A Sequential Pattern Mining Approach to Extract Information from BIM Design Log Files

Saman Yarmohammadi[a], Reza Pourabolghasem[b], Arezoo Shirazi[a], Baabak Ashuri[c]

[a] Ph.D. Student, Economics of the Sustainable Built Environment (ESBE) Lab, School of Building Construction, Georgia Institute of Technology, USA
[b] Ph.D., School of Electrical & Computer Engineering, Georgia Institute of Technology, USA
[c] Associate Professor, Economics of the Sustainable Built Environment (ESBE) Lab, School of Building Construction, Georgia Institute of Technology, USA
E-mail: Saman.yar@gatech.edu, pourabolghasem@gatech.edu, Arezoo.shirazi@gatech.edu, Baabak.ashuri@coa.gatech.edu

Abstract –

Building Information Modeling (BIM) and its implementation in Architecture, Engineering and Construction (AEC) industry is rapidly growing all over the US. The growing utilization of BIM application has resulted in accumulation of tremendous volumes of computer-generated design data. Such vast datasets provide practitioners with an opportunity to extract valuable information regarding the process of design. Design log files, in particular, are rich data sources that contain model development data automatically recorded throughout the design process. However, the existing studies are mostly restricted to mining structured data and hence, lack the capabilities to handle unstructured temporal data sources such as design log files. The main objective of this paper extract implicit process information from design log data by implementing a tailored sequential pattern mining approach. For the purpose of this research, we examined the feasibility of utilizing Revit journal log files as a non-intrusive data collection mechanism to capture modelers' interactions with the software and detect common command execution structures during model development sessions. To this end, user-model interaction data such as modeler characteristics, command type, and command time were extracted from the data sample's journal files using a text file parser. After careful data cleaning, the final set of temporal data were transformed into sets of multiple character strings. We used an efficient implementation of Generalized Suffix Trees (GST) data structure to identify common command execution sequences among several modelers. The results of the study identified several shared command sequences among five modeler. This study proposes a novel pattern mining methodology to extract useful information from time-stamped design log data and enables project managers to obtain valuable insight into design development processes.

Keywords –

Building Information Modeling; Data Mining; Sequential Pattern Mining; BIM Log Files

1 Introduction

Design log data in general and Autodesk Revit© journal files, in particular, are unstructured text files that BIM tools create when modelers use the application. These files capture all modeling activities that occur during a design session, as well as system information, such as memory performance and operating system [1]. Revit journal files are largely used to diagnose and troubleshoot technical problems with the software. In this research, however, we examine the feasibility of utilizing these log files as a non-intrusive data capturing mechanism for documenting modeller-software interactions and recording model development events. The information extracted from Revit journal files will be utilized to examine our research hypothesis that there are common command execution patterns among BIM modelers who work on similar projects. In the context of this research, a journal log file is regarded as a database of ordered modeling events (commands) recorded with a concrete notion of time. A frequent pattern is an ordered set of individual commands that occurs more than a threshold number of times (i.e., minimum support) in the original sequence database. The following questions are of particular interest in this research: What types of commands sequences do modelers execute frequently? What structures are formed from various commands at each stage of modeling and how? What command pattern sequences are common among different modelers? Is there a resemblance between the modeling behaviour of BIM users who work on similar types of projects?

In recent years, there has been an increasing amount

of studies on the topic of mining information from BIM-generated documents. In particular, a considerable literature has accumulated around the topic of extracting patterns from geometric and semantic information stored in virtual building models. For instance, Hu et al. developed a multivariate linear regression model to predict man-hour quantity for steel fabrication projects in the planning phase [2]. The utilized data was extracted from historical Tekla BIM models. In another study, Abdelmohsen et al. developed a cost analysis and reporting system that utilizes elements' dimensions extracted from BIM models [3]. The researchers used Industry Foundation Classes (IFC) formats and Solibri Model Checker (SMC) to obtain quantity take off data, and integrate BIM and cost models. The main emphasis of these research efforts was on utilizing data stored in BIM models to enhance non-design practices, such as production planning, cost estimation, and site layout analysis. However, these studies are limited to utilizing information stored as physical and functional characteristics of elements after the model is already created. As a result, little is known about the implicit information that can be obtained by studying BIM model development data throughout design processes.

