Unifying Modeling Language-Merise Integration Approach for Software Design

Issar Arab[1]
Department of Informatics, Technical University of Munich,
Garching/Munich, Germany

Safae Bourhnane[2], Fatiha Kafou[3]
School of Science and Engineering
Al Akhawayn University in Ifrane, Ifrane, Morocco

***Abstract*—Software design is the most crucial step in the software development process that is why it must be given a good care. Software designers must go through many modeling steps to end up with a good design that will allow for a smooth development process later. For this, designers usually have to choose between two main modeling methodologies: Merise and UML. Both methodologies are widely used; however, each one has its own advantages and disadvantages. This paper combines both techniques and merges their advantages to come up with an approach that would help software designers make the best of both methodologies. This integration mainly targets the software design step in general but can be specifically applied to database design. It presents the weaknesses and strengths of each one of UML and Merise as two techniques used in database modeling and design. Later in this paper, a comparing of UML and Merise diagrams is lead and based on it, the decision on which of the two is the best at each step of the modeling process.**

***Keywords*—UML; Merise; modeling; design; databases**

I. Introduction

Database is what all software developers are concerned with in the first place. If you have a well-designed database, you can be sure that the entire development process will go as smoothly. For the purpose of designing the best database, tools and frameworks like Merise and UML can be used for data modeling. However, none of the existing frameworks is perfect. That is why it is thought that presenting a new approach based on UML and Merise would help having a good database design just by applying little effort and avoiding the drawbacks of each technique.

Although UML is the methodology that is widely used, it, definitely, has some disadvantages that make its usage tedious to some extent. UML is very complex with more than 13 diagrams and more than 100 types of classes [1]. This makes it hard to adopt and even harder to master. UML is also time consuming. It takes a lot of time to manage and maintain UML diagrams [2]. On top of that, software developers do not benefit from UML diagrams as much as you would hope, because they work with code and programs rather than pictures and diagrams. UML is rather beneficial for project managers that are concerned with the way the software tool would work [3].

Merise, on the other hand, is not as widely used. It also has a set of advantages and several disadvantages. Merise does a great job with the modeling and the conception of small databases. But, when designing large databases, it may not be the best methodology to opt for. Also, it is limited to the 3rd normal form [4]. In addition to that, it is best suited to work with modeling sequential tasks and does not deliver a good result when dealing with distributed ones. It is not meant to model semantic data.

The new approach comes to circumvent the disadvantages mentioned above for both methodologies through creating a new process to model the system in general and databases in particular. This integration of both techniques has less limitation than each methodology when applied by its own.

The rest of the paper is organized as follows: Section II presents the work that has been previously done in the same field. Section III consists of a comparison of UML and Merise. The description of the work done is presented in Section IV. Finally, Section V describes the suggested final process to follow.

II. Related Work

Knowing that MERISE is a methodology that is mainly used in France and that is being adopted in European engineering community more than other communities, the author tries in [5] to make the methodology more suitable for English speaking users. However, this work does a perfect job in trying to spread MERISE in English speaking community, by somehow translating the existing elements of the methodology, but doesn't in any way try to hide the disadvantages and limitations of the methodology itself.

In [6], the author presents the different concepts of UML as an object-oriented modeling language. These concepts definitely have many problems and limitations, which actually don't exist in the first methodology. But UML does have advantages that, in contrast, don't exist in MERISE. Hence, comes the idea of combining the two approaches by integrating the diagrams from each to satisfy the user's need in different scenarios. In the next sections, the paper will present how the integration is to be devised.

III. Unified Modeling Language vs Merise

Before getting on with the integrated approach, it is necessary to look at the comparisons between UML and Merise that have been done in the literature.

UML and Merise are not completely similar. Each one has a different concept. UML, for example, takes care of the object-oriented modeling, while Merise works best for relational databases. Even though UML is more widely used

than Merise, both methodologies can nonetheless be used in modeling and conceiving databases.

On the one hand, Merise is said to constitute a real methodology that respects the standards. Earlier, in 2003, Merise was divided into three main pillars: steps to follow, formalism, and organization. However, some of these aspects did not survive in front of the advancement of technology and needs of the recent applications. The "steps to follow" for example, is no longer needed in order to have a good methodology, while the importance of "formalism" persists.

