

“TRANSFORMING DATA FLOW DIAGRAM TO USE CASE DIAGRAM”

Shinde Ajaykumar D.*

Dr. M.S. Prasad**

Abstract:

In this paper we present an approach that transforms the DFD to the Use Case diagram of UML. DFD is a structured approach which provides the functional view of the system whereas Use Case diagram is an object oriented approach provides the functional view of the system under consideration. 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 transformation of one approach to the other will be beneficial for the developers. For this we present an approach that will transform Data Flow Diagrams (Major tool of Structured Approach) with Use case diagram.

Key Words: DFD (Data Flow Diagram), Object Oriented Approach, Structured Approach, UML (Unified Modeling Language), UCD (Use Case Diagram)

* Dept. of Computer Studies, CSIBER, Kolhapur, Research Student

** Professor, Bharati Vidyapeeth's Institute Of Management And Entrepreneurship Development, Erandavane, Pune

1. Introduction

Many organizations are using the software that was designed and developed at least a decade ago. The approach used to design and develop these systems was procedure oriented and the same approach was reflected in documents [18]. The procedure oriented approach has become outdated and the focus is shifting to object oriented. As replacing the existing software puts extra burden on the users in terms of cost. Many customers are continuing with existing software. It is possible to implement changes in the code so as to shift software from procedure oriented to object oriented, but this will create a new problem document and code can not match. The only feasible solution for up-gradation and maintenance is to preserve system design and incorporate it with latest software development strategies [7]. It is possible to generate design using reverse engineering with the help of available code. But if frequent changes are made to code it becomes inconsistent with the design. The user also feels the original system is irreplaceable and trustworthy [9].

In procedure oriented approach DFD is treated as main artifact for system representation. A DFD is must for each and every system designed using procedure oriented approach. The main advantage in using DFD is it shows dynamic approach of the system [9]. Procedure oriented is being replaced by object oriented approach and is becoming the only approach for design and development. Many organizations are shifting from procedure oriented to object oriented approach [11]. . For design of object-oriented systems and creation of model, Unified Modeling Language[17] has now become the industry standard [2][3]. UML is a collection of diagrams used to represent different aspects of the system under consideration. UML allows us to represent static structure of the system as well as dynamic behavior of the system.

In [12] Liu and Wilde, have proposed type base and global base object finder methodologies for identifying object from non-object-oriented languages. In [8] Jacobson and Lindstrom, discuss reverse engineering strategies for object-oriented model to incorporate changes. Newcombe and Kotik [13] present a tool for abstract object-oriented model generation. Subramanian and Bwirne [15] generate objects from FORTRAN code. They discuss constraints like private, virtual, and pure virtual. Cimitile and others [4] and De Lucia and others [5] present approaches that revolve around data stores. Authors propose approaches that consider functions and subroutines, interacting with tables, data-store and use them as objects methods. De Lucia and others in [6] propose an approach to recover class diagram from system code that is highly data intensive.

From the above discussion it is clear that all the techniques are dependent on code. In our approach, we are more interested in procedure oriented design than code. In design, we have observed that in literature, both structured design to non-UML design and structured design to UML design transformations exist.

2. Data flow diagram and Use case diagram

a) Notations used in Data Flow Diagram

The notations for DFD were proposed and popularized by Yourdon, DeMarco, and others are described below:


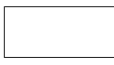
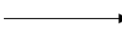

symbol	Name	purpose
	processes	To show work to be carried out in the system
	Source and consumer of data	To show external users of system
	Data flow	To show data flowing through the system
	Data store	To show where the data is stored

Table 1 : symbols used to draw DFD

b) Use Case Diagram

The Use case diagram is drawn to identify the primary elements and processes that form the system. Primary elements are termed as "actors" and the processes are called "use cases", Use case diagram shows which actors interact with each use case.

A use case diagram captures the functional aspects of a system. More specifically, it captures the business processes carried out in the system. As we discuss the functionality and processes of the system, we discover significant characteristics of the system that we model in the use case diagram. Due to the simplicity of use case diagrams, and more importantly, because they are shorn of all technical jargon, use case diagrams are a great tool for user meetings. Use case diagrams have another important use. Use case diagrams define the requirements of the system being modeled and hence are used to write test scripts for the modeled system.

