Merging of Data Flow Diagram with Unified Modeling Language

Kirti Tiwari, Alpika Tripathi, Shipra Sharma, Vandana Dubey

Department of Computer Science, Amity University, Lucknow, India

Abstract- In this paper we present an approach that combines the DFD with the UML diagrams. DFD is a structured approach which provides the functional view of the system whereas UML is an object oriented approach provides the structural view of the embedded system. Structured methods are very commonly used by the developers and if there is need to expand the functionality of the systems then object oriented approach is used which is very useful. So the combination of both approaches will be beneficial for the developers. For this we merge Data Flow Diagrams (Major tool of Structured Approach) with Unified Modeling Language diagrams (Use-Case & Class Diagrams).

Index Terms- Class Diagram, DFD (Data Flow Diagram), IOD (Initial Object Diagram), Object Oriented Approach, Structured Approach, UML (Unified Modeling Language), UCD (Use-Case Diagram)

I. Introduction

The structured methods are used to represent the dynamic and behavioral view of the system. It is a tool for system analysis that designs the system as a hierarchy of functions. The structured approach is based on graphical tools (such as DFD, decision tree, decision tables etc.) application. The most important diagram used in structured approach is Data Flow Diagram (DFD). Other diagrams like Structure chart, State machine, ER Diagrams are not very useful. DFD's are having certain advantages over them. The hierarchal structure of DFD provides different abstraction level which is very useful in system designing [1].

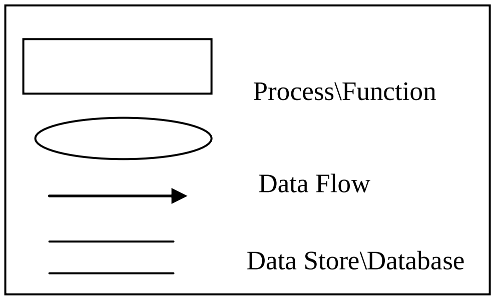

Figure1. Symbols used in DFD

DFD is a graphical representation that consists of nodes and directed arcs in Fig1. A node may be a data store, an auxiliary node, a process or a terminator that is either an input or an output of the system. Arcs correspond to data flows, are represented by the arrows. The designer must label the nodes and arcs. DFD is a language very widely used for process modeling in structured requirement analysis. DFD reflect the structure of the system and are also used for the stepwise refinement of a system [2].

The object oriented approach combine data and associated methods in to a single unit called object for developing complex systems. Unified Modeling Language (UML) has now become the most important tool for model Object-Oriented systems and model creation. UML is a collection of diagrams that is used to model the different aspects of object oriented software. UML is widely used for modeling complex systems [3]. In our approach, we are using Use-Case diagrams and Class diagrams to combine with DFD.

A. Use Case Diagrams

Use-Case diagrams are very important tool of UML and the developers can rely upon Use-Case diagrams for analysis of the system. It is very simple to understand hence can be very useful in requirement gathering. The Use-Case diagrams separate the system into actors and use cases in Fig 2. Actors represent roles of the users of the system. These users can be humans, other software systems, other computers or pieces of hardware. The actors must be external to the part of the system that is partitioned into use cases. The Use–Case diagram is a very useful tool for describing the behavior of the system. This behavior is described in a textual manner [4].

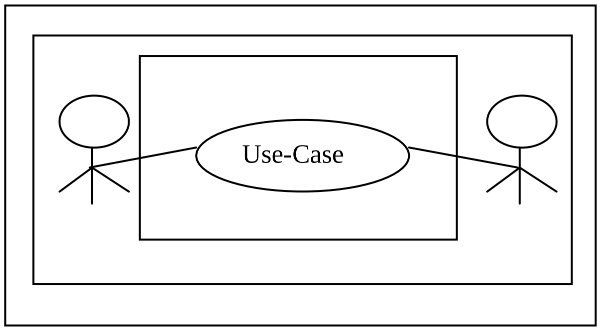

Figure2. A Basic UCD

B .Class Diagrams

Class diagrams are one of the most important tools for object-oriented design. UML Class Diagram is used to represent the system's static structure. The purpose of this diagram is to depict the classes with in a model [4]. In most UML models these types include:

- a class
- an interface
- a data type
- a component

A class is represented by a rectangle containing three compartments stacked vertically Fig 3. The top compartment shows the class's name. The middle compartment shows the class's attributes. The bottom compartment shows the class's operations. When drawing a class element on a class diagram [5].

