Synthesis of System Test Cases from UML Models

M Sarma\(^a\), R Mall\(^a\)

\(^a\)Department of Computer Science and Engineering, 
Indian Institute of Technology Kharagpur, West Bengal - 721302, India

Many modern systems are extremely complex and state-based, making it necessary to ensure satisfactory state coverage during system testing. However, achieving reasonable coverage of system states during testing is a non-trivial problem, since the number of system states for practical systems is usually very large, and system developers therefore often do not construct a system state model. To overcome this, we have used UML use case, sequence and class level statechart models to generate a set of sequences of scenarios that can achieve adequate coverage of system states.

1. INTRODUCTION

The aim of system testing is to ensure that a fully developed and integrated system is error free. System testing is often considered to be the most complex and intricate among all types of testing. This situation can be attributed to the fact that system testing involves testing a fully integrated system that is often large, complex, and state-based. For such system, system testing often overwhelms manual test case design efforts. Therefore, automatic design of system test cases is assuming crucial importance. There are essentially two main approaches to automatic design of system test cases. One approach attempts to design system test cases from requirements specification and the other from code. Test case design solely based on requirements specification completely ignores system implementation aspects. On the other hand, test case design from program code is cumbersome and difficult to automate. To overcome these shortcomings, presently grey box approach is being advocated [1]. In the grey box approach, system test cases are designed from design documents.

In recent years, Unified Modeling Language (UML) [2] has emerged as the de facto standard for modeling software systems and has received significant attention from researchers as well as practitioners. The importance of UML models in designing test cases has been well recognized [3-8]. UML diagrams can be used to capture different views of a system. The different views that can be modeled using UML are: users’ structural, behavioral, implementation and environmental views [9]. The UML structural models have traditionally been used for testing object-oriented programs at unit and integration level [10]. The behavioral view, on the other hand, has been recognized to be particularly useful to test a system at cluster and system levels [10].

A number of approaches for generating test cases from UML models have been proposed. Several researchers [4,5,7,11,12] have pointed out that use case diagrams are a set of important artifacts for the purpose of system test case design. Frohlich and Link [12] presented a method to generate test cases from textual description of use cases. They first transform use cases into a state machine representation in which the states are abstractions representing the interval between two successive messages sent to the system by a user. In their approach, the test cases are derived in a formal and partly automatic way and requires certain manual annotations of the UML models. Another approach to generating system level test cases based on use case models and state diagrams has been proposed in [7]. This is a black box approach which is based on the assumption that different user inputs to the sys-
Table 3
Memory comparison.

<table>
<thead>
<tr>
<th>Data structure</th>
<th>Sigmod dataset</th>
<th>XML-movies dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meta-Tree</td>
<td>5,08,528</td>
<td>37,424</td>
</tr>
<tr>
<td>Prefix string</td>
<td>6,32,292</td>
<td>1,61,988</td>
</tr>
<tr>
<td>Adjacency List</td>
<td>6,16,472</td>
<td>1,58,808</td>
</tr>
<tr>
<td>Adjacency Matrix</td>
<td>5938243K</td>
<td>394101K</td>
</tr>
</tbody>
</table>

There are different ways of defining the support threshold. We implemented the algorithm for finding both the root occurrence and the document occurrence. Root occurrence considers the number of occurrences under a single document, whereas the document occurrence counts the number of occurrences as the number of documents that the particular pattern exists in (not considering how many times it occurs within a document).

We considered the XML documents for evaluation purpose. The XML documents are converted into tree database which serves as the input for our algorithm. We have developed this step in Perl. Experiments were conducted using real datasets. Two popular sets among those are SIGMOD records from UW repository [17] and IMDB (Internet Movie DataBase) [18].

Table 3 shows the memory comparison which gives the space complexity comparison in the Meta-Tree with respect to other data structures: prefix string representation, Adjacency list, and Adjacency matrix representations.

For both the real datasets SIGMOD and IMDB records we find the root occurrence of the frequent patterns. Graph in Figure 3a shows the performance comparison of two algorithms Efficient XMLMiner and Unot (Asai Oct. 2003) with SIGMOD dataset. The performance graph is plotted for the time taken for execution by varying the minimum support value from 0% to 100%. Graph in Figure 3b shows the performance graph obtained for IMDB dataset (IMDB). We observed that our approach EfficientXMLMiner took lesser execution time for all supports given as input for both the real dataset.