Current applications of textual data mining techniques in AEC literature can effectively extract patterns from information stored in structured relational datasets or unstructured documents [4]. However, these approaches are restricted to data that lack time dimension [5]. Because of this shortcoming, these methods are inherently limited to analyze sequential design log data [6]. Soibelman and Kim outlined the steps necessary to apply data mining and Knowledge Discovery in Databases (KDD) as tools to extract novel patterns in design and construction fields [7]. In two consecutive studies, Caldas et al. [8] and Caldas and Soibelman [9] proposed text mining-based approaches to automatically classify unstructured construction documents. The presented approaches were implemented to classify several product description documents into categories defined based on CSI MasterFormat classification hierarchy. Another example of implementing data mining algorithms to analyze unstructured textual data is construction cost overrun prediction by William and Gong [10]. The authors used a stacking ensemble model of several classifiers to forecast the level of cost overruns. These approaches present valuable insights on how to overcome several challenges of handling unstructured textual information such as dealing with the high dimensionality of data, removing incorrect and irrelevant data points, and identifying features to represent the data in an analysis-friendly format. However, the existing methods fail to address how to incorporate the chronological dependencies of temporal data and hence, are not suitable to analyze design log files.

The primary objective of this study is to extract implicit process information from design log data by implementing a tailored sequential pattern mining approach. For the purpose of this research, we examined the feasibility of utilizing Revit journal log files as a non-intrusive data collection mechanism to capture modellers' interactions with the software and detect common command execution structures during model development sessions. Throughout this paper, several steps necessary to collect, prepare, and analyze data to extract frequent command patterns are elaborated. A major international design firm provided the data for this exploratory study. This study contributes to the state of practice by providing new insights into what additional design process information can be retrieved from Revit journal files. The novel method created in this research contributes to the body of knowledge by incorporating chronological dependencies of textual records into the existing pattern matching models. Section 2 describes the methodology used in this paper to analyze time-stamped design log data. We outline the implementation details of the proposed approach in section 3. In section 4, we provide results obtained from conducting an experiment on real design project data. Section 5 is conclusions and future research directions.

2 Research Approach

"Figure 1" presents the flowchart of the process to extract necessary information from Revit journal files and identify common frequent command execution sequences. This process consists of three major parts. First, a large number of Revit journal files that belong to a design project are collected. These text files are parsed to extract and store necessary data items. Second, we transform the obtained data to construct long strings of characters and generate input vectors. Finally, we construct Generalized Suffix Tree data structures for each user. A Depth-First Search (DFS) is later conducted on the GST structures to find common command sequences for each and among the five users.

2.1 Data Collection

We utilized modeling log data provided by an international design firm. The research team had access to data from 10 healthcare project designed in 2013 and 2014. The provided database consists of over 5,000 Revit journal files that were later parsed to extract useful entries, amounting to over 10 Gigabytes of structured data. In addition to the modeling events, journal files contain information about the user and the project to which the model belongs. The steps that were followed to extract required information from the journal files are outlined in section 3.

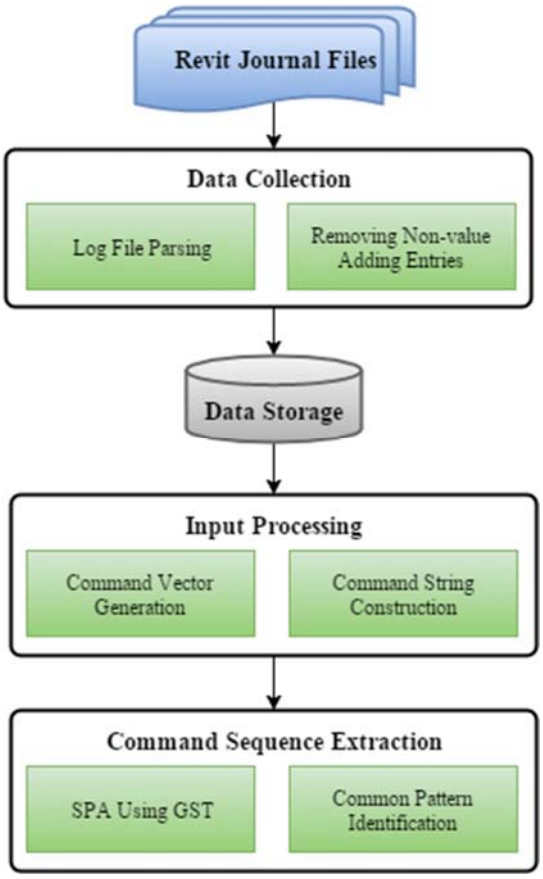

Figure 1. Flowchart of Common Command Sequence Extraction from Revit Journal Files