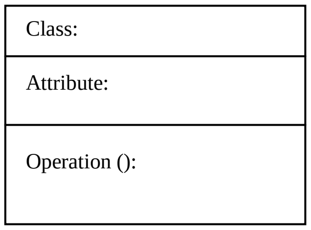

Figure3. A Basic Class Diagram

II. BACKGROUND

Researchers have already studied and presented the work related to the integration of DFD and UML. We are only mentioning some of the important approaches are-

Tran, Khan, Lan proposed a framework that works on the 3 levels of DFD. At first level of abstraction it transforms the DFD into UCD. At second level of abstraction framework transforms DFD into Interaction Diagram. At third level of abstraction ER Diagrams are used in place of DFD & framework transform ER Diagram into UML Class diagram. At first level of abstraction it mapped the process in DFD into Use-Case, external entities into actor and Data Store into Classes. Data Flow Mapping is not resolved by the framework. At level-2 Data Flow variations are transformed into UML interaction Diagram. At 3rd level components of ER diagram are including entity, association entity, attributes and relationships are mapped to class association, attributes and operations [6].

Shiroiwa, Miura, Shioya proposed a Meta model approach, in this approach authors propose DFD meta model that describes the DFD semantics formally. Author also discuss the generation of UML models (Use-Case Diagram, Class Diagram, Sequence Diagram) using the DFD meta model. Proposed meta model is used as a mediator between DFD and UML [7].

Tan, Yang, Blan proposed a systematic transformation of functional analysis into Object-oriented design and implementation. In this approach, an enhanced data flow diagram called DF net was proposed which is used to specify use cases from requirements. The DF net is transform to generate object oriented designs. Transformation between DF net was done in different steps. In the first step processes dealing with data stores executed through DF net are grouped together with external entities and data buffers and processes that are sharing same data are grouped together. In the next step classes are generated from different groups [8].

Truscan, Fernandes, Lilius [9] propose a model based transformation to integrate structured and object oriented approach for modeling the embedded systems. They propose a software modeling workbench (SMW) that gathers requirement, create Use-Case diagrams and transform it into non UML Initial Object Diagrams. Then it transforms IOD into DFD & DFD into class diagram Fig 4. His approach was not very easy to implement for other systems. He presents a design flow that has several phases.

The phases are:

a.) He Extract the application requirements
b.) Then Creates the Use-Case Diagram
c.) After creating Use-Case he specifies each Use-Case in a textual manner
d.) Then the UCD is transformed into an IOD
e.) Based on their functionality of the objects, group, and split and discard them to refactor the IOD.
f.) IOD is then transformed into the DFD
g.) Build a data dictionary by identifying the data flows
h.) Using Activity Diagram specify the behavior of the process
i.) DFD is transformed into the object diagram
Or
j.) DFD view is transformed into the Class diagram

The proposed model was specific only for the IPV6 Router. It was specifically designed for IPV6. Rules are specific only for IPV6 case study. The author merge the DFD with UCD, Initial object diagram, Activity Diagram, Class Diagram and a DFD like object diagram. Approach is very complicated to implement for the other complex systems.

Some other approaches are also there but they are not very much useful and practical.

