https://doi.org/10.31871/WJIR.9.1.28

World Journal of Innovative Research (WJIR)
ISSN: 2454-8236, Volume-9, Issue-1, July 2020 Pages 106-114

Harmonization of Tools and Techniques for System Development

Nengak Iliya Sitlong, Francisca Ogwuleka Nonyelum

***Abstract*— This article looks at the similarities between the two most common system development tools, Data Flow Diagram DFD and Unified modelling Language UML diagrams. It explored the intricacies of the function of notations and meanings of the symbols used in the two modelling tools by pointing out the similarities and contrast between them, then subsequently established facts of how the two modelling tools can be blended together to leverage a combined benefits of the two modelling tools as an entity in system development process. The research also put side by side the techniques for process modelling which includes interview, questionnaire, survey group and observation. Hence, it provided detailed analysis to justify when one can be preferred over another and made further recommendation on the most efficient model to adopt during process modelling. The article chose as a case study a coach system captioned World Wide Tour Management System (TMS) operating in most part of Europe but predominantly in the UK which is broad enough to provide scenarios for demonstrating the benefits of using Use Case Diagram (UCD), Class Diagram, Communication Diagram and Sequence diagram during both analysis and design phase of process development. In conclusion it was established that DFD are most suited for understanding of functionality of the system operations at requirement gathering stage where as UML models seamlessly reveals all software objects necessary for constructing the proposed system under consideration.**

***Index Terms*— Tools, Technique, Process Modelling, Harmonization.**

I. Introduction

The focus of this work is to explore and present the features of system development tools and techniques as they evolve over time from the early stage of Data Flow Diagrams (DFDs) to the emergence of UnifiedModelling Language (UML) diagrams and come up with ways of harmonizing them with the aim of achieving the combined benefits of the two structured modelling processes in business process modelling. DFD is an organized framework that models the operations of a system from a generalized perspective thus abstracting the underlying details of the processes embedded within the system and then subsequently refines them in a stepwise manner [5]. Conversely, UML is an object oriented hierarchical model that models the detail processes of a sophisticated system into software objects using varieties of analysis and design tools and diagrams relative to the design stage or context of the system under consideration.

Early business process modelling notation adopting the object oriented analysis and design approach such as the Object Modelling Technique (OMT) pioneered by (Loomis, 1987) used Data-Flow-Diagrams (DFD's), Class diagrams, Decision trees and Structured English to model business processes into software objects. DFD's models business processes satisfactorily by abstracting the complexities in the detail structures of the business starting with a context diagram usually referred to as the Level Zero (0) [7]. The context diagram which models the entire system is further subdivided into more refined levels usually call Level one up to level three where it becomes clearer to hand over to the software developer.

DFD's have now been replaced by Unified Modelling Language (UML) diagrams that logically structure the hidden details formally abstracted by the DFD's into set of object oriented diagrams such as the Use Case Diagram (UCD), Class Diagrams, Communication, Sequence, Activity Diagrams amongst others [10].

A. Focus of this Study

The essence of this work is to justify where systems development tools such as DFD symbols relates to UML diagram symbols to effectively and efficiently model business processes to leverage its full benefits with consideration to the right technique to adopt.

B. Important Definitions to this Discuss:

i. **Model:** In business context a model represent the manner in which a business should operate, taking into account the business goals, purpose, strategies and productivity.

ii. **Business Process Model:** Business process model provides the standard view of the business goal.

iii. **Business Rules:** Business rules are the guidelines about how business transactions are run. Business rules are grouped as:
- Derivation rule (for instance compute the result of).
- Constraint rules (example allow or disallow access).
- Existence rules (for example, ensure the existence of customer object.

iv. **UML:** Unified Modelling Language (UML) is a modelling language made up of notations and set of meaning and structural rules for guiding its use [12].

UML Diagrams that are of Relevance to this Discussion Includes:

Class Diagrams: A class diagram consist of description

Nengak Iliya Sitlong, Computer Science Department, Federal College of Education, Pankshin Plateau Sate, Nigeria
Francisca Ogwuleka Nonyelum, Cyber Security Department, Nigerian Defence Academy, Kaduna, Nigeria

for a collection of objects and the data sequence associated with them and their respective behaviors, a class can be information, organization, actor, or products.

Activity Diagram: Activity diagrams describe the flow of processes in a business modelling. The flow may contain receive, guard, process-Clauses.

Use Case Diagram (UCD): Use case Diagrams are used to represent the relationship between different use cases. A use case represents the interaction between an actor (user) and the system's function. Use case may have include (mandatory path) and extend (optional channel).

Communication diagram: This represents the interaction between an actor with the boundary (example GUI) and the imaginary background processes call a control that mediates messages between other entities.

Sequence Diagram: This inherits from the communication diagram and it shows the details of movement of messages between processes and their live lines [6].

v. **Data Flow Diagram (DFD)** language consist of four symbols namely process, data flow, data store and external entity.

Process: A process is an operation executed for a particular business purpose.

Data Flow: Data Flow can be a unit of data (such as "Product price") or an organized piece of information such as customer detail.

Data Store: Data Store is a repository of data organized in a specific format, and data can be retrieved from or added to it.

