ISSN 2522-9400 **European Modern Studies Journal** *Vol 8 No 1*
available at www.journal-ems.com
DOI: 10.59573/emsj.8(1).2024.28

Analyzing Data Flow: A Comparison between Data Flow Diagrams (DFD) and User Case Diagrams (UCD) in Information Systems Development

Arwa Y. Aleryani
Associate Prof., Independent Researcher, Canada

Abstract. This research endeavors to investigate the process of eliciting data flow by employing both data flow diagrams and user case diagrams. Through an in-depth analysis of existing literature and comprehensive comparisons between these two diagrammatic representations, the study concludes that the most effective approach for systems analysts involves the concurrent utilization of both types of diagrams. These two diagrams will enhance the understanding of data flow dynamics within the system, encompassing the data flow between system functions and the seamless flow of data among those engaged with the system.

By leveraging the strengths of both data flow diagrams and use case diagrams, analysts can achieve a more holistic and nuanced comprehension of the intricate data flow relationships inherent in complex systems.

Keywords: Data Flow, Data Flow Diagrams, User Case Diagrams, Information Systems Development

Introduction

In contemporary information technology landscapes, a predominant trend has been the utilization of structured approaches for modeling and documenting information systems (Yourdon, 1989). Originally, these systems were conceived and crafted using languages that, with the passage of time, have become obsolete (Jilani et al., 2011). These structured methodologies primarily aim to encapsulate the dynamic and behavioral facets of a system, employing graphical tools such as Data Flow Diagrams (DFD), decision trees, and decision tables (Yourdon, 1989).

Nowadays data availability and access to various platforms are changing the nature of Information Systems (IS) studies. Such studies often use large datasets from various platforms, which may incorporate structured and unstructured data (Arpan & Yogesh 2020).

In the present era, software companies are engaged in a competitive race to deliver high-quality software products with increased efficiency. To ensure the excellence of their products, these companies are directing their efforts towards every stage of the Software Development Life Cycle (SDLC). Among these stages, the design phase holds paramount importance, demanding significant effort and attention from developers (Jacob, et al. 2016).

Achieving top-notch software involves a meticulous definition of the problem at hand and the subsequent creation of an optimal solution during the design phase. In the function-oriented approach, this phase centers around the utilization of Data Flow Diagrams (DFD) and Structure charts. Conversely, the object-oriented approach places emphasis on the use of Unified Modeling Language (UML) during the design phase.

Among the various structured tools, the Data Flow Diagram (DFD) stands out and is still the most pivotal, offering a hierarchical representation that proves invaluable in system design (3). This hierarchal structure not only facilitates a nuanced understanding of the system but also enables abstraction at different levels, a crucial aspect in the intricate process of system design (Jilani et al., 2011). While other diagrams like Structure charts, State machines, and ER Diagrams exist within the structured paradigm, they are perceived as less instrumental compared to the versatile DFDs (Tiwari et al., 2012).

In a parallel evolution, the object-oriented approach has emerged as a powerful paradigm for developing intricate systems by amalgamating data and associated methods into cohesive entities known as objects (Yourdon, 1989). At the forefront of modeling object-oriented systems and creations stands the Unified Modeling Language (UML), which has evolved to become the quintessential tool in this domain (Ober, 2000). UML encompasses a comprehensive set of diagrams that serve to model different dimensions of object-oriented software, with Use-Case diagrams emerging as a pivotal element in system analysis (Tiwari et al., 2012).

Nowadays, data flows in huge amounts from a variety of different sources, including text, images, and videos. Data flow has become a standard solution for managing increasingly complex business processes (Aleryani (a), 2020).

In the contemporary landscape, UML's widespread adoption attests to its efficacy in capturing the intricacies of complex systems (Ober, 2000,). Developers, relying on UML, find Use-Case diagrams to be particularly instrumental, offering a visual means for insightful analysis of system functionalities and interactions (Tiwari et al., 2012). As the technological tapestry continues to evolve, these modeling approaches stand as vital pillars in the ongoing quest for robust, efficient, and adaptable information systems.

User Experience (UX) stands as a crucial factor in comprehending the intricacies of data flow, playing an increasingly pivotal role in contemporary system development. Leveraging UX is instrumental in fostering a profound understanding of data flow within the system, and its widespread adoption is a testament to its effectiveness. In this context, tools such as Data Flow Diagrams (DFD) and User-Centered Design (UCD) emerge as invaluable assets, offering a user-friendly interface that facilitates meaningful discussions with clients.