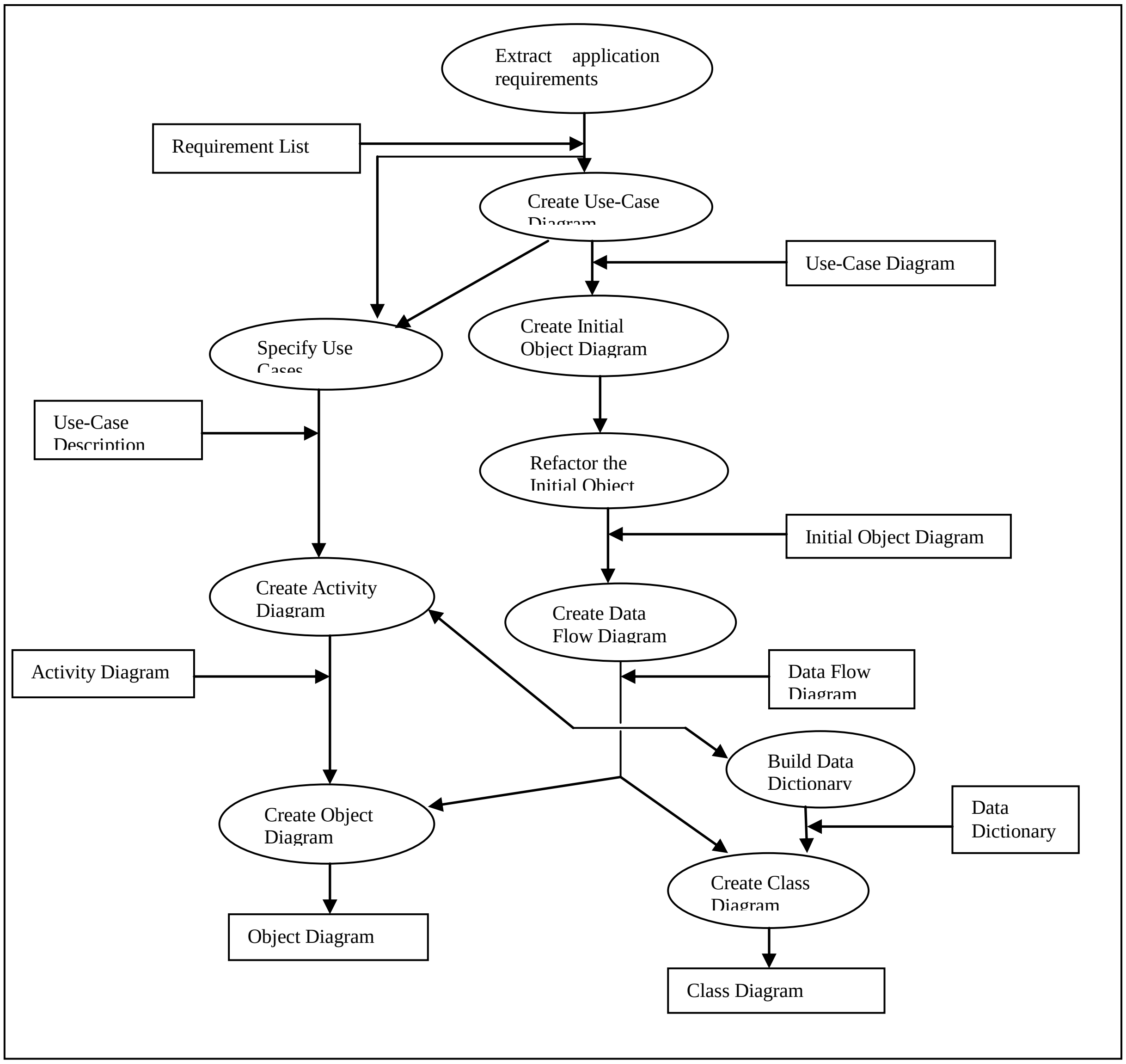

Figure4. Model for DFD-UML integration

III. MERGING OF DFD WITH UCD & CLASS DIAGRAM

Our basic approach is to combine the DFD with the UML diagrams (UCD & Class diagram) as an extension of UML model Fig 4.

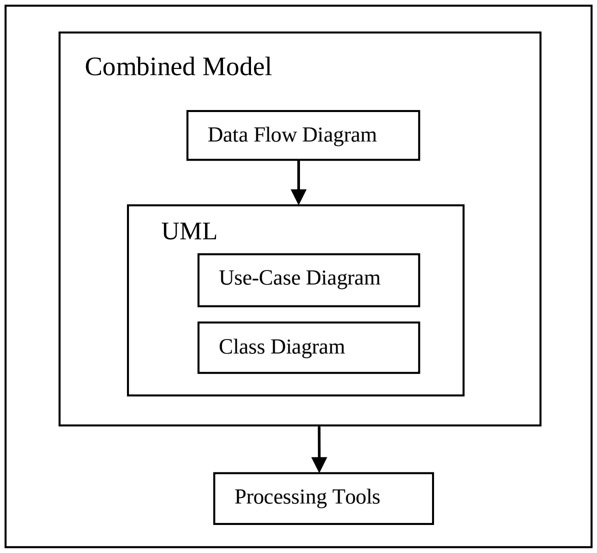

Figure5. Combined Model

In this approach we have merged the DFD with the UCD and Class diagrams. In first step we have chosen the Use-Case diagram as the basic diagram to analyze the system under consideration Fig 5. UCD is the most sufficient diagram to analyze the system. Therefore we have started our process with UCD.

In next step Use-Case diagrams are transformed into the basic level DFD Fig 6. Since the actors in UCD represents the user of the system hence they are transformed into the Data Stores and sinks, Use-Cases describes the behavior of the system and action that has taken place hence very much similar to the processes in DFD. Therefore Use-Cases are transformed into Processes and Association lines are transformed into Data Flows. DFD's can represent the behavior and architecture of the analyzed system very well. To represent the system components and system behavior in detail Basic level DFD's are not sufficient therefore we will transform the basic level DFD into the context level DFD to express the system more specifically. Context level DFD will give the developers more clear view of the system.

