrics were the easiest ones to compute. For the character count metric, each end of line was counted as one character and all indentations were removed. The difficulty with the token count metric was in deciding what counts as a token. While in the case of the TPM the definition of a token could be taken directly from the language definition, the XML-based XSLT was not so straightforward: counting raw XML tokens would not be an accurate representation of the XSLT program complexity. For a more accurate measurement, the following rules were used:

- An opening tag counts as 1 token plus attributes.
- A closing tag counts as 1 token.
- A stretch of character data between two tags counts as 1 token.
- Each attribute of an opening tag counts as at least 2 additional tokens:
  - 1 token for the attribute name;
  - if the value of the attribute is an XPath [13] expression, the number of XPath tokens in it is used; otherwise, the value counts as 1 token.
- The <?xml version="1.0"?> preamble and the opening and closing tags of the xsl:stylesheet element are not counted.

Note that the 'character mapping' and 'country code mapping' programs contain repeating fragments, one fragment per table entry. During token counting, only one such fragment contributed to the program token count, the rationale being that the effort required for writing the second and each of the subsequent fragments is no longer proportional to the number of tokens in the fragment.

The metrics are presented in Table 2. With respect to the number of lines of code, XSLT programs are significantly larger than the equivalent TPM programs ($t(5) = 3.65$, $p < 0.05$); generally twice as large. In the given set of problems, the ratio varies from 1.36 to 7. Four tasks out of six have the XSLT to TPM line count ratio greater than 2.

Task name	Lines of code		Char. count		Token count	
	XSLT	TPM	XSLT	TPM	XSLT	TPM
Tag extraction	12	5	286	93	20	33
Identity transf.	7	1	197	13	8	7
Name splitting 1	30	11	801	213	81	73
Name splitting 2	41	18	1090	341	124	117
Char. mapping	24	18	1090	341	124	117
Country code map.	15	11	523	249	25	38
Total	129	60	3612	1117	321	327

Table 2. Metrics of XSLT and TPM evaluation programs.

In the character count metric, XSLT programs are also significantly larger ($t(5) = 4.27$, $p < 0.01$): about three times larger than the corresponding TPM solutions. The overall ratio is 3.2, five tasks out of six having the ratio above 3 and the remaining task showing 2.1. This also indicates that XSLT program lines are about 1.5 times longer on average than those of the TPM.

Finally, token counts do not show significant differences between XSLT and TPM. Four tasks are 'longer' in the XSLT version, while the other two contain more tokens when implemented in TPM. The overall token counts are almost the same.

In addition, it should be noted that XSLT-based solutions make extensive use of the built-in XPath function library for string processing. The functions employed include *contains*, *substring-before*, *substring-after*, *starts-with*, *substring*, and a rather complex *normalize-space*. At the same time, owing to the sufficient expressiveness of its core language, TPM has no built-in function library. If a similar utility library was provided to the TPM programmer, it could further contribute to the reduction of TPM program sizes, in particular in the number of tokens used.

Because a utility library is domain-specific and is not part of the data transformation architecture, it is beyond the purpose of TPM language. The above results show that even without such a library TPM programs are on par with or significantly less complex than equivalent XSLT programs, depending on the metric used.

CONCLUSION

TPM underwent subjective and objective comparative usability evaluation alongside XSLT – a widely used system for tree-structured data transformation. This study was based on a set of six sample data transformation tasks. The subjective evaluation analysed the questionnaires filled out by the participants following their hands-on experience with both systems in solving sample tasks. The objective evaluation measured three software metrics of the efficient solutions of the sample tasks.

This evaluation study comes to validate not only the implementation offered by the TPM, but moreover, the architecture and the language behind it. The usability of TPM is perceived by study participants to be generally higher than that of XSLT, and in certain aspects significantly better. The significantly higher perceived usability is in particular due to the TPM's better language consistency, design, and efficiency. These language-related traits follow directly from the properties of the proposed architecture [10,11]. Note however that these results should be taken with caution, as only 6 participants returned completed questionnaires for both systems, and 2 more returned one questionnaire each.

Software metrics also confirm higher usability of the TPM language. Therefore, both subjective and objective evaluation results indicate that the architecture behind TPM allows it to significantly surpass XSLT in terms of usability.

The usability study described here has three major benefits. Firstly, it suggests that the limitations of XML and XSLT can be effectively addressed. Secondly, it shows that the metrics for software usability can be successfully employed to compare data transformation languages. Thirdly, the current practice of using XSLT beyond the area it was intended for comes at a cost in terms of reduced usability. This justifies the efforts put into developing specialised languages for metadata processing, such as TPM which are both easy to learn and use.

ACKNOWLEDGEMENTS

The support of Enterprise Ireland for the ADSA project and Science Foundation Ireland for funding under the NTSRC development grant is gratefully acknowledged.

REFERENCES

1. Andersson, T. *A survey on software quality metrics*. Abo Akademi Univ. Unpublished manuscript, (1990).
2. Basili, V.R. Qualitative software complexity models: A summary. In *Tutorial on Models and Methods for Software Management and Engineering*. IEEE, CA, (1980).
3. Beazley, D.M. and van Rossum, G. *Python Essential Reference*. Que, (2001).
4. Brooke, J. SUS: A 'quick and dirty' usability scale. In *Usability evaluation in industry*, pages 189-194. Taylor & Francis, London, (1996).
5. Dushay, N. and Hillmann, D. Analyzing metadata for effective use and re-use. *NSDL Meeting*, WA, (2003).
6. Fraser, B. and Gluck, M. Usability of geospatial metadata or space-time matters. *Bulletin of the American Society for Information Science*, 25(6): 24-28. (1999).
7. Kearney, J.K., Sedlmeyer, R.L., Thompson, W.B., Gray, M.A. and Adler, M.A. Software complexity measurement. *Communications of the ACM*, 29(11):1044-1050, (1986).
8. Nielsen J. *Designing Web Usability: the practice of simplicity*. Indianapolis, New Riders Publishing, (1999).
9. Perlman, G. Web-based user interface evaluation with questionnaires. http://www.acm.org/~perlman/question.html.
10. Schmidt, N. A Common Architecture for Manipulating Tree-Structured Data. PhD thesis. University College Dublin, Ireland, (2003).
11. Schmidt, N. and Patel, A. Rule-driven processing of tree-structured data using pointed trees. *Computer Standards and Interfaces*, 25(5):463-475, (2003).
12. Venners, B. Programming at Python speed: A conversation with Guido van Rossum, part III. Published online by Artima Software, http://www.artima.com/intv/speed.html.
13. World Wide Web Consortium. *XML Path Language (XPath)* 1.0, W3C Recommendation, (1999).
14. World Wide Web Consortium. *XSL Transformations (XSLT)* Version 1.0, W3C Recommendation, (1999).