ISSN 2278-3091
Volume 8, No.3, May - June 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse70832019.pdf
https://doi.org/10.30534/ijatcse/2019/70832019

Similarity Syntax Rules between DFD and UML Diagrams

Norhanim Selamat[1], Rosziati Ibrahim[2]
[1]Universiti Tun Hussein Onn Malaysia (UTHM), Malaysia, norhanim@uthm.edu.my
[2]Universiti Tun Hussein Onn Malaysia (UTHM), Malaysia, rosziati@uthm.edu.my

ABSTRACT

Today, individual and society relies on innovative software systems. This situation needed software engineering to produce dependable and trustworthy systems economically and rapidly. During analysis and design, requirements specification can be documented using two (2) mostly methodologies which are SDLC (software development life cycle) and OOSAD (object-oriented system analysis and design). In SDLC, DFD (data flow diagram) specification is used, meanwhile UML (unified modelling language) specification is used for the object-oriented approach. Nowadays, there is an intensive research for model transformation, integrating and merging from structured to object-oriented approach. Most researches are concentrated on converting the existing DFD to UML diagram. In the end, they produced a new UML diagram as a design material and the implementation of new systems. It is also serves as a documentation for the upcoming development of new systems. This paper proposes a framework which is acquired from the syntax rules for DFD and UML. Established along the similarity methods, this paper will define the syntax rules of DFD Level 0, DFD Level 1, UML Use Case Diagram and Sequence Diagram. Then, the method is tested with the case study of DFD and UML diagrams. Furthermore, the factors involved in the similarities are found. And so, the similarity can be utilized to generate requirements specification from DFD to UML Diagram.

Key words: DFD, UML, use case diagram, sequence diagram

1. INTRODUCTION

Today, every individual and society depend on an innovative software system. This situation requires software engineering to produce a system that is reliable and trustworthy in line with the economy and rapid technological developments such as [1] and [2]. During requirements engineering process, requirements elicitation is the second important stages after determining the feasibility study for a software system. Furthermore, requirements specification phase began when the developers start modelling needs of users, who use certain methods (either structured approach or object-oriented approach). During this phase, descriptive user requirements is significant for producing complete and precise requirements such as [3] and [4].

Structured methodology based on the Waterfall model is a commonly used approach in the system development life cycle (SDLC). The development of information systems using SDLC as stated by Dennis et. al [5], must follow four phases including planning, analysis, design, and implementation. Then, every activity and deliverables should be completed in each of the phases before proceeding to the next stage. The structured approach is very useful to model and document legacy systems [6]. The data flow diagram (DFD) is the most well-known diagram in the structured approach requirements specification [8].

In addition, analysts can use object-oriented approach. This approach provided to accelerate the response to the dynamic business environment in the evolution of the system [7]. Object-oriented technique is called unified modelling languages (UML). Kendall and Kendall [3] stated that four (4) phases in UML which are problem analysis, identification, analysis, and design. Meanwhile, UML has a group of diagrams that are employed to model the different characteristics of object-oriented software.

Various researchers have proposed some preliminary methods and a framework to transform DFD with UML diagrams [8]–[12]. However, they are converting the existing DFD to UML diagram. In the end, they produced a new UML diagram as a design material and the implementation of new systems. It is also serves as a documentation for the upcoming development of new systems. In this paper, we propose a framework, which is acquired for the syntax rules for DFD and UML. Established along the similarity methods, this paper will define the syntax rules of DFD Level 0, DFD Level 1, UML Use Case Diagram and Sequence Diagram. Then, the method is tested along the case study of DFD and UML diagrams. Furthermore, the factors involved in the similarities are found and the similarity can be utilized to generate requirements specification from DFD to UML Diagram. Noted that DFD [13] and UML standard [14] are referred.

The rest of this paper is structured as follows. The discussion of related work in Section 2. Then, syntax rules for DFD and UML are stated in Section 3. Section 4 presents a case study for extracting the similarity syntax rules (SSR). Section 5 discusses the framework for SSR. Finally, in Section 6, we conclude the paper.

2. RELATED WORKS

During the phase of the requirements specification, two (2) approaches can be used for system's requirements which are DFD for structured approach and UML for object-oriented approach. DFD technique is also known as process modelling. It has been widely used in requirements specification such as in [15] and [16]. According to Dennis et al. [5] and Kendall and Kendall [17], the data store usually appears in the next level diagram that is DFD level 0. This diagram decomposes a single process from context diagram that includes entities and processes. Then, each process at DFD level 0 can be described more clearly in the next level of DFD, called as DFD level 1. When the diagram is needed to determine the user's requirements, then, decomposed can be expanded to the nearest bottom diagram.

Generally, syntax and semantic errors can occur in DFD which are considered as two fundamental problems that differ in the requirements specification. Syntax errors can be identified using clear rules compared with semantic errors. So, these errors are uncomplicated to find and fix in a set of DFD. According to Ibrahim and Yen [18] analysts should follow a set of rules to evaluate DFD in terms of accuracy and consistency of the diagrams.

