Carleton University

COMP4905 – Honours Project

Entity Relationship Diagram Design Tool

– A JavaFX application

Name: Gajaneethan Selvarajah (100867303)

Supervisor: Lou Nel

Date: 15.12.2017
Abstract

This report provides information about Entity Relationship Diagram Design Tool. The Entity Relationship Diagram Design Tool is used to design Entity Relation model and users can generate and save functional dependencies and database schema for an Entity Relationship model. Entity Relationship Diagram Design Tool is a useful application for users who are leaning Entity Relationship model and database schema; especially, students, who are taking COMP3005. Moreover, this project describes about what is the motivation for implementing the Entity Relationship Diagram Design Tool, features, design patterns, challenges that encountered during the implementation, improvements and features that can be added to the Entity Relationship Diagram Design Tool and how users can install the application.
Acknowledgment

I would like to thank Professor Lou Nel who provided guidance and feedback on this project as a supervisor. Online JavaFX tutorials helped me to understand about the basics of JavaFX as a beginner to JavaFX.
# Table of Contents

1. Introduction .................................................................................................................. 7

2. Motivation ....................................................................................................................... 9

3. Objectives ...................................................................................................................... 10

4. Features ........................................................................................................................ 11
   4.1 User Friendly Graphical User Interface ...................................................................... 11
   4.2 Validate Entity Relationship model design .............................................................. 13
   4.3 Print Entity Relationship model design ..................................................................... 13
   4.4 Generate Functional Dependencies ......................................................................... 13
   4.5 Generate Schema ...................................................................................................... 14

5. Design ............................................................................................................................ 15
   5.1 Graphical User Interface .......................................................................................... 15
   5.2 Model Classes .......................................................................................................... 18
   5.3 View Classes ............................................................................................................ 19
   5.4 Controller Classes ................................................................................................... 19
   5.5 Software Design Pattern .......................................................................................... 20
6. Challenges................................................................................................................. 21
7. Future Improvements................................................................................................. 22
8. How to add new feature............................................................................................. 23
9. Install Instructions ...................................................................................................... 24
10. Summary.................................................................................................................... 29
11. References.................................................................................................................. 31
12. Abbreviations............................................................................................................. 33
List of Figures

Figure 1. Entity Relationship Diagram Design Tool

Figure 2. Users should select an option

Figure 3. Users should set datatype for each attribute

Figure 4. Illustration of Model-View-Controller design pattern (Image source is [12])

Figure 5. Import Project in Eclipse

Figure 6. Import Existing Project into Workspace

Figure 7. Choose the “erddesigntool” project

Figure 8. Open the Java Perspective

Figure 9. Run the Entity Relationship Diagram Design Tool application
1. Introduction

Entity Relationship Diagram Design Tool is a JavaFX based application which allows users to design Entity Relationship (ER) model via a Graphical User Interface (GUI). This Entity Relationship Diagram Design Tool is a useful application for users who are leaning Entity Relationship model and database schema. Especially, students, who are taking COMP3005 (Introduction to Database Systems), can use this tool to design the Entity Relation model and generate functional dependencies and database schema.

FIGURE 2. Entity Relationship Diagram Design Tool
Entity Relationship model is graphical representation of entities and their relationship to each other\textsuperscript{[1]}. Also it is used to represent cardinalities between relationship and entity. Attributes can be added to entities and relationships which becomes a column in the table later in the schema generating process. A relationship can be represented into a new table if the cardinality is “N to N” or included into one entity if the cardinality is “1 to 1”, “1 to N” or “N to 1” between two entities.

JavaFX is a new graphical user interface framework intended to support desktop application and web application\textsuperscript{[2]}. Also it is used to create rich internet applications that can run on a large variety of devices. JavaFX supports many operating systems, such as, Windows Vista, Windows 7, Windows 8, Windows 10, MacOS and Linux. Moreover, JavaFX is intended to replace Swing as the standard graphical user interface library for Java SE and it makes use of FXML, which is a new XML based mark-up language created by Oracle Corporation, for defining User Interfaces (UIs).

The Entity Relationship Diagram Design Tool has GUI, which is built in JavaFX, where users can draw entities, attributes and relationships and connect them via straight lines. Also users can move the drawn entities, relationships and attributes on the GUI. Additionally, users can print the entity relationship model so they can have it as reference document and able to compare the schema with model if they needed later on.
2. **Motivation**

Entity Relationship model, functional dependencies and database schema are very new concepts for COMP3005 students as they start learning during the course. So building a tool, which allows them to draw and practice the Entity Relationship model, would be a useful to them and they can easily understand those concepts. Also as a user, the tool should be user friendly and able to handle errors. Moreover, the tool should be able to describe the errors in details so that student can understand and learn from mistakes.