A use case diagram is quite simple in nature and depicts two types of elements: one representing the business roles and the other representing the business processes. Let us take a closer look at use at what elements constitutes a use case diagram.


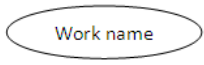

Element	Name	Meaning
 <<role>>	Actor	External user of the system
	Use case	Work carried out in the system/ functionality expected by user
	connector	Connection between user and use cases

Table 2 : Symbols used in Use case diagram

3. Representation of DFD in Framework

All transformation processes are dependent on representation of DFD. In order to simplify the transformation process we have designed and used a framework [14]. A data flow diagram uses very limited number of symbols and may be represented as a set of symbol sets. In the framework the DFD is represented as a graph using atomic relational grammar [1]. In the framework [14] DFD is defined as

DFD = $\{\{SS\}, \{PS\}, \{DS\}, \{TS\}, \{RS\}, \{PR\}\}$ where,
DS represents set of data flows

PS represents set of processes that may be either atomic or aggregate

TS represent set of data stores

SS represent set of source consumers

RS represents the set of relationships

PR represents the set of productions

The framework treats DFD as directed graph in which the sets $\{SS\}$, $\{PS\}$, $\{TS\}$ are the vertices and the set $\{DS\}$ is the set of edges. $\{RS\}$ is a set of relationships between the atomic elements of DFD and $\{PR\}$ is set of productions derived from elements of $\{RS\}$. A member of the set $\{SS\}$ will be a start symbol [14].

a) Transformation of DFD to Use Case Diagram

For transformation of DFD to use case diagram strategy is defined in this paper. Every transformation strategy should be based on concrete rules [10][16]. The transformation strategy presented in this paper is also based on rules. As all diagrams in procedure oriented and object oriented design are represented as graphs. We have designed a strategy that is based on patterns, as patterns strategy deals with graphs and representation of graph is based on concrete or abstract syntax of source or target model language [16]. The transformation strategy adopted in this paper constructs a intermediate model using tagged language [14]. Transformation rules are framed and applied to entire model rather than a specific location in the model. Application of the transformation rule generates a new model; same set of rules is applied iteratively to all matching locations in the source model. The transformation rules are applied in phases where each phase has a specific purpose and it invokes a definite rule of transformation. The transformation rules are organized according to the source language and target language i.e. DFD and use case diagram. The rule application strategy used here is unidirectional [16] i.e. transformation rule are used to transform source model to the target model reverse is not possible.

The transformation strategy adopted in this paper is Hybrid; it is a mixture of direct manipulation, relational approach and graph transformation approach [16]. In direct manipulation approach API and internal model representation is provided. In relational approach the type of the element is specified using relational constraints. This approach also has relational specification and mapping rule. In graph transformation approach LHS and RHS sides are used. The LHS pattern is matched in the model being transformed and replaced by the RHS pattern. As these properties are incorporated in the framework, it becomes hybrid strategy for transformation [16].

The transformation strategy designed in this paper starts with the scanning of existing model/graph. The scheme adopted for transformation is model-to-model mapping of symbols.

This approach offers an internal model representation plus some API to manipulate it. Since the diagram is represented internally as distinct sets of symbols in the framework, each symbol set is scanned separately. Framework goes on scanning each and every element from DFD and identifies its type. The scanning process identifies attributes of the elements and is very essential process for mapping the symbols from DFD to use case diagram.

As the first step of transformation, the framework starts scanning source consumer set(SS set). For each source and consumer in the data flow diagram a new actor is added to the actor set in use case diagram. The source consumer in DFD and actors in use case diagrams represent the external users of the system. The attributes of source and consumer from DFD are recorded as attributes of actors in use case diagram. After completion of scanning of the source and consumer set the frameworks starts scanning the process set.