A. Transformation of UCD into DFD:

$$\begin{tikzcd}
\text{Actors} \arrow[r] & \text{Data Stores, sinks} \\
\text{Use Cases} \arrow[r] & \text{Process} \\
\text{Association Lines} \arrow[r] & \text{Data Flows}
\end{tikzcd}$$

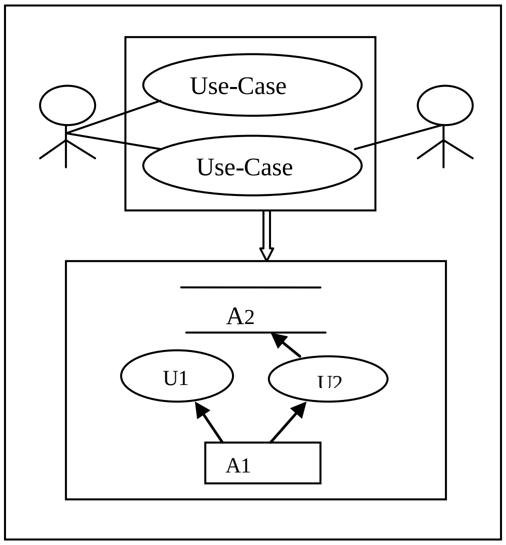

Figure6. UCD to DFD transformation

The basic level DFD is elaborated into context level DFD for the refinement of the system and then transformed into the Class diagram Fig.7. In Class diagrams data plays the central role. We will classify and encapsulate the data into classes along with their corresponding methods. The Data store and terminators are transformed into classes, processes are transformed into the member function since they define the action and Data flows are used as the association between the classes.

B. Transformation of DFD into Class Diagram:

Data Stores →	Classes
Terminator →	Classes
Process →	Member Function
Elements of Data stores →	Attributes of Classes
Data Flow →	Association between Classes

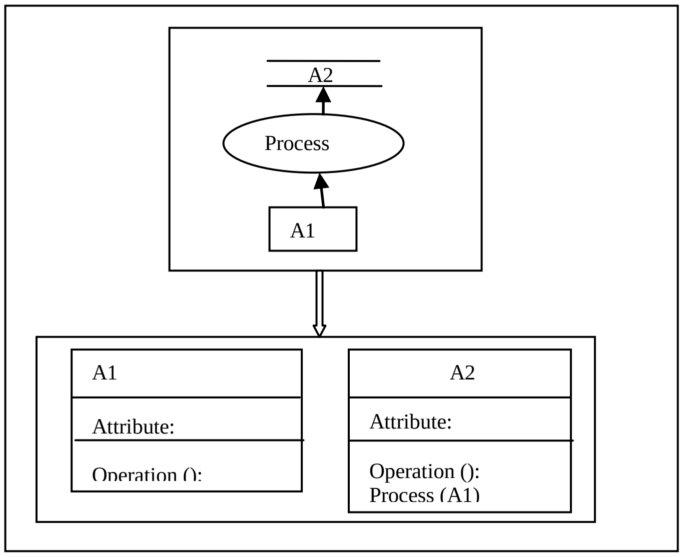

Figure6. DFD to Class Diagram transformation

Since both the UML and DFD models are having user friendly view, the approach of merging both models will be very useful to get the advantage of both models to express the important features of the system. We have used Use-Case diagrams because it is easy to read notation and simplicity. DFD's are much more expressive then Use-Case diagrams and hence they can be used to represent the behavior and architecture of the analyzed system. Class diagrams are very much useful in describing the static structure of the system and can be very convenient for the developers to get the full view of the system.

IV. CONCLUSION

This paper present the merging of Data Flow Diagram with UML diagrams. We developed a model in which the Use-Case diagrams and Class diagrams re to be combined with the Data Flow diagrams. Use Case diagrams are very reliable to analyze the system and represent the dynamic view of the system. We transform Use Case diagram into basic level DFD. It is more expressive then UCD. To represent the system components and system behavior in detail basic level DFD is transform to context level DFD and then to Class diagram. It will represent the structural view of the system. It is just a theoretical approach. In future it can be implement on some complex applications. It can also be modified by adding more UML diagrams for the specific systems if required.