In order to achieve the user friendly interface, JavaFX is one of the good choices. At the same time, I do not have any previous experience in JavaFX. So on the one hand, implementing a tool using JavaFX helps me to learn about JavaFX and on the other hand, students get a tool which is useful for their studies. Moreover, other than the graphical interface part, the Entity Relationship Diagram Design Tool logic in implemented using Java because Java is one of the most popular programming languages and I have gained a lot of experience in Java through academic courses and co-op work terms.

Learning JavaFX is a new asset that I gain via this project and it adds more value to my academic career. Furthermore, independently learning JavaFX from scratch is good experience and helps to improve my self-learning skill which is a valuable skill for my future career. For this project, I need to expand my database related skills apart from the learning in COMP3005 course materials that requires completing this project and it is another valuable skill I gain via this project.
3. **Objectives**

A GUI is a very important component of an application because if the GUI is not user friendly then the application would be less useful regardless of how many features it has. In other words, “A user interface is like a joke because if you have to explain it, it is not that good”\[^3\]. In order to enforce usefulness of the Entity Relationship Diagram Design Tool, the GUI needs to be user friendly and easy to use where users can able to find about how to use the tool without explaining it to them.

Users should be able to complete the Entity Relationship model without any problem and when errors occur, it has to be descriptive and understandable by users, even if the user is not a technical person. Moreover, when user enters into an error state, they should be able to return back to previous state so they can correct the error and continue further. At the same time, users should not be allowed to enter into next state if there are any errors in the current state; in other words, users should correct errors in the current state in order to process further.

Users should be able to validate the current state of the Entity Relationship model and generate and save functional dependencies as “.txt” file if there are no validation errors. Also users should have the ability to make decision on whether to make a new table or not for a relation when the cardinality between two entities is either “1 to 1”, “1 to N” or “N to 1”. The Entity Relationship Diagram Design Tool should have a delete feature so users can delete an entity, attribute or relation if they do not need it, at the same time, they should be able to undo the last deletion as well.
Users should be able to generate and save the database schema as ".sql" (Structured Query Language) file if there are no validation errors and before generating schema, users should have the ability to make decision on whether to make a new table or not for a relation when the cardinality between two entities is either “1 to 1”, “1 to N” or “N to 1”. Moreover, users should be able to print the current state of the Entity Relationship model so they can compare the schema with Entity Relationship model and use it as reference document.

4. **Features**

Entity Relationship Diagram Design Tool has the following features: entities, relations, and attributes can be placed in the drawing area by drag and drop, users can delete any selected component, users can undo the last deletion action, ER model design can be validated any state and error message will be displayed if any validation errors occur, print the current state of the ER model design, generate and save functional dependencies as “.txt” file, generate and save database schema as “.sql” file and users has the choice to make decision on whether to make a new table or not for a relation if the cardinality between two entities is either “1 to 1”, “1 to N” or “N to 1”.

4.1 **User friendly Graphical User Interface**

The GUI components can be placed in the drawing area by dragging and dropping on an entity, relation or attributes. The line component, which connects the “entity – relation”,

Gajaneethan Selvarajah
“entity – attribute” or “relation – attribute”, will be added automatically. By default, the lastly placed component is selected when user place the another component it will be automatically connected with lastly placed component. Also each component on the drawing area can be moved by dragging them and if the component is connected with another component then the line between those components will be automatically resized. The attribute component contains two check boxes two represents “key” and “Not Null” so users can choose either the attribute is a Key or Not Null or both.

When launching the Entity Relationship Diagram Design Tool, relation and attribute components are disabled. The attribute component will be enabled when an entity component is placed and the relation component will be enabled after the second entity component is placed. Moreover, the line, which connects the entity and relation, has two radio buttons (named as “1” and “N”) where users can choose the cardinality. The drawing area will grow dynamically if needed which means when users place a component in the edge of the drawing area then it will grow and a vertical or horizontal bar will appear if the drawing area grows more than size of the screen.

