## Analysis Methods

This involves first of all the collection of information on the field passing through the need and flow of information within a particular information system in order to design software closely to the users’ needs.

W¶¶wwwwwwwqqe will step by step look through or present the different types of methods under the large set: The object oriented methods and the functional methods. At the end we will precise the best method suitable in the realization of our system by giving the raisons d’être of our choice.

## 3.10 Object Oriented Methods

The Object oriented methods (OOMs) describe the static structure of the objects, their classes and their relations. ¶One can mention here the following OOMs: ¶OMT method, UML method and UP.

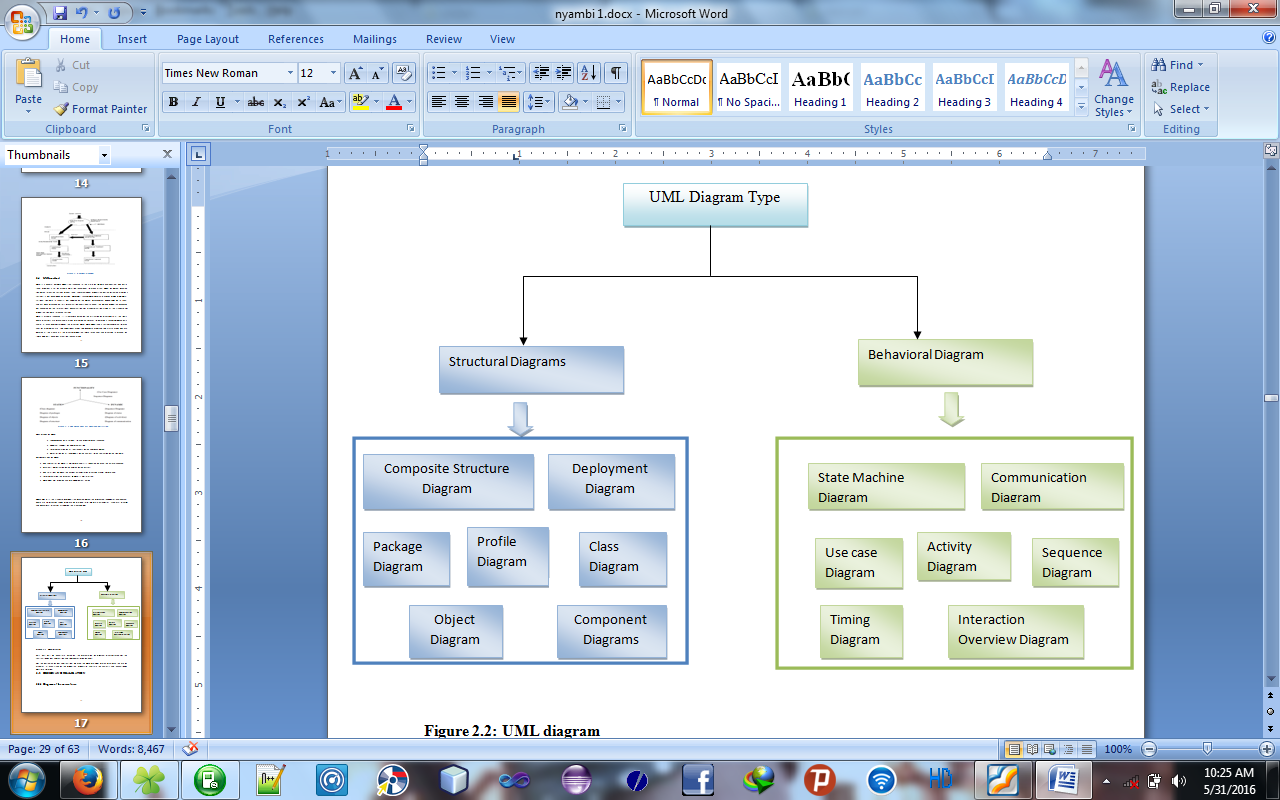
### 3.10.1 OMT Method

The Object Modelling Technique (OMT) is an object modelling method for software modelling and designing. It was developed around 1991 by Rumbaugh, Blaha, Premerlani, Eddy and Lorensen as a method to develop object-oriented systems and to support object-oriented programming (ESPINASSE, 1980). OMT was developed as an approach to software development. The purposes of this modelling according to Rumbaugh are:

* Testing physical entities before building them (simulation);
* Communication with customers;
* Conception (alternative presentation of information);
* Reduction of complexity

### 3.10.2 Unified Modelling Language (UML) Method

UML is a language of modelling unified object in an object oriented environment developed in response to the call for the proposal launched by the Object Management Group (OMG) with the goal of defining the standard notation for the modelling of applications built using objects. ¶The principal authors of UML are Grady Booch, Ivvar Jacobson and Jim Rumbaugh.



*Figure xxxxxx: Overview of UML (static3.creately.com, 2016)*

Some advantages of UML ¶ are:

* Formal and standardized language, it allows proceeds of precision and constitutes a pledge of stability. This is what encourages the use of the tools;¶
* Powerful support of communication;¶
* Implementation of all the richness of the object approach;¶
* Description of all the models from the analysis to the realization of the software;¶
* Standardization of the concepts objects.¶

Some limits of UML¶ are:

* The semantics of UML is not formalized.¶ It is specified by using the natural language;¶
* Difficult optimization of the choice of the classes;¶
* Various categories of diagrams are not formalized;¶

### 3.10.3 Unified Process (UP) Method

Unified Process (UP) is a management method in the life cycle of software development and thus for object-oriented software. This is a generic method, iterative and incremental unlike the sequential method MERISE or SADT. This method is the general precept methods with the abbreviations: RUP, UPA XUP, EUP, 2TUP, AM, DCU. Thus, an embodiment according to UP, to transform the software needs of users, must necessarily have the following characteristics:

* UP is based on components
* UP uses UML
* UP is driven by use cases
* UP centric architecture
* UP is iterative and incremental.

Some advantages of UP¶ are:

* Use case sensitive
* Architecture centric
* Iterative and incremental.

Some limits of UP are:

* It is used only at the beginning of the whole process to create business requirements.
* The final application reflects the businesses processes, but there exist no closer bond between them.
* A small change in the business process leads to a fundamental change of the created information system.