External Entity: An External Entity is an object that is external to the system but communicates with it, this corresponds to the actor in a use case [8].

II. REVIEW OF RELATED WORKS

So many related works have been carried out in trying to point out the similarities that exist between DFD and UML diagrams. [16] outline the interwoven nature of DFD symbols and UML symbols basing the analysis on UCD and Class diagrams. It was established in the preceding research that the External entity in a DFD plays exactly the same role as the actor in a UCD which is of course external to the system too but interacts with it, these external entities or actors could be individuals, or any other gadgets. Further, the use cases in the UCD corresponds directly to the processes in a DFD and are easily interpreted for requirement gathering by the software developer [6]. The two modelling languages are both hierarchical structured modelling techniques. However, DFD uses uniform class of symbols in all stages of process modelling with supportive structures like decision table, decision tree and structured English. Unlike DFD, UML diagram structures a system in an object oriented perspective by bundling together its attributes with its methods or behaviors using separate representations at several stages of its refinement such as communication diagrams, sequence diagrams activity diagrams etc. This article concur with the established fact by [19] that the data store in a DFD has no difference with a class in UML diagram. This is obvious due to data storage capabilities which offers room for data manipulation through the application of different operations such as search, remove, and add amongst other operations on its data, classes in UML does exactly the same [14].

Atif, (2011), proposed that DFD are more communicative than UCD in presenting user requirements therefore UCD should be substituted by DFD in representing user requirements. The research further stated that the context diagram of DFD translates into UCD by equating data stores to actors, processes to use-cases and data flows into associations. Finally, the data stores will then be expanded into classes in building class diagrams, and all the associated operations to the data stores regarded as functions for the class and associated data units as attributes. This makes a lot of sense to this work because is in tune with the set goal.

A. Additional Correlations between DFD and UML Model Notations

This work explores further points of intersection between UML diagrams and DFD other than the above mentioned ones. Firstly, UCD without the partitions or swim lanes added to it is similar to a logical DFD because they both do not say where or who carries out what operation. Therefore they are both inadequate at their basic states. Partitions in UCD correspond to a physical DFD with the added details.

Further, the Use-Case-Description can be liken to the Structured-English of DFD because they both play the same complimentary role of giving detail explanation to concept that are complex to comprehend on the models produced such as their semantics.

UML sequence diagram can be deduced from a DFD by mapping the external entity to an actor, external entity data flow as a boundary, process to a control, and the sinks or data stores to the entities of the sequence diagram and finally the data flows to and from processes and data stores as messages passed across.

In activity diagram, DFD processes can map into the series of **Activities**, **Send** to an actor, **Receive** to a data store or sink, **Decision Point** to a process with two emanating data flows, the guards can express control flow mutual exclusiveness which is lacking in DFD processes, activity diagram **Control Flow** corresponds to a single data flow to or from a DFD process. This is logical because activity diagrams are mostly used during analysis phase to model the processes in a use case diagram than they are used in design phase which is also true of DFD's. They are more useful in modelling the system functions than focusing on software objects involved which is what UML design is for.

To better understand the above explained systems development tools, it is imperative to introduce certain practical scenario. The case study for this purpose is a Tour Management System (TMS) captioned Wide World Tour Management System.

III. WIDE WORLD TOUR MANAGEMENT SYSTEM (TMS) CASE STUDY

"Wide World Coach Tours operates coach tours of varying durations, mostly in the UK but also to European cities. The company operates a fleet of coaches and maintains a list of associates as drivers and as tour leaders; most but not all associates are self-employed. Administrative staff and booking agents are employed by Wide World, as are the managers of the 20 branches throughout the UK. Wide World publishes tour information on their website and in brochures which are sent to places like libraries as well as to previous customers and in response to requests. Bookings can be made online or in the branches, or by post to the company's headquarters.

There are existing systems to deal with scheduling of drivers and coaches once a tour has been organised and booking numbers are known, and to handle enquiries and bookings for places on coach tours. There is a simple database which stores tour information, for use within the website, but it will need to be expanded. Tour leaders currently plan their tours offline and input outline information, enough to support website enquiries and bookings, but there is increasing need for tour leaders to be able to plan their tours interactively within the system, with better access to up-to-date lists of venues and hotels where discounts have been negotiated. Better information would also help branch and customer service staff when answering booking enquiries. Wide World has concluded that it is time for a new system to work alongside the existing systems."