On the other hand, methodologists claim that UML presents a very good formalism with a high level of standardization, but it is lacking the process to follow in addition to the organization to be a real methodology. Besides, Merise works best with organizational information systems while UML is designed for object-oriented based information systems. That is why the two methods actually complement each other and can be used at the same time.

The purpose of this paper is to combine these two methodologies and to prove that they can together be leveraged in the modeling of the same project.

IV. Description of the Work Done

It goes without saying that the order of the UML diagrams to be used is not fixed as it depends on the type of the application and the style of the designer or developer.

In this paper, an attempt to unify the process of software design is made, by making all the steps standardized and clear.

In the first place, the classes that constitute the system are identified then the actors of the application are looked at. Right after that, the exchanged messages between the actors of the system are studied, their sequence as well as the order in which these messages appear. Then, further light is shed on the set of activities that are performed within the application.

At each step, a comparison of the diagrams used in each methodology is lead, and then the assessment.

The figure below (Fig. 1) summarizes the process as described above.

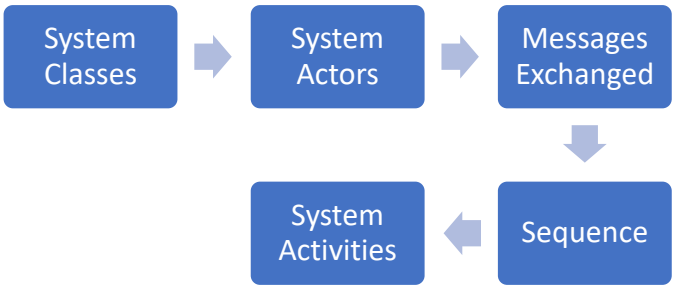

Fig. 1. Software design process.

A. UML Class Diagram vs Conceptual Model of Merise

1) UML Class Diagram

a) Definition

It provides a general overview of the final system by describing the classes involved in the system and by explaining the relationships between them. It allows the users to go from domain specific data structures to a detailed design of the final product. The main components of the class diagram are [7]:

Class: grouping of objects with the same characteristics

Method: part of a class that shows the behaviors of a certain object of that class

Attribute: part of a class that represents the static properties of an object of the same class

Multiplicity: indicates that one of the related classes refers to the other and it can take many values.

Relationship: represent the logical relationship between classes. There are many types of relationships in the class diagram of UML.

Object: instances of a specific class

Access Level: data privacy is determined by assigning an access level to it

b) Example

Fig. 2 shows the different components mentioned in the previous section.

Fig. 2. UML class diagram.

As shown in the figure above, it represents a simple class diagram that consists of two classes: Person, and Address. Each class has attributes (e.g. Name, Address, …) and operations or methods (in this case only the class Person has a method that is called BuyCar()). The multiplicities in this example mean that an address is associated to one person maximum, while a person does necessarily have one address.

2) Merise Conceptual Model

a) Definition

At this level of the modeling process, the *entity/relationship schema* is used in Merise. A typical entity/relationship diagram would contain two main components as its name suggests: *entity* and *relationship*.

An entity, short for entity type, can be compared to a class in the context of UML, but it only contains *properties (attributes)*. In general, an entity can be defined independently of the rest of the data and corresponds to one row in a database table [8], [9].

In an E-R diagram, the entity needs to have a unique identifier which could be one or a set of properties (e.g. orderid + date to characteristic of an order).

The *relationship* (or association) links together one or more entities and can itself contain additional attributes.

b) Example

Fig. 3 shows a graphical representation of the E-R diagram.

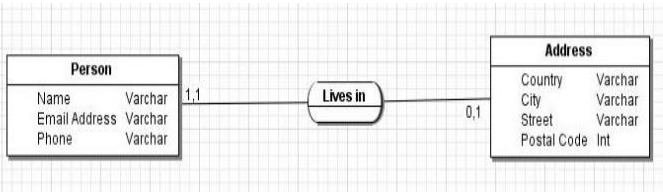

Fig. 3. Merise conceptual diagram.