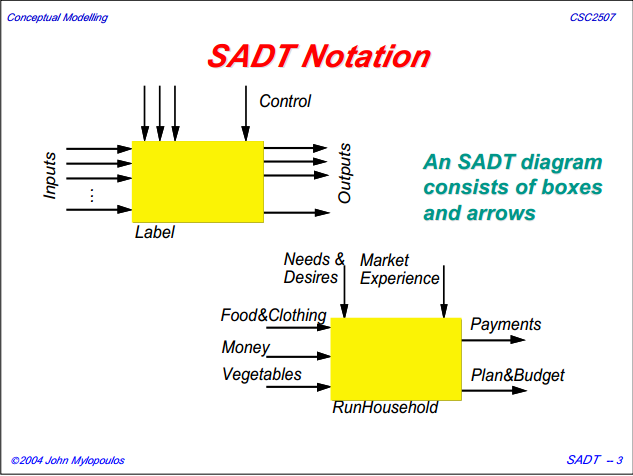
## 3.11 Functional Methods

The functional methods have their origin in the development of the procedural languages. ¶More directed towards the managements than towards the data, they highlight the functions to be ensured and propose a hierarchical, downward and modular approach by specifying the bonds between the various modules. ¶With the evolution of systems and programming languages, these methods took into account the modelling of the data and the problems arising from real time.¶

### 3.11.1 SADT Method

Structured Analysis and Design Technique ( ¶SADT) Method is a method of American origin developed in 1977 by DOUG ROSS then introduced in Europe since 1982 by Michel GALINER. ¶It is a multi-field language which supports the communication between users and originators.

¶As a method of functional analysis and the most known management of projects, SADT presents strong points and weak points.

¶SomeSome advantages of SADT Method¶ are:

Its simplicity ¶

* Its adequacy to capture the user's needs ¶
* Its capacity with being able to produce solutions on several levels of abstraction.¶

Some limits of SADT Method:¶ are:

* Its analysis is concentrated much on the functions, the coherence of the data being neglected ¶
* The rules of decomposition are not explicit.¶ The decomposition differs according to analysts ¶
* Its difficulties of taking account of the non-hierarchical interactions in the complex systems ¶
* Lastly, the volatility of the functions makes that the system is in perpetual D-design ¶

### 3.11.2 The MERISE Method

The MERISE (Methode d’Etude et de Realisation Informatique pour les Systems d’Entreprise) method was launched around 1977 through a national consultation launched by the French Ministry of industry with the aim to create a company of data processing consultant in order to define a method of design of information system. ¶The Merise method is based on separation of data and treatments to be carried out in several conceptual and physical models.

¶The Merise method recommends three levels of abstraction; ¶the conceptual level, the organizational level and the physical level.¶

**The conceptual level:** ¶The conceptual level defines the finalities of the company. ¶It is on this level that objectives to reach and constraints which weigh on the company are identified. ¶It generally constitutes the most stable level and the first level of development. ¶At the conceptual level, one distinguishes the Conceptual Data Model (CDM) and the Conceptual Treatments Model (CTM).¶

**¶The organizational or logical level:** ¶The organizational level describes the organization which it is desirable to be set up in the company to achieve the laid down objectives. ¶The purpose of it is to provide a diagrammatic representation of the organization of the company. ¶One has heard of the Logical Data Model (LDM) and the Organizational Treatments Model (OTM). ¶The organizational level is less stable and constitutes the second level of invariance.¶

**The physical level:** ¶The physical level describes the means which will be implemented to manage the data and to activate the treatments. ¶It is organized around the Physical Data Model (PDM) and the Operational Treatments Model (OTM). Table 1 below represents the three levels:

**¶**

Table 1: Representation of the levels of perception of Merise

|  |  |  |
| --- | --- | --- |
| **Levels** | **Static (Data)** | **Dynamic (Treatments)** |
| **Conceptual** | CDM | CTM |
| **Physical** | PDM | OpTM |
| **Logical and Organizational** | DLM | OTM |

Some advantages of MERISE Method are:

* ¶Merise is considered like a method of design of information systems on the plan of its general organization. ¶For this reason, it has many advantages:¶
* ¶Merise allows the comprehension and the formalization of the needs for the trade ¶
* Merise supports the dialogue between originator and owner, building particularly in the projects of integrated system development of management.¶
* ¶Merise allows the general modelling of the data for construction of a database.¶
* ¶Merise ensures the formalization of the user's needs within the framework of a schedule of conditions, before the work of design.¶

Some limitations of Merise Method are:

¶In spite of its many advantages, the Merise method was often criticized as being a Franco-French historical method. ¶Its disadvantages can be analyzed around three points:¶

* ¶Merise is more turned towards the engineering of general design than towards the software genius ¶
* Difficulty in maintaining the system.¶
* Not easily evolutionary system.¶¶

¶

### Choice of the method

MERISE is considered a method of design of IS (Information System) of organizations, more turned towards the comprehension and formalization of the needs of trade towards the realization of software. In this sense MERISE claims more engineering of IS trades than software genius. MERISE is ideal for:

* *Modeling of data for construction of a relational database.*
* *Modeling processes trades of IS automated partly by software.*
* *The formalization of the user needs within the framework of schedule of user conditions, for the design of an adapted software.*

To design our information system, we will use the MERISE method.¶ The choice of this method is based on the fact that MERISE ensures the separation of data and treatments to be carried out in several conceptual and physical models, which ensures the longevity of the system.¶ In addition, MERISE method allows the comprehension and the formalization of the needs for the trade. ¶It supports also the dialogue between originator and owner building, particularly in the development of the integrated management system.¶ Lastly, the MERISE method allows the general modeling of the data for the design of a data base.¶

### 3.3.4 Various models of MERISE

#### 

#### 3.3.4.1 The Flow diagram

The flow diagram also called Conceptual Model of the Communication makes it possible to supplement the diagram of context by breaking up the organization of the system into a series of internal actors or domains. ¶In this diagram, information are represented by arrows whose orientation indicates the direction of the flow of information

#### 

#### 3.3.4.2 Data dictionary

¶After the analysis of the various entities, the data dictionary has to be established. ¶It is about a table in which all the data are listed with precise details: ¶N°, reference, significance, type and size. ¶All these data result from the information collected during the interview and the information retrieved.¶ The dictionary below represents these data.