Meanwhile, UML is usually used to state common idea for requirements specification. UML use case and sequence diagrams are mostly used to illustrate the functional requirements of a system in the object-oriented approach. Many researchers such as [19] and [20] said that UML basically provides a lot of syntaxes, but not enough semantics and it lacks the rigor of formal modelling languages. Nowadays, a formal definition of a common Meta Object Facility (MOF) based metamodel stated in [14] can be specified for the abstract syntax rules and semantics of the UML. This requirement needed to fulfill the primary goal of UML for advancement in the industry by supporting object visual modelling tool interoperability. While the semantics are to be recognised by computers in a technology independent way.

Various researchers have proposed some preliminary methods and a framework to transform DFD with UML diagrams. Madanayaneke et al. [12] proposed software prototype of transformation between roles and goals from requirements into UML Use Case. They are using Ontology Based Framework as a basis for their work. This prototype also could execute other transformation such as DFD Level 1 to state diagram and sequence diagram. Jacob et al. [8] presented the transformation technique between DFD and three UML diagram namely; use case diagram, class diagram and activity diagram using ten academic projects. They conclude that the mapping techniques helps the designer assist between function-oriented design and object oriented design methodologies. However, some weakness appears during mapping which is mismatches among elements used in notation transformation.

Handigund and Nagalakshmi [10] proposed a methodology of Use Case Flow Diagram, which can act as an association for use case, sequence and composite diagrams. They are using elements of a DFD because the diagrams are equivalent to the gap of object methods that is a use case to be conceptualized from class diagrams. They also present sufficient syntax to derive the semantic of Use Case Flow Diagram for the connection. Fries [7] proposed a framework for the transformation of DFD and ERD to UML diagram including use case, sequence, and class diagram. Fries [7] proposal converted the existing DFD to UML diagram. However, a manual decision used to encounter the transformation. Tiwari et al. [11] presented an approach that merges the DFD with the UML diagrams. Amalgamation of the diagrams can help the developer to analyse the system. They are using transformation technique to produce a combination model, including Use case to DFD and DFD into Class Diagram. Table 1 presents the analysis of the existing DFD and UML framework.

Table 1: Analysis of the Existing DFD and UML Framework

Author	Diagrams	Frame-work	Technique
Madanayanake et. al (2017) [12]	DFD Level 0, UML Use Case, and UML Sequence	Yes	Transformation
Jacob et. al (2016) [8]	DFD Level 0, DFD Level 1, and UML Use Case	No	Transformation
Handigund & Nagalaksmi (2014) [10]	DFD Level 0, and UML Use Case	Yes	Transformation
Fries (2012) [7]	DFD Level 0, DFD Level 1, UML Use Case, and UML Sequence	Yes	Transformation
Tiwari et. al (2012) [11]	DFD Level 0, DFD Level 1, and UML Use Case	No	Merging

3. SYNTAX RULES FOR DFD AND UML

In this section, we describe syntax rules for similarities between the DFD Level 0 and Use Case Diagram as stated in Table 2. Meanwhile in Table 3 shows syntax rules for similarities between the DFD Level 1 and Sequence Diagram.

Table 2: Syntax Rules for Similarities between DFD and UML Use Case Diagram

DFD Level 0	UML Use Case
Rule 1: External Entity, Process	**Rule 1:** Actor, Use case
Rule 2: At minimum one input data flow for an external entity, At minimum one output data flow for an external entity, At minimum one input data flow for a process, At minimum one output data flow for a process, At minimum one input data flow for a data store, At minimum one output data flow for a data store, Output data flows usually have different name than input data flows for a process, Every data flow connects to at minimum one process, Data cannot move directly from one data store to another data store.	**Rule 2:** An actor can be associated with other actors using a specialization or superclass relationship, An actor is placed outside the subject boundary, An actor can provide input to the system, An actor receives output from the system, A use case can have a relationship (extend/ include) with another use case A use case is placed inside the system boundary, A relationship has a modality of null or not null as a property, A relationship is dependent or independent.
Rule 3: A specific name that is a verb phrase, a number and a description for a process, A specific name that is a noun and a description for the external entity,	**Rule 3:** A specific name that is a noun and a description for the actor, A specific name that is a verb and a description of the use case

Based from Table 2, from our define syntax rules, the elements involved in the similarities are detected. All of syntax referred to DFD [13] and UML standard [14]. The elements are external entity and process for DFD while, actor and use case for Use case Diagram in Rule 1. Furthermore, Rule 2, and 3 will follow during creating the elements in the diagrams.

Based from Table 3, the elements of DFD Level 1 are data flows and data stores while, the message and object for Sequence Diagram in Rule 4. Furthermore, Rule 5 and 6 will be followed during creating the elements in the diagrams.

Table 3: Syntax Rules between DFD Level 1 and UML Sequence Diagram

DFD Level 1	UML Sequence
Rule 4: Data flows, Data stores	**Rule 4:** Message, Object
Rule 5: At minimum one input data flow for an external entity, At minimum one output data flow for an external entity, At minimum one input data flow for a process, At minimum one output data flow for a process, At minimum one input data flow for a data store, At minimum one output data flow for a data store, Output data flows usually have different name than input data flows for a process, Every data flow connects to at minimum one process, Data cannot move directly from one data store to another data store	**Rule 5:** An actor can send message to another actor or object, An actor can receive messages from another actor, An actor is placed on the top of the diagram, An object can send message to another actor or object, An object can receive messages from another actor or object, An object is placed on the top of the diagram, A message to pass on information from one object to another object,
Rule 6: A specific name that is a noun and a description for the data store, A specific name that is a noun and a description for the data flow	**Rule 6:** A specific name that is a noun and a description for the object, A specific name that is a verb and a description of the message

4. A CASE STUDY FOR EXTRACTING THE SIMILARITY SYNTAX RULES (SSR)

We use the Library system in this section, as a case study to describe the similarity syntax rules (SSR) between DFD Level 0 and use case diagram based on Rule 1 until Rule 3 of syntax rules as stated in Table 2. Figure 1 illustrates the DFD Level 0 of the Library system. While, Figure 2 states the Use Case diagram of library system.