The seamless integration of UX into the system development process proves particularly advantageous during client interactions. DFD and UCD, being intuitive tools, enhance communication with clients by simplifying complex concepts related to data flow. System analysts, recognizing the paramount importance of UX, are well-advised to advocate for its incorporation right from the early stages of requirements elicitation. By doing so, the gap between business analysts and system users is effectively bridged, ensuring a more cohesive and user-centric approach to system development (Aleryani (b), 2020).

Data Flow Diagram (DFD)

The inception of the Data Flow Diagram (DFD) can be attributed to Larry Constantine, the visionary pioneer of structured design, who introduced this seminal concept in the 1970s (Kendall & Kendall 1999). A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used to visualize data processing (structured design). In a DFD, data elements flow from an external data source or internal data store to an internal data store or external data source, via an internal process. The DFD does not provide any information about the timing of operations, or about whether the operations will run in series or in parallel (Ganesh & Prabu 2020).

DFD presents business processes with the data that are shared and exchanged between processes internally and with external entities. Dennis et al. (Kazi & Kazi 2022) described the basic elements of DFD: external entities, data flows, business processes, and data stores (presented in table no 1).

Central to the efficacy of DFD is its hierarchical structure, which not only imparts a systematic organization but also facilitates different levels of abstraction, proving to be an asset in the intricate process of system design (Donald, 2015). Beyond its structural significance, Data Flow Diagram stands as a fundamental artifact, offering a clear and insightful representation of a system's architecture (Kendall & Kendall 1999). It serves as a visual map that elucidates the intricate interplay of various system components.

Moreover, the significance of the DFD transcends its role as a standalone artifact; it acts as a linchpin, providing foundational information that is harnessed by other artifacts to illustrate the dynamic aspects of the system (Kendall & Kendall 1999; Jilani et al., 2011). Essentially, the Data Flow Diagram is a graphical portrayal of how data traverses through the intricate web of an information system (Coronel & Morris 2018). This visual narrative elegantly captures the influx of data from external sources into the system, showcasing the intricate pathways through which data seamlessly moves from one process to another, thereby providing a comprehensive and illuminating snapshot of the system's data flow dynamics. In essence, the Data Flow Diagram not only embodies the structured approach's essence but also emerges as a powerful lens through which the intricate symphony of data movement within a system can be discerned and comprehended.

DFD Component

Data Flow Diagram (DFD) serves as a comprehensive visual tool for illustrating the intricate interplay of processes, data stores, and external entities within a business or system, revealing the connecting data flows that constitute the system's dynamics (Elmasri & Navathe 2017). The DFD employs a concise set of symbols, each laden with specific meanings:

1. Squares or Ovals (External Entities): Representing individuals or groups external to the system's control, these symbols depict the origins and destinations of information, providing a tangible representation of where information emanates from and where it traverses.

2. Circles or Rounded Rectangles (Processes): These symbols encapsulate processes within the system, showcasing components that transform inputs into outputs. The symbolism in the name of each process typically elucidates its function, often adopting a verb-object phase for clarity.

3. Arrows (Data Flows): Symbolizing the movement of data, whether electronic or physical items, arrows in DFDs embody the pathways through which packets of information flow. The directionality of arrows signifies whether data or items are entering or exiting a given process.

4. Open-ended Rectangles (Data Stores): Encompassing both electronic and physical stores, these rectangles represent data storage mechanisms. Whether used for short-term or long-term data accumulation, data stores are pivotal elements in the DFD schema.

Use Case Diagram (UCD)

Introduced by Ivar Jacobson in 1986, a Use Case Diagram (UCD) serves as a crucial methodology in system analysis, aiming to identify, clarify, and organize system requirements within the framework of Unified Modeling Language (UML).

Within UML's array of behavioral diagrams, the Use Case Diagram focuses on external interactions, providing a high-level overview of system functionality.

Use case diagrams provide a high-level view of how a system will be used as viewed from the perspective of a third party (actors) during the design phase. Use case diagrams are used to define the behavior of the system as it is implemented. These internal and external agents are known as actors. So, use case diagrams consist of actors, use cases and the relationships between them. The diagram is used to model the system/subsystem of the application. A single-use state diagram captures a particular function of the system (Ganesh & Prabu 2020).

Use Case Diagram Components

Use Case Diagram (UCD) is an object-oriented diagram. It shows that the system is related to external entities. UCDs are explained extensively about system requirements and functionality.

