An improvement in data management based on an existing hotel room booking model

0960229

Abstract

The aim of this model is a functioning data management system for a room booking model by using the fundamental concepts of Empirical Modelling (EM). This is an improved model based on an existing data model built by C. Y. Chung[1]. To begin, this report will: 1) identify the issues of original model and then 2) produce some suitable solutions to overcome the issues, 3) identify the limitation of the EDDI notation according to the improved model, lastly 4) discuss the multi-notation aspect in the EM models and Web applications.

1 Introduction

This report introduces a data management model by using a database notation in EM. The database notation is called EDDI[2] which is a database interpreter using an Eden definition. The notation extends a functionality of Eden in processing, managing and manipulating data. The database also provides a sustainable purpose that keeps the model to work continuously in any time.

This room booking model is built with EDDI notation as an improvement of the existing data model which was done by C. Y. Chung[1] as an undergraduate project. The report describes the limitation in the existing model from data storage and the fundamental of EM aspects, and then provides finding solutions to address the limitation in previous model. The improvement in the existing model involves 1) integrating non dependency data and stored these data into a database notation (EDDI), 2) solving the latent issues behind the existing model, such as the misunderstanding of EM fundamental concepts.

Furthermore, this report will then discuss the limitation of EDDI, the EM models and Web applications from a multi-notation aspect. If the database is functioning well in EDEN, the more models associate with the database that can be put on Web Eden as the Web applications.

2 The improvement based on the existing model

2.1 Limitation in the existing model

According to Chung's project, one of the major limitations in the model is the issues of data storage. These include[1]:

i. When a new reservation is made, the existing booking states are not able to update the reservation instantly. For instance, when the state of room101 is updated from vacancy (presented in background as white) to reserve (presented in background as yellow), the room101 is not able to change its background colour immediately. Therefore, it is unable to tell that the states have changed from interface.

ii. The data storage size in the existing model is limited so it can only store 6 weeks data of the booking states. When more data is needed, the size of list must be enlarged first before the additional data can be added into the list.

iii. The data of the booking states are stored in a list without specifying the date. When the data is changed, it is difficult to identify which state is likely to be changed. For example, the list in the model looks like below[1],

```
ma101 = [van,res,cop,van,res,van,cop,res,res,]
```

The segment above shows the booking states in a list room101, but there does not contain extra information to support the states, such as check in date, check out date.

In order to solve these issues, the improved model needs a reconstruction programme to meet the concepts of EM that will solve the issue i. It also needs a database that enables the model to store data within an unlimited storage size, more importantly the database should be compatible with Eden. This

will solve the issue ii and iii. As introduced in the section 1, EDDI notation is suggested by Dr. Beynon which has many successful examples working well with Eden.

2.2 Adding the EDDI notation

EDDI is like other EM notations that share the same variables with EDEN. The share variable in the improved model is the name of the table in EDDI called "*recordsTable*". This variable is a list in EDEN which contains all the data retrieved from EDDI. Therefore, the manipulation of the data in EDEN is similar to processing list functions and list data type.

In the model, there has one table stored in the EDDI initially for testing the database functionalities. The table contains five columns which are check in date, check out date, room number, name of the booking person, and the status of the room, such as "*reserve*" or "*occupy*". All the default room statuses are "*vacancy*" unless they are retrieved from database. This is because the database only stores the reservation and occupation of the rooms (See Figure 1).

```
-------------------------------------------------------------
dateFrom     | dateLeave    | roomNo | name  | status  |
-------------------------------------------------------------
[2010,01,01] | [2010,01,02] | 102    | Woods | occupy  |
[2010,01,24] | [2010,01,30] | 103    | Anna  | reserve |
[2010,01,24] | [2010,01,27] | 104    | Lee   | reserve |
[2010,02,12] | [2010,02,18] | 201    | John  | reserve |
[2010,01,28] | [2010,01,30] | 102    | Joe   | occupy  |
[2010,02,10] | [2010,02,15] | 302    | Mary  | occupy  |
-------------------------------------------------------------
(6 rows)
```

Figure 1: The EDDI database

This model has one data stored method which is typing the information into the textbox on the interface (See Figure 2).

In Figure 2, there has three buttons to function the database, such as appending and removing data. Each button triggers the corresponding procedure when it is clicked. For example, when the check in button is clicked, it triggers the "*checkIn*" procedure which reads all the data in the database. If the input data and the stored data in the database are identical, the input data will not be saved into database. This is to avoid the duplicate data stored in the database at the same time. The duplicate data may cause problems to the model when reading the database. If they are not identical, then the input data will be stored into the database. The execution of storing data is using "append" statement.

Figure 3 shows the statement indicates the name of the table "*recordsTable*" with a list of the data where the data is bound.