In developing DFD Level 0 diagram for the Library system as shown in Figure 1, we have recognised two (2) external entities to show parts of an automated library circulation system. These external entities can receive information from or provide information to the system. Students specify key inputs to the systems and librarians provides information to the system. We have identified five (5) separate processes.

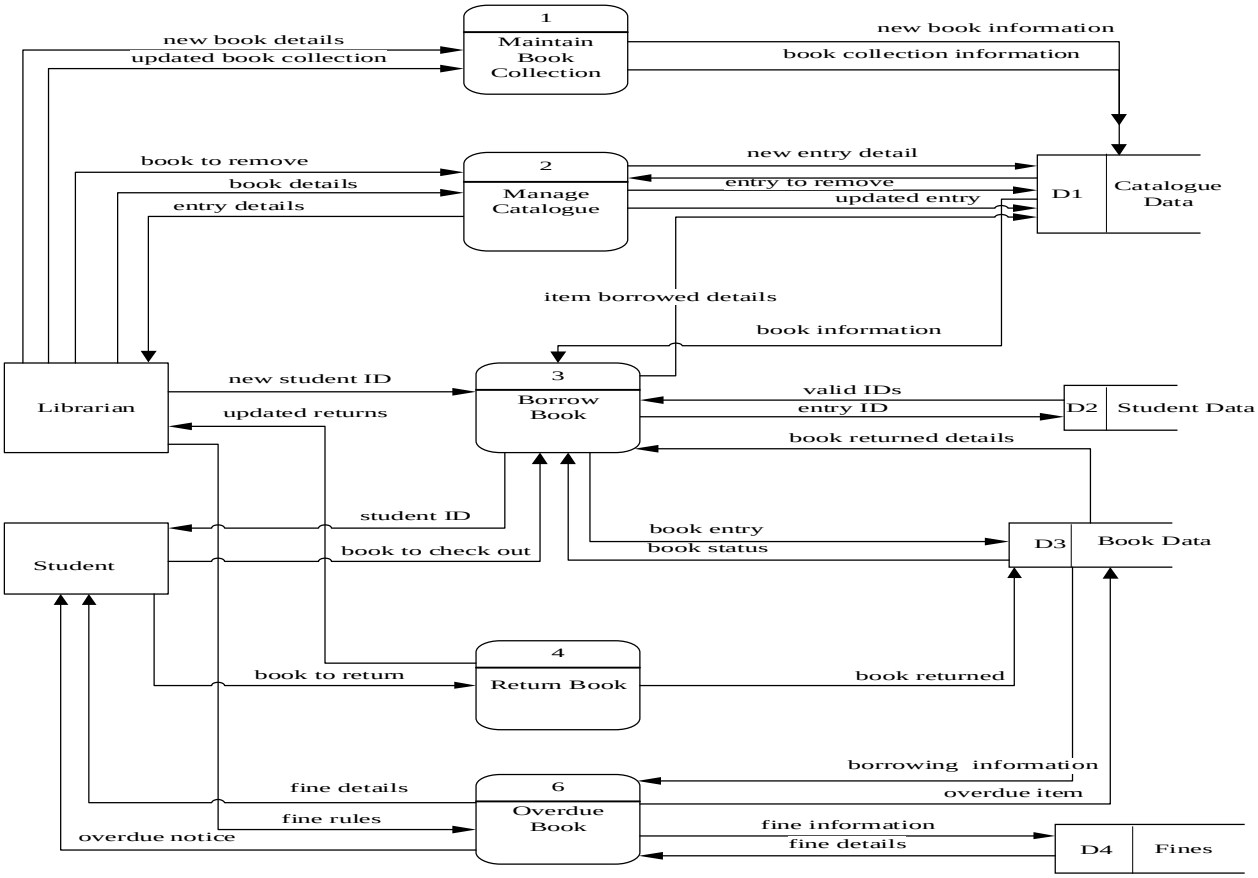

Figure 1: DFD Level 0 of Library System

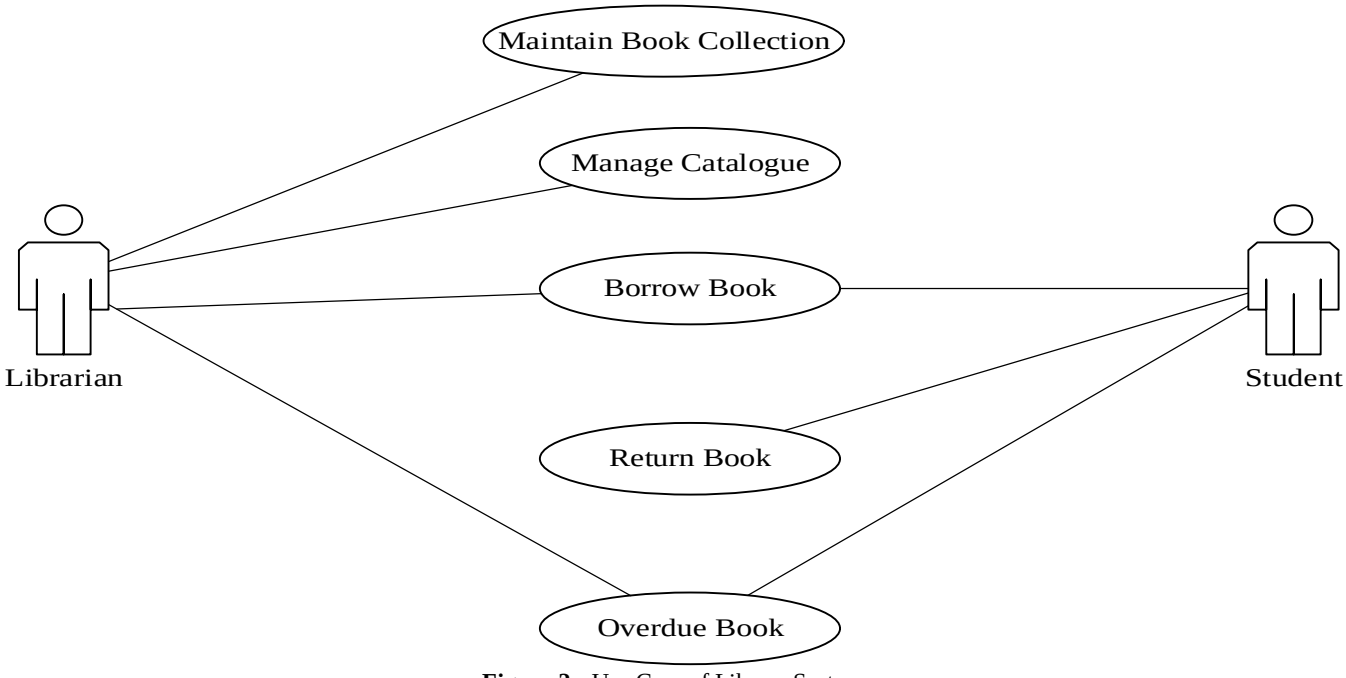

Figure 2: Use Case of Library System

While in the DFD level 0 diagram, these major functions correspond to actions such as the following:

- Process 1: maintenance book collection activities handle with inserting and deleting books from the library's book collection.
- Process 2: students can be searched managing catalogue on the website including book's availability.
- Process 3: borrowing activities built around registering student as a borrower, checking books out and borrowers' status.
- Process 4: returning activities deal with checking returning books by students.
- Process 5: the librarian can verify either the students have any overdue books or unpaid fines.

Meanwhile, the primary actors in the developing Use Case diagram are Librarian and Student. While the primary business processes are the following:

i. Librarian maintain book collection activities which handle with inserting and deleting books from the library's book collection.