1. Actor (Represented by a Stick Figure): Denoting a person, group, organization, or external system interacting with the system, actors encompass entities ranging from individuals to networked devices. The stick figure serves as a graphical representation.

2. Use Cases (Represented by a Horizontal Ellipse): Describing sequences of actions that deliver measurable value to an actor, use cases offer a dynamic perspective on system functionality.

3. Associations (Represented by Lines): Depicting interactions described by a use case, associations are visualized through lines connecting actors and use cases, with optional arrowheads indicating the direction of the initial invocation or designating the primary actor.

4. System Boundary (Enclosed by a Rectangle): The system boundary demarcates the scope of functionality within the diagram, encapsulating all depicted use cases. Anything within this boundary represents the functionalities encompassed by the system under consideration.

Literature Review

Wulandari and Widiantoro (Wulandari & Widiantoro, 2017) studied using data flow diagram to support the user experience in application development. In the intricate process of application development, developers navigate through multiple crucial stages, with one pivotal step being the articulation of the application's workflow – a comprehensive portrayal often encapsulated in a Data Flow Diagram (DFD). Simultaneously, the developer must meticulously attend to the user's delight, encapsulated in the realm of User Experience (UX). Elevating the user's interaction involves a meticulous presentation of each element within the application, fostering an environment where users effortlessly engage with the app. This not only enhances user convenience but also contributes significantly to the overall user satisfaction and usability of the application.

In a different approach, Tiwari and the coauthors (Tiwari et al., 2012) proposed a synergistic integration of DFD and Unified Modeling Language (UML) diagrams, envisioning enhanced benefits for developers. Their innovative model seamlessly merged data flow diagrams with UML diagrams, specifically incorporating use-case and class diagrams. While their work remains theoretical, it presents a conceptual framework for leveraging the strengths of both DFD and UML in unison.

Munassar and his coauthor (Munassar & Govardhan 2011) delved into a comparative analysis between the Traditional Approach and the Object-Oriented Approach in Software Engineering Development. Their findings highlighted the limitations of the traditional approach, noting its plethora of models tailored for diverse projects but lacking the flexibility required for Object-Oriented projects. In contrast, Object-Oriented Software Engineering (OOSE) demonstrated adaptability through its employment of UML notations such as use case, class diagram, communication diagram, development diagram, and sequence diagram. The comparison illuminated that the traditional approach relied on common processes contingent upon project size and type, while the Object-Oriented approach leaned on team experience and project complexity via the number of objects involved.

Jilani and the coauthors (Jilani et al., 2011) conducted a survey focusing on the transformation techniques from Data Flow Diagrams to Unified Modeling Language (UML). Their observations highlighted a prevalent trend in rule-based transformation approaches, which, while widely employed, exhibited incompleteness and abstraction issues. The researchers noted that the informal syntax of DFD led to varied interpretations among analysts and designers, complicating the transformation process. This underscores the need for more comprehensive, standardized transformation techniques to bridge the gap between DFD and UML effectively.

Research Methodology

This study endeavors to delve into the distinctions between the data flow diagram (DFD) and the use case diagram (UCD), seeking to illuminate their respective roles and functionalities. Additionally, the research aims to grasp the implications of the escalating data flow within contemporary systems on the efficacy of their utilization. By examining the interplay between data flow diagrams and use case diagrams, the study seeks to provide insights into how the evolving landscape of data flow influences the effectiveness of these diagrammatic tools in current system environments.

Analysis & Discussion

Through an extensive literature review and drawing on the author's experience, it is evident that both Data Flow Diagrams (DFD) and Use Case Diagrams (UCD) share a graphical nature that is inherently understandable to customers. These diagrams effectively communicate system functions and illustrate the flow of data within the system. However, as systems grow in complexity, DFDs may face challenges, becoming less intuitive and potentially overcrowded with sub-functions.

Analyzing the comparison presented in Table no1 and Table no2, it becomes apparent that both DFDs and UCDs follow a similar modeling approach, with minor distinctions like the representation of external entities/actors transitioning from rectangles to person symbols. Notably, UCDs lack the explicit depiction of data stores, a characteristic present in DFDs.

Comparatives between DFD & UCD

By studying previous literature, comparisons can be made between the two schemes, including various elements, as shown in Table no.1 and Table no. 2.