Table 2: Representation of the data dictionary¶

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N°** | **Reference** | **Significance** | **Type** | **Size** |
| 1 | x\_id | Identifier | int | 50 |
| 2 | ads\_id | Advert identification | vachar | 50 |
| 3 | ads\_name | Advert name | vachar | 255 |
| 4 | ads\_price | Advert price | vachar | 50 |
| 5 | ads\_comment | Advert description | varchar | 2000 |
| 6 | ads\_status | Advert status | varchar | 50 |
| 7 | ads\_view | If an advert is viewed or not | int | 20 |
| 8 | ads\_pubDate | Advert publication date | timestamp | Current timestamp |
| 9 | Folder time | The time the advert folder is created | varchar | 50 |
| 10 | ads\_id1 | Advert’s photo identifier | varchar | 50 |
| 11 | ads\_photo | Advert’s photo | varchar | 50 |
| 12 | cat\_id | Category identifier | int | 20 |
| 13 | cat\_name | Category name | varchar | 50 |
| 14 | cat\_icon | Category icon | varchar | 30 |
| 15 | cat\_alt | Category alt | varchar | 10 |
| 16 | cat\_url | Category uniform resource locator | varchar | 30 |
| 17 | cat\_ads\_number | Category advert number | int | 30 |
| 18 | Id | Command identifier | int | 10 |
| 19 | client\_name | Commanding client’s name | varchar | 50 |
| 20 | client\_tel | Commanding client’s telephone number | varchar | 20 |
| 21 | client\_email | Commanding client’s email | varchar | 50 |
| 22 | payment\_type | Type of payments solicited by client | varchar | 50 |
| 23 | command\_code | Command’s code | varchar | 50 |
| 24 | command\_date | Command’s date | varchar | 50 |
| 25 | command\_price | Command’s price | int | 10 |
| 26 | cm\_id | Craftsman id | varchar | 128 |
| 27 | cm\_title | Craftsman title | varchar | 10 |
| 28 | cm\_fmane | Craftsman first name | varchar | 50 |
| 29 | cm\_lname | Craftsman last name | varchar | 50 |
| 30 | cm\_email | Craftsman email | varchar | 60 |
| 31 | cm\_region | Craftsman region | varchar | 50 |
| 32 | cm\_city | Craftsman city | varchar | 50 |
| 33 | cm\_regDate | Craftsman registration date | timestamp | Current timestamp |
| 34 | cm\_phone1 | Craftsman first phone number | int | 20 |
| 35 | cm\_phone2 | Craftsman second phone number | int | 20 |
| 36 | cm\_address | Craftsman address | varchar | 50 |
| 37 | cm\_category | Craftsman category | varchar | 50 |
| 38 | msg\_id | Message identifier | int | 20 |
| 39 | msg\_sender | Message sender | varchar | 50 |
| 40 | msg\_sub | Message subject | varchar | 2000 |
| 41 | msg\_email | The email to which the message goes to | varchar | 50 |
| 42 | msg\_tel | The telephone number to which the message goes to | int | 20 |
| 43 | msg\_cont | Message content | varchar | 5000 |
| 44 | msg\_date | Message date | timestamp | Current timestamp |
| 45 | pc\_id | Page configuration identifier | int | 50 |
| 46 | pc\_name | Page configurarion name | varchar | 50 |
| 47 | pc\_content | Page configuration content | varchar | 1000 |
| 48 | reg\_id | Region identifier | int | 20 |
| 49 | Reg\_name | Region name | varchar | 50 |
| 50 | Id | Sms\_request identifier | int | 5 |
| 51 | Keyword | Sms keyword | varchar | 100 |
| 52 | Msg | message | varchar | 500 |
| 53 | sp\_id | Support identifier | int | 2 |
| 54 | sp\_name | Support identifier | varchar | 30 |
| 55 | sp\_number1 | Support identifier first phone | int | 20 |
| 56 | sp\_number2 | Support identifier second phone | varchar | 20 |
| 57 | ads\_id | Advert’s tracking identifier | varchar | 100 |
| 58 | track\_status | Tracking status | varchar | 255 |
| 59 | Location | location | varchar | 255 |
| 60 | Comment | Comment about tracking | varchar | 255 |
| 61 | Date | Tracking date | timestamp | Current timestamp |
| 62 | user\_id | User identifier | int | 20 |
| 63 | user\_title | User title | varchar | 50 |
| 64 | user\_fname | User first name | varchar | 50 |
| 65 | user\_lname | User last name | varchar | 30 |
| 66 | user\_username | User username | varchar | 50 |
| 67 | user\_password | User password | varchar | 30 |
| 68 | temp\_pass |  | varchar | 50 |
| 69 | user\_email | User email | varchar | 50 |
| 70 | user\_region | User region | varchar | 60 |
| 71 | user\_city | User city | varchar | 50 |
| 72 | user\_regDate | User registration date | timestamp | Current timestamp |
| 73 | user\_verCode | User verification code | varchar | 50 |
| 74 | user\_phone1 | User first phone number | int | 20 |
| 75 | user\_phone2 | User second phone number | int | 20 |
| 76 | user\_address | User address | varchar | 50 |
| 77 | user\_sec\_question | User security question | varchar | 50 |
| 78 | user\_sec\_answer | User security answer | varchar | 50 |

#### 3.3.4.3 Conceptual Data Model (DCM)

¶The conceptual data model is a static representation of the information system of the application which highlights its semantics. ¶The purpose of it is to write in a formal way the data which will be used by the information system. ¶It is thus, the representation of data to ease comprehension. ¶This aspect covers the words which describe the system as well as the bonds which exist between these words. ¶The formalism adopted by the MERISE method to carry out this description is based on the concepts "entity-association".¶

¶The rules of design of the conceptual data model make it possible to lead to a standard chart which eliminates the redundancies and the ambiguities. ¶But before conceiving our MCD, we will review some basic concepts of MERISE method.

#### 3.3.4.4. The entity or object¶

¶**¶**An entity is the representation of a material or immaterial element having a role in the system which one wishes to describe. ¶Each entity is made up of properties making it possible to describe it. ¶An entity is represented by a rectangle separated in two fields: ¶The top field contains the wording and the field of bottom contains the list of property of this entity. For example, if we consider our e-gallery application we wish to realize in this work, we will have the entities as represented below

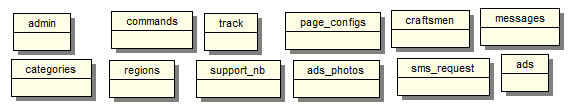


Figure 2: List of the entities

#### 3.3.4.5The identifier

¶¶It is a property which makes it possible to identify in a single way an object. ¶It can also be defined as a particular property of an object such as there do not exist two occurrences of this object for which this object could take the same value. ¶The attributes of a class of entity making it possible to indicate in a single way each authority of this entity are called absolute identifier. ¶The conceptual data model proposes to underline the identifiers. ¶Thus, each class of entity must have at least an identifying attribute, and the whole of these attributes identifiers must be indicated with the creation of the entity. ¶If we consider the case of our e-gallery platform, we will have identifiers as represented below.