Table 3.0 Data Dictionary of the case study is as represented below:

Use Case Name:	*Add Venue to Tour*	
Primary Actor:	*Tour Leader*	
Secondary Actors:	*Tours Manager*	
Other Stakeholders:	*TMS*	
Business Goal:	*To add a venue or hotel to an existing tour*	
Precondition	*Tour leader, Tour, and Venue all exist on the system*	
Success Condition	*Venue or hotel added to the tour*	
Main Path		
	1. Tour Leader selects Add Venue to Tour	2. System displays the Tour Selection screen, showing tour name, start date and duration, for each tour currently assigned to the Tour Leader
	3. Tour Leader selects a tour from the list	4. System displays Tour Details screen, showing tour name, tour type, description, start date, duration and venue names for the selected tour
	5. Tour Leader selects Add Venue	6. System displays Search Criteria screen, giving search fields of venue code, venue name, location, type (hotel, restaurant, museum, gallery) and required date range
	7. Tour Leader inputs search criteria and selects Search	8. System displays list of venues that meet the criteria, with availability in the required date range
	9. Tour Leader selects venue code	10. System displays full details of selected venue
	11. Tour Leader selects Confirm	12. System displays 'Add another venue?'
	13. Tour Leader selects No	
Variant Paths		
No suitable venue in system		8a. System displays error message 'No venues meet the criteria, please amend search fields and try again'
	8b. Return to 7.	
Multiple venues	13a. Tour Leader selects 'Yes'	13b. Go to 6.

IV. MODELS OF THE PROPOSED SYSTEM USING UML SYSTEMS DEVELOPMENT TOOLS

A. Use Case Diagram

Understanding what is model in a Use Case diagram requires the knowledge of what a Use case is. A use case is a single role that can be carried out by a user of a system (an actor). Collection of use cases represents the different aspects of the system's operations that may be executed by the actor [13].

A Use Case Diagram models how the use cases in a system relates with one another to coherently accomplish the systems or organization's set goals of satisfying the user's needs with an optimum minimal effort. It depicts use cases that must always be done in some instances stereotyped as (include) and the one that is optionally executed with the stereotype (extend) [20].

Use Case is used to representing the interaction between an actor and the systems role to be performed in a pictorial form that depicts the actor with a stick person usually skeletally drawn to create an impression of human features and the role drawn with an oval shape. The use case is labeled using a verb phrase and the actor with a noun.

Use case is a useful technique in system analysis and design for concise and adequate requirement gathering during the early stages of business modelling process either to upgrade an already automated system or to automate a manual system [17].

Use case generates a valid acceptable architecture of a system that will later be used to identify the needed entities that forms classes and relationship between them as well as their attributes and possible operations.

Most importantly use case forms the basis for system operations documentation that provides a reference point for smooth transaction.