ii. The librarian can manage catalogue that searched by Students on the website including book's availability.
iii. The librarian can register a student. The student can do borrowing activities, which are checking books out and get status from systems.
iv. The student can do returning activities such as checking returning books.
v. The librarian can verify either the students have any overdue books or unpaid fines.

Typically, a use case is initiated by actors. For example, `Overdue Book` is initiated by the `Librarian`. A use case can interact with actors other than the one that initiated it. The `Overdue Book` although initiated by the `Librarian`, interacts with Student by sending reminder emails to them who have overdue books. Figure 2 shows all the association between actors and use cases.

Table 4: Similarities of Elements for DFD Level 0 and UML Use Case Diagram

DFD Level 0	Use Case

External Entity
`Librarian`
`Student`
	Actor
	`Librarian`
	`Student`

Process
`Maintain Book Collection`
`Manage Catalogue`
`Borrow Book`
`Return Books`
`Overdue Book`
	Use case
	`Maintain Book Collection`
	`Manage Catalogue`
	`Borrow Book`
	`Return Books`
	`Overdue Book`

Therefore, we can detect both diagrams have similarities from system users (`Librarian` and `Student`) and functional requirements (`Maintain Book Collection`, `Manage Catalogue`, `Borrow Book`, `Overdue Book` and `Return Books`). Similarities elements of DFD from the structured approach and Use case diagram from the object-oriented approach is summarized in Table 4.