In the example above, the same classes as in the previous UML class diagram are kept to show the differences that exist between the two modeling techniques when dealing with system classes. The first difference that pops up is in the multiplicities. They still mean the same in both diagrams, but they inversed. Another difference resides in the fact that associations in Merise can have attributes, thing which does not exist in UML.

3) What to use and why

If a translation from the conceptual model to the class diagram of Merise was to be done, nothing much would be done: in fact, each relationship will be transformed to an association, each entity will be a class and relationships with attributes will be transformed into a class association with the same attributes.

Although the differences are not that big, it is recommended to use the UML class diagram for the following reasons:

- The multiplicities in the class diagram are more intuitive and make more sense to the designers that are new to the domain. It is easier to understand that a tutor has a program rather than a program is owned by a tutor.
- The class diagram gives a better illustration and overview of the system because it presents not only the attributes of the objects but also their data types in addition to behaviors and their return data types.
- UML class diagram is closer to the implementation as it lets you think about the code and the things to be implemented in the coding phase. This saves a huge amount of time in the implementation.
- The UML class diagram is more for object-oriented languages (java, visual basic, .net …), and the object-oriented paradigm is gaining a lot of popularity among programmers these days.

B. UML use Case Diagram vs Conceptual Model for Communication of Merise

1) UML Use Case Diagram

a) Definition

Use case diagrams give a general overview of the usage requirements of the final system [10]. They are mainly used to represent the stakeholders of the entire project. It is also helpful in the deployment phase as programmers find it easy to go from actual use cases from the diagram to functions in the system. The Use Case diagram consists of the following components:

Use cases: they are horizontal ellipses that represent the sequence of actions that are done by a user and that would be of additional value to them.

Actors: The main users of the system, they can be humans or external entities (operating systems).

Associations: They represent the relationship between the actors and the use cases, between use cases (include, extend), or even between users (inheritance).

System Boundary: Represented by a rectangle drawn around the use cases. Their main goal is to delimit the scope of the project.

Packages: Packages are totally optional. They are used to group use cases of the same type together allowing for a better organization of the entire diagram [11].

b) Example

Fig. 4 is a simple use case diagram that shows the different actions performed by the student and professor in a university.

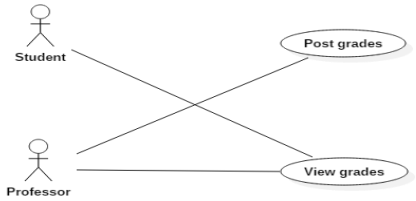

Fig. 4. UML use case diagram.

Concerning the example in the previous figure, it presents two main actions that are performed within a university by the professor and the student. It can be inferred from the diagram that the professor posts and views the grades while the student can only view the grades.

2) Conceptual Model for Communication

a) Definition

This model is complementary to what is called “Context Diagram”. The “Context Diagram” shows the external entities that interact with the system to be designed [12]. The Conceptual Model for Communication completes this diagram in the sense that it decomposes the system into many internal actors who exchanges messages between them.

Graphically, the actor is represented by an ellipse whereas the messages are represented by arrows [13].

b) Example

In Fig. 5, the organization is composed of 2 internal actors who are the professor and the student, and they are interacting with the system through performing two main actions that are Post Grades and View Grades. The actions done by each actor can be inferred just like in the previous diagram.

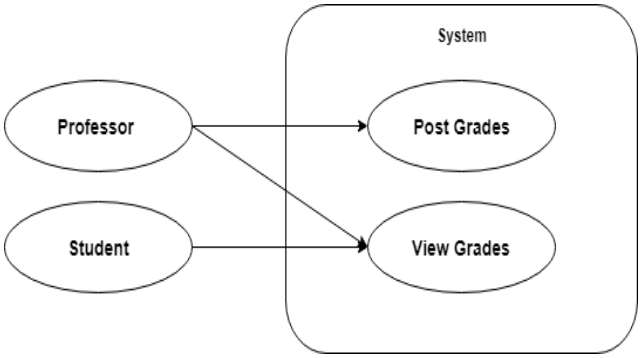

Fig. 5. Conceptual model for communication.