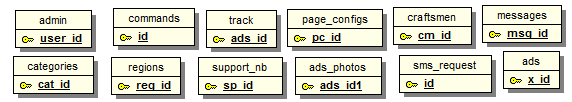
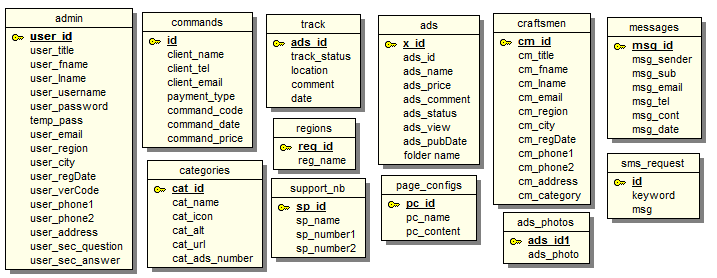


Figure 3: Identifiers of the model

#### 3.3.4.6 The properties (attributes)

¶¶A property or attribute is elementary information that is non-deductible from other information, which presents an interest for the studied field. ¶Each value taken by a property is called occurrence. ¶¶For example if we consider the e-gallery platform, we will have the following entities and their properties:



**Figure 4: List of attributes**

#### 3.3.4.7The occurrence of entity

According to the definition of an entity, we know that the knowledge of a value of the identifying heading determines the knowledge of the values of the other headings. The whole of these values is called occurrence of entity. An occurrence is thus a value taken by a property. The following figure presents the occurrences of entity faults.

Table 3: Representation of the occurrences of the entity “commands”

|  |
| --- |
| **COMMANDS** |
| 10 |
| 30 |
| 128 |

#### ¶3.3.4.8The Association

An association (also called relation) is a semantic bond between two or more entities. ¶It is derived from clear existence of the relationship between the given entities and it can carry one or many attributes. ¶¶An association which connects the same class of entity is known as recursive. ¶That which connects two entities is called binary association. ¶When association connects three classes of entity, it becomes a ternary association and finally an association which connects N classes of entities is an n-ary association. ¶One can possibly add properties to associations. ¶If we take for example the e-gallery platform, one can add association “place” between the entities commands and ads to mean a customer can place a command for one or many adverts (works of arts). ¶The figure below illustrates this association clearly.¶¶

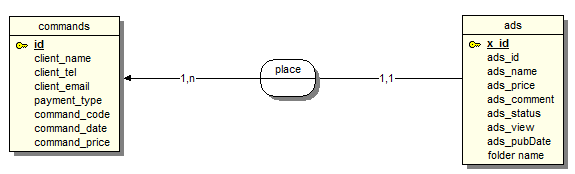


Figure 5: Example of association

#### 3.3.4.9 The cardinality

¶¶The cardinalities make it possible to characterize the bonds which exist between an entity and the relation to which it is connected. ¶The cardinality of a relation is composed of a couple comprising a maximum terminal and a minimal terminal, interval in which the cardinality of an entity can take its value. ¶The minimal terminal (generally 0 or 1) described the minimum number of times that an entity can take part in a relation. ¶The maximum terminal (generally 1or N) described the maximum number of times that an entity can take part in a relation. ¶The figure below which describes the cardinalities between the “place” between the entities commands and ads to mean a one or many commands can be place for one work of art.

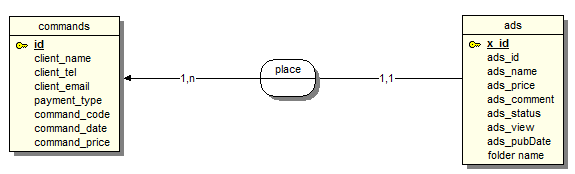


Figure 6: Example of cardinality

#### 3.3.4.10 Rules of standardization

**¶¶**¶The goal of the rules of standardization is to lead to a valid conceptual data model. ¶These rules are inter alia: ¶

* The name of an object, an association or an attribute must be single. ¶In our DCM, we took care that all the objects, associations and the attributes are characterized by their uniqueness.¶
* Each object must have an identifier. ¶All the objects which we used have each an identifier.¶
* An object has at least an attribute. ¶This rule is perfectly complied in the design of our objects.¶
* An association can have or cannot have property.¶
* The properties should not be redundant.¶
* The minimal cardinality is always 0 or 1 and the maximum cardinality is always 1 or N.

#### 3.3.4.11 Normal forms

¶There are three normal forms which must be applied in the order:¶

* **¶First normal form**:¶

¶Its aim is to eliminate the properties which have several values. ¶In our CDM, this problem does not arise.¶

* **Second normal form ¶**

Its aim is to eliminate the properties which depend only on part of the identifier. ¶When the identifier is composed, all the properties must functionally depend completely on this identifier and not partly.¶

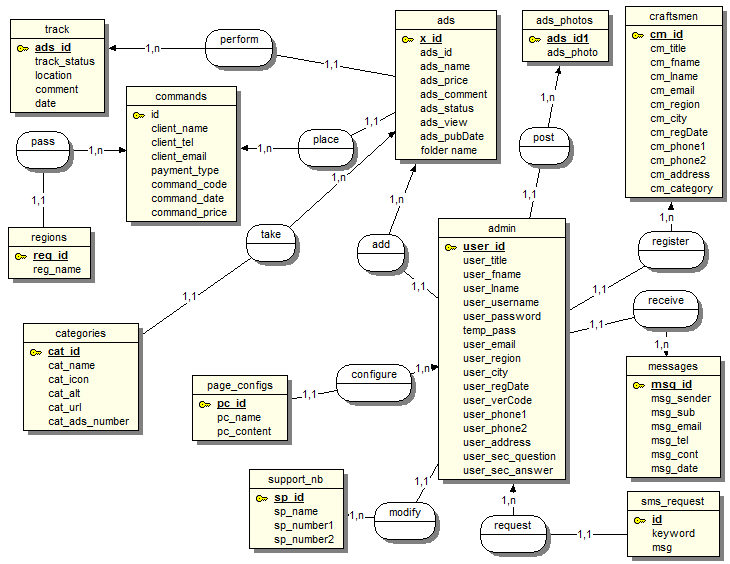
¶Example starting from the identifier of commands, one can know the client’s name who has place the command, the telephone number, the command code and many others.¶

* **Third normal form ¶**

Its aim is to eliminate the properties which depend on another property other than the identifier. ¶We do not have this problem in our DCM.

#### 3.3.4.12 Conceptual Data Model (DCM)

The formalization of data is the major point of MERISE. It is important to expand this approach to data for a large number of sets of manipulated information, requires work necessary, to ensure consistency and optimization of storage, costs and processing information. This complexity of the sets of information that can be found in Computer Management has necessitated the creation of a method that takes into account the structure of information.