Table 5: Similarities of Elements for DFD Level 1 and UML Sequence Diagram

DFD Level 1	UML Sequence

Data Store
`Student`
`Catalogue`
`Borrowed Item`
	Object
	`Student Database`
	`Catalogue`
	`Book`

Data Flow
`new student ID`
`entry ID`
`valid IDs`
`student ID`
`book to check out`
`borrower details`
`book information`
`book entry`
`book returned details`
`book status`
	Message
	`RegisterBorrower (NewStudentID)`
	`RegisterBorrower (EntryID)`
	`RegisterBorrower (ValidID)`
	`CheckBorroweredBook (StudentID)`
	`CheckBorroweredBook (CheckOutBook)`
	`CheckBorroweredBook (BorrowerDetails)`
	`CheckBorroweredBook (BookInformation)`
	`GetBorroweredStatus (BookEntry)`
	`CheckBorroweredBook (BookReturnedDetails)`
	`GetBorroweredStatus (BookStatus)`

From the comparison of similarities syntax rules (SSR) between DFD and UML Use Case Diagram as shown in Table 2, we conclude that similarity of elements in Rule 1 clearly indicated in Table 4. Additionally, Rule 2 and, 3 also support the similarities when both diagrams are developed into a complete and consistent manner. Therefore, by introducing the similarities of elements of informal syntax rules, it can be used to determine the diagrams that are illustrated correctly.

We proceed the case study of the Library system to see how the DFD Level 0 diagram can be further decomposed. The third process in Figure 3 is called `Borrow Book` which does the borrowing book activities. The `Librarian` who will register students as a borrower (3.1). After the registration process, the `librarian` will examine the book borrowed by the Student (3.2). The `Student` with valid ID and does not have any overdue books or any fines can borrow the books. Then, `Student` will make a book checkout and accept the Student's status. The similarities elements are shown in Table 5.