3) What to use and why

Both the use case diagram and the conceptual model for communication show the set of internal actors that exist in the system and the actions that are performed by those.

It is recommended to use the conceptual model for communication if there is an extensive interaction not only between the internal actors, but also between those and other external entities. The conceptual model for communication explicitly shows the interaction between the entities. This implies that even if the system in question has a significant number of entities that are communicating with each other, it can easily be represented in a nice and readable diagram.

However, if the focus needs to be done on each individual actor (to limit the privileges and describe them), then it would be more suitable to use the use case diagram. This latter focuses on the user rather than the actions done. It takes a better care of the privileges given to each actor which consequently affects the actions to be performed by that actor.

C. UML Sequence Diagram vs Merise Data Flow Diagram

1) UML Sequence Diagram

a) Definition

Sequence Diagram is a high-level interaction diagram that shows how operations are carried out between the different parts that exist in the system [14]. Graphically, the messages exchanged during the interactions are ordered vertically in an increasing chronological order. The vertical line that represents time is called the *lifeline*. It extends as long as the life of the actor in question within the system. The horizontal axis shows the different objects involved in the interactions the diagram shows. Each of those objects is called a participant and has its own lifetime [15].

b) Example

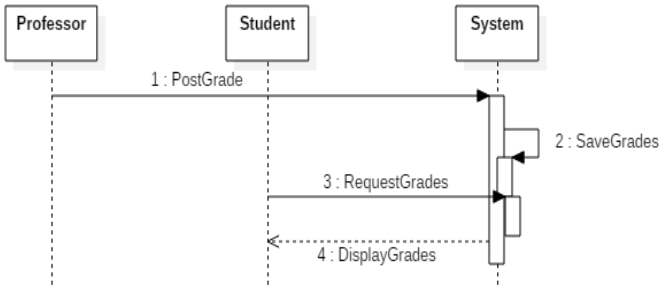

Fig. 6. UML sequence diagram.

The example above (Fig. 6) has the same actors as the use case diagram. The sequence diagram shows the messages that are exchanged between the two actors and the system. The professor posts the students' grades to the system which saving them later to the database. To view their grades, the students request the grades from the database which then replies by displaying the grades.

The messages shown in the diagram follow a chronological order, meaning that the first message sent is displayed in the top of the diagram and has the ID number 1 and so on.

2) Merise DataFlow Diagram

a) Definition

This diagram shows which activities are related to each other and how they are involved in solving the problem stated [16]. At this stage, this diagram is done without taking into consideration the actual behavior of the system (scheduling, synchronization …). It shows the activities and relationships between them in a non-sequenced manner [17].

b) Example

Fig. 7 shows an example of a Merise Data Flow Diagram.

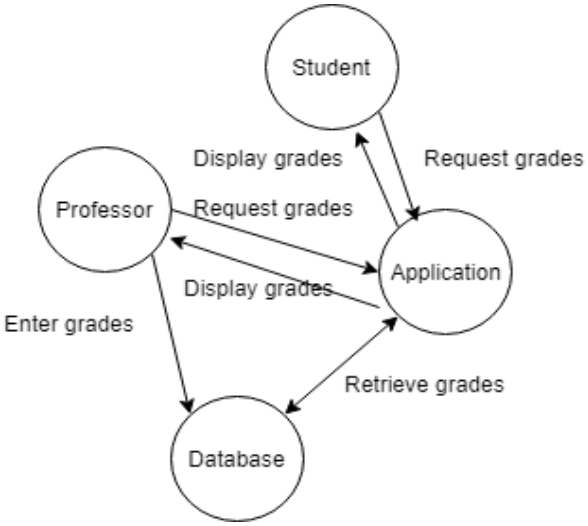

Fig. 7. Merise data flow diagram.

3) What to use and why

It is clear that both diagrams are about exchanging messages. The main and obvious difference would be that UML's sequence diagram looks more structured and organized because it takes into consideration the time and it shows the activities in a chronological order.