Users can delete any selected component at any state of the Entity Relationship Diagram Design Tool and the deletion is cascading. When users delete a relation then all attributes of the relation will be also deleted. Moreover, when users delete an entity then all attributes and connected relations will be also deleted. At the same time, users can undo the last deletion action if they need and all the deleted components will be restored in the same way that they were before the deletion. It will prevent users from losing their progress due to any accidental deletions.
4.2 Validate Entity Relationship model design

Validation is another useful feature in the Entity Relationship Diagram Design Tool. Users can validate their design at any state of the ER model design. The validation will ensure the following: are there any entities, relation or attribute components without name or name with invalid pattern which is not allowed in database, entities without key, entities without attributes and attributes of an entity or relation with duplicate name. Moreover, the validation message will be descriptive and combine all errors at once instead of displaying one by one to users.

4.3 Print Entity Relation model design

Users can print the current ER model design anytime; in this case, users will be prompt to choose a printer. This will allow the users to store their work into a hard copy without completely losing the work if they happened to close the application before completing the ER model. The printing feature is not affected by the validation feature because printing the current ER model design is not related to functional dependencies or schema generation functionalities. Moreover, the printing the document is used as reference of the ER model design.
4.4 Generate Functional Dependencies

Generating and saving the functional dependencies is one of the major features in the Entity Relationship Diagram Design Tool. After successfully completing and validating the ER model design, user can generate the functional dependencies for their ER model. If there any “1 to 1”, “1 to N” or “N to 1” cardinality between entities in the ER model then users will be prompted to make decision whether to make a new table or not for the relation between those entities. After generating functional dependencies, users can save the generated functional dependencies into a “.txt” file. Also users can store the “.txt” file by browsing into any directory in their machine.

4.5 Generate Schema

Generating and saving the database schema is the major feature in the Entity Relationship Diagram Design Tool. After all ER model design validation passed, there will be two steps to generate database schema. First, if there any “1 to 1”, “1 to N” or “N to 1” cardinality between entities in the ER model then users will be prompted to make decision whether to make a new table or not for the relation between those entities. Second, users need to choose the datatype, i.e. Integer, Text, Blob, etc., for the attributes of each entity because in the database schema, the datatype must be provided for each database table column. Finally, users can generate the database schema and save the schema as “.sql” file. Also users can store the “.sql” file by browsing into any directory in their machine.
5. Design

Entity Relationship Diagram Design Tool is built on JavaFX and Java. GUI component is built using JavaFX and the rest of the application logic is written using Java. JavaFX is used to create rich java applications and it replaces the Swing as the standard library. FXML is used to define UI elements where it can be linked with a controller class. FXML is an XML-based UI markup language.

FXML is used to define ER model components in Entity Relationship Diagram Design Tool and the FXML can be loaded into the main application via FXMLLoader. Overall Entity Relationship Diagram Design Tool design is split into two phases. First, handling ER model components and process them in order to convert to database tables and columns (i.e. ER model phase). Second, handling database tables and columns objects, which are derived from the phase one, in order to generate functional dependencies and database schema (i.e. Database phase).

5.1 Graphical User Interface

The GUI component is split into two interfaces using two FXML files, “ER_Drawing_Interface” and “Table_View_Interface”. The first FXML file is used to create the ER model drawing area where users can draw the ER model components. Entity (represented by rectangle), Attribute (represented by Ellipse) and Relation (represented by square, angled with 45 degree). Moreover, the ER model drawing area contains action buttons for “Delete”, “Undo”, “Validate Design”, “Print ER Diagram” and “Save Schema and Functional Dependency".
The second FXML file is used to display table view for the ER model design. The table view provides the options to make decisions whether making a new table or not if the cardinality is “1 to 1”, “1 to N” or “N to 1” between entities. If there is more than one entity with those types of cardinalities then a “Next” button will show up and users can make choice for each entity by clicking the “Next” button. An example is shown in FIGURE 2.

![Table view](image)

**FIGURE 2. Users should select an option.**

Once the decision making part is completed, users can save the functional dependencies as “.txt” file and there will be a Browse window when click on “Save Functional Dependencies”
button. Moreover, users will be prompted to set datatypes for all attributes of each entity. If there is more than one entity then a “Save & Next” button will be appeared and user can set datatype for attributes for each entity one by one. Also there is a “Previous” button as well, in case users want to update the previously set values. An example is shown in FIGURE 3.

FIGURE 3. Users should set datatype for each attribute.
Once all the datatypes are set then the “Save Schema” button will be enabled and users can save the schema as “.sql” file and there will be a Browse window when click on “Save Schema” button. The saves schema file can be run in a database and that will create all tables which users modeled via Entity Relationship Diagram Design Tool.