ACKNOWLEDGMENT

The authors are very thankful to their respected Mr. Aseem Chauhan, Additional President, Amity University, Lucknow, Maj. Gen. K.K. Ohri, AVSM (Retd.), Director General, Amity University, Lucknow, India, for providing excellent computation facilities in the University campus. Authors also pay their regards to Prof. S.T.H. Abidi, Director and Brig. U.K. Chopra, Deputy Director, Amity School of Engineering, Amity University, Lucknow for giving their moral support and help to carry out this research work.

References

[1] Atif A. A. Jilani, Muhammad Usman, Aamer Nadeem,Zafar I. Malik,Zahid Halim,"Comparitive Study on DFD to UML diagrams Transformations", journal of WCSIT, vol. 1,no. 1,10-16,2011.
[2] Fanchao Meng, Dianhui Chu, Dechen Zhan, "Transformation from Data Flow Diagram to UML2.0 Activity Diagram",1/10, 2010 IEEE Journal.
[3] Ileana Ober, "More meaningful UML Models", TELELOGIC, 2000 IEEE Journal.
[4] UML Use Case Diagrams, Engineering Notebook, C++ Report, Nov-Dec 98.
[5] *"An introduction to structure diagrams in UML 2"*,Donald Bell , IT Architect, IBM Corporation.

[6] Tran,T. N., Khan, K. M.,Lan,Y.C.(2004). "A framework for transforming artifacts from Data Flow Diagrams to UML". In: Proceedings of the 2004 IASTED International Conference on Software Engineering (Innsbruck, Austria, Feb. 17-19, 2004). ACTA Press, Calgary, AB, Canada IASTED International Conference on Software Engineering (Innsbruck, Austria, Feb. 17-19, 2004). ACTA Press, Calgary, AB, Canada.

[7] Shiroiwa, M., Miura, T., Shioya, I.(2003). "Meta Model approach for mediation", In: Proceedings of the 27th Annual International Computer Software and Applications Conference (COMPSAC'03) IEEE Press, 480-485, New York.

[8] Tan, H.B.K., Yang, Y., Blan, L. (2006 February). "Systematic Transformation of functional analysis model in Object Oriented design and Implementation." IEEE Transaction on Software engineering, 32(2), Pp111-135.

[9] Truscan, D., Fernandes, J.M., Lilius, J. (2004). "Tool Support for DFD-UML based transformation." In: Proceedings of the IEEE International Conference and workshop on the engineering of Computer-Based Systems (ECBS'04) (Brno, Czech Republic, May 24-27, 2004), Pp 378-397 IEEE Press, New York.

[10] UML (Unified Modeling Language): Superstructure Version 2.1.2 Object Management Group (OMG). Document 07-11-02. Pdf from http://www.omg.org/spec/UML/2.1.2/ [Accessed: March 1,2008].

[11] Soon-Kyeong Kim and David Carrington, "A Formal Object-Oriented Approach to defining Consistency Constrraints for UML Models", Proceedings of the 2004 Australian Software Engineering Conference(ASWEC'04),2004 IEEE.

AUTHORS

First Author– Kirti Tiwari has completed her M.Tech in Computer Science Engineering from Amity University, Lucknow in 2012 and B.Tech degree from Integral University, Lucknow in 2009. Her area of interest includes Software engineering and Networking. Email ID- kirtitiwari9@gmail.com.

Second Author – AlpikaTripathi received M.Tech in Computer Science from Amity University Lucknow in 2010. She is currently working as a Lecturer in Department of Computer Science Engineering, Amity University, Lucknow. She has guided number projects and thesis in graduate and post-graduate level program. She has produced several national and international publications.
Email ID- alpika2k@gmail.com.

Third Author – Shipra Sharma received M.Tech. in Computer Science Engineering from Amity University Lucknow in 2010. She is a Lecturer in Department of Computer Science Engineering, Amity University, Lucknow. She has guided number projects and thesis in graduate and post-graduate level program. She has produced several national and international publication. Her research interests include Wireless Sensor Network, Software Engineering and Artificial Intelligence.
Email ID- shipra.sharma1510@gmail.com.

Fourth Author –Vandana Dubey received M.Tech in Computer Science from Amity University, Lucknow in 2010. She is currently working as a Lecturer in Department of Computer Science Engineering, Amity University, Lucknow. Her research interests include Advanced Computer architecture and Computer Organization.
Email ID- vandanashuklaec05@gmail.com.

Correspondence Author –Kirti Tiwari has completed her M.Tech in Computer Science Engineering from Amity University, Lucknow in 2012 and B.Tech degree from Integral University, Lucknow in 2009. Her area of interest includes Software engineering and Networking.Email ID- kirtitiwari9@gmail.com