However, it can be observed that the use of the data flow diagram brought by Merise will do a better job in giving a general view about the communication of the objects within the system:

- The sequence diagram somehow gives an idealistic representation of the messages exchanged between instances, while the view given by the data flow diagram is more realistic.
- In the data flow diagram, there is no need to follow a specific order the thing that allows for a certain level of flexibility. This way, the user will be able to see different scenarios and choose a specific instance to initiate the scenario.
- The data flow diagram is easy to master with few symbols and notations compared to the complex UML sequence diagram. Plus, it is more intuitive and easy to explain to project managers or clients who, not necessarily have a computer science background.

D. UML Collaboration Diagram vs Merise Dataflow Diagram of Merise

1) UML Collaboration Diagram

a) Definition

The collaboration diagram is similar to the sequence diagram. The difference is that the collaboration diagram is object-centered whereas the sequence diagram is time-oriented [18], [19].

b) Example

Fig. 8 shows a simple example of a UML collaboration diagram.

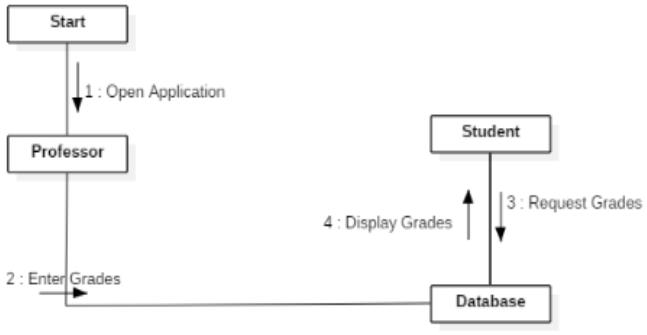

Fig. 8. UML collaboration diagram.

2) What to use and why?

In this step of the modeling, no Merise diagram is introduced. However, one can opt for an intermediate solution in this case. It is recommended to use the UML collaboration diagram, and if not applicable (for the specifications of the application in question), simply replace the objects by the usual entities used in Merise.

That is because:

- The collaboration diagram shows more details about the messages between objects/entities.
- There might be a chronological order introduced to the diagram.
- Actors can be included in the diagram
- It gives a clear and structured overview of the system in a later step of the design

E. UML Activity Diagram vs Merise MCT

1) UML Activity Diagram

a) Definition

The activity diagram consists of activities, states and transitions between those. It shows how activities coordinate to achieve and provide certain services and defines the main events of the system needed to make a given service, and how those events relate to each other.

It is an advanced flowchart that combines other details such as the actors, the starting point, and the finishing point of the system.

In addition to that, it captures the dynamic flow of the system [20], [21].

b) Example

Fig. 9 illustrates an example of UML's activity diagram.

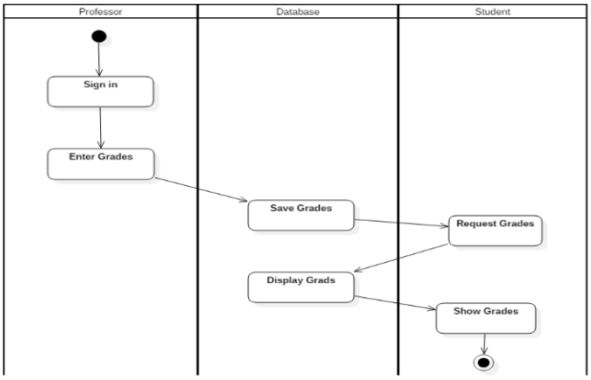

Fig. 9. UML activity diagram.

2) Merise MCT

a) Definition

The conceptual model of treatment is one of the most famous diagrams in Merise. It allows for the treatment of the dynamics of the information system meaning the event-driven operations that are carried out within the system.

This diagram helps then to model the activities of the system using clear schemas. It simply defines what should be done without giving any idea about how, when or where.

The components below describe the MCT diagram [22]:

Process: A subset of the enterprise activities. This means that the entity uses many processes within the same activity.

Operation: Is a set of actions executed after an event or a conjunction of events.

Event: An event represents the change in the external universe of the information system or in information system itself.

b) Example

The example showed in Fig. 10 presents a Merise conceptual treatment model.