2.2 Input Processing

The records of five modelers that each had over 100,000 log entries are used to conduct further analysis. We treated the log entries for each user as a long string. The commands that each user issued were modelled as characters. The final string for each user was constructed by putting these characters in the original order they were recorded. Input vectors of modelers were then generated by joining the constructed command strings in chronological order.

2.3 Command Sequence Extraction

Log files (or transaction logs) have for long been studied in the data mining community. Log files can be generated in different applications, such as retail transactions data, web access logs, etc. Agrawal and Srikant introduced Generalized Sequential Pattern (GSP) algorithm to mine shopping patterns [11]. The authors studied a large database of customer transaction data, where each transaction consisted of customer-id, transaction time, and items bought in the transactions. Utilization of downward closure property (i.e., Apriori) provides GSP with the advantage of reducing search space compared to an exhaustive search approach. However, the time and memory performance of GSP is relatively low since a huge number of candidates must be generated and stored in each repetition for evaluation [12]. PrefixSpan algorithm overcomes this issue by taking advantage of pattern-growth methodology [13]. This algorithm greatly reduces the candidate sequence generation efforts by following a divide-and-conquer approach called prefix-based projection. The approaches proposed in these studies can extract valuable information from sequential datasets. However, general Sequential Pattern Mining (SPM) methodologies do not preserve the exact order of elements in a sequence. As a result, some of the elements in extracted sequences may not necessarily be consecutive in the original string of transactions.

For the purpose of this study, a special GST-based string processing algorithm was used. This method is particularly useful as it maintains the order in which the executed commands are recorded. Xiao and Dunham first proposed applying GST data structure to mine web access log data [14]. The authors analyzed clickstream data generated based on the access made by internet users to find frequent web page traversal patterns. The technique proposed in this study achieved a high level of adaptability to large databases through dynamic compressions and effective pruning. Guerbas et al. introduced an improved version of GST algorithm with an optimized data structure to extract page visit patterns of internet users [15]. The primary objective of the authors was to improve the search experience of users by predicting what pages they intend to view next. To this end, the research team utilized GST algorithm to find common web navigational patterns among users with similar interests. The methodology utilized in our study is a modified version of Guerbas et al.'s algorithm tailored for mining journal log files. The algorithm's implementation details are outlined in the following section.

3 Implementation of the Proposed Design Log Files Mining Approach

In this section, we present a detailed explanation of our proposed approach to process and analyze information embedded in Revit journal files using

Table 1 Examples of data items as recorded in Revit journal files

Column name	Example
Date and Time	'E 19-Apr-2014 22:22:53.175; 0:< Jrn.Command "Internal" , "Save the active project with a new name , ID_ ' 1:< System (MB) [Available / Total] [Revit Memory Usage (MB)]
Project Path	'H 19-Apr-2014 22:23:05.063; 2:< Jrn.Data "File Name" _ , "IDOK", "..\studentunion\Atlanta\gatech.rvt"
General Command	Jrn.Command "AccelKey" , "Cancel the current operation , ID_CANCEL_EDI 'C 19-Apr-2014 22:22:48.142; 0:< idle0_doc ' 0:< Setup Tool transaction group (wall)
Specific Command	Jrn.Command "AccelKey" , "Cancel the current operation , ID_CANCEL_EDITOR" 'C 19-Apr-2014 22:22:48.142; 0:< idle0_doc ' 0:< Setup Tool transaction group (wall)
Revit Version	' 5:< File was saved in Autodesk Revit 2014 (Build: 20130123 1540(x64)) ' 6:< ::8:: Delta VM: Avail -20 -> 8386966 MB, Used +16 -> 334 MB, Peak +8 ' 0.000370 7:<<<ET::repairModificationDatesOutOfSequence
View Type	Jrn.Directive "WindowSize" _ , "[Project1]", "Floor Plan: Level 1" _ , 1322, 625