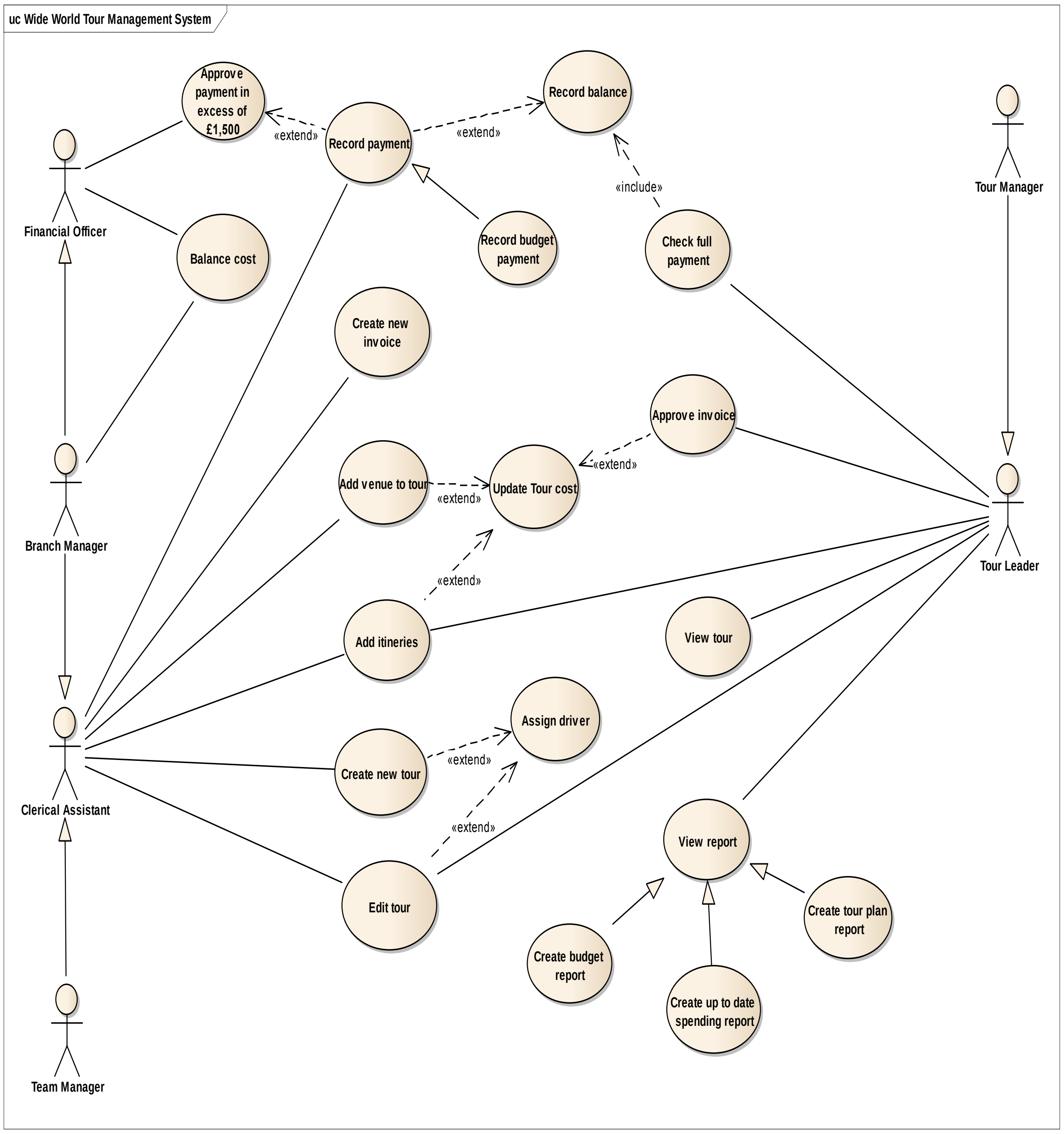

Figure 3.1: The Use Case Diagram for Wide World Add Venue to Tour Case Study

B. The Proposed System's UCD Design Considerations

In drawing the wide world tour management system Use Case diagram the actors identified are: Tour Manager, Tour Leader, Financial Officer, Branch Manager, Team Manager and clerical Officer.

A "Clerical Officer" generalizes the functionalities of both team manager and branch manager and both managers have their specialized attributes, hence the generalization flow control arrow from both branch manager and team manager to the clerical officer actor.

A branch manager also performs the role of a financial officer in a branch therefore financial officer generalizes part of branch managers role.

A Tour Manager performs all the role of a tour leader while tour leaders are absent hence the generalization relationship from tour manager to tour leader.

The lines between use cases and actors shows the different roles associated with the respective actors.

As part of the decisions made while drawing the Use Case diagram a clerical officer records payments, while he does that he records balance if such exist which does not always happens therefore is an extend operation, if in the process the payment made is in increase of £1,500 then it first needs approval from the financial officer before it is recoded therefore record payment extends approve payment use case. A clerical officer create new tour records driver may be added during tour creation or edited later hence this operation extends add driver use case, venues and itineraries could letter be added to the tour by the clerical officer or the tour leader and in the process tour cost may be updated immediately or latter therefor the two operations extends update cost use case. Tour is also edited by tour leader but not created.

New invoices are created by clerical officer but approved by tour leader and in the process sometimes update tour cost, therefore an extend operation on update cost use case.

Before tour commences tour leader check record of full payment this process will always include checking balance record to ensure no balance is left.

Tour leader views record of tour and as well view report. Report generalizes different other reports such as create budget report, up to date spending report and tour plan report all these are shown as individual use cases because they are performed separately.

A financial officer balances cost for the whole wide world tour whereas a branch manager balance cost for a branch as shown in the use case diagram. This forms the summary of decisions considered in drawing the wide world tour use case diagram.

V. USE CASE REALIZATION FOR 'ADD VENUE TO TOUR' USE CASE

A. Analysis Class Diagram

A class diagram is made up of entities call classes and the association between them. An Analysis Class Diagram is a Class Diagram that represent classes with their attributes specified as public, usually prefixed with a plus sign, with their respective data types such as string, integer etc. However, at this stage the focus is on the entities that exist in the system and not how they behave or respond to dynamic nature of the system, therefor operations of the classes are not included [15]. It also points out which entities are related to one another and the possible multiplicities in relationships (for instance 1..*, 0..*) but no details of the order in which they interact.

Class Diagrams can be useful in system analysis and design in representing complex concept that has properties and behaviors (for instance clients, Staff, Information etc.) and how they communicate with each other to form a coherent system. Class Diagram in summary describes the static structure of a system in system analysis and design [4].

