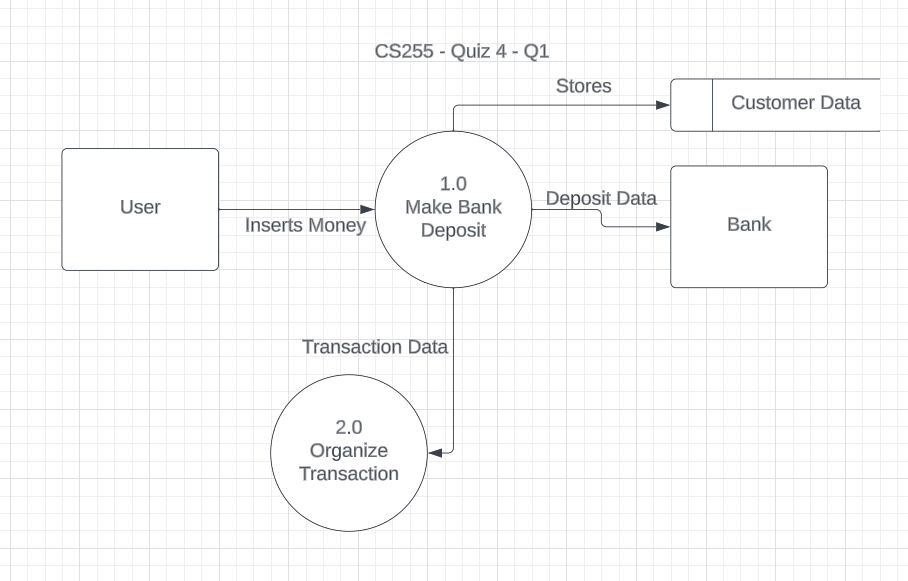
Nicholas LaMantia - cs255 - Quiz 4

7.25 You stop decomposing a DFD when the following six conditions are satisfied: (1) each process is a single decision or calculation or a single database operation, such as retrieve, update, create, delete, or read; (2) each data store represents data about a single entity, such as a customer, employee, product, or order; (3) the system user does not care to see any more detail, or when you and other analysts have documented sufficient detail to do subsequent systems development tasks; (4) every data flow does not need to be split further to show that different data are handled in different ways; (5) you believe that you have shown each business form or transaction, computer screen, and report as a single data flow; or (6) you believe there is a separate process for each choice on all lowest-level (interface) menu options for the system.

**Explain through example i.e. provide an illustration of a “simple” business transactional flow from user initiation to user actions completion addressing each category above (30 min).**

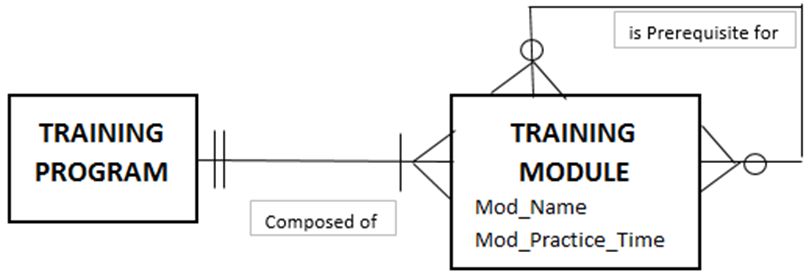


In the diagram, the theory of a finished DFD is addressed. Each process must be a single decision or calculation, which is seen above. Making a bank deposit and organizing the transaction are single calculations and/or decisions. The data store in this diagram, the customer data, represents a single entity, the customer’s data. The system user doesn’t need any more data, nor cares to see any more. As well, every data flow is handled in one way in this DFD, meaning nothing needs to be split further to show that different data needs to be handled in different ways. I believe to have created a DFD that shows each business form or transaction, computer screen, and report as a single data flow. There is a singular flow that you can map that encapsulates this DFD and the processes within it. Finally, there is a separate choice for all menu options, because there is only one menu option being represented, which is to make a deposit.

*10/23 // Level N looks like each process is a single decision, which is what the diagram shows. The Make Bank Deposit process only has the ability to make the bank deposit using the money inserted by the user. Similarly, the Organize Transaction process has the single ability to organize the data from the transaction (how much money was deposited to what User) and compile that into a receipt. Each data store should only represent one entity, which it does. The Customer Data is linked to the individual customer. Bank is a source and a sink. I believe that this diagram represents a level N, “DONE”, diagram because in theory, it passes the requirements listed. In your feedback, you mentioned separating the second process (Organize Transaction). I believe there is no need for this because as I have it defined, it only does one thing. I realize this model isn’t to scale by any means, but it’s just something to help visualize the different criteria in action. // (From Re-submit - 1)*

*10/26 // The organizing transaction process means that it takes the money deposited, the date deposited and the account information, and organizes it onto one receipt. This is supposed to be one process. As well, the make bank deposit process is one process as it only accounts for taking the money from the user and depositing it into their account. That is one process which cannot be broken down further. The customer data data store only references the bank, given the user account information. The user doesn’t care for any more information. The data flows do not need to be split further, as each leads to a single action or sink. Also, there is a separate process for each choice on all options for this system. The points above represent the fact that this diagram is an illustration of a “done” diagram. //*

8.49 This model below defines two entities, Training Program and Training Module, with a one-to-many relationship (composed of) between them, and a unary optional (because some modules do not have a prerequisite, and some modules are not a prerequisite to other modules) many-to-many relationship (Is Prerequisite for) on the Training Module entity. This is represented in the E-R below:



**Explain and provide a modified E-R diagram as to how the model would change if you added a training evaluation module that has a prerequisite for completion of the training module that is evaluated. The evaluation module is described by MOD\_NAME, MOD\_SCORE, and MOD\_TIME to COMPLETE. (30 min).**

The ER diagram below represents if there’s an entity added called Training Evaluation Module (TEM). In the TEM, there are attributes called mod\_name, mod\_score, mod\_time, and complete. This entity has a one-to-one optional relationship (has) with the training module as one training module may or may not be evaluated. The only thing that would change from the above model to the below model is the addition of a new entity containing a new relationship and attributes.

