CO255 - Software Engineering Lab Assignment 1

Date of submission : 15/01/2018

Project Members

1. Ashwin Joisa
   Roll number : 16CO104
   Mobile number : +91 9482851775
   Email id : ashwinjoisa@gmail.com

2. Praveen Raj
   Roll number : 16CO115
   Mobile number : +91 9481276415
   Email id : hlpr98@gmail.com

Problem Title : Automated Lab Program Evaluator

Problem Statement

This software aims to automate the process of evaluating the programs written by the students in the lab, during the lab classes, surprise tests, mid-semester and end-semester examinations. This reduces the workload on the teachers and the teaching assistants by a considerable amount. This scheme of evaluation is also better as it is impartial and all corner cases can be tested, unlike in the case of a person doing it manually, for each and every student. The marks obtained can be seen instantaneously by the student, and the the teacher can view all the students' marks properly formatted. This automation ensures that no evaluation has to be done manually during and after the lab or examination is over.

Work Envisaged

1. Study the Software Development Life Cycle Processes and identify the right process or the above.
2. Define an overall software project plan clearly defining the phases, duration and tasks.
3. Develop Functional Requirement Specifications and Software Requirement Specifications.
4. Develop high and low level design.
5. Construct and unit test the programs.
6. Perform system and integration testing, do basic performance testing. Fix bugs and retest the system. Trace the tested system back to requirements to ensure completeness of system.
7. Final release of the software.

PART I : Software Crisis

Case 1 : Imagine that your software has failed at the customer's site. List out the possible reasons for failure and write possible solutions.

1. Incompatibility of the platforms : This software is designed for linux and unix environment (for example Ubuntu OS). To overcome this problem, different models of the software can be developed for different platforms.
2. Program written by the user may run into an infinite loop, which may cause the system to hang : One possible solution, is to terminate the program being tested, after a specified amount of execution time.
3. Program written by the user may accidently try to access a part of the main memory which is prohibited by the OS, or may not exist. To overcome this, we display a “Runtime Error” along with the error message(from the OS) obtained by running the program.
4. Program to be tested may be in a language, not supported by the software : This could be resolved, by specifying the host of languages supported by the software.

Case 2 : Imagine that your software is not delivered to the customer on the given time and customer is willing to cancel the project. List out the possible reasons for delay in development and write some possible efforts that you would make to convince the customer in order to retain the project.

Possible reasons for delay in development of the software:

1. No clear vision for the project : The lack of clarity about vision, would result in a disconnect between the ultimate end user’s needs, resulting in changing scope over time.
2. Poor initial project estimation : A project with poorly estimated requirements and scope, if enters the production cycle, would suffer huge delays, since the developers would struggle to understand the true scope of the project.
3. Adding to the Scope : Repeated modifications of the scope would only result in delays and not otherwise.
4. Poor project tracking and management
5. Poor leadership and governance : A poor leadership often results in demotivated developers and in sloppy management of the project’s progress, thus resulting in delays.