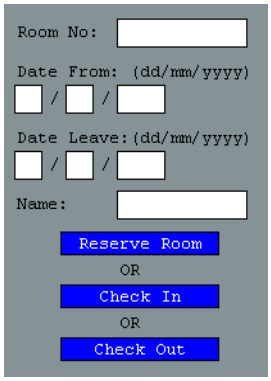

Figure 2: The interface for storing information into the database.

```
append recordsTable,
    [checkInDate,checkIn_LeaveDate,checkInRoom,name,"occupy"];
printInfo = "Data is saved.";
writeln(printInfo);
readRoomRecords();
```

Figure 3: Append statement for adding data into database.

2.3 The fundamental concepts of EM for the improved model

Chung's project also mentioned some error messages that are generated by EDEN, DONALD and SCOUT notations in the model. These errors refer to the three key concepts of EM which are observables, agents and dependencies. Also the issue i as mention in section 2.1 refers the same problem of misunderstanding the concepts.

First of all, the non-dependency data have to store in the database to ease the implementation of the model, such as name of the person.

Secondly, reconstructing the dependency in the existing model, such as booking status "vacancy", "reserve" and "occupy", they are originally defined as non-dependency variables. For example, Figure 4 shows a non- dependency definition in the existing model. If the colour of variable "*cop*" is changed into "green", the cop variables in the ma101 list will then be replaced to "green" (See Figure 5). This causes the model unable to identify the replaced variables "green" after the original variable "cop" changed its colour.

Therefore, the dependency declaration can prevent the same problem occurred in the improved model. Figure 6 shows one declaration of the dependency used in my model.

```
cop = "yellow";
res = "red";
van = "grey";

ma101 = [van,res,van,cop,res,
van,res,cop,res,]
```

Figure 4: dependency problem in the existing model

```
cop = “green”;
res = “red”;
van = “grey”;

ma101 = [van,res,van, green,res,
van,res, green,res,]
```

Figure 5: A result in changing the colour of variable “cop” in the existing model

```
vacancy is "white";
occupy is "red";
reserve is "yellow";
```

Figure 6: An example of the dependency definition

3 Limitation of notations

3.1 Limitation of EDDI

There are only 3 attribute types which are real (REAL), integer (INT) and string (CHAR) in EDDI. In my model, it requires to store some dates like duration of reservation and occupation in the database. Since there are not many choices about the attribute types, the dates are stored as string.

When EDEN retrieves the data from the database, the dates cannot be used for calculating the duration between two dates straight away. In order to make a comparison, the dates then need to be converted from string to integer. This is one inconvenience in using this notation.

Furthermore, all the data processing are computed under EDEN, the role of EDDI in this model only plays as a data storage. This has limiting EDDI on operators, such as, union, difference, intersection, selection and join, also increasing the complexity of EDEN implementation. For example, almost all functions in EDEN have to cooperate with the database procedure which runs by a “*for*” loop, in order to read all the data or update the data in the database. While the model is operating, there would have innumerable “*for*” loops running. When the size of the database becomes larger, it may slow the performance of the execution in reading all the data in the database.

4 Multi-notation aspect

Many reports[1,3] argues the separation concerns between various notations and confusing programming format within each notation. However, this is also happening in real life, an application may be built by different programming languages sometimes, such as Web applications. The web applications usually associate with a database to increase the usability of the application itself. This is also a popular and common approach on the Web. A programmer usually requires the understanding of a server, JSP or PHP, SQL, even JavaScript. It is also not a trivial concept to build up the knowledge of Web applications at first.

On the other hand, each notation has its distinguishing feature and purpose in EM. When these notations join together, it could have various extensions to build a model. For example, Chung’s model had many fascinating functions, such as displaying a room layout, changing the location of the furniture in a room. This is because a 2D graphic is supported by Donald. Moreover, it is one of advantages in EM that Web applications could not overcome easily.

5 Further works

As mention in section 4, the web application usually operates with a database together. If the EDDI notation is able to process data as much like the database using in the Web applications. The further work could be putting the database-managed EM model on the Web Eden for extending the usability of the model itself.

6 Conclusion

To conclude, the improved model has overcome the limitation of the existing model by using EDDI notation. However, EDDI itself has few limitations when associating with EDEN. Although the data are converted its types between string and integer many times when processing the data, the notation is still functioning well in the improved model.

Acknowledgements

I would like to use this opportunity to thank Dr. Meurig Beynon who spent his time explaining the concepts of EM and problems of the previous model. Also I would like to thank my personal tutor Dr. Steve Russ who helped me to proofread my draft report and give me useful advices to the report.

Many thanks to my course-mate Yu Guan helped me to clarify my programming logic. In order to make the model runs successfully.

References

[1] C. Y. Chung. The use of Empirical Modelling in Data Modelling for Hotel Management.

[2] EDDI – the Eden Definition Database Interpreter. Url:http://www2.warwick.ac.uk/fac/sci/dcs/research/em/notations/eddi/

[3] 0523756. What Empirical Modelling tools should look like. *The Fith Warwick Empirical*

Modelling Bulleting (WEB-EM5).
Url:http://www2.warwick.ac.uk/fac/sci/dcs/research/em/publications/web-em/05