We tested the scalability of the EfficientXMLMiner and sensitivity of the run time to the size of the database as well as to the minimum support value that considers document occurrence of the frequent patterns. The scalability is tested using both SIGMOD and xml-movie datasets. Graph from Figure 4a shows the scalability graph \( \text{minsupp} vs. \text{running time} \) with respect to number of documents and Graph from Figure 4b is the scalability graph, dataset size Vs. running time, with respect to the minimum support value for SIGMOD records. The scalability plot that shows the scalability with respect to the number of documents for XML-movie dataset is presented in Figure 4c. These scalability graphs show how stable the algorithm is, if we vary the number of documents (size of dataset) or the minimum support value.

From these graphs, we notice that the performance of EfficientXMLMiner is more stable. As the minimum support value increases, the number of frequent induced substructures decreases. The time required for mining increases with increasing number of frequent induced substructures. So, the execution time decreases with increasing minimum support value. The execution time increases with the increasing number of documents, as it requires more time for processing. The graphs clearly exhibit the above stated inferences. From the experiments we conducted, we can conclude that the EfficientXMLMiner is an efficient way for Frequent induced subtree (or frequent induced substructure) mining for Labeled Rooted Unordered trees.

6. Conclusion and Future work

In this paper, we have looked at the issue of mining frequent induced substructures from
Meta-Tree based Mining of XML Documents for Frequent Induced Subtrees

Figure 3. Performance graph in comparison with Unot using real datasets SIGMOD and IMDB.

Figure 4. Scalability graph for Efficient XMLMiner with respect to Dataset size and minimum support value.
Santhi Thilagam et al., database of XML documents which are modeled as labeled rooted unordered trees. In the proposed approach, we reduce the space as well as the database scan complexities by compressing the huge tree database into compact data structure. The proposed algorithm mines frequent induced subtrees without generating any candidate subtrees which is the main bottleneck in the Apriori method. Overall, the mining procedure takes a single database scan. One way to measure the data mining algorithms is in terms of its speed. The reasoning is that the faster the algorithm runs, the larger the data set to which it can be applied. The larger the data set is, the better the accuracy of the predictive model will be. All the existing algorithms like Unot [15] are based on Apriori principle which is time consuming. So, considering this speed as the measurement, the proposed EfficientXMLMiner algorithm performs better than the other existing algorithms for mining frequent induced subtrees. The time and space complexity analysis and the experimental results show that our algorithm is more efficient than the existing algorithms and is also scalable for varying data size and minimum support values.

We can extend our algorithm to employ user-defined constraints and the use of condensed representations. This leads to constraint-based mining of maximal patterns and closed patterns [19] for trees. This algorithm can be applied to other real world semi structured databases that can be modeled as tree structure such as profile analysis of Web directory and motif discovery in chemical compounds. The analysis and the extraction of the knowledge on the Web using tree and graph mining combined with text mining are the most challenging problems.

REFERENCES

5. J. Han, J Pei, and Y Yin. Mining Frequent Patterns without Candidate Generation, In Proceedings of ACM-SIGMOD Internet Conference Management of Data (SIGMOD-00), Dallas, TX, pages 1-12, 2000.
12. Y. Chi, S. Nijssen, R. Muntz, J. NKok. Frequent Subtree Mining - An overview, Funda-
mental Informatica Special Issue on Graph and Tree Mining, IOS Press, pages 1001-1037, 2005.


17. SIGMOD XML Data Sets (SIGMOD): http://www.acm.org/sigmod/record

18. The Internet Movie Database (IMDB) http://us.imdb.com/

19. Y. Chi, Y Yang, Y Xia, R. R Muntz. CMTreeMiner: Mining Both Closed and Maximal Frequent Subtrees, Pacific Asia Conference on Knowledge Discovery and Data Mining (PAKDD’04), May 2004.

P Santhi Thilagam received her Bachelor Degree in Computer Science and Engineering in 1991 from Thiagarajar College of Engineering, Madurai, India and Master Degree in Computer Science and Engineering in 2000 from Anna University, Chennai, India. Since 2004, she has been pursuing the Doctoral degree and working as a Lecturer in the Dept of Computer Engineering, National Institute of Technology Karnataka, Surathkal, India. Her research interests are mainly in the areas of database systems and data mining.

Ananthanarayana V S received his Bachelor Degree in Computer Science and Engineering in 1990 from Karnataka University, India, his Master Degree in Computer Science and Engineering in 1995 and his Doctoral Degree in 2001 from Indian Institute of Science, Bangalore, India and Post Doctoral Fellow in 2005 from Memorial University of New Found Land, Canada. He is currently the Professor and Head of Department of Information Technology, National Institute of Technology Karnataka, Surathkal, India. His research interests are in the areas of database systems, data mining, distributed computing and web services.