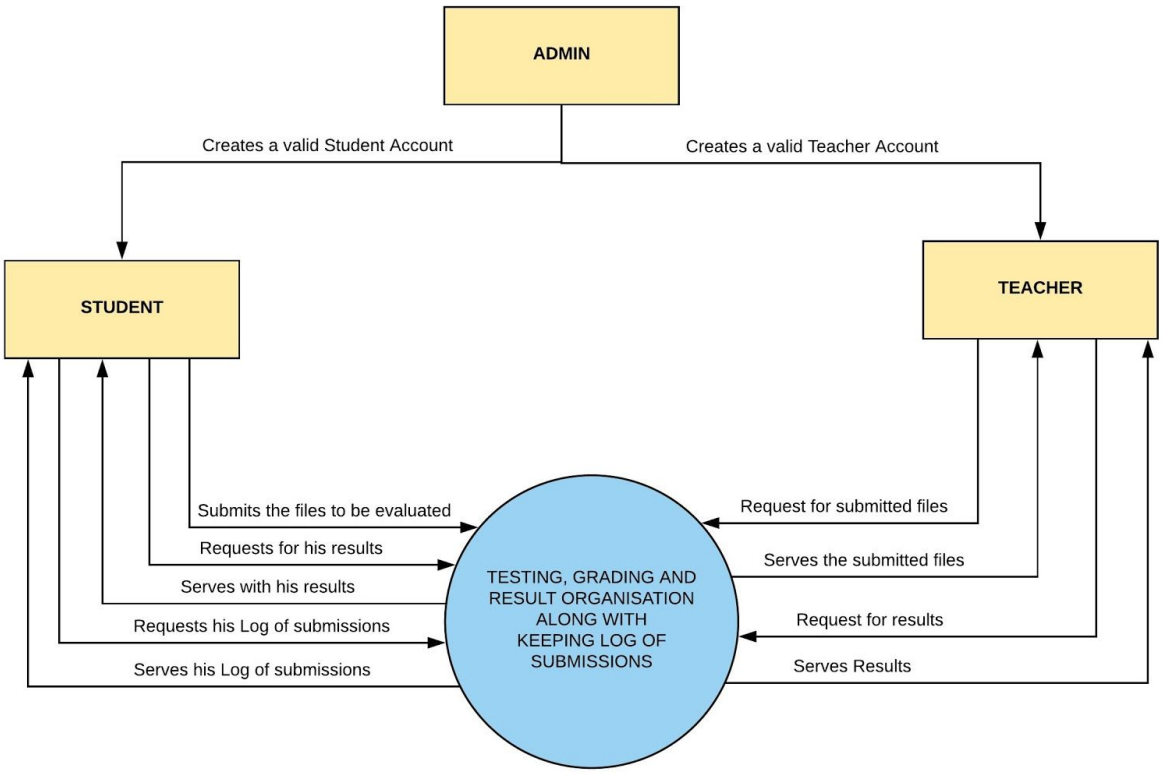
Reason to convince the customer to retain the project:

1. As mentioned in the problem statement, this software reduces the workload on the teachers and the teaching assistants by a considerable amount, since the manual work of testing need not be done.

2. This scheme of evaluation is also better as it is impartial and all corner cases can be tested, unlike in the case of a person doing it manually, for each and every student.
3. The marks obtained can be seen instantaneously by the student, and the the teacher can view all the students' marks properly formatted. This automation ensures that no evaluation has to be done manually during and after the lab or examination is over.

PART II - Context Diagram, Data Flow Diagram and ER-Diagram

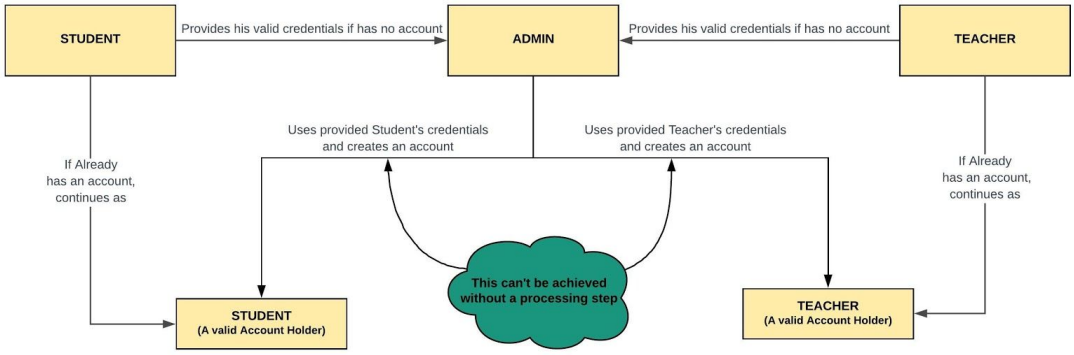
Question 1 : Draw a context diagram for your problem statement.



Question 2 : Imagine that you have recruited a fresh graduate to draw Data flow diagram for your problem statement. List out minimum five possible errors he / she may make while drawing the data flow diagram and explain it with a clear pictorial representations.

1) **Illegal data flows :-** One of the patterns of data flow analysis is that all flows must begin with or end at a processing step. This makes sense, since presumably data cannot