Table 1. DFD & UCD Components

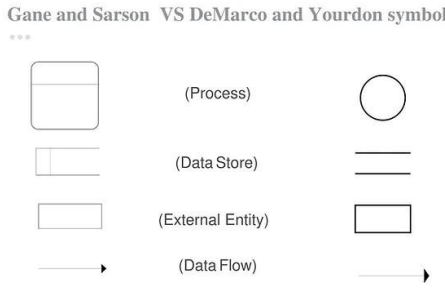
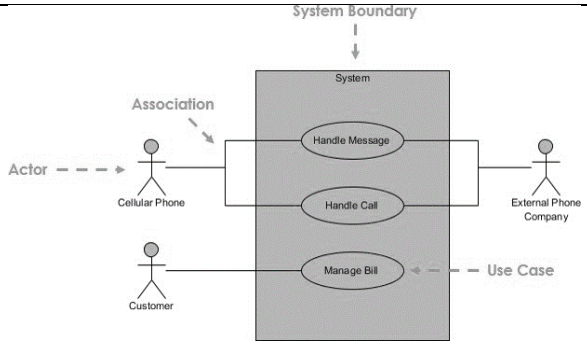
DFD nations	UCD nations
	
From ART BUSINESS ANALYSIS https://www.artofba.com/post/data-flow-diagrams-dfd-explained	From visual paradigm website https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

Table 2. Comparison between DFD & USD)

	Data Flow Diagram (DFD)	Use Case Diagram (UCD)
History	proposed by Larry Constantine (1970s).	Proposed by Ivar Jacobson (1986).
View	Functional view from system's perspective.	Functional view from actor's perspective.
Purpose	Primarily used for representing the flow of data within a system. It focuses on processes, data stores, and data flow between them.	Primarily used for capturing system functionalities and interactions between external entities (actors) and the system.

Abstraction Level	Provides a high-level view of the system's data flow and processes, allowing for abstraction and decomposition.	Offers a high-level view of system interactions without delving into internal system details.
Components	Involves processes, data stores, data flows, and external entities.	Involves actors, use cases, associations, and system boundaries.
Representation of Processes	Processes are represented as circles or rounded rectangles, illustrating transformations of inputs into outputs.	Processes are not explicitly represented; rather, use cases depict system functionalities.
System Boundary	The scope is defined implicitly by the diagram structure.	The system boundary is explicitly represented, encapsulating all use cases.
Handling Large Systems	Can become complex and crowded when dealing with large systems, especially with numerous sub-functions.	Generally maintains simplicity but might lack the detailed process view needed for complex systems.
External Entities/Actors	External entities are represented as squares or ovals, indicating individuals or groups outside the system.	External entities are represented as stick figures, emphasizing their role in interacting with the system.
Data Stores	Includes open-ended rectangles to represent data stores.	Does not explicitly represent data stores; data-related aspects are often abstracted.
Communication	Emphasizes the flow of data between processes, offering insights into system functionality.	Emphasizes the interactions between actors and the system, providing a user-centric view.
Use in Development Lifecycle	Often used during the system design and analysis phase, providing a structured view of data flow.	Frequently employed during the requirements gathering phase, aiding in communication between stakeholders.
Advantages	Simplicity and clarity. Focus on Processes and the flow of data between them. Hierarchical Structure, allowing for a top-down approach to system analysis. Ease of updates as the system evolves or changes. Integration with other modeling techniques and methodologies.	Easy to understand a clear high-level view of system functions and user interactions. Serves as an effective communication tool between system developers and stakeholders. Focus on user interactions and representing the system from the user's point of view. Useful in defining system boundaries.
Disadvantages	Limited detail is needed for a comprehensive understanding of complex systems. Ambiguity in the interpretation of symbols and processes. Leads to misunderstandings if not properly documented or explained. Not Suitable for small systems Doesn't capture the timing of processes. May not capture control flow. Potential for inconsistency may arise if there is a lack of standardization leading to confusion and misinterpretation.	Simplicity can lead to oversimplification neglecting important details. Limited to functional aspects and may not capture non-functional requirements. Do not provide details about the behavior or sequence of steps. Not suitable for large systems to capture all interactions. Ambiguity in relationships between actors and use cases can sometimes leading to misunderstandings. Do not represent the dynamic aspects of the system's behavior during runtime. Overemphasize user interactions and neglect interactions between system components or subsystems.
Summary	- Neither DFD nor UCD is inherently more important; their significance varies based on project needs and development phases. - Combining both diagrams offers a comprehensive understanding of the system from technical and user-centric perspectives.	