5.2 Model Classes

The model classes are grouped according to phases. For the ER model phase, each ER model components are represented by a model class; for example, Entity component is represented as “EREntity” class. Each model classes in this phase are extended to “StackPane” super class because the model classes needs to have many JavaFX components and “StackPane” allows binding many components together. For instance, the “ERAttribute” model class contains “TextField”, “Ellipse” and two “CheckBoxes”.

For the Database phase, database table is represented by “DatabaseTable” class and database table column is represented by “DatabaseColumn” class. Moreover, a “DatabaseColumnTO” (i.e. Database Column Transfer Object) is used when transforming ER model attribute into database column. Additionally, two transformer classes, such as “EntityToTableTransformer” and “RelationToTableTransformer”, are used in between ER model phase and Database phase in order to transform data. In other words, the transformer classes transform data from ER model phase to Database phase.
5.3 View Classes

View classes are built by using FXML because it is more convenient to build and maintain and Scene Builder tool can be used to build GUI components. Scene Builder is a visual layout tool that allows designing JavaFX application UI quickly without coding and UI components can drag and drop into work area and the properties can be modified as well. Moreover, two FXML files are used; such as, “ER_Drawing_Interface” for ER model phase and “Table_View_Interface” for Database phase. Additionally, a Cascading Style Sheets (CSS) file is applied for GUI components’ styling purposes where the following properties are customized to make nice looking GUI: font, font size, color, padding, border color, background color, maximum length, and minimum length.

5.4 Controller Classes

Controller class logic is implemented in Java and Controller classes are split based on ER model phase and Database phase. ER model phase controller classes, such as ERDrawingController and ERModelShapeController, handle drawing the ER model components and buttons action events. Moreover, they manage the connectivity of the ER model components when users drag and drop in the drawing area and update the current GUI view upon any event or action performed by users. The “ERDrawingController” is loaded in “ER_Drawing_Interface” FXML so all the GUI components in the “ER_Drawing_Interface” can be accessed using ”@FXML” annotation and “fx:id” value, defined in FXML, rather creating new instances in the controller class.
Database phase controller classes, such as TableViewController and SchemaGenerateController, handle the transformation of ER model components into database components, validate ER model, generate functional dependencies, save functional dependencies, generate database schema and save database schema. Moreover, the controller classes responsible for displaying validation error message window and file browsing window to save functional dependencies and schema files.

### 5.5 Software Design Pattern

Software design pattern is a formalized best practice that can be used to solve a common problem when designing an application by the programmers. The software design pattern shows relationship and interaction between objects and classes without specifying the final application objects or classes that are involved\(^{[11]}\). Also the software design pattern may be viewed as a structured approach to computer programming intermediate between the levels of a programming paradigm and a concrete algorithm\(^{[11]}\).

The Entity Relationship Diagram Design Tool is built based on Model-View-Controller software design pattern. In Model-View-Controller software design pattern, the controller receives user inputs then manipulates model objects and updates the view accordingly\(^{[12]}\). In the application, classes are grouped into model, view and controller classes. Controller classes manipulate model objects based on users’ actions and update the view. Consequently, Model-View-Controller design pattern is the best fit for Entity Relationship Diagram Design Tool.
6. Challenges

Major challenge in this project is lack of knowledge in JavaFX because I had to learn JavaFX from scratch to complete the project. Learning JavaFX was the initial milestone in this project; consequently, I had to dedicate additional working hours for this project. Moreover, independently learning JavaFX required a lot of online researches and practicing JavaFX tutorials. JavaFX has wide range of functionalities and learning all of those in the short term is difficult so I had to isolate what needed and what not needed for this project.

Interacting with FXML and controller classes was difficult to understand because the components defined in the FXML need to have “fx:id” value in order to access those components.
in the controller class. Connecting entities, relations and attributes also challenging and it is not enough to set edge position equal to center of a ER model component because if users move a component then the component will become disconnected from the line. Consequently, I had to explicitly use JavaFX binding between ER model component and line and this will allow connecting those components while moving and the line will dynamically grow to satisfy the distance.

Implementing Delete and Undo functionalities was harder to succeed because deleting attributes is straight forward; at the same time, deleting entities and relations need to be cascading. If users delete entities then attributes and relations that are connected to the entity should also be deleted. Similarly, if users delete a relation then all the attributes also need to be deleted. On the other hand, when users undo a deletion, all the deleted components need to be restored with the same position where they were before the deletion and reconnect all connected components back.

7. Future Improvements

The following improvement can be implemented to Entity Relationship Diagram Design Tool in the future: Save the ER model design into database, export ER model design as XML file, import the ER model design from a XML file, run the generated schema against an external database, and make entities, relations and attributes as resizable components.