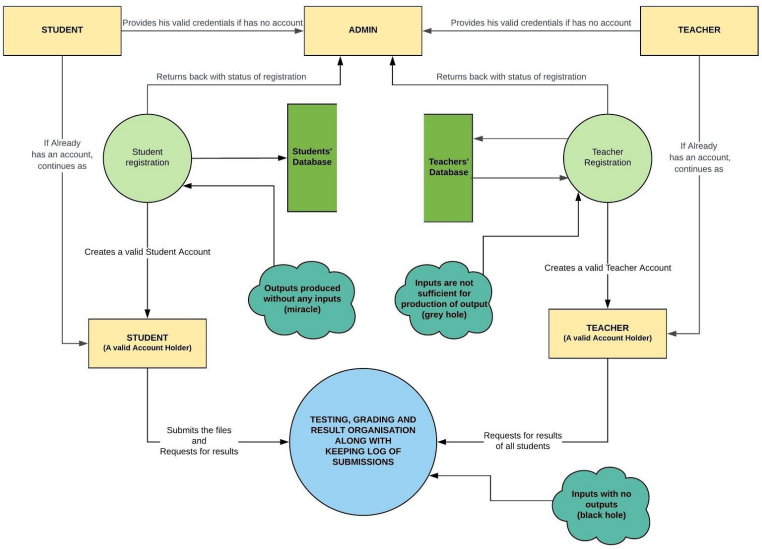
simply metastasize on its own without being processed. This simple rule means that the following mistake can be fairly easily identified and corrected in a DFD.



2) Diagramming mistakes: Black holes, grey holes, and miracles: -

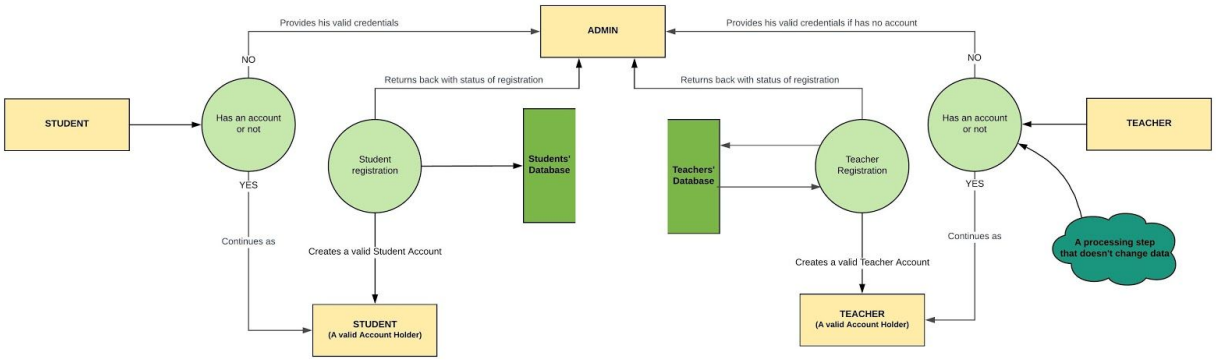
A second class of DFD mistakes arise when the outputs from one processing step do not match its inputs.For instance:

- A processing step may have input flows but no output flows. This situation is sometimes called a *black hole* .
- A processing step may have output flows but now input flows. This situation is sometimes called a *miracle*.
- A processing step may have outputs that are greater than the sum of its inputs - e.g., its inputs could not produce the output shown. This situation is sometimes referred to as a *grey hole*.

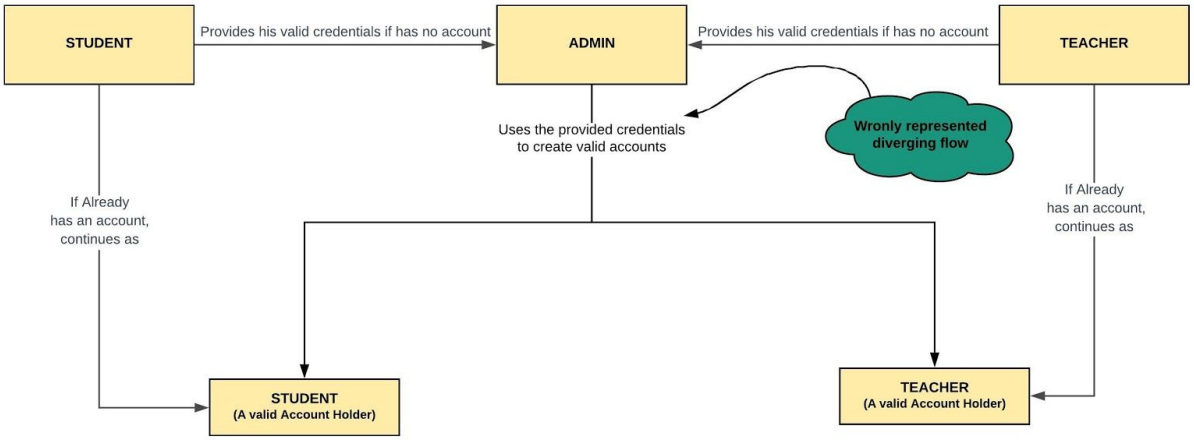


3) DFDs are not flow charts:-

Flow charts often show both processing steps and data "transfer" steps (e.g., steps that do not "process" data); DFDs only show "essential" processing steps. Flow charts might (indeed, often do) include arrows without labels: DFDs never show an unnamed data flow. Flow charts show conditional logic; DFDs don't, i.e the processing steps that do not change data, do not belong on the Data Flow diagram.