Conclusion

Upon a comprehensive review of existing literature (refer to the table no.), it is evident that the data flow diagram (DFD) significantly contributes to comprehending the dynamic exchange of data between external sources, functions, and data stores. The DFD serves as an easily understandable visual representation, actively involving end users in grasping the system's evolution and effectively eliciting functional requirements.

On the other hand, the use case diagram emerges as a valuable tool for elucidating the intricate interactions between system users and functions. It not only provides clarity on these interactions but also facilitates the seamless integration of end users into the system development process, ultimately enhancing the overall user experience (UX).

In examining the drawbacks associated with each diagram, a noteworthy observation is that the synergistic use of both models effectively mitigates these limitations. By leveraging the strengths of the data flow diagram in depicting data flow and the use case diagram in illustrating user-system interactions, a more comprehensive understanding emerges. This integration not only addresses the individual shortcomings but also enhances the overall effectiveness of system analysis and design.

In conclusion, the judicious combination of data flow diagrams and use case diagrams proves to be a strategic approach, leveraging their respective strengths to create a holistic representation of system functionality. This integrated methodology not only overcomes individual limitations but also fosters a more robust framework for system development and user engagement.

References

Aleryani A. (2020a). A data analysis perspective by the Business Analyst and Data Scientist; *International Journal of Scientific and Research Publications (IJSRP), 10*(09). http://dx.doi.org/10.29322/IJSRP.10.09.2020.p10525

Aleryani, A. (2020b). The Impact of the User Experience (UX) on the Quality of the Requirements Elicitation. *International Journal of Digital Information and Wireless Communications (IJDIWC), 10*(1), 1-9.

Arpan, K. & Yogesh, D. (2020). Theory building with big data-driven research – Moving away from the "What" towards the "Why". *International Journal of Information Management, 54*.

ART BUSINESS ANALYSIS via https://www.artofba.com/post/data-flow-diagrams-dfd-explained

Jilani, A., Usman, M., Nadeem, A., Malik, Z., & Halim, Z. (2011). Comparative Study on DFD to UML Diagrams Transformations. *World of Computer Science and Information Technology Journal (WCSIT), 1*(1), 10-16.

Coronel, C. & Morris, S. (2018). *Database Systems: Design, Implementation, and Management* (13th ed.).

Donald S. (2015). Understanding Data Flow Diagrams, file:///C:/Users/arwaa/Downloads/Understanding_Data_Flow_Diagrams.pdf

Elmasri, R. & Navathe, S. (2017). *Fundamentals of Database Systems* (7th ed.).

Ganesh, R., & Prabu, D. (2020). Determination of Internet Banking Usage and Purpose with Explanation of Data Flow Diagram and Use Case Diagram. *International Journal of Management and Humanities, 10*(1). http://sdiwc.net/digital-library/the-impact-of-the-user-experience-ux-on-the-quality-of-the-requirements-elicitation

Kazi, Z., & Kazi, L. (2022). Software Project Duration Estimation Based on COSMIC Method Applied to Data Flow Diagram. *Int. Arab J. Inf. Technol., 19*, 639-651.

Wulandari, W. & Widiantoro, A. (2017). Design Data Flow Diagram for Supporting the User Experience in Applications. *International Journal of the Computer, the Internet and Management, 25*(2), 14-20.

Kendall, K. E. & Kendall, J. E. (2013). *System Analysis and Design* (9th ed.). Prentice Hall.

Munassar, N. & Govardhan, A. (2011). Comparison between Traditional Approach and Object-Oriented Approach in Software Engineering Development. *IJACSA, International Journal of Advanced Computer Science and Applications, 2*(6).

Ober, I. (2000). More meaningful UML Models. *TELELOGIC, 2000 IEEE Journal.*

Jacob, P., Ilyas, M., Jose, J., & Jose, J. (2016). An Analytical approach on DFD to UML model transformation techniques. *International Conference on Information Science (ICIS),* Kochi, India, 2016, pp. 12-17. http://dx.doi.org/10.1109/INFOSCI.2016.7845292.

Tiwari, K., Tripathi, A., Sharma, S., & Dubey, V. (2012). Merging of Data Flow Diagram with Unified Modeling Language. *International Journal of Scientific and Research Publications, 2*(8).

Visual paradigm website via https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

Yourdon, E. (1989). *Modern Structured Analysis*. Yourdon Press. Upper Saddle River, NJ.