In the second step of transformation, process set(PS set) is taken. For each process in process set of DFD framework adds a new use case in the use case set of use case diagram. While scanning the process set the framework looks for the expansion attribute of the process in DFD. If the expansion attribute of process is true, it implies that the process is expanded in the next level of the DFD, i.e. the process expanded is made up of many other sub processes. For each level of the DFD the scan process is applied iteratively to the next level of the DFD. The framework opens the expanded DFD and starts scanning it. For each process in DFD a new use case is added to the use case diagram. In case of expanded processes the framework maintains a relationship between expanded and processes in new DFD as generalization. The generalization relationship is used to show relationship between more general and more concrete use case. The generalization relationship may be named as extends or uses depending upon, whether it adds something new to existing use case or it is integral part of existing use case.

In the third step of transformation, framework starts scanning relationships set(RS set). Relationship set has start and end attributes. If the relationship is either starting or ending on the data store the relationship is not recorded in relationship set of the use case diagram. This is because use case diagrams do not have data stores concept. For all the other relationships it goes on adding a new relation in the relation set in the use case diagram. The fourth step of scan starts on the data store set(TS set) of DFD. As data stores are not part of the use case diagrams this entire set is skipped. In the last step framework scans data flows; data flow coming into the data stores and the data flows that are coming out of the data stores(DS set) are omitted from the use case set. All the other data flows are stored as links in the use case set.

b) Algorithm for Transformation of DFD to Use Case

We have proposed an algorithm for transformation of DFD to use case diagram. It starts by reading the symbol set from DFD. It looks at the shapes and other characteristics of symbol to identify its type i.e. rectangle is source consumer of information, circle is process, arrow is a data flow with start and end characteristics, parallel lines represent data stores. This information is used for mapping the symbols from DFD to use case diagram.

Algorithm DFDtoUCD	//if circle in DFD is expanded
Input : DFD subsets {SS}, {PS}, {DS}, {TS}, {RS}	in next level
Output : use case set	Else
//start reading the DFD from first symbol set to	Get next level DFD
	Call algorithm recursively

<pre> last symbol set If symbol set is empty return else For each symbol in DFD set // if the symbol from DFD set is rectangle then add stickman symbol to use case set If symbol.type = source consumer then Add actor symbol to use case set. //if symbol from DFD is circle then add oval to use case set Else if symbol.type = process then // if circle in DFD is not expanded in next level If process.expanded = false then Add use case to use case set </pre>	<pre> //If the symbol from DFD is arrow Else if symbol.type = dataflow then //do not record links with datastore If dataflow.source != datastore or data store.destination != data store Then Record link to use case set //do not record links with datastore Else if relation.start != data store or relation.end != data store then Record relationship to the use case set //do not consider the datastore symbol for conversion Else if symbol.type = data store then Skip the symbol Get next symbol Endif </pre>
<p>Algorithm-1: Algorithm to transform DFD to Usecase diagram.</p>	

The algorithm starts reading the DFD. As the DFD is represented using sets, it starts reading these sets one by one. It reads the terminal symbol sets first. After reading the symbols from terminal set, it identifies the type of the symbol. Once the type is known, it finds out the mapping symbol from use case set and adds it to the use case. The mapping set is given in Table-3. For the data store symbol there is no equivalent symbol. All data store symbols and the links associated with data store symbol are skipped by the algorithm. The mapping process obtains the basic symbols required for drawing the use case diagram. The framework accepts these symbols and by looking at the link set goes on drawing the final use case diagram. The mapping symbol set used by the algorithm for transformation of DFD to use case diagram is given in Table 3. While mapping the symbols from DFD to use case the framework makes it sure that the symbol is not duplicated. This helps us in reducing the number of diagram elements and removing redundancy of element in the translated diagram.








DFD		Use Case Diagram	
Symbol	Meaning	Symbol	Meaning
	Source and consumer of data		User of the system
	Processes in the system		Use case in the system
	Data flow and its direction		Link between user & use case
	Storage of the data	No corresponding symbol available in use case model. The data store becomes class in class model.	

Table 3: Mapping of Symbol Set from DFD to Use Case Diagram

The mapping of the symbol is done by looking at the purpose of each symbol e.g. the process symbol in DFD use used to show the functionality performed by the system similarly the use case symbol in use case diagram is used to show functionality in the use case diagram. As the purpose of both the symbols is same the process symbol from DFD is mapped to use case symbol in use case diagram. The source and consumer of data in DFD plays the role of the user. Actor in the use case diagram is user of the system. The source consumer symbol is mapped to actor symbol in use case diagram. A data flow is a link between two elements in the DFD. The data flow in DFD is mapped to a link between the elements of the use case diagram. As use case diagram has no concept of data store all the data stores and their associated links are dropped from the use case set.