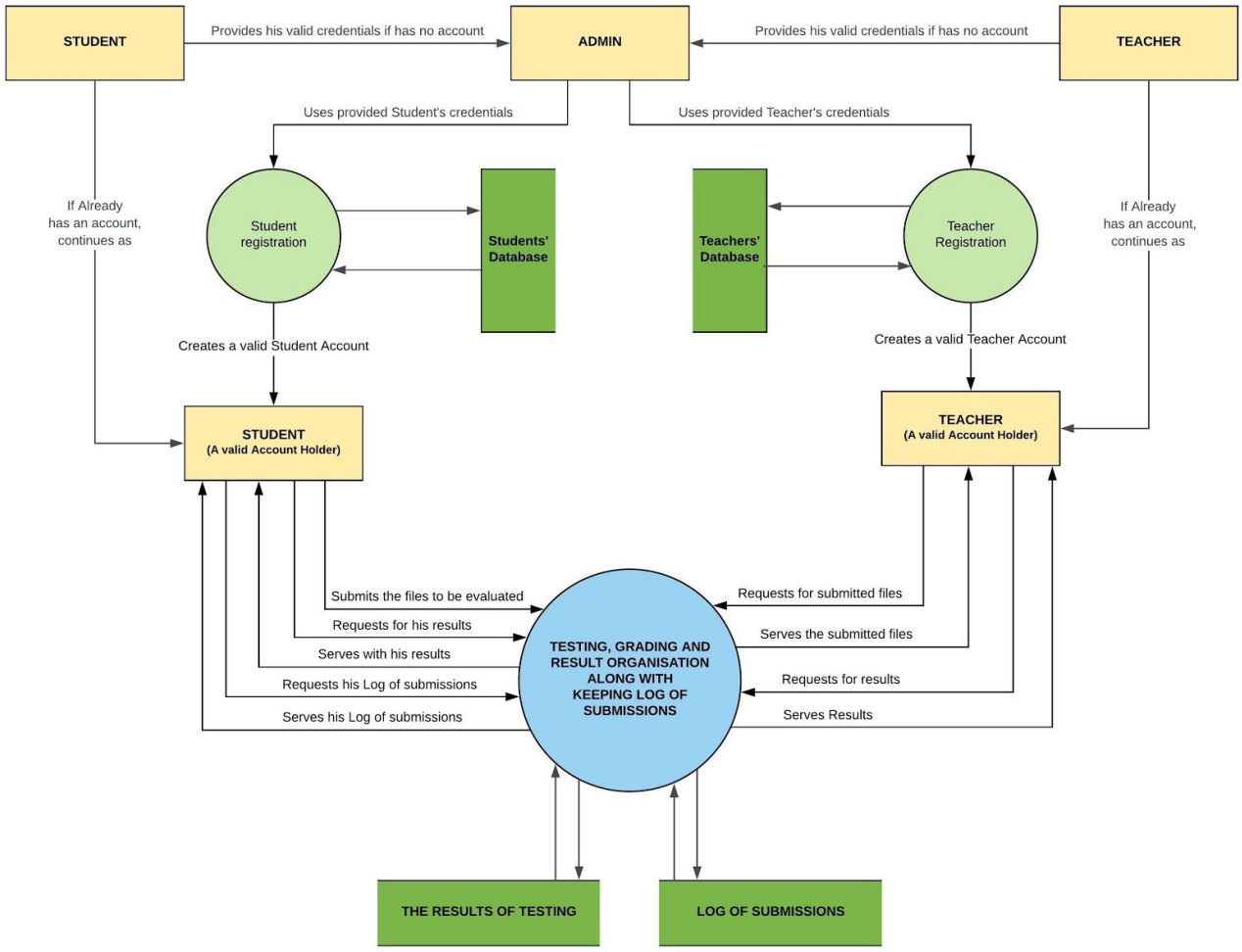


4) Diverging data flows should be replaced by individual flows



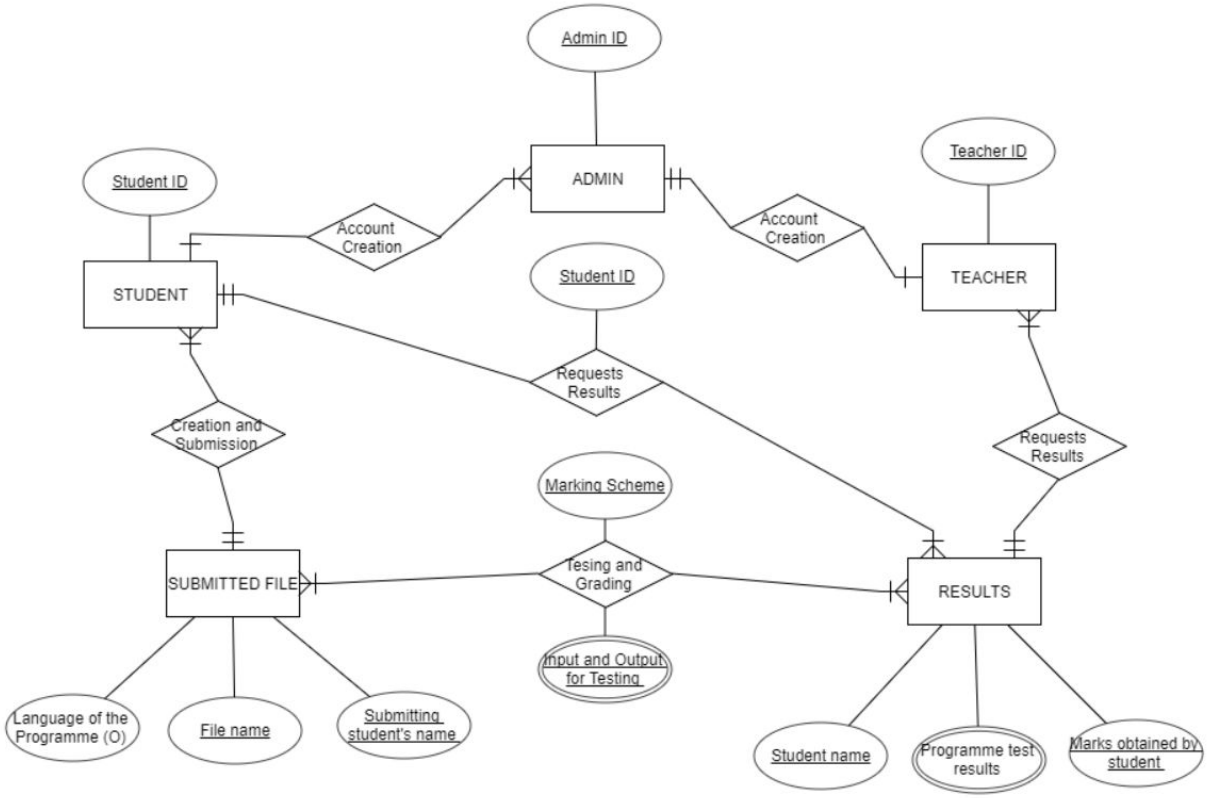
5) Creating unbalanced decomposition (or explosion) in child diagrams. Each child diagram should have the same input and output data flow as the parent process. An exception to this rule is minor output, such as error lines, which are included only on the child diagram.

Actual Data Flow(Level-1) Diagram of the project



Question 3 : Construct an ER-diagram for your problem statement. Imagine that customer is changing his / her requirements after two or three days which may result in adding new entities or updating the existing entities. At this situation, you are instructed to draw a new ER-diagram. List out the possible change in requirements and corresponding changes to be made in the ER diagram (entities and relationships).

ER diagram for the project :-



Possible changes in requirements and corresponding changes to be made in the ER diagram :-

a) **Change in requirements:-** Removal of the Admin control.
Change in ER diagram:- Removal of the entity ADMIN along with the relationships originating from it.

b) **Change in requirements:-** The entity TEACHER may contain more than one teachers.
Change in ER diagram:- Changing the cardinality of relation between TEACHER and RESULTS to many-to-many.

c) **Change in requirements:-** A student must submit exactly one file for evaluation.
Change in ER diagram:- Changing the cardinality of the relation between STUDENT and SUBMITTED FILE to one-to-one.