KariAnn Harjo

ITSE 1450

Module 5 Questions

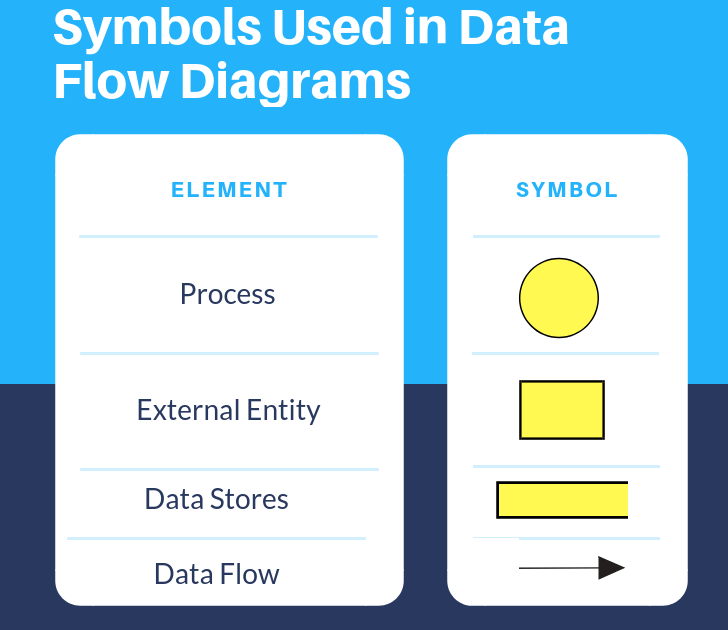
1.

Logical and physical models are both essential stages in systems design, serving distinct purposes. The logical model focuses on representing the abstract structure and relationships within a system without considering its physical implementation details. It captures what the system is supposed to do, outlining entities, attributes, and relationships. In contrast, the physical model delves into the actual implementation specifics, detailing how the system will be constructed. This includes data storage, access paths, and other technical considerations. While the logical model emphasizes the "what" and "why," the physical model is concerned with the "how" and "where." Both models work in tandem, ensuring a comprehensive understanding and execution of a system's design.

2.

A Data Flow Diagram (DFD) in the Software Development Life Cycle (SDLC) serves as a visual representation of the flow of data within a system. It illustrates the sources and destinations of data, how data moves through different processes, and where data is stored. By providing a clear and concise overview of system components and their interactions, DFDs aid in understanding the system's workings, enabling stakeholders to identify inefficiencies, redundancies, and potential areas for improvement. Throughout the SDLC, DFDs help guide analysis, design, and decision-making, ensuring the development of an efficient and effective system.

3)



4.

When drawing Data Flow Diagrams (DFDs), it's essential to maintain consistency in naming and symbols throughout the diagram. Each DFD should be kept simple and uncluttered, ensuring that all data flows have a clear source and destination. It's also crucial to assign unique names to each process, data flow, data store, and external entity to avoid confusion. The flow of data should logically progress, typically from top to bottom or left to right, avoiding overlapping lines. Lastly, when decomposing a high-level DFD to a lower level, the data flow between them should be balanced, meaning inputs and outputs should match at each level.

5.

A context diagram is the highest level in a Data Flow Diagram (DFD) and provides an overview of an entire system, illustrating the system's interaction with external entities but not showing the internal processes. In contrast, Diagram 0, one level below the context diagram, offers a more detailed view by breaking down the system into its primary processes, showcasing data flow between these processes, data stores, and external entities. While the context diagram represents the whole system as a single process, Diagram 0 provides insight into the main sub-processes and how they interact with one another and with external entities.

6.

In a context diagram, the data store symbol is not used. The context diagram provides a high-level overview of the system and its interaction with external entities, focusing solely on the flow of information between the system and these external entities. Therefore, it only utilizes the process, entity, and data flow symbols, and avoids detailing internal data storage or its intricacies.

7.  
Levelling a DFD involves breaking down the main processes into more detailed subprocesses in a hierarchical manner, ensuring clarity and specificity as one moves from higher to lower levels. Beginning with a context diagram (the top level), which provides an overview of the entire system and its interactions with external entities, one then creates Diagram 0, which expands the context diagram to show main processes. Subsequent levels further decompose these processes into finer details, with each level providing greater granularity. Each level should maintain a consistent flow of data and should avoid any sudden jumps in complexity.

8.  
Balancing a DFD refers to ensuring that the flow of data and processes at one level of the DFD is consistent and accurately represented in the next level. To balance a DFD, one should ensure that every data flow entering or leaving a process in the higher-level DFD appears in the next detailed level. This means that the inputs and outputs to each process should remain the same across levels. It's crucial to verify that no data is added or lost when moving from one level of detail to another, maintaining a cohesive representation of the system's functionality.

9.  
A data element, often referred to as a data item or field, is the smallest indivisible unit of data that holds specific meaning. It represents a singular fact or piece of information, such as a customer name, product price, or order date. Data elements serve as foundational building blocks for more complex data structures, ensuring that data is correctly organized, processed, and stored within a database or information system.  
  
10.  
A decision table is a visual representation used in both software development and business decision-making that presents a systematic way of listing all possible conditions or scenarios for a given problem and their corresponding actions or outcomes. It aids in clarifying complex business rules by detailing every possible condition and its consequent action, ensuring that all scenarios are considered, avoiding ambiguity, and facilitating more accurate and consistent decision-making processes.