Current architecture and implementation for UMLx

System Overview

The structure of UMLx is shown in Fig. 1, which comprises 7 modules to support the analyses. Specifically, Model File Parser module is responsible for parsing the input XML files into the directed graphs and associating nodes with the corresponding UML diagram elements. Element Consolidation module is to consolidate the elements that share the same name across different diagrams. Path Profiler module traverses the directed graphs to identify the paths that are the potential transactions. Pattern Tree Builder module creates a tree structure for the input patterns, which will be used later in pattern matching process. Pattern Matching module matches each of identified paths against the pattern tree and associates operational characteristics with the transaction when a path is matched and determined as a transaction. Identified transactions are input to Performance Evaluator module to be compared with the empirically understood transactions in order to calculate identification and categorization accuracy. Also the identified transactions are input to Transaction Analyzer module to perform calculations according to the defined metrics. The end users are able to review the results at different stages of the procedure. A layered architecture view also provided in Fig. 2.

System Implementation

For the purpose of standardizing our proposed approach to analysing UML diagrams and improving its efficiency, the proposed procedure and algorithms are implemented as a tool called UMLx, which is an abbreviation for UML analytics. This web-based UML diagrams analytic tool is implemented with Nodejs/MongoDB for the back-end logic and data storage, HTML/Javascript/CSS for the front end, and R scripts for statistical analyses. The Model File Parser module is currently implemented based on the format of the XML files exported from Enterprise Architect, to extend the use of the tool to other UML modeling tools, for example, Visual paradigm, an individual parser needs to be implemented and integrated to be able to analyse specific types of XML files.

Fig. 1. UMLx system architecture.

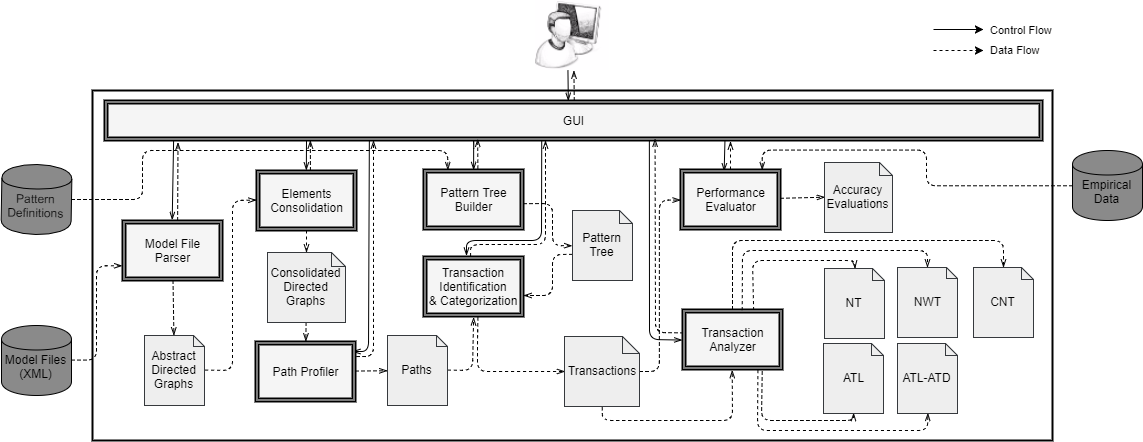
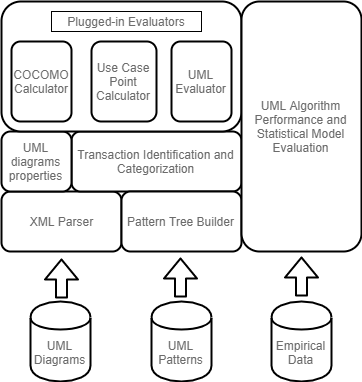
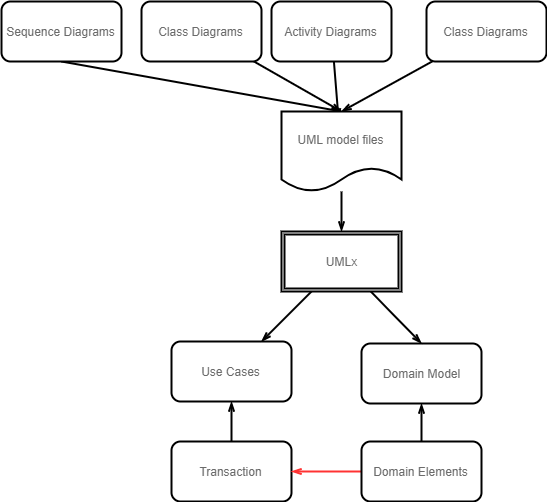


Fig. 2. Layered Architecture View



**P.S. Don’t share this document to other people.**

Fig. 3 Data Structure



Goals of this semester (the items tagged with 1-5 with its importance)

1. UI and API development:
   1. Restructure User Interface to make the data presented in more efficient way.(5)
   2. Add animations, dialogs, and pages to make UI more robust. (4)
   3. Improve the APIs, for example, check validity of the uploaded files and parameters, restructure the data base query functions for better efficiency, put comments on the apis for its readability. (3)
   4. Add the user management feature: including user registration/login, assign different resources (repos) for different users. (5)
2. UML analytics:
   1. Develop the UML diagrams parser for xml files exported from Visual Paradigm. (4)
   2. Reiterate the existing algorithms for better dealing with the corner cases. (4)
   3. Identify the relationships between the UML diagram elements: associating elements across different diagrams. (5)
   4. Write parsing algorithms for new diagrams, for example, activity diagrams. (5)
3. Model Training (this part may require domain knowledge of software engineering and software size estimation).
   1. Identify the most significant factors to predict effort, SLOC, etc. For this, you will need to understand meaning of different attributes of a project and their effect on effort. (3)
   2. Calibrate the parameters and the weights for the models. (5)
   3. Evaluate significance of the parameters, goodness of fit, and prediction accuracy. (4)
   4. Statistical measures to deal with limited data points. (3)
   5. Iterate data sources to collect data. For example, evaluate software engineering documents to collect relevant information. (5)