4. Examples for transformation of DFD to use case diagram

The framework [14] provides a toolbox to draw a data flow diagram. The elements from toolbox have to be selected by the users and paste on the drawing area. While connecting the symbol a line must start within the boundaries of the element and also must end in the boundaries of the element. The drawing procedure validates the connection between atomic elements of the diagram. i.e. if we try to connect sources and consumers to each other, error message is displayed and connection is rejected. All validations are based on the relationship between elements using ARG [1]. When the drawing is over the diagram is saved using a tagged language as shown in section 4(a).

a) Representation of DFD in Framework

The DFD is represented in the framework using tagged language. In this section we present tagged language generated by the framework for the Figure 1. The DFD is represented as multiset of symbols with attributes the symbol sets are written separately. A set of relationship is additional set written as a result of drawing it includes the binary relationships between the symbols in DFD.

Consider DFD drawn in Figure 1, it is represented in the framework as follows. Internal representation of DFD in figure 1 is given below it. It uses a tagged language for representation of DFD.

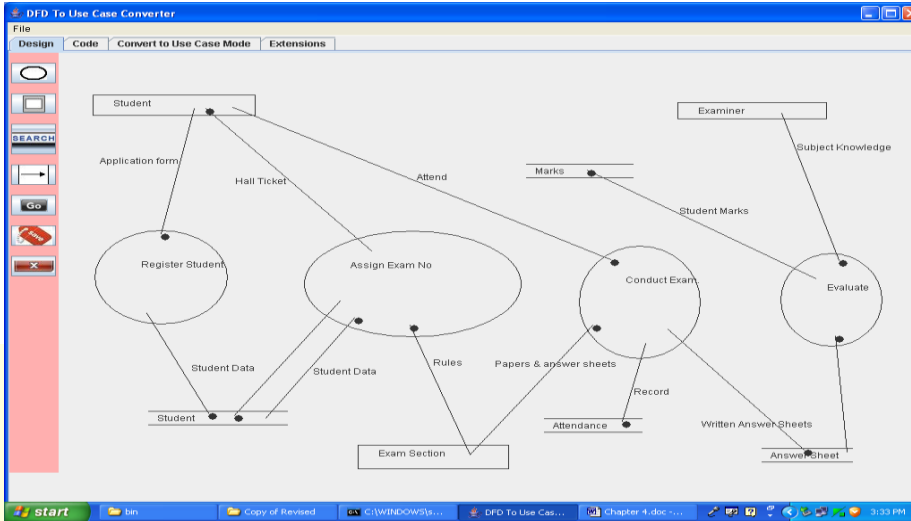


Figure 1 Level 1 DFD for Examination System

<pre> <DFD> <Source-Consumer> <%rectangle id=0 nm=" Student ", x=93 , y=62 , w=181 , h=30 %> <%rectangle id=1 nm=" Examiner ", x=746 , y=73 , w=166 , h=24 %> <%rectangle id=2 nm=" Exam Section ", x=389 , y=579 , w=168 , h=35 %> </Source-Consumer> <Relationship> <%relation start=Student, end=Register Student %> <%relation start=Assign Exam No, end=Student %> <%relation start=Student, end=Conduct Exam. %> <%relation start=Examiner, end=Evaluate %> <%relation start=Register Student, end=Student %> <%relation start=Assign Exam No, end=Student %> <%relation start=Student, end=Assign Exam No %> <%relation start=Exam Section, end=Assign Exam No %> <%relation start=Exam Section, end=Conduct Exam. %> <%relation start=Conduct Exam., end=Attendance %> <%relation start=Conduct Exam., end=Answer Sheet %> </pre>	<pre> <%Joiner x1=206 , y1=82 , x2=169 , y2=267 , nm="Application form" , x=101 , y=164 %> <%Joiner x1=404 , y1=292 , x2=219 , y2=82 , nm="Hall Ticket" , x=252 , y=194 %> <%Joiner x1=249 , y1=80 , x2=671 , y2=305 , nm="Attend" , x=455 , y=188 %> <%Joiner x1=862 , y1=89 , x2=926 , y2=306 , nm="Subject Knowledge" , x=879 , y=144 %> <%Joiner x1=153 , y1=384 , x2=222 , y2=532 , nm="Student Data" , x=203 , y=470 %> <%Joiner x1=369 , y1=366 , x2=251 , y2=535 , nm=" " , x=0 , y=0 %> <%Joiner x1=286 , y1=539 , x2=385 , y2=391 , nm="Student Data" , x=339 , y=475 %> <%Joiner x1=514 , y1=593 , x2=447 , y2=402 , nm="Rules" , x=473 , y=462 %> <%Joiner x1=514 , y1=593 , x2=651 , y2=402 , nm="Papers & answer sheets" , x=542 , y=464 %> <%Joiner x1=710 , y1=429 , x2=684 , y2=544 , nm="Record" , x=697 , y=505 %> <%Joiner x1=735 , y1=408 , x2=887 , y2=586 , nm="Written Answer Sheets" , x=773 , y=553 %> <%Joiner x1=935 , y1=589 , x2=922 , y2=418 , nm=" " , x=0 , y=0 %> <%Joiner x1=900 , y1=333 , x2=645 , y2=173 , nm="Student Marks" , x=748 , y=238 %> </Line> < DataBase> <%Database nm="Marks", x1=577 , y1=164 , x2=696 , y2=164 %> </pre>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<pre> <%relation start=Answer Sheet, end=Evaluate %> <%relation start=Evaluate, end=Marks %> .</Relationship> <Process> <%oval id=0 nm="Register Student", x=95 , y=262 , w=148 , h=138 %> <%oval id=1 nm="Assign Exam No", x=329 , y=263 , w=242 , h=156 %> <%oval id=2 nm="Conduct Exam.", x=636 , y=285 , w=135 , h=166 %> <%oval id=3 nm="Evaluate", x=861 , y=296 , w=113 , h=137 %> </Process> <Line> </pre>	<pre> <%Database nm="Student", x1=155 , y1=529 , x2=310 , y2=529 %> <%Database nm="Attendance", x1=597 , y1=540 , x2=706 , y2=540 %> <%Database nm="Answer Sheet", x1=840 , y1=584 , x2=941 , y2=584 %> </DataBase> <ID relationship> <% sid=0 eid=0 %> <% sid=0 eid=1 %> <% sid=0 eid=2 %> <% sid=1 eid=3 %> <% sid=2 eid=1 %> <% sid=2 eid=2 %> </ID relationship> </DFD> </pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

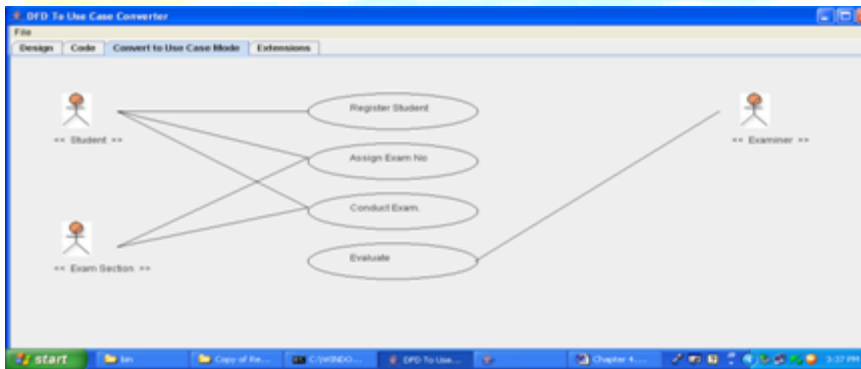


Figure 2 Conversion of Level 1 DFD to Use Case Diagram

The representation of DFD in the framework uses multi-sets of symbols. The tagged language records these sets separately using a definite format. The sets include set for source and consumers, set for processes, set for data stores, set for data flows and the data flow set is used to generate another set called relation set. The output of drawing of data flow is shown with the tagged language. Figure 2 shows conversion of level 1 DFD shown in figure 1. The conversion of DFD to use case diagrams uses the algorithm described in section 3(b).