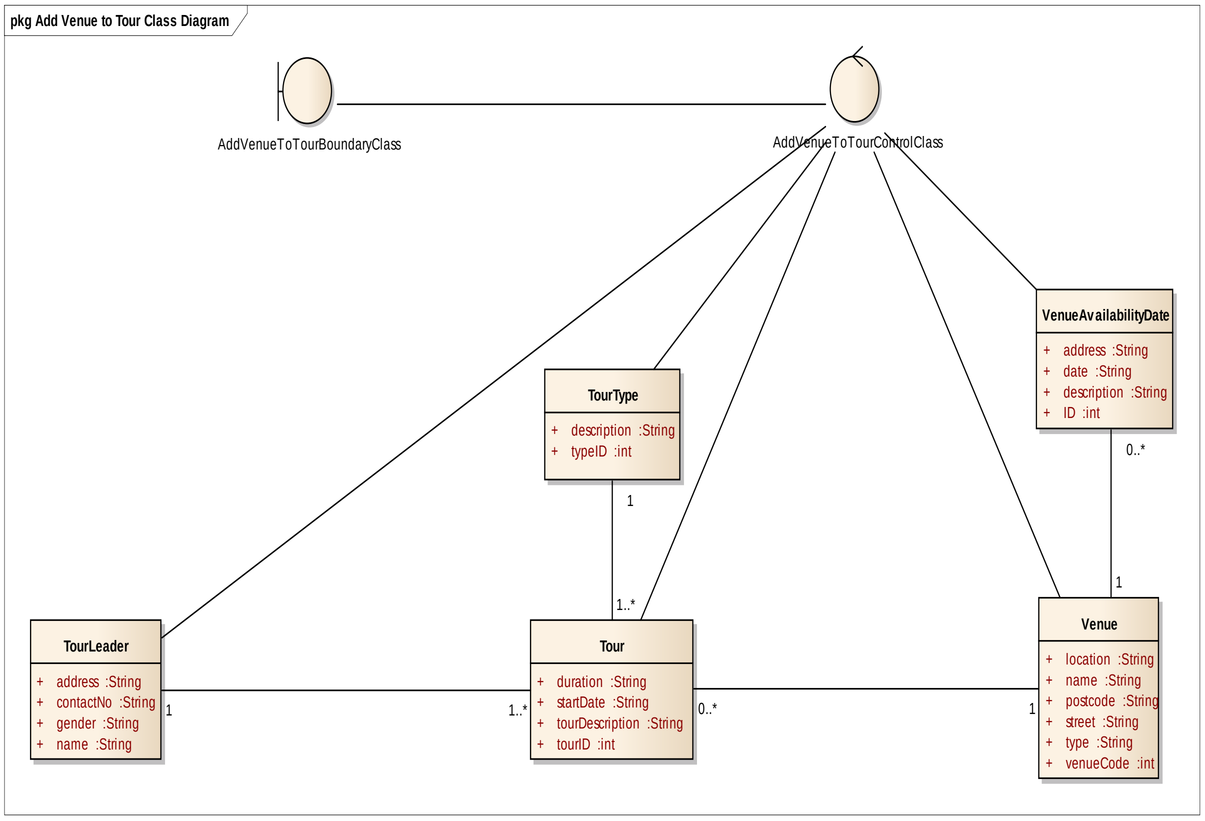

Figure. 3.2: Analysis Class Diagram of the Wide World 'Add Venue To Tour'

A. Summary of the Decisions Made in Drawing the 'Add Venue to Tour' Analysis Class Diagram

First of all everything starts from the boundary class which may represent some sort of interface that the user can interact with, in this case the tour leader.

The class to look at next is the Control Class which represent an imaginary system's operation class that connects all of the other classes in the class diagram together and thus disconnecting them from the Boundary class to produce a loosely couple design that ensures flexibility and scalability in operations performance. It does not have any properties or operations.

Further, Tour Leader, Tour and Venues are stored in the system, therefore the need for a Tour Leader, Tour and Venue Classes. These classes have the listed attributes shown in the 'TourLeader', 'Tour' and 'Venue' classes as represented in the class diagram with their corresponding data types.

Some of the attributes are of type string because the need to hold a descriptive value composed of characters while others are integers because they are expected to hold numeric values.

The attributes are prefixed with a '+' implying they are public at this stage because is an analysis class diagram and no operations are provided too because they are normally not known yet.

The Tour Leader class collaborate with a Tour class to identify the tour assign to a Tour Leader which may be one or more tours therefor and association line is drown between them with the multiplicity of 1 and '1..*'.

The tour has types (for instance Hotel, Restaurant, Museum and Gallery) therefore Tour Types has attributes as shown in the class diagram and as such it can be a class of its own and it interacts with the Tour class to identify a tour type assign to a tour leader.

It is possible for the Tour class to have zero, one or more tour type replicated over several tours therefore the multiplicity 1 and '0..*' on the association line between Tour Type class and Tour class.

Venues are assign to tours based on specific days or dates therefore they may not be available for another tour on a day already assigned to another therefore the list of days a venue is available needs to be kept hence the need for 'VenueAvailability' class as included in the above class diagram with its attributes.

The Venue class collaborate with the 'VenueAvailability' class to identify when a venue is available which may be zero, one or more availabilities hence the multiplicity 1 and '0..*' on the association between Venue class and VenueAvailability class.

VI. COMMUNICATION DIAGRAM

The communication diagram models the changes in state of the logical operations govern by certain business rules outline in a class diagram during conceptual design process in order to actualize an initially set precondition. This is made possible through the representation of the manner in which object instances interact by sending messages to and fro one another to complete the system operations [9].

Communication diagrams are useful in system analysis and design as a tool for depicting system logic and semantics in the way objects dynamically behave during run time or program execution which could not be realized with class diagrams. It is used to show details of interaction between objects.