¶Figure 7: The Data Conceptual Model.

#### 3.3.4.13. Data logical model

¶The data logical model describes the structure of the data. ¶It aims at:¶

* ¶ Avoid the inconsistencies in the data:¶ a pensioner has one code for example
* ¶ Avoid the redundancies of information:¶ that is avoiding that same information is stored in various tables
* ¶ Avoid the zero values:¶ they are either the unknown values, or of the known but not available or still of the inapplicable forms.¶ Thus, makeign the joints difficult to specify.¶

¶The generation of the RDLM starting from the DCM passes by a certain number of elements and imposes rules which should scrupulously be complied with.¶

#### 3.3.4.14. Rules of passing from the DCM to the DLM

**¶Rule 1**: Any entity becomes a table in which the attributes become columns. ¶The identifier of the entity then constitutes the primary key of the table.¶

**Rule 2**: An association of the type 1: n disappears and becomes a foreign key of the table of the side 0:1 or 1:1 which refers to the primary key of the other table

**¶Rule 3**: An association of the type n: ¶n (that is which has positioned maximum cardinalities with “n” on the two sides of association) results in the creation of a relation of which the primary key is made up of the foreign keys referring the relations corresponding to the entities bound by association ¶

**Rule 4**: ¶A binary association of type 1:1 is represented by a binary association of type 1: ¶n. ¶Except that the foreign key is seen imposing a constraint of unicity in addition to one possible constraint of vacuity.¶

**Rule 5**: Any non-association on the type 0: n can be seen as another table with primary key becoming all the foreign keys.

¶The observation of the rules of passing from the CDM to the LDM enabled us to generate the following DLM. For our application, we can have the LDM below:

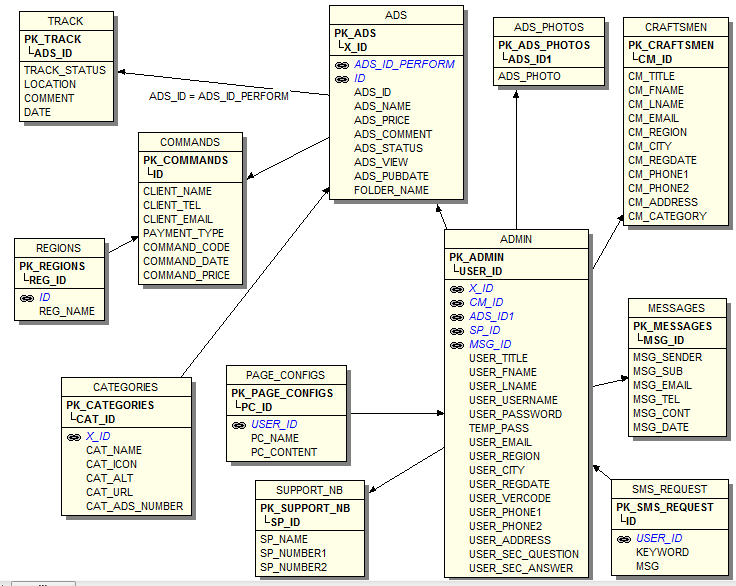


Figure 8: the Data Logical Model

## 3.12 Choice of Method

Research on this work has presented: OMT, UML, UP, SADT and MERISE as some of the principal models that can be used in designing an application. As a methodology to be used in this work, UML has been chosen to design our application. Automatically, UML will use the UP method because UP uses UML notations. The reason why UML is chosen is because in UML, the dynamic (behavioral) and static (structural) things are fused into the system’s entity to realize good and desirable results. This creates interdependency between the static and the dynamic things. It also provides precision and stability of the system. Hence, it is faster in building our application using the UML to MERISE method. The MERISE method on the other hand, separates static approach system from the dynamic approach. It uses data models in representing the static system and treatment models in representing the dynamic system, it is not a method made specifically for software development like UML but rather, it (MERISE) is generally used thus making the building of the application slower and costlier because more material are used to attain the same but less reliable result in quality and quantity.

## 3.13 Application of UML Method

As it is often said, a picture is worth a thousand words, this absolutely fits while discussing about UML. UML is a pictorial standard and modeling mechanism for specifying, visualizing, constructing, and documenting the artifacts of software systems. So beyond reasonable doubts, UML will help us better realize our application and understand its functionality.

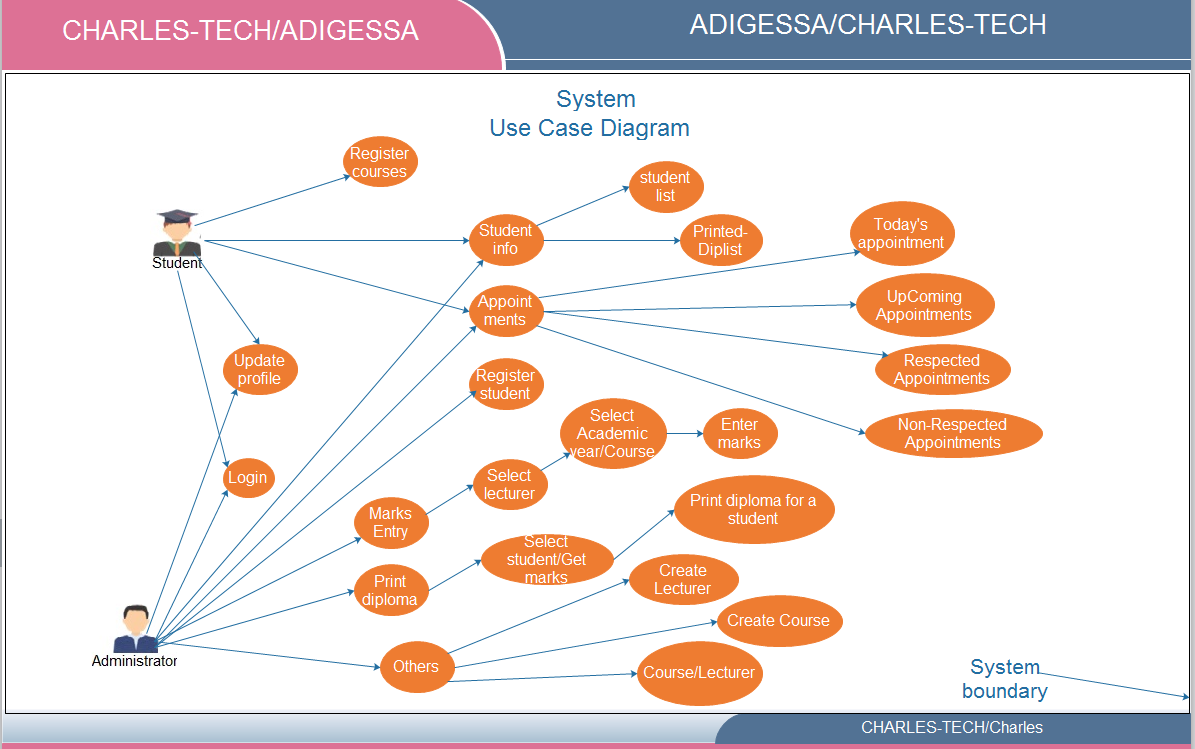
### 3.13.1 Actors

An actor is an external human or software agent closely to the system and interacts with that system by its role played. In our case we have the following actors:

* **Admin (Supper Administrator/Administrator) :** Is any administrator of the computer unit in UBa in charge of the administration of the system, printing of diplomas and also filling forms containing students’ information to be used for further processing of diplomas (automatic generation of diplomas and sending of SMS alerts to students’). He configures all the system to work efficiently and controls the access of the other actors (administrators and students). He can also make sure that names of all the students’ whose diplomas are printed should be displayed on the website so as to keep students’ informed. The supper administrator may be different from the admin in that, he can carry out further modifications on processed diplomas and print them.
* **School (UBa):** It is the institution issuing diplomas to students. So it passed instructions to the various admins to process and print students diplomas so that they can be authenticated and dispatched.
* **Appointments:** it is a software agent associated with the system that is, automatically created by the system based on students’ registration information entered by the admin.
* **Courses:** it is a software agent associated with the system that is, they are courses per program, per school of the University of Bamenda which all already exist in the school database of courses. After they are selected by students’, they are studied to determine their performance
* **Marks:** it is a software agent associated with the system that is, gotten from the performance of each and every student. They are entered into the system by the administrators. These marks permit the system to automatically generate diplomas.
* **Students’:** Are the main actors of our system, this is so because the system is functioning for their seek and the rate at which they visit the system is far more than the rate any other actor visits it. If they are not there, it will be needless for the super admin and/or the admin to carry out all the processing. So they need to be informed and also interact with the system by constantly consulting the web application.
* **Student courses:** it is a software agent associated with the system that is, they are selected by a student based on the required number per semester and per form B of each student which will subsequently help to determine the student’s average, GPA and overall performance
* **Teachers:** these are lecturers who teach the courses that are offered by students’. They do not sign in to the system but are assigned these courses inside the system by the admin/superadmin after their credentials are entered into the database.

## 3.16 Use case diagram

A single use case diagram captures a particular functionality of a system (ADIGESSA). So to model an entire ADIGESSA, many diagrams are needed.

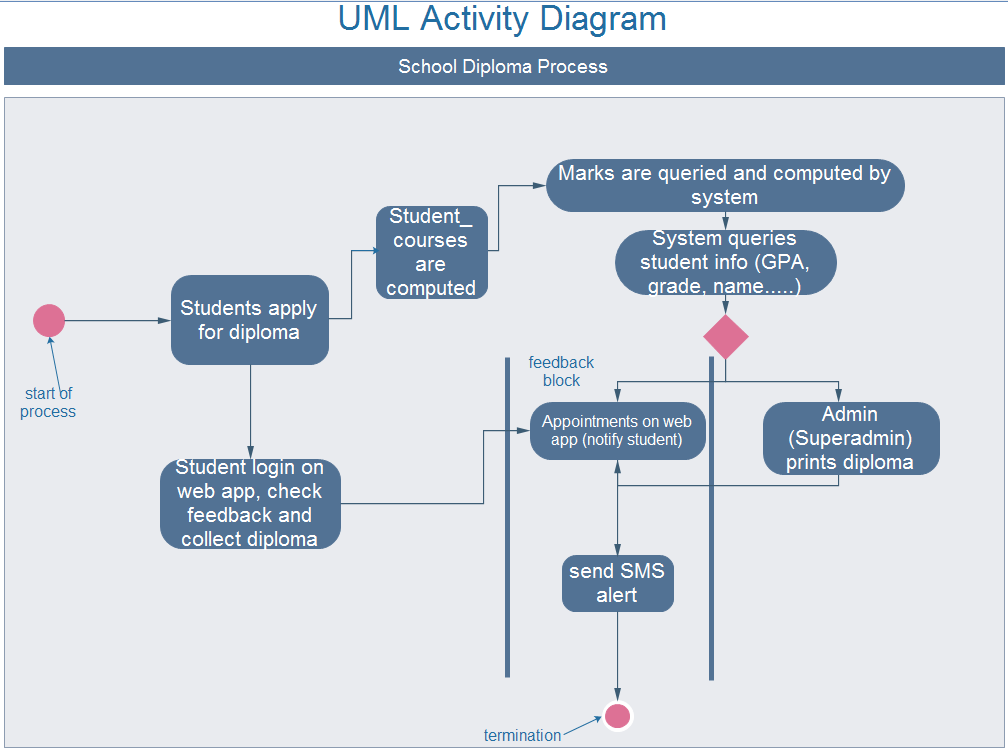


*Figure 7: Use case diagram for our Web Application*

From fig 9 above, an Admin/Superadmin logs into the system to update profile for example students select (register) course while Admin/Superadmin can delete a student, restore a student or check the student as a student with printed diploma. The Admin/Superadmin and students can view student info and appointments. The Admin/Superadmin can enter marks, print diplomas, and carry out other activities like assigning courses to lecturers.

## 3.17 Activity diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the  
system. Activity diagram is basically a flow chart to represent the flow from one activity (a function performed by the system) to another activity.



*Figure 8: activity diagram of a diploma delivery process*

From the activity diagram above, when a student has completed studies, he/se applies for a diploma. Based on the courses the student selected marks for the student is computed. The system further queries the student’s information such as GPA, grade, name and insert into the diploma template. The superadmin then prints the diploma. After it is printed, the student is notified with an appointment found on the web application and with an SMS alert. The student can then come and collect the diploma where and when as it is précised in the message he/she received

## 3.18 State machine diagram

Also known as state chart diagram, it describes different specific states of a component/object in a system. These states are controlled by external or internal events (send request, confirm request, and dispatch).

