**UMLx System Tutorial:**

1. **Source code structure**

data/: stores the UML diagrams collected from the selected projects, there might be more to be added in the future. The files with surfix .xml are exported from Enterprise Architect for the UML diagrams.

db\_backup/: the backup of current database, which is based on MongoDB.

documents/: the documents about the system configuration, tutorials of system structure, or functionality explanations.

evaluators/: this folder contains high level analysis for the extracted information from UML diagrams. For example, to use the number of transactions to calculate Use Case Point. Each of the JS files is an Evaluator, which is used as a plugin to register in UMLEvaluator.js.

model\_drawers/: include the methods to visualize the extracted models, for example, user-system interaction model, transaction, etc.

model\_platforms/: this folder contains the low level operations to parse a specific format of the exported files for UML models. For example, ea/XMI2.1Parser.js parses the information from the xml files exported from Enterprise Architect.

node\_modules/: includes the installed nodejs plugins. We usually don’t touch the folder.

public/: stores the uploaded data (the exported UML files) from the web portal, also the outputs from different analysis.

Rscript/: includes the R scripts for statistical analysis for the extracted information from UML models.

Temp/: store temporary files.

Views/: include templates for web pages, written with the Jade plugin of Nodejs.

UMLEstimator.js: used to receive commands for model calibration.

UMLEvaluator.js: the abstraction for the evaluators. It currently registers the evaluators under the evaluator/ folder, for example, COCOMOCalculator.js, UMLModelEvaluator.js, and UseCaseCalculator.js.

UMLFileManager.js: the UML model file manager module, which manages uploaded model files.

UMLModelExtractor.js: start the procedure for extracting user-system interaction model from UML diagrams.

UMLModelInfoManagerMongoDB.js: includes the database management functions, for example, querying model or diagram information.

UMLxServices.js: includes the Restful APIs to request the analytical data, including the Url access entries for the backend functionality.

1. **Programming Language:**

Nodejs, Javascript, HTML, CSS, R scripts

1. **Dependencies:**

NodeJs (with modules: multer, admzip, express,path,jade)

MongoDB

1. **Platform:**

Windows or Mac OS or Linux

1. **Setup:**
2. install mongoDB. (<https://docs.mongodb.com/manual/installation/>)
3. create the folder C:\data\db, otherwise the db was not able to start
4. started MongoDB
5. restore the backup database, using: mongorestore <Project Dir>/db\_backup, or restore the individual collections, using: mongorestore –drop –db model\_info\_collection <Project Dir>/db\_backup/model\_info\_collection.
6. Install nodejs. (https://nodejs.org/en/download/)
7. Install dependencies using `npm install` from project dir <Project Dir>
8. Start mongodb:
   1. Navigate to the database folder of MongoDB. The default folder for MongoDB installation is : ~/Documents/db.
   2. Using command to start mongoDB : mongod –-dbpath .
9. Start web service:
   1. Navigate to your project root folder.
   2. Using command to start UMLxServices: node ./UMLxServices.js
10. Access: (<http://127.0.0.1:8081/>) to load the home page.
11. Data Setup:
    1. mongodump –out /data/backup.
    2. mongorestore –drop –db db\_name <project dir>/db\_backup/db\_folder\_name

Notes: sometimes you may mess up the database for some reason. You can delete the mongodb and restore the backup database. This tutorial is just the skeleton of the process of setting up the project. There might be some variations for different operation systems and environments, or some of the instructions may not be very accurate. In those cases, you may need to do some researches for those steps.