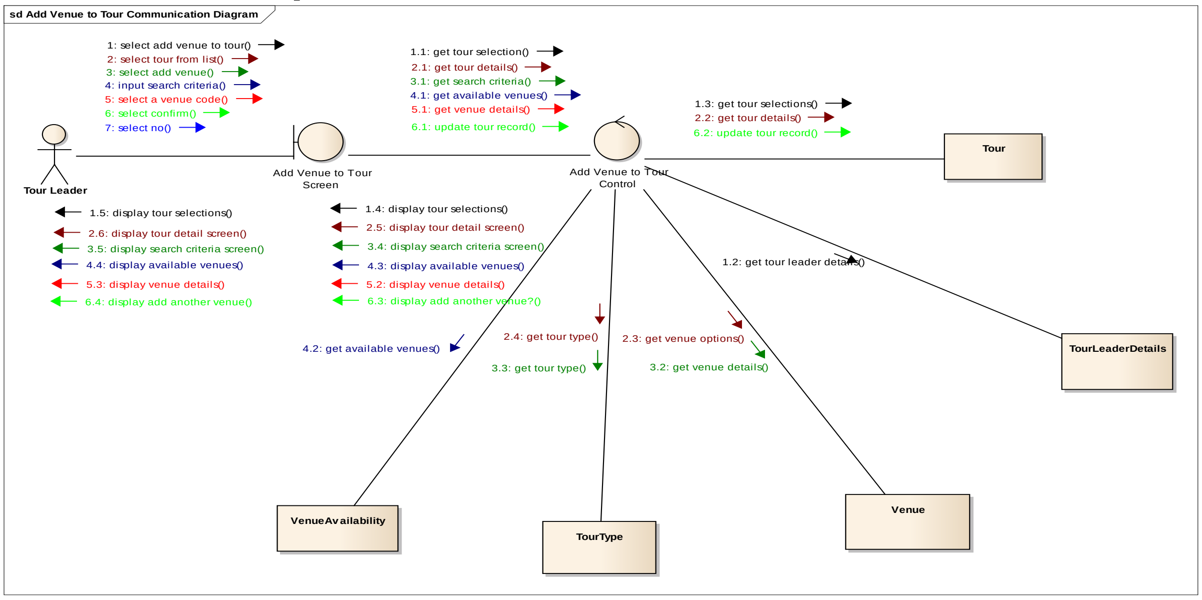
sd Add Venue to Tour Communication Diagram
1: select add venue to tour()
2: select tour from list()
3: select add venue()
4: input search criteria()
5: select a venue code()
6: select confirm()
7: select no()
1.1: get tour selection()
2.1: get tour details()
3.1: get search criteria()
4.1: get available venues()
5.1: get venue details()
6.1: update tour record()
1.3: get tour selections()
2.2: get tour details()
6.2: update tour record()
Tour Leader
Add Venue to Tour Screen
Add Venue to Tour Control
Tour
1.5: display tour selections()
2.6: display tour detail screen()
3.5: display search criteria screen()
4.4: display available venues()
5.3: display venue details()
6.4: display add another venue()
1.4: display tour selections()
2.5: display tour detail screen()
3.4: display search criteria screen()
4.3: display available venues()
5.2: display venue details()
6.3: display add another venue?()
1.2: get tour leader details()
4.2: get available venues()
2.4: get tour type()
3.3: get tour type()
2.3: get venue options()
3.2: get venue details()
TourLeaderDetails
VenueAvailability
TourType
Venue

Figure 3.3: Communication Diagram for the Use Case 'Add Venue to Tour'

A. The Communication Diagram Design Considerations

First the process started by an actor who initiated the process, Tour Leader, and the boundary provides the interface for interaction with the user depicted as the 'Add Venue to Tour Screen' shown on the communication diagram. When the tour leader's input get into the system it needs to be coordinated which explains the existence of the 'Add Venue to Tour Control' in the diagram.

Tour, Venue, VenueAvailability, TourType and TourLeaderDetails are entities stored in the system for access therefore they all need to be included as shown in the communication diagram.

The sequences of operations in the 'Add Venue to Tour' communication diagram are described by numbered messages passed across by objects from one entity to another in a sequential order. Each operation starts with a unique whole number value from the actor, Tour Leader, (for example 1, 2, ….7) and continues by an incremental decimal point at regular interval (for instance 1.1,1.2,..1.5) to the final destination of that message sequentially as shown in the communication diagram. The directional arrow preceding each massage shows the flow control of the message from its origin to its final destination.

The reasons why the different entities listed are needed are already emphasized in the analysis class diagram which is where the communication diagram represented here is derived from. To make the diagram readable and self-explanatory a unique consistent decimal numbering and directional flow control is maintained all through the design process of the communication diagram thus making the diagram easy to interpret, the diagram speak for itself as a means for justification.

VII. SEQUENCE DIAGRAM

Sequence diagram models the order and timing of messages transmitted between a collections of objects. Operations in sequence diagram may be grouped into a synchronous or asynchronous pattern. The cycle of operations in a synchronous group must be completed and a confirmation of successful completion acknowledge at the next phase before the subsequent operation starts. In an asynchronous cycle another operation may start before acknowledgement of completion of the earlier operation is received. This is normally depicted in sequence diagram using 'Life Lines' represented in solid bars with extended dotted lines showing where an operation starts and where it ends [13].

Sequence diagram is useful in system analysis and design for representing operations that require sequential order of execution in a system in order to avoid unusual occurrences and ensure proper management of system resources such as memory [8]. For instance an operation may expect the information from a previously completed operation as its input to start up its processing, and if this process is not properly sequenced it will give room for some irregularities. Therefore sequence diagram clearly address such situation in system analysis and design.