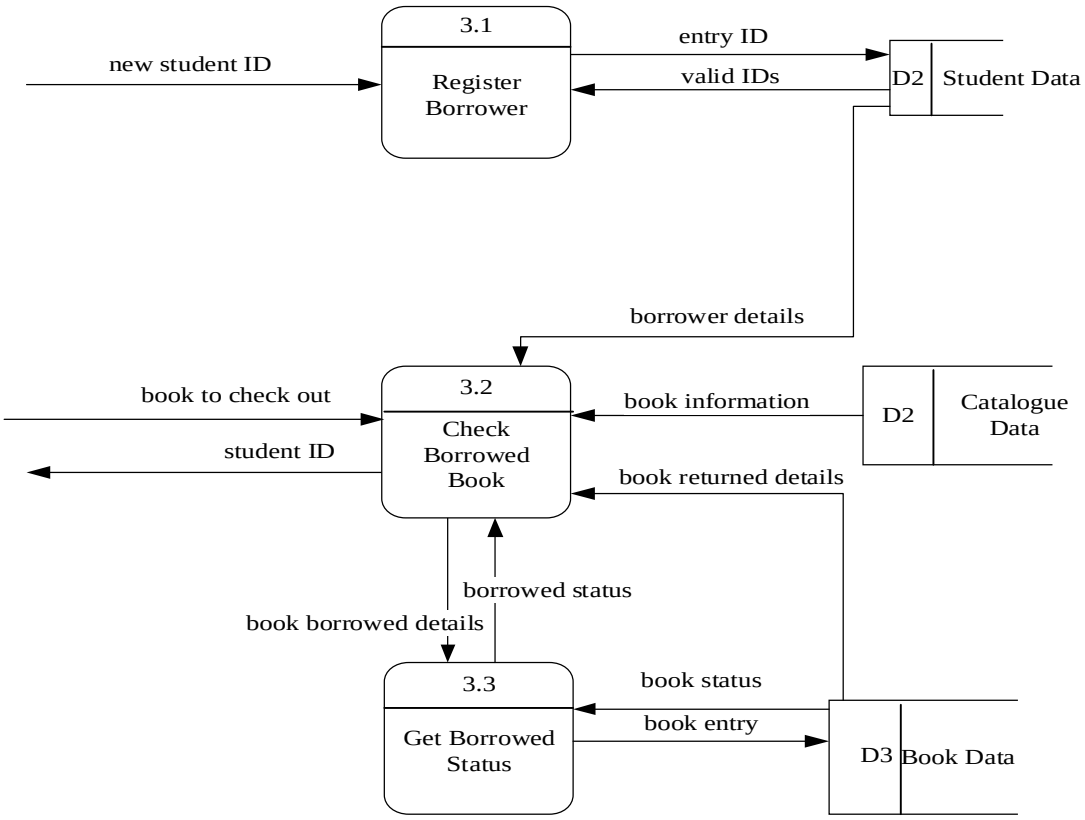

Figure 3: DFD Level 1 of Library System

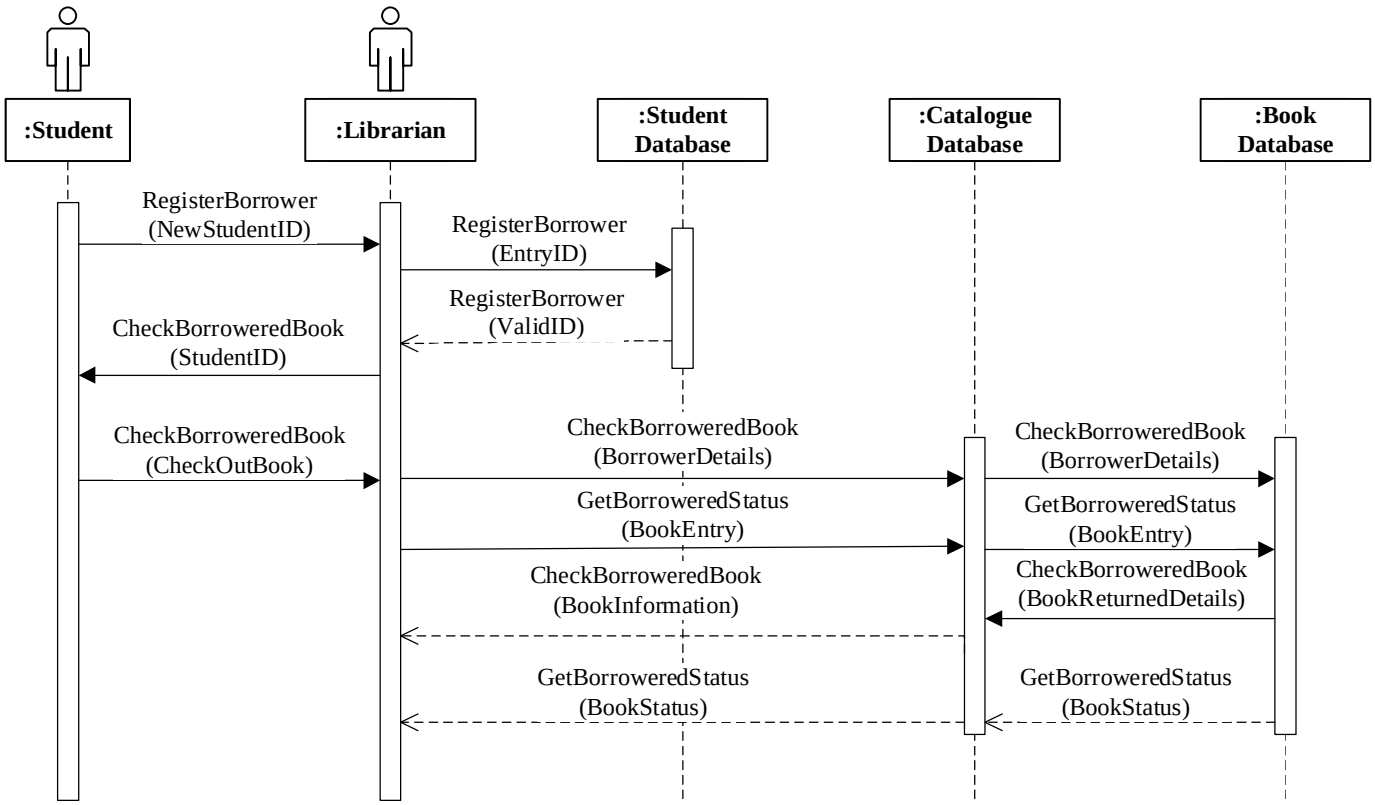

Figure 4: Sequence Diagram of Library System

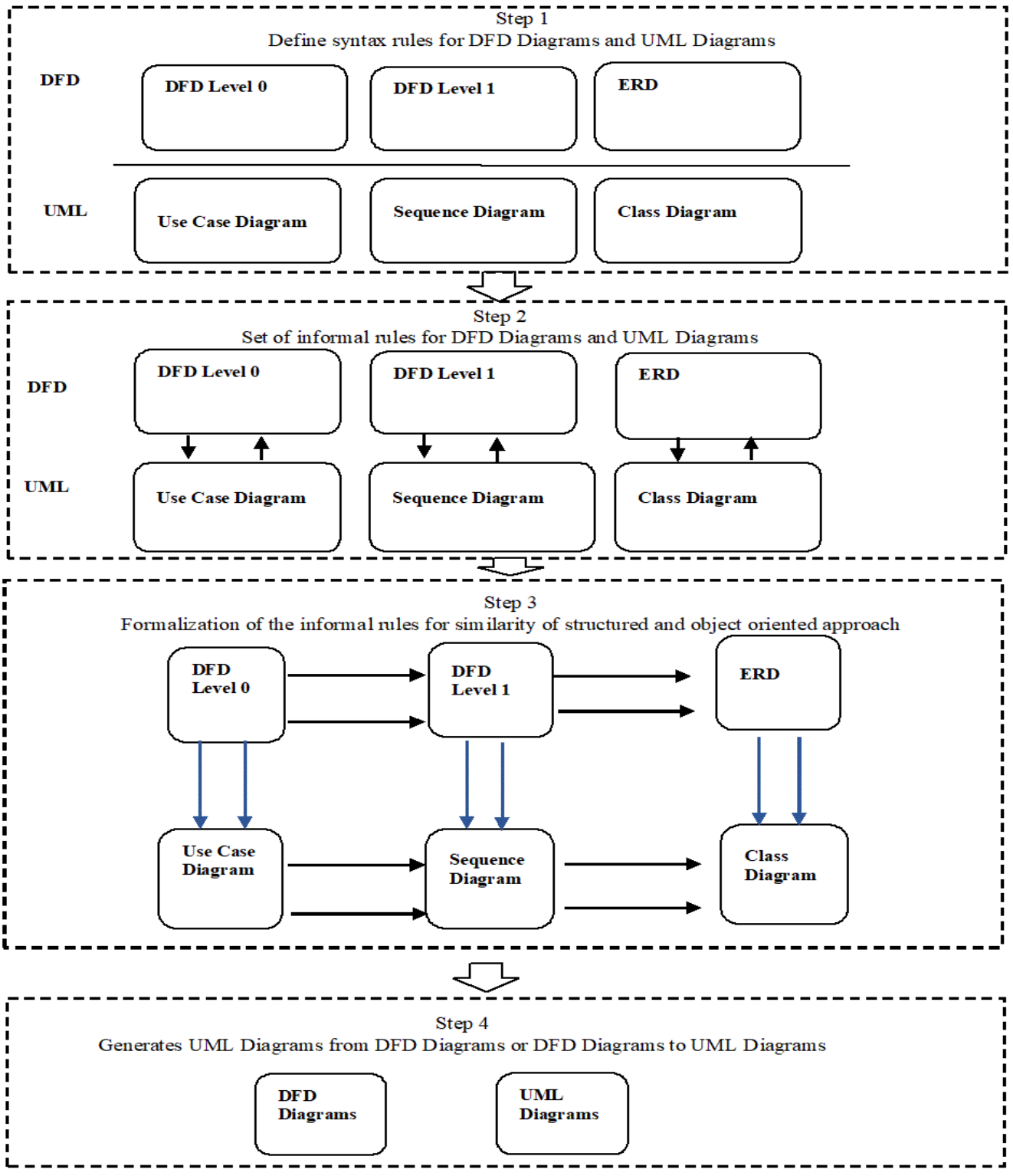

Figure 5: Framework of Consistency and Similarity Rules for Structured Approach with Object Oriented Approach

Figure 4 shows the UML sequence diagram. In this case study, we draw use case `Borrow Book` for students who have valid IDs and do not have any overdue books and fines. `Librarian` executes the procedure. This procedure allows the student as one of `Student` type to send the message `CheckBorrowedBook(CheckOutBooks)` to ask the `Librarian` to perform the process when the student gives the `Librarian` the books to check out. The `Librarian` in return register the Student. Then, the `Librarian` activate `RegisterBorrower(EntryID)` procedure in the Student Database to execute the `EntryID` data to validate the student's ID number. This process is performed during checking borrowed book until the StudentID and BookStatus data displayed to student. From the case study in Figure 3 (structured approach) and Figure 4 (object-oriented approach), we can identify similarities for DFD Level 1 and UML Sequence Diagram. Both diagrams describe the data used during the development of the Library system. Similarities elements of DFD Level 1 from the structured approach and UML Sequence diagram from the object-oriented approach are summarized in Table 5.

5. THE FRAMEWORK FOR SIMILARITY SYNTAX RULES (SSR)

Based on Table 3 and 4, the framework is proposed in Figure 5 for similarity between DFD and UML. There are four (4) main steps in defining the framework of consistency and similarity rules, namely, define syntax rules for DFD diagrams and UML diagrams, set of informal rules for DFD Diagrams and UML Diagrams, formalization of the informal consistency and similarity rules and generate diagrams between structured and object-oriented approach.