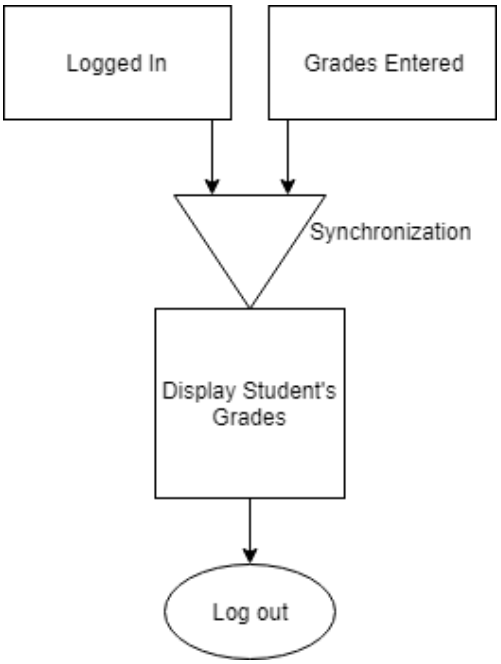

Fig. 10. Merise MCT.

3) What to use and why?

Both the activity diagram of UML and MCT of Merise can be used in the same stage of the conception and design phase. They both show the flow of activities in the system to be conceived. But, UML's activity diagram tends to be more powerful for the following reason:

- The separation between the actors of the system will be of a great help in trying to really understand the system. It makes it clear for the user which activity is done by which actor.
- The MCT provides the use of some rules that when added may result in increasing the complexity of the diagram.
- The MCT does not clearly identify the initial and final events while it is really important to state when the flow of activities starts and when it ends.

V. Entire Process to Follow and its Limitations

Now that all the recommendations regarding the usage of UML and Merise diagrams were given, it is time to discuss the entire process to follow.

This is done through providing the steps of the entire process that is suggested in this paper. Fig. 11 summarizes the suggested process.

As mentioned previously in this paper and as shown in the diagram, It is recommended to start either with UML use case diagram or the Merise conceptual model for communication. Then, go for UML class diagram. For the following step, it is suggested to opt for Merise dataflow diagram. As for the forth step, UML activity diagram is said to do a better job, then end up with an integrated approach that combines collaboration and dataflow diagrams.

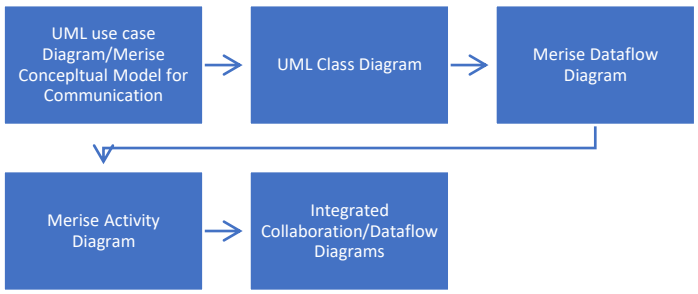

Fig. 11. Entire process to follow.

In the process of comparing Merise and UML diagrams, many challenges were faced. The first one was to find common aspects and features in both methodologies. This required closely looking at the applications of each one seperatly. The second challenge was mainly about keeping the required functionalities at each stage of the design process. The last challenge resides in keeping the process as efficient as it initially is while modifying the diagrams used.

The limitations of this new approach reside in the fact that it has not been based on experiments. It is based on studying each step of the design process along with the diagrams associated with it. Besides, this new approach may not suit all types of projects and all communities. People used to Merise digrams will have hard time merging it with another approach, and the same goes for software designers who are more of UML users.

VI. Conclusion and Future Work

UML and Merise are two methodologies that are used by software designers. Unlike what many practitioners in the domain think, these two modeling methods are not very different from each other. UML and Merise complement each other in a way allowing for their integration which means that they can be used at the same time with the same application that is why software designers always have problems choosing the framework to use.

This paper came up with an approach to unify UML and MERISE approaches in a way that reduces the drawbacks and takes advantages of each one of them.

This new approach aims at helping software designers by simplifying the design process and making it smooth.