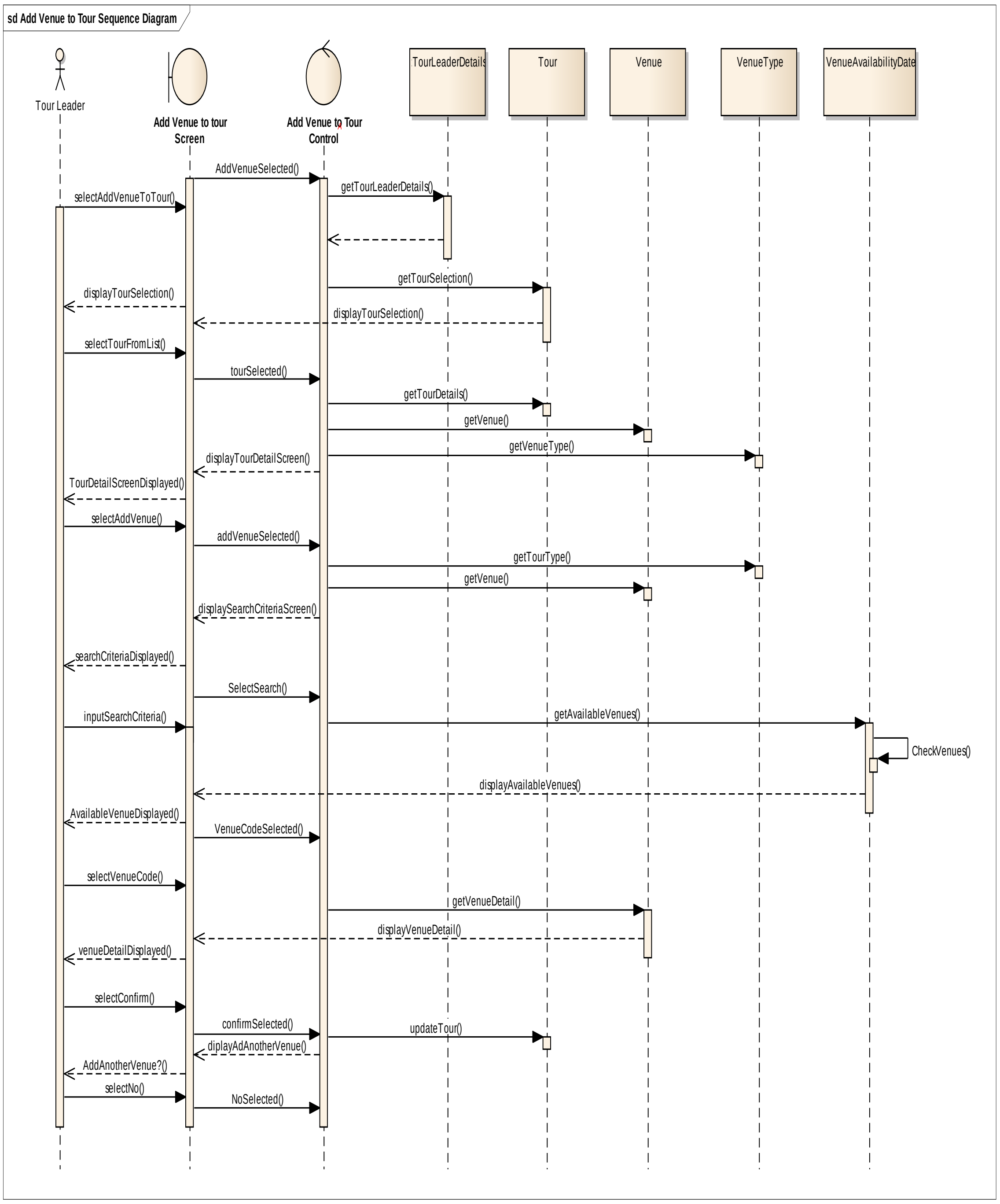

Figure 3.4: Sequence Diagram for the Use Case 'Add Venue to Tour'

A. The Sequence Diagram Design Decisions

In drawing the 'Add Venue to Tour' Sequence Diagram the following decision are considered necessary:

The actor that represents the Tour Leader is needed, a boundary to provide intractable interface between the system and the tour leader is required and a way of demarcating the boundary from the other entities is eminent hence the presence of the 'Add venue to Tour Control'.

The other entities added includes Tour, TourLeaderDetails, Venue, VenueTypes and VenueAvailability.

These entities are necessary because if the Tour Leader key-in his details from the 'Addvenue to Tour Screen', the control will firstly be directed to the TourLeaderDetail entity for identification after which the list of tour associated with the tour Leader will be queried from the 'Tour' entity and return to the control and back to the boundary for presentation to the tour leader.

To get a search criteria the Venue, and VenuType, entities has to be accessed and the generated result returned to the tour leader to choose a search criteria.

Tour Leader may have to choose a venue therefore he has to ensure the venue is not already occupied hence the 'VenueAvailability' entity has to be check for availability of the venue chosen from the 'Venue' entity.

Thereafter the available venue is chosen and the 'Tour' entity is updated with the selected venue.

The process can be repeated if another venue is to be added or it may be terminated otherwise.