The Entity Relationship Diagram Design Tool does not have any feature to save users’ current work other than printing into a hard copy. This feature will help users to restore their
current progress back instead starting from scratch again. In this case, the application can be connected to database and store the current ER model design into a database table. Moreover, the ER model components have a fixed size and users cannot change the size but having resizable components would be a valuable feature so users can manage the drawing area space efficiently.

Another valuable feature would be exporting the ER model design into a XML file. This feature would help to load the XML file into other external applications which read XML inputs and the external application can interact with Entity Relationship Diagram Design Tool. Similarly, adding import ER model design from a XML file would also be a nice feature so the Entity Relationship Diagram Design Tool can interact with external applications.

8. **How to add new feature**

As described in the Design section, the Entity Relationship Diagram Design Tool uses Model-View-Controller software design pattern. Model-View-Controller design pattern allows efficient code reuse and maintenance. In case adding a new major feature, the implementation would affect the model, view and controller classes. For example, if a new view needs to be added then the implementation may require create new FXML file, a controller class for the FXML file and one or more model classes as needed.

As listed in the “Future Improvements” section, if the export XML file feature is required to implement then the changes need to be applied in the ER model phase which may need to add new functions in “ERDrawingController” or “ERModelShapeController” class and update
the “ER_Drawing_Interface” FXML. Similarly, the same changes needs to be implemented if the import XML feature is required to be added. Moreover, if the GUI components’ style is required to be updated then the “application.css” file needs to be updated with new style property values.

9. **Install Instructions**

1. Download and install Eclipse Oxygen (IDE for Java EE developers).

   [http://www.eclipse.org/downloads/packages/release/Oxygen/1A](http://www.eclipse.org/downloads/packages/release/Oxygen/1A)

2. Make sure you have jre1.8.0_40 or higher is installed in your system (Recommended is jre1.8.0_144); Otherwise you might get an error similar to this:

   [https://stackoverflow.com/questions/31065172/does-java-1-8-0-31-has-the-javafx-scene-control-alert-class-how-can-i-import-it](https://stackoverflow.com/questions/31065172/does-java-1-8-0-31-has-the-javafx-scene-control-alert-class-how-can-i-import-it)

3. Extract the submitted honour project zip file into a temporary directory

4. Launch Eclipse

5. Click File menu and select “Import…”
6. Under the “General” section, select “Existing Projects into Workspace” option
7. Click on “Browse” button and navigate to the temporary directory that you extracted the project zip and choose “erddesigntool” project
FIGURE 7. Choose the “erddesigntool” project

8. Click on “Window” then select “Perspective” then click “Open Perspective” and select “Others…” then select “Java” and click “Open”
9. Right Click on the “erddesigntool” project then click “Run As” and select “Java Application”. Now the Entity Relationship Diagram Design Tool will launch…
FIGURE 9. Run the Entity Relationship Diagram Design Tool application

10. Summary

Entity Relationship Diagram Design Tool is implemented for users who are beginners to ER model and database schema especially for COMP3005 students. Entity Relationship Diagram Design Tool’s GUI components are built using JavaFX and other application logics are
built on Java. The application is split into two phases: ER model phase and Database phase in order make the application design to be easy maintainable. Moreover, Model-View-Controller software design pattern is used as entire application design.

The Entity Relationship Diagram Design Tool has user friendly GUI, so users can easily use the application features. Users can draw the ER model by drag and drop components into drawing area, print the current ER model design, delete components, undo the last deletion, validate ER model design, generate functional dependencies, save functional dependencies as “.txt” file, generate database schema and save database schema as “.sql” file.

Helping people who are new to database and ER model is main focus in this application and increasing their learning by practicing via this tool. At the same time, learning new skills for my academic career is another motivation for this project. Moreover, I was able to learn some ethics as a student and a software developer. For example, meeting deadlines, satisfy customer expectations, developing software by following software design pattern and independently learning new skills.

Some features can be added in the future to the Entity Relationship Diagram Design Tool. Such as, able to save the ER model design into a database, able to export ER model design as XML file, able to import the ER model design from a XML file, able to run the generated schema against an external database, and making entities, relations and attributes as resizable components. Overall, the Entity Relationship Diagram Design Tool is useful application for COMP3005 students and uses who like to learn and design ER model and database schema.
11. References


https://en.wikipedia.org/wiki/Software_design_pattern

https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
12. Abbreviations

CSS – Cascading Style Sheets

ER – Entity Relationship

GUI – Graphical User Interface

SQL – Structured Query Language

UI – User Interface