Figure 3 Expansion of Process3 to Level 2 DFD

Figure 3 shows the expansion of process 3 from level 1 DFD. The expansion of a process in framework is done by selecting a process and clicking on 'Go' button in tool box.

b) Representation of Level2 DFD in Framework

Representation of level-2 DFD shown in figure 3 in the framework

<pre> <DFD> <Source-Consumer> <%rectangle id=0 nm=" Exam Section ", x=274 , y=431 , w=186 , h=51 %> </Source-Consumer> <Relationship> <%relation start=Exam Section, end=Start Exam %> <%relation start=Exam Section, end=Record Attendance %> <%relation start=Start Exam, end=Record Attendance %> <%relation start=Start Exam, end=Record Attendance %> <%relation start=Record Attendance, end=Attendance %> <%relation start=Record Attendance, end=End Exam %> <%relation start=Record Attendance, end=End Exam %> <%relation start=End Exam, end=Answer Sheets %> </Relationship> <Process> <%oval id=0 nm="Start Exam", x=181 , y=68 , w=144 , h=129 %> <%oval id=1 nm="Record Attendance", x=509 , y=97 , w=165 , h=141 %> </pre>	<pre> <Line> <%Joiner x1=290 , y1=439 , x2=249 , y2=182 , nm="Papers & answer Sheets" , x=129 , y=287 %> <%Joiner x1=434 , y1=446 , x2=570 , y2=222 , nm="Rules" , x=515 , y=322 %> <%Joiner x1=293 , y1=138 , x2=516 , y2=152 , nm=" " , x=0 , y=0 %> <%Joiner x1=531 , y1=192 , x2=425 , y2=310 , nm=" " , x=0 , y=0 %> <%Joiner x1=642 , y1=174 , x2=822 , y2=169 , nm="Time" , x=721 , y=167 %> <%Joiner x1=881 , y1=208 , x2=907 , y2=345 , nm="Return Answer Sheets" , x=895 , y=295 %> </Line> < DataBase> <%Database nm="Attendance", x1=350 , y1=306 , x2=490 , y2=306 %> <%Database nm="Answer Sheets", x1=845 , y1=341 , x2=979 , y2=342 %> </DataBase> <ID relationship> <% sid=0 eid=0 %> <% sid=0 eid=1 %> <% sid=-1 eid=1 %> </pre>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

```
<% oval id=2 nm="End Exam", x=814 ,
y=100 , w=155 , h=145 %>
</Process>
```

```
<% sid=-1 eid=2 %>
</ID relationship>
</DFD>
```

Figure 4 is transformation of level2 DFD shown in figure 3. The procedure for transformation is same as that of level1 DFD to use case diagram.



Figure 4 Transformation of expansion of Process3 from fig.

Figure 5 shows the relationship between process in level-1DFD and its expansion in level 2 DFD. The process in level 1 and its sub-processes are in relations. The only relationship possible between these processes is generalization. Figure 5 shows this relationship between a process and its sub-process using use case notations.