In the first step, there are two (2) DFD elements for Level 0, two (2) DFD elements of DFD Level 1, three (3) DFD elements of ERD, two (2) UML elements for use case diagram, two (2) UML elements for sequence diagram, and three (3) UML elements for class diagram will be defined from each syntax rules diagrams. In step 2, from the DFD elements that follow to each DFD diagrams and their consistency rules will be set of informal rules. Afterward, a set of similarity rules will be determined, which integrated from DFD rules and UML rules. Then, in step 3, formalization of several consistency rules and several similarity rules between those diagrams will be developed. Similar approach will be adopted for forming the formal rules as discussed in [18], [19], [21], and [22]. Lastly, in step 4, a tool will be developed to generate UML diagrams from DFD diagrams or DFD diagrams from UML diagrams.

6. CONCLUSION AND FUTURE WORKS

This paper has discussed a comparison of a set of syntax rules between DFD (Level 0 and Level 1) and UML (Use Case Diagram and Sequence Diagram). The rules are then used to check the similarities between both diagrams that are DFD represented for the structured approach, meanwhile UML Use Case Diagram represented for the object-oriented approach.

For future works, the formalization for syntax rules and similarities rules between the DFD and UML diagrams namely; DFD Level 0, DFD Level 1, ERD, UML Use Case Diagram, Sequence Diagram and Class Diagram will be developed.

ACKNOWLEDGEMENT

The authors would like to thanks the Malaysia Ministry of Education for supporting this research under the Scheme of Academic Training Initiative (SLAI) and Universiti Tun Hussein Onn Malaysia (UTHM) for funding this research under E015501, Research Management Centre (RMC).