VIII. SYSTEM DEVELOPMENT PROCESS TECHNIQUES

A. Interview

Interviewing a focus group of the individuals, users, legal advisers and designers of system to be implemented will produce more viable and reliable responses, but getting access to these right set of people to interview them may proof hard due to their busy schedules, on the other hand the users of this technologies are often constrained in one way or the other therefore may not be comfortable to grand an interview or participate in a focus group which obviously spelt out a great disadvantage for the choice of interview as a technique for the system development processes [1].

Interview technique adopts a more flexible repetitive tools approach of extracting and grouping responses but it uses open ended questions hence making interaction process between the system analyst and the participant spontaneous and free for detail opinions expression rather than a "Yes" or "No" answers as in the case with questionnaires but this makes the analysis phase of the findings tedious, complex, and difficult (Bernard, 1995).

B. Survey Technique

Considering the demanding nature of the schedules of the survey groups to be administered on the system that is to be implemented, who are partly the direct users, legal advisers and designers of the technology, survey questionnaire will be a most suitable technique to use since it can be completed at a convenient time and place and then later picked up by the system analyst, this makes data collection process more efficient [11].

C. Observation Technique

Observation technique is based on an interpretive research concept, practically interpretive research employs two separate roles namely the outside observer role and the participating observer role [3]. Reports generated from an interpretive outlook should not be perceived as devoid of bias, this is due to system analyst's prejudice in the collection and analysis of data [2]. During a focus group observation research, system analysts find it impossible to resist the urge to guide their participants understanding of the ongoing process, a process described as "double Hermeneutic" [18].

Weighing the nature of this research and the above points about the respective techniques, this research recommends carrying out a survey (qualitative method) in its data collection processes during system development processes.

IX. CONCLUSION

DFD and UML have a lot in common as pointed out by the preceding discussion, they both have strength and weaknesses, DFD works better at analysis phase such as requirement gathering, whereas UML handles software objects more naturally because it is designed for that purpose. Therefore combining the two system development structured modelling tools together; with a painstaking questionnaire survey technique of system development process will certainly leverage the full potentials of system process modelling activities into a reliable and efficient software base solution.

REFERENCES

[1] Adelopo, I. (2010). *The impact of corporate governance on auditor independence: A study of audit committees in UK listed companies*. PhD, De Montfort University, 45-60.

[2] Alain, P. (1993). *Survey Research Methodology in Management Information Systems: An Assessment*. Center for Research on Information Technology and Organizations: UC Irvine, 6-30.

[3] Berlo, A. (2005). *Ethics in domotics. Gerontechnology*. Research Methods in Anthropology.2nd ed, 3(3), 165-170. London: Sage.

[4] Cook, S. and Daniels, J. (1994). *Designing Object Systems: Object-Oriented Modelling with Syntropy*. UK:Prentice Hall International Ltd.

[5] [5] Dennis, A., Wixom, B. and Roth, R., (2012). *Systems Analysis & Design*. 5th ed. 111 River Street, Hoboken, NJ: John Wiley & Sons, Inc.

[6] Desfray, P. and Raymond, G. (2014). *Modelling Enterprise Architecture with TOGIF: A Practical Guide Using UML and BPMN*. 225 Wyman Street, Waltham, MA 02451, USA: Elsevier Inc.

[7] Eckert, C. (2005). *Risk across design domains: research methodology, problem analysis, evaluation*. Melbourne, 15-18

[8] Hook, G. (2011). *Business Process Modelling and Simulation*. Winter Simulation Conference, IEEE, 13-16.

[9] Iran, T. (2017). *Fundamentals of Software Engineering*. IPM International Conference, 7th IPM, 24-28.

[10] Jilani, A. (2011). Comparative Study on DFD to UML Diagrams Transformation. *World of Computer Science and Information Technology Journal (WCSIT), 2(5)*, 10-16.

[11] Kothari, C. (2009). *Research methodology: Methods and techniques*. 2nd ed. New Delhi: New age.

[12] Kuskela, M., and Haajanen, J. (2007). *Business Process Modelling and Execution*: Tools and Technologies Report for SAOMes Project. Julkaisija-Utgivare Publishers.

[13] Lampathaki, F., Koussouris S. and Psarras, J. E. (2013). *Business Process Reengineering*. Decision Support System Laboratory, NTUA.

[14] Noran, S. O. (2000). *Business Modelling: UML vs IDEF*. Griffith University.

[15] Sommerville, I. (2011). *Software Engineering*, 9th ed. Pearson.

[16] Tiwari, K. (2012). Merging of Data Flow Diagram with Unified Modelling Language. *International Journal of Science and Research Publications*, 1 (2), 1-6.
[17] Ullmer, B. and Ishii, H. (2000). Emerging frameworks for Tangible User Interfaces. *IBM systems journal*, *39(4)*, 36-41.
[18] Walsham, G. (1995). Interpretive case studies in IS research: Nature and method. *Eur. Journal of. Information Systems, 4(2)*, 74-81.
[19] Wazlawick, R. (2014). *Object Analysis and Design for Information Systems: Modelling with UML, OCL and IFML*. 225 Wyman Street, Waltham, MA, 02451, USA: Elsevier Inc.
[20] Yan, Z. (2007). *Business Process Modelling: Classifications and Perspectives*. Leipzig, Germany.