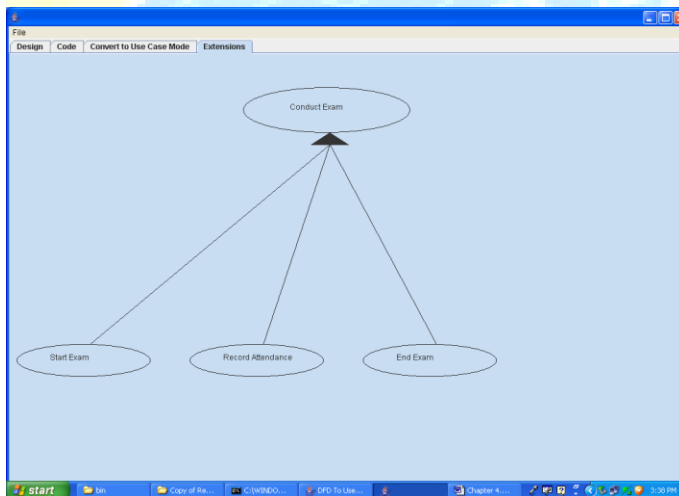


Figure 5 Relationship between Process3 and it's Expansion

Conclusion

As software industry is shifting from procedure oriented paradigm to object oriented. It is becoming essential to find out commonalities and differences between these two approaches. An attempt has to be made to establish connection between these approaches. The approach suggested in this paper is an attempt to establish connection between these approaches. The transformation strategy presented in this paper allows the user to draw DFD also it understands syntax and semantics of DFD. The main advantage of this framework is it stores diagram using tagged language which is easy to read and understand. On the other hand because storage is done in textual format it saves disk space. The transformation algorithm presented in this paper converts DFD to correct Use case diagram. The framework also understands relationship between leveled DFD's and this information is used to establish generalize relationship between use cases.

References :

- [1]. Antti-Pekka Tuovinen Object-Oriented Engineering of Visual Languages, PhD Thesis, Series of Publications A, Report A-2002-, Helsinki, February 2002, 185.
- [2]. Booch, G., Jacobson, Rumbaugh, J. I. (2005). Unified Modeling Language User Guide. (2nd Edition), Addison Wesley, Upper Saddle River, NJ.
- [3]. Booch, G., Rumbaugh, J., Jacobson, I. (1999). The Unified Modeling Language User Guide. Addison Wesley.
- [4]. Cimitile, A., De Lucia, A., Di Lucca, G.A., Fasolino, A.R. (1997). Identifying objects in legacy systems. In: Proceeding Of 5th International Workshop on Program Comprehension, Dearborn, Michigan, IEEE CS Press, pp. 138-147.
- [5]. De Lucia, G.A., Di Lucca, G.A., Fasolino, A.R., Guerra, P., Petruzzelli, S. (1997). Migrating Legacy Systems towards Object-Oriented Platforms. In: 13th International Conference on Software Maintenance ICSM'97.
- [6]. De Lucia, G.A., Fasolino, A.R., Carlini, U. D. (2000). Recovering Class Diagrams from Data-Intensive Legacy Systems. In: 16th IEEE International Conference on Software Maintenance ICSM'00.
- [7]. Dietrich, W., Nackman, I., Gracer, L. (1989), Saving a legacy with objects. In Proceedings of OOPSLA, pp. 77-88.
- [8]. Jacobson, I., Lindstrom, F. (1991). Re-engineering of old systems to an object-oriented architecture. In: Proceedings of OOPSLA, pp. 340-350.
- [9]. Kendall, K. E., Kendall J. E. (1999). System Analysis and Design. (3rd Edition), Prentice Hall.
- [10]. Krzysztof Czarnecki and Simon Helsen, *A classification of model transformation approaches* OOPSLA' 03 Workshop on Generative Techniques in the Context of Model-Driven Architecture.
- [11]. Larman, C. (2004). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design. (3rd Edition), Addison Wesley Professional.
- [12]. Liu S., Wilde, N. (1990). Identifying objects in a conventional procedural language: an example of data design recovery. In: Proceedings of Conference on Software Maintenance, San Diego, CA, IEEE CS Press, pp. 266-271.
- [13]. Newcombe, P. and Kotik, G. (1995). Reengineering procedural into object-oriented systems. In: Proceeding of 2nd Working Conference on Reverse Engineering, Toronto, Canada, IEEE CS Press, pp. 237-249.
- [14]. Shinde Ajaykumar and Dr. M.S. Prasad *Automic Relational Grammer(ARG) as a means for Representing Data Flow Diagram* International Journal of Information Systems, ISSN 2229-5429, Vol. III Issue II, December 2012.
- [15]. Subramaniam, G.V. and Bwirne, E. J. (1996). Deriving an object model from legacy FORTRAN code. In: Proceeding of International Conference on Software Maintenance, Monterey, CA, IEEE CS Press, pp. 3-12.

- [16]. Thu Nga Tram, Khaled M. Khan, Yi-Chen Lan, *A Framework for Transforming Artifacts From Data Flow Diagram to UML*, Technical Report No. CIT/04/2004. IASTED International Conference on Software Engineering Innsbruck, Austria.
- [17]. 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].
- [18]. Yourdon, E. (1989). *Modern Procedure oriented Analysis*. Yourdon Press. Upper Saddle River, NJ.