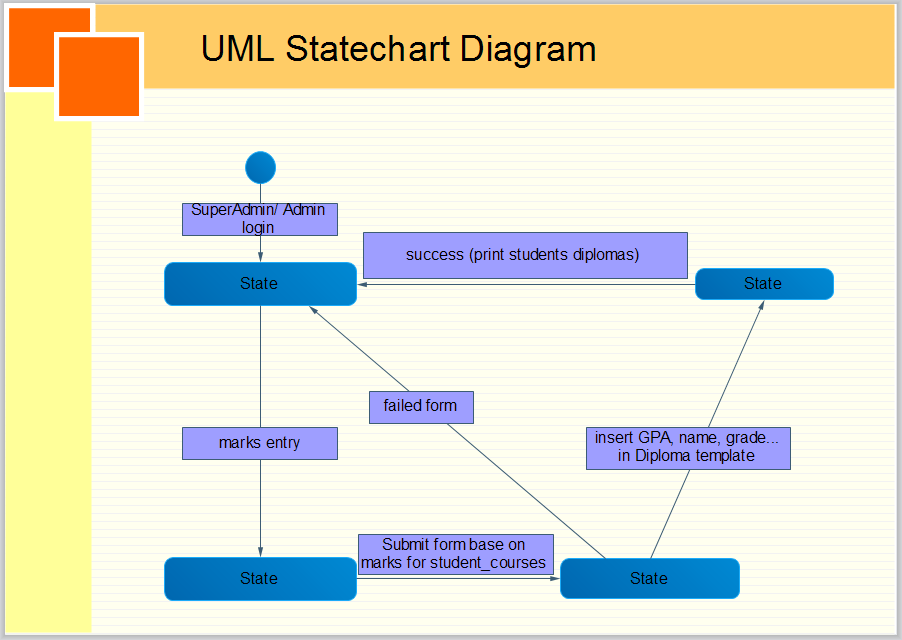


Figure 9: state machine diagram showing the various states to generate a diploma

On the state chart diagram in fig 11 above, in order to generate a diploma in ADIGESSA, the administrator logs in and authenticates. The mark entry menu is click to open the form to fill the student marks. The form is submitted and automatically the GPA is calculated and with other information from the students table like date of birth, they are automatically inserted into the diploma’s template and the diploma is ready for printing in the print diploma menu. But when the submitted form fails, the process to create a diploma terminates.

## 3.19 Sequence diagram

Sequence diagram as one of the interaction over view diagrams, emphasizes on time sequence of messages flow from one object to another.

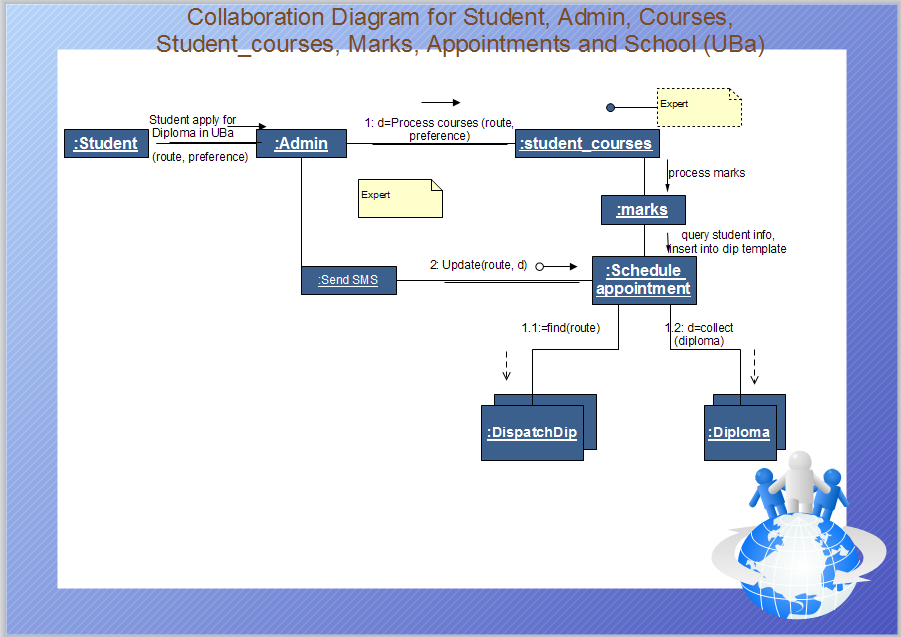


*Figure 10: Login sequence diagram*

Figure 10 above shows the login process for an administrator and a student. When the user logs in to authenticate, a check on information provided is carried out in the database and if successful, then the menu page of ADIGESSA is loaded for the administrator or a respond (students’ home page) is displayed. When the diploma is processed and displayed on the system, the information is cached. In case the initial login process fails, the student or the administrator starts all over.

## 3.20 Communication diagram

Communication diagrams, formerly known as collaboration diagrams in UML 1.x, can be used for because they provide a birds-eye view of a collection and collaboration of collaborating objects.

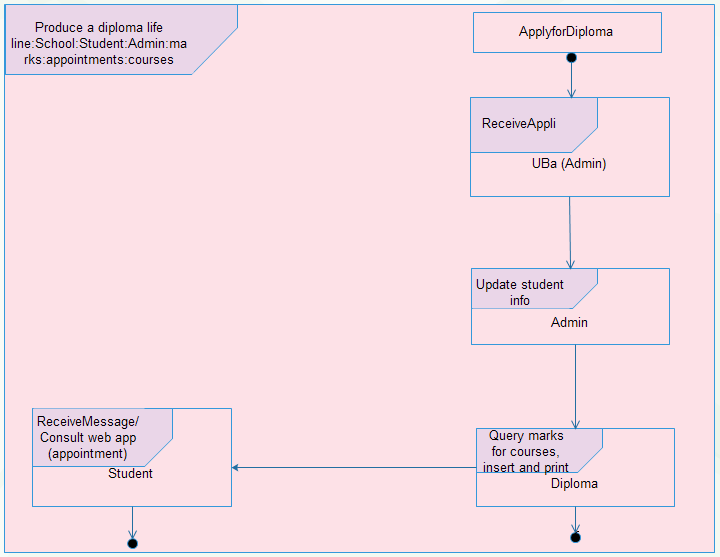


*Figure 11: Communication diagram showing objects organization*

As can be seen on figure 13 above, there is a logical and sequential organization of the student, administrator, student\_courses, marks, appointments, SMS alert and diploma in order for it to be obtained. Before a student’s diploma is produced, he applies to the school (administrator), a schedule of all the processes to produce the diploma takes place. The marks are queried from the courses the student is offering. Along with other student credentials, the marks are inserted in the diploma template and is printed and dispatched.

## 3.21 Interaction overview diagram

An interaction overview diagram is a form of activity diagram in which the nodes and notations represent interaction diagrams.

