IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-8, Aug 2018 E-ISSN: 2347-2693

DFD Schema: A Versatile Approach for XML Based Representation of DFD

T.R. Shah

Department of ICT, Veer Narmad South Gujarat University, Surat, India

*Corresponding Author: proftejas@gmail.com

Available online at: www.ijcseonline.org

Accepted: 12/Aug/2018, Published: 31/Aug/2018

***Abstract*—** The Requirement Engineering phase begins with inception and elicitation of functional, non functional requirements and concludes iteratively with modeling and specification. Requirement Engineering demands the coarse level of requirement specification by primary objectives, design constraints and appropriate artifacts of a system. In system development life cycle (SDLC), a system model is analysed and developed using Data Flow Diagram (DFD). DFD is graphical diagram for analyzing, specifying, creating and visualizing the model of a system. The formal requirement analysis and specification method like DFD experiences the problem of ambiguity with different notation and complex graphical presentation. This paper introduces DFD Schema; an XML based versatile specification approach for the structural representation of DFD of a system. Its definition was motivated by lack of available structured and open formats that describe data flow of system with its artifacts. This schema can be used in an interoperable way to transfer data flow requirements.

***Keywords*—**Requirement Engineering, System Development Life Cycle (SDLC), Data Flow Diagram, DFDS, Data Flow Diagram Schema, XML

I. INTRODUCTION

Requirement Engineering (RE) means activities involved in incepting, discovering, analysing, specifying, documenting and maintaining a set of requirements for a system [1]. The RE is evolved as most critical and complex processes in software development life cycle and as a consequence, many errors are introduced in the requirements' phase, caused by incorrectly analysed, poorly written, ambiguous, unclear or missed requirements. Failure to specify the requirements correctly can lead to major delays, budget overruns, layoffs. Good efforts have been made for exploration of alternative elicitation paradigms beyond a pure automation approach as well as semi-automated requirement elicitation [2].

Requirements specification is one of the most essential RE phase during which incepted, elicited and analyzed requirements are precisely documented. The conventional approach of RE may produce the document (like word document) during the initial requirement phase of a project which consists of many graphical diagrams. The consequential manual specification typically becomes inconsistent, incomplete, ambiguous and hard to trace [3].

The DFD modelling aids in describing boundaries of system and provides detailed representation of system components through graphical techniques. To record and document the requirements, the natural language is most influential and communicative medium used by business analyst and stakeholders. The natural language documents can be combined with more formal requirement representation (e.g. DFD model, UML model, mock-ups).

A DFD is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated [4].

A DFD is a graphical tool and model which allows system analyst and users to show the flow of data in an information system. The key principle is ensuring balancing of every level and decomposition the system until system analyst and user can provide detail description of process. It is having components from where the information captured, stored and transferred to. A DFD is incredibly important for the rejuvenation of old legacy systems. However, DFD lacks formalism and as an impact ambiguity and inconsistencies may present. Formal representation of DFD and its formal semantics help in making unambiguous requirement specification and design.

The natural language unstructured documents appear to be well suited for modelling, articulated and specifying the requirements, but the specification might be ambiguous, inconsistent and incomplete [5]. During software development phase, a huge number of unstructured text documents from various stakeholders become available. It is very difficult to perform testing based on various testing techniques mentioned in [6] to validate textual requirements. Therefore, the structured specified requirements and more

specifically entities, processes and flow of the system shall be recorded in standardized format like XML.

A. Limitations of DFD

Some of the problems and limitations with DFD modelling and graphical representation are mentioned below.

- **Time Consuming:**

The DFD simply takes a long time to create, so long that the analyst may not receive support from management to complete it. The DFD go through a lot of modification before handed over to users, so makes the process little slower. The semi structured data format reduces the delay in communication over the network.

- **Ambiguity in Understanding:**

Different DFD models have different symbols like in Gane and Sarson process is represented as rectangle where as in DeMarco and Yourdan symbol it is represented as eclipse. These differences in pictorial representation cause ambiguity in system understanding at times.

- **Ambiguity in interpretation of Notation:**

The different notation like Gane and Sarson, DeMarco and Yourdan are using different symbols. These graphical notations are interpreted by practitioners, designers in ambiguous way. A well defined semantics or DFD formalism could help to reduce such inconsistencies and uncertainty. It make the programmers little confusing concerning the system.

- **Control and Process Timing:**

It does not show information about process timing, sequential or parallel process execution. The UML activity workflow diagram as unified model