GST string mining algorithm. To begin this process, we need to parse journal files and extract necessary data items. This step is particularly challenging as there is no documentation available on public domain that specifies how and where different data components are recorded. The data items extracted for the purpose of this study are modeller name, date of the modeling session, project name, command execution time, general and specific command description, view type, and the Revit version utilized by the modeller. To find this information in journal files, we use regular expression matching operations. “Table 1” shows examples of each data item as they appear in journal files. We manually search the files to identify the local format using which each data instance (e.g., project name, command, and view type) is recorded. Once patterns are identified, we use a text processor to extract and store information in a Comma-Separated Values (CSV) file as shown in “Table 2”. Non-value adding commands, such as “cancel the current operation”, are removed from stored entries to improve the quality of obtained command patterns.

Testing whether a sequence of commands occurs frequently in a database needs to be performed in an efficient manner. That is why we opted for GSTs. Suffix trees are very efficient data structures and often provide linear time solutions to challenging string problems [16]. “Figure 2” shows the algorithm utilized to search for frequent command execution sequences where the original order of data is conserved. First, we assign unique characters to each specific command name to generate the necessary input strings. We generate long strings by joining these characters in accordance to the original order in which corresponding commands are recorded in each modeling session. Once these strings are constructed, we use suffix trees to represent all suffixes of the string set. In our

Table 2 Sample of structured processed data

Modeller Name	Time	Project Name	General Command	Specific Command	Revit Version	View Type
modeller	10:30:07	StdUnion	"Internal"	"Show or hide recent files	2014	Floor Plan
modeller	10:33:50	StdUnion	"StartupPage"	"Open an existing project	2014	Floor Plan
modeller	10:36:51	StdUnion	"Internal"	"Print the active window	2014	sheet
modeller	10:45:25	StdUnion	"Internal"	"Activate this viewport	2014	sheet
modeller	10:48:04	StdUnion	"Internal"	"Modify View Templates	2014	sheet
modeller	10:48:07	StdUnion	"Internal"	"Manage Links	2014	sheet
modeller	10:48:09	StdUnion	"KeyboardShortcut"	"Steering Wheels	2014	3D view

```
Input: a csv file containing recorded entries, minimum length of the common command patterns
l, minimum number of strings sharing the pattern m
Output: common command patterns
01- Construct input vectors for each by transforming command sequences
02- Build generalized suffix tree for all strings
03- For each leaf node f
04-     current node ← f
05-     While root is not reached
06-         p ← parent (current node)
07-         add string id of the leaf node f to the node p if this id has not been added to it yet
08-         if the number of string ids added to p>=m and ||path(p, root)||>l then
09-             return path(p, root)
10-         end if
11-         current node ← p
12-     end While
13- end For
```

Figure 2 Using generalized suffix trees to detect frequent command execution sequences (Adapted from Guerbas et al. [15])

implementation, the DFS step generates an ordered list of leaf nodes of the suffix tree. The leaves corresponding to each internal node are, therefore, a consecutive sub-list of this ordered list of leaves. The DFS saves the start and end position of the leaves (based on their DFS order) for each pattern (i.e., internal node) in a helper hash table. The subtraction of the end position and the start position will give the number of repetitions of the specific pattern. Additionally, this way we can access all the instances of that pattern in the original string by having access to the leaf nodes and their corresponding suffix index. This technique enables us not only to identify patterns, but also calculate their frequency. These substrings, then, are filtered based on simple heuristics (e.g., minimum length of the substring, minimum frequency, etc.). At the end of this step, we have a limited number of most frequent substring of commands that is common between all the users. We call these substrings as primitives. We repeat this process for all users to extract their frequent command patterns. Finally, the retrieved patterns are compared against each other to identify the ones that are common among different users. The results obtained from implementing this approach on the provided Revit journal files are presented and discussed in the following section.