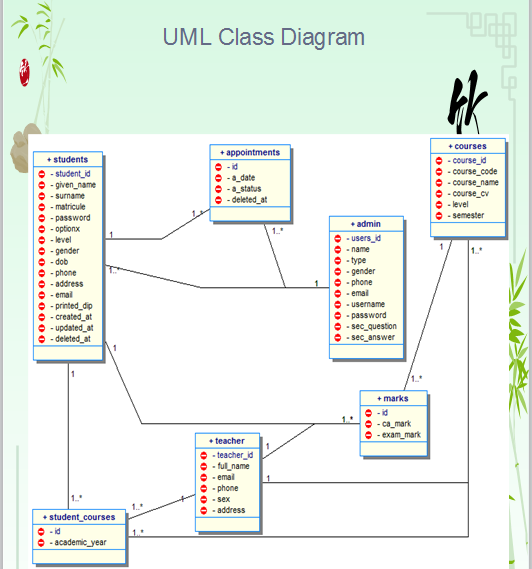


*Figure 12: interaction overview diagram of a diploma delivery process*

Figure 12 above gives a general view how the various activities involved in by administrators, students, school, marks, appointment and courses, interact with each other. When a student applies for a diploma, the school admin receives application, check student info and student credentials are queried and inserted on a diploma template for printing, and the student is given an appointment via an SMS alert or via the web application.

## 3.22 Class diagram

The class diagram is a static diagram. It represents the static view of an application. So a collection of class diagrams represent the whole system. Class diagram is not only used for visualizing, describing and documenting different aspects of system but also for constructing executable code of the software application.

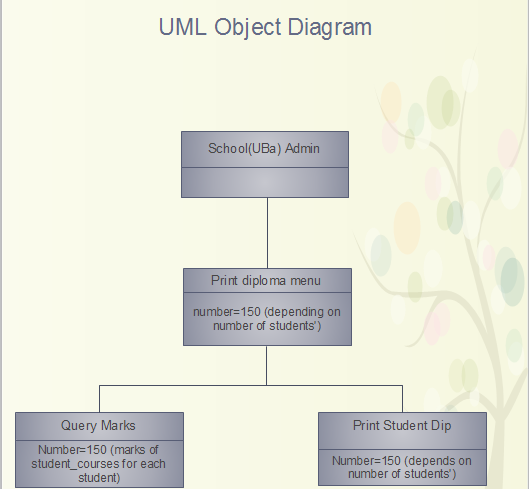


*Figure 13: Main application class diagram*

The class diagram in fig 13 above shows all the entities of ADIGESSA and their respective attributes. The relationship that exit between one or more entities (tables) can be clearly seen from the lines linking the individual tables and the nature in which they related can be seen from the cardinality of different tables. As can be seen, one appointment date can be given to one or many students, one or many student can be registered by an admin, one or many student\_courses can be offered by a student, one or many students can merit a particular mark, one or many student courses is taught by a teacher (lecturer), one or many students’ marks is allocated by a teacher, one or many appointments can be given by an admin, one or many marks belong to a course, one or many course is taken by a teacher and one or many student\_courses originates from the courses table.

## 3.23 Object diagram

Object diagrams are derived from class diagrams (instance of a class diagram) so object diagrams are dependent upon class diagrams and represent a static view at a particular moment.

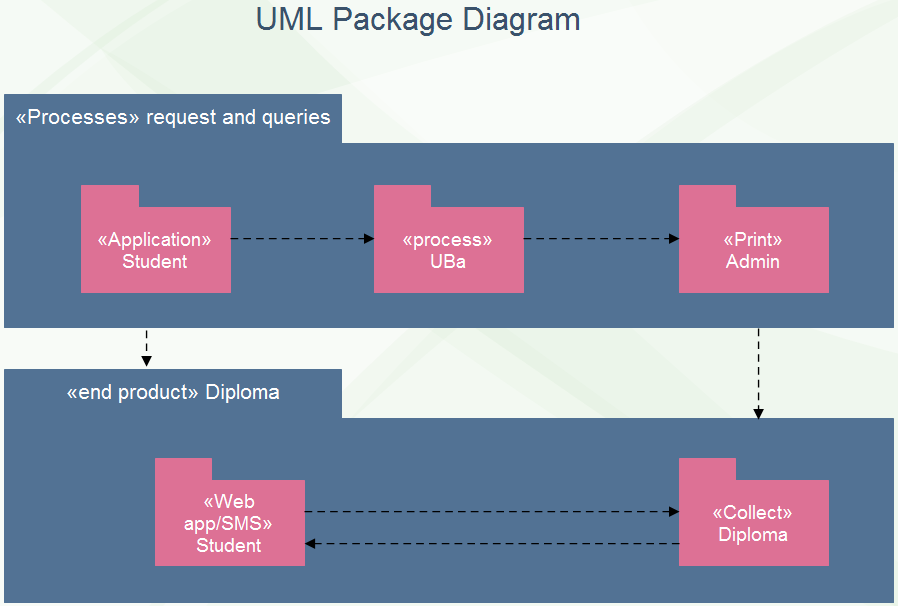


*Figure 14: object diagram showing an instance of printing diplomas*

The object diagram in fig 16 above is an instance of entities of the class diagram above using arbitrary values which can vary. The flow of information about diplomas starts from the school right down to administrators in the computer unit printing and students’ collecting diplomas. Copies of different diplomas are then produced and SMS sent to students’.

## 3.24 Package diagram

**Package diagram** is **UML** [structure diagram](http://www.uml-diagrams.org/uml-25-diagrams.html#structure-diagram) which shows [packages](http://www.uml-diagrams.org/package-diagrams.html#package) and dependencies between the packages. As a **namespace**, a package can **import** either individual members of other packages or all the members of other packages.



*Figure 15: package diagram*

From figure 15 above, when a student applies for a diploma in school, the application is processed and diploma printed. The student then come and collects it. Especially in case the diploma is not printed, or the student not updated on the status of its diploma via SMS or students’ home page, then he has to complain, comes back and consult the web application and SMS alert for updates.

## 3.25 Composite structure

Composite Structure Diagram is one of the new artifacts added to UML 2.0. It shows the internal structure (including parts and connectors) of a structured classifier or collaboration. It is used to explore run-time instances of interconnected instances collaborating over communications links to achieve some common objectives.

****

*Figure 16: composite structure diagram overview*

On figure 16 above, we can see that all the parts of the internal structure of our application are connected to the central processor showing that all the processing of all what we have been saying from the very first UML diagram cannot be accomplished without the computer (central processor). And for the end product which is the diploma to be achieved at the quickest time possible, different parts of the application has to harmoniously work.