REFERENCES

1. Lile, R., **Towards a New Critical Role of Information Systems in the Modern Decision Making Process**, Int. J. Adv. Trends Comput. Sci. Eng., Int. J. Adv. Trends Comput. Sci. Eng., vol. 8, no. 1.1 SI, pp. 48-53, 2019. https://doi.org/10.30534/ijatcse/2019/1081.12019
2. Ismail, N. A., & Daud, S. N. **Exploratory Factor Analysis of Pre-Purchase Construct for Airlines e-Ticketing System in Malaysia in The context of e-Transaction**. Int. J. Adv. Trends Comput. Sci. Eng. t al., Int. J. Adv. Trends Comput. Sci. Eng., vol. 8, no. 2, pp. 255-258, 2019. https://doi.org/10.30534/ijatcse/2019/25822019
3. M. Hagal and F. Kandemili, **Reducing Missed Requirements Issues: Complete, Unambiguous and Necessary Requirements Elicitation**, Int. J. Adv. Res. Comput. Sci. Softw. Eng., vol. 7, no. 1, pp. 10–14, 2017. https://doi.org/10.23956/ijarcsse/V7I1/01110
4. J. Melegati, A. Goldman, F. Kon, and X. Wang, **A model of requirements engineering in software startups**, Inf. Softw. Technol., vol. 109, pp. 92–107, 2019. https://doi.org/10.1016/j.infsof.2019.02.001
5. A. Dennis, B. H. Wixom, and R. M. Roth, **Systems analysis and design**, 5th ed. New Jersey: John Wiley & Sons, Inc., 2012.
6. H. Zhang, W. Liu, H. Xiong, and X. Dong, **Analyzing data flow diagrams by combination of formal methods and visualization techniques**, J. Vis. Lang. Comput., vol. 48, pp. 41–51, 2018.
7. T. P. Fries, **Reengineering Structured Legacy System Documentation to UML Object-oriented Artifacts**, in Information Systems Reengineering for Modern Business Systems : ERP, Supply Chain and E-Commerce Management Solutions, R. Valverde and M. R. Talla, Eds. Hershey: IGI Global, pp. 30–53, 2012. https://doi.org/10.4018/978-1-4666-0155-0.ch002

8. P. M. Jacob, Muhammed Ilyas H, J. Jose, and J. Jose, **An Analytical approach on DFD to UML model transformation techniques**, in 2016 International Conference on Information Science (ICIS), pp. 12–17, 2016. https://doi.org/10.1109/INFOSCI.2016.7845292
9. M. Dahan, P. Shoval, and A. Sturm, **Comparing the impact of the OO-DFD and the Use Case methods for modeling functional requirements on comprehension and quality of models: a controlled experiment,**" Requir. Eng., vol. 19, no. 1, pp. 27–43, Mar. 2014. https://doi.org/10.1007/s00766-012-0155-2
10. S. M. Handigund and S. R. Nagalakshmi, **An ameliorated methodology for the design of panoptic usecase flow diagram to constellate UML's usecase and sequence/communication diagrams**, in Proceedings - 2014 4th International Conference on Advances in Computing and Communications, ICACC 2014, pp. 153–156, 2014. https://doi.org/10.1109/ICACC.2014.43
11. K. Tiwari, A. Tripathi, S. Sharma, and V. Dubey, **Merging of Data Flow Diagram with Unified Modeling Language**, Int. J. Sci. Res. Publ., vol. 2, no. 8, pp. 1–6, 2012.
12. R. Madanayake, K. Asanga Dias, and N. Kodikara, **Transforming Simplified Requirement in to a UML Use Case Diagram Using an Open Source Tool**, Artic. Int. J. Comput. Sci. Softw. Eng., vol. 6, no. 3, 2017.
13. C. Gane and T. Sarson, **Structured Systems Analysis: Tools and Techniques**. New York, NY: Improved Systems Technologies, Inc., 1977.
14. Object Management Group, **OMG Unified Modeling Language TM (OMG UML), Superstructure**. Needham, MA 02494, USA: Object Management Group (OMG), 2011.
15. S. Tegginmath, **Data and Process Modelling: Investigating the Gap between Education and Industry Expectations in New Zealand**, M.P. Thesis, School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, New Zealand, 2017.
16. H. Xiong, H. Zhang, X. Dong, L. Meng, and W. Zhao, **DFDVis: A Visual Analytics System for Understanding the Semantics of Data Flow Diagram**, Commun. Comput. Inf. Sci., vol. Vol. 727, pp. 660–673, 2017 https://doi.org/10.1007/978-981-10-6385-5_55
17. K. E. Kendall and J. E. Kendall, **Systems Analysis and Design**. New Jersey: Pearson Education, Inc, 2013.
18. R. Ibrahim and S. Y. Yen, **A Formal Model for Data Flow Diagram Rules**, J. Syst. Softw., vol. 1, no. 2, pp. 60–69, 2011.
19. N. Ibrahim, R. Ibrahim, M. Z. Saringat, D. Mansor, and T. Herawan, **Consistency rules between UML use case and activity diagrams using logical approach**, Int. J. Softw. Eng. its Appl., vol. 5, no. 3, pp. 119–134, 2011.
20. R. Hazra and S. Dey, **Consistency between Use Case, Sequence and Timing Diagram for Real Time Software Systems**, Int. J. Comput. Appl., vol. 85, no. 16, pp. 975–8887, 2014. https://doi.org/10.5120/14924-3444
21. H. Aman and R. Ibrahim, **Formalization of Transformation Rules from XML Schema to UML Class Diagram**, Int. J. Softw. Eng. Its Appl., vol. 8, no. 12, pp. 75–90, 2014.
22. R. Ibrahim, H. Aman, R. Nayak, and S. Jamel, **Consistency check between XML Schema and class diagram for document versioning**, Int. J. Adv. Sci. Eng. Inf. Technol., vol. 8, no. 6, pp. 2590–2597, 2018. https://doi.org/10.18517/ijaseit.8.6.5007