Table 3 Statistics of the dataset used

Data set	Period	Number of entries
Modeler 1	2013	126,815
Modeler 2	2014	122,813
Modeler 3	2014	114,290
Modeler 4	2013	112,082
Modeler 5	2014	106,887
All records	2013-2014	5,568,243

4 Experiments and Results

This section describes the results of experiments conducted and discusses the patterns obtained. We present some statistics about the dataset used. We comment on the patterns discovery results obtained using GST and how they support our initial hypothesis regarding the existence of common command execution patterns among BIM modelers.

A dataset containing 5,000 Revit journal files was used to implement our approach. We started the analysis by processing these files using a text parser. We also filtered out all noisy data (e.g., “cancel the current operation”, “delete”) and entries related to errors. The processed data was stored in .csv format amounting to over 5.5 million records. Data from five architects that model interior systems of healthcare projects were selected to conduct analysis. “Table 3” provides some statistics about the utilized data.

Table 4 Top three most executed individual commands

	Frequency		
Modeler	Move selected objects or their copies	Align references	Create a line
Modeler 1	19.30%	11.92%	7.89%
Modeler 2	15.27%	13.56%	14.23%
Modeler 3	18.19%	6.21%	11.07%
Modeler 4	15.57%	15.98%	13.07%
Modeler 5	14.31%	6.94%	10.00%

Prior to using GST, we conducted some preliminary analysis to identify the most frequently executed individual commands. “Table 4” shows the top three commands for each modeller. These commands were

consistently found to be the most frequent events for all users.

In the next step, we constructed a GST data structure for each modeller's command string to identify shared sequences. Several arbitrary minimum frequency threshold values were tested, among which we selected 250 and 500. The minimum length of extracted common command patterns was also set to three. Interesting patterns obtained for the two threshold values are presented in "Table 5".

The primitives extracted for minimum threshold of 250 are longer and represent meaningful modeling activities. Pattern 1 seems to correspond to commands executed to create and extend multiple lines. In this case, modelers have hidden a number of objects to gain easier access to the elements they want to modify. The second pattern appears to show cases where modelers make copies of different elements and visualize their dimensions. The third pattern captures commands used to make copies of a specific object in the model and modify them.

The obtained patterns became shorter when we increased the minimum frequency threshold. This observation was expected since longer sequences tend to match less frequently. The first pattern corresponds to cases where modellers navigate through different viewpoints to select and copy certain objects. The third pattern also captures command sequences used to change visibility of different layers. Contrary to these two patterns, it is not clear what specific activity the second command sequence represents. We saw more examples of such repetitive sequences for both thresholds. Presence of such patterns may be because of noisy input data or consecutive execution of similar commands by modelers.

5 Conclusion and Future Work

In this paper, we proposed an efficient approach to extract modeling development information embedded in design log files produced by Autodesk Revit. We have reviewed all steps of this process and made an effort to make a contribution at each step. The first step in Revit journal mining consists of different phases. The first phase is to identify the format in which different information items are stored. Our code accepts the raw journal files and produces structured csv files as shown in "Table 2". We also clean the obtained data by removing non-value adding entries, such as cancel and error messages. The conclusion we derive from our investigation at this level is that using the suggested approach will help in processing unstructured journal log files and producing good quality input data for mining algorithm.

In the second step, we use GST data structures to find common command sequences among BIM users. First, we transform command sequences into character-based input vectors. Then, the transformed data is utilized to construct GST. Frequent command patterns are identified by conducting DFS on the trees ("Figure 2"). Extracted patterns for different users are compared against each other to identify shared sequences. The conclusion at this step is that GST-based string mining approach is an efficient method to extract common command patterns among several modelers.

Table 5 Common command sequences extracted using GST

	Pattern 1	Pattern 2	Pattern 3
Extracted Pattern 250	1. Select objects to modify 2. Hide selected elements 3. Create a straight detail line or a detail arc 4. Rotate selected object(s) 5. Trim/Extend two lines or walls to make a corner	1. Copy the selection and put it on the Clipboard 2. Move copies of selected objects 3. Create aligned Dimensions	1. Select objects to modify 2. Create an object similar to selected object 3. Move selected objects or their copies 4. Align references 5. Finish Sketch
Extracted Pattern 500	1. Activate this viewport 2. Copy the selection and put it on the Clipboard 3. Deactivate the currently active viewport	1. Move selected objects or their copies 2. Move selected objects or their copies 3. Move selected objects or their copies	1. Deactivate the currently active viewport 2. Activate this viewport 3. Control visibility and appearance of objects (applied only in the active view)

Finally, we conducted experiments on Revit journal files provided by a design firm to verify our proposed approach. Our code successfully processed the raw log files and extracted common command patterns for five architects who model interior systems of healthcare projects. The obtained results also confirmed our initial hypothesis that there are frequent command execution sequences shared among BIM modelers who work on similar projects.