As future work, this new approach needs to be implemented in a real-world project and tested in terms of performance. A good way of doing that could be by testing each approach, MERISE and UML, and comparing it to the one suggested in this paper. These tests based are to be performed on a real-life project that is complex to enough in order to push these methodologies along with the new approach to the limit, and hence be a proof of concept. An interesting metric to measure this performance would be the number of iterations done in each methodology before getting the appropriate model to implement. Besides, the feedback of the software designers will also be valuable in assessing the performance of each technique.

REFERENCES

[1] D. E. Avison. (1991). MERISE: A European methodology for developing information systems. Available online at: https://link.springer.com/article/10.1057/ejis.1991.33

[2] Bernd Bruegge and Allen H. Dutoit (1999). Object-Oriented Software Engineering Using UML, Patterns and Java. Available online at: http://dbmanagement.info/Books/MIX/POO_Software_Engineering_Using_UML_Patterns_and_Java_3rd_Edition.pdf

[3] Keng Siau, Qing Cao (2003). How complex is the Unified Modeling Language? Available online at: https://dl.acm.org/citation.cfm?id=960145

[4] Tom Mens, Ranghild Van Der Straeten, and Jocelyn Simmonds. Maintaining Consistency between UML Models with Description Logic Tools. Available online at: http://www.cs.toronto.edu/~jsimmond/docs/mcc/TomMensEtAl.pdf

[5] Murray Cantor (1998). Object-Oriented Project Management with UML. Available online at: https://www.wiley.com/en-us/Object+Oriented+Project+Management+with+UML-p-9780471253037

[6] Third Normal Form (3NF). Available at: https://ww.1keydata.com/database-normalization/third-normal-form-3nf.php

[7] The Class Diagram. Available at: https://www.ibm.com/developerworks/rational/library/content/RationalEdge/se04/bell/

[8] UML 2 Class Diagrams: An Agile Introduction (2014). Available at: http://ww.agilemodeling.com/artifacts/classDiagram.htm

[9] Yann Thierry-Mieg (2007). Database Design & Modeling: Entity / Relationship. Available at: https://pages.lip6.fr/Yann.Thierry-Mieg/old/EFREI-DBMS/07-Design-E-R.pdf

[10] Margaret Rouse (2014). Use Case Diagram. Available at: http://whatis.techtarget.com/definition/use-case-diagram

[11] Edwin Obenauf (2017). UML use case diagram example. Available ate: http://studiootb.com/uml-use-case-diagram/uml-use-case-diagram-uml-example-adorable-photo-what/

[12] Chris Adams (2016). What is a system diagram and what are the benefits of creating one. Available at: http://www.modernanalyst.com/Careers/InterviewQuestions/tabid/128/ID/1433/What-is-a-Context-Diagram-and-what-are-the-benefits-of-creating-one.aspx

[13] Sabah Al-Fedaghi, Ala's Alsaqa, Zahra'a Fadel (2009). Conceptual Model for Communication. Available online at: https://arxiv.org/ftp/arxiv/papers/0912/0912.0599.pdf

[14] Donald Bell (2004). The Sequence Diagram. Available at: https://www.ibm.com/developerworks/rational/library/3101.html

[15] Sequence Diagram. Available at: https://en.wikipedia.org/wiki/Sequence_diagram

[16] What is a Data Flow Diagram? Available at: https://www.lucidchart.com/pages/data-flow-diagram

[17] Data flow diagram. Available at: https://www.smartdraw.com/data-flow-diagram/

[18] Margaret Rouse. Collaboration Diagram. Available at: http://searchsoftwarequality.techtarget.com/definition/collaboration-diagram

[19] UML 2 Communication Diagramming Guidelines. Available at: http://agilemodeling.com/style/collaborationDiagram.htm

[20] UML – Activity Diagrams. Available at: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm

[21] Activity Diagram for Inventory Management System (UML). Available at: https://www.lucidchart.com/pages/activity-diagram-for-inventory-management-system-UML

[22] Merise (data-processing). Available at: http://wikipedia.qwika.com/fr2en/Merise_(informatique)