This study contributes to the state of knowledge by proposing a tailored string mining algorithm capable of extracting meaningful information from time-stamped design development data. We contribute to the state of practice by enabling design project managers to gain an unprecedented insight into the evolution of design. It is suggested that the possibility of using command patterns to characterize modelers is investigated in future studies. For instance, it might be a good idea to modify the proposed approach to calculate the average time it takes BIM users to executed different command patterns. Such empirical assessment of modellers' behaviour can potentially be used to provide customized training to improve designers' performance. Moreover, calculated average times can help design manager to improve project outcomes by choosing an optimal team configuration.

Acknowledgement

The authors take this opportunity to thank Professor Charles M. Eastman, who leads Digital Building Laboratory (DBL) at the Georgia Institute of Technology, for his support of this research project. In addition, we are grateful to Dr. John Haymaker, Mr. Dan Chasteen, and Mr. Josh Emig who kindly helped us throughout the progress of this study.

References

[1] Autodesk Revit. On-line: http://www.autodesk.com, Accessed 15/03/2016.

[2] Hu X. and Lu M. and AbouRizk S. BIM-based data mining approach to estimating job man-hour requirements in structural steel fabrication. In *Proceedings of the 2014 Winter Simulation Conference*, pages 3399–3410, IEEE Press, 2014.

[3] Abdelmohsen, S. and Lee J. and Eastman C. Automated cost analysis of concept design BIM models. *Designing Together: CAAD Futures*, 2011.

[4] Manyika J. and Chui M. and Brown B. and Bughin J. and Dobbs R. Big data: The next frontier for innovation, competition, and productivity, 2011.

[5] Baars H. and Kemper H. Management support with structured and unstructured data—an integrated business intelligence framework. *Information Systems Management*, 25(2): 132-148, 2008.

[6] Doan A. and Naughton J. and Ramakrishnan R. and Baid A. Information extraction challenges in managing unstructured data. *ACM SIGMOD Record*, 37(4): 14–20, 2009.

[7] Soibelman L. and Kim H. Data preparation process for construction knowledge generation through knowledge discovery in databases. *Journal of Computing in Civil Engineering*, 16(1): 39–48, 2002.

[8] Caldas C. and Soibelman L. and Han J. Automated classification of construction project documents. *Journal of Computing in Civil Engineering*, 16(4): 234–243, 2002.

[9] Caldas C. and Soibelman L. Automating hierarchical document classification for construction management information systems. *Automation in Construction*, 12(4): 395–406, 2003.

[10] Williams T. and Gong J. Predicting construction cost overruns using text mining, numerical data and ensemble classifiers. *Automation in Construction*, 43: 23–29, 2014.

[11] Agrawal R. and Srikant R. Mining sequential patterns. In *Data engineering 1995. Proceedings of the Eleventh International Conference on*, pages 3–14, IEEE, 1995.

[12] Verma M. and Mehta D. A comparative study of techniques in data mining. *International Journal of Emerging Technology and Advanced Engineering*, 4(4), 2014.

[13] Pei J. and Han J. and Mortazavi-Asl B. and Pinto H. and Chen Q. and Dayal U. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *icccn*, page 0215, IEEE, 2001.

[14] Xiao Y. and Dunham M. Efficient mining of traversal patterns. *Data & Knowledge Engineering*, 39(2): 191–214, 2001.

[15] Guerbas A. and Addam O. and Zaarour O. and Nagi M. Effective web log mining and online navigational pattern prediction. *Knowledge-Based Systems*, 49:50–62, 2013.

[16] Gog S. and Beller T. and Moffat A. and Petri M. From Theory to Practice: Plug and Play with Succinct Data Structures. In *sea*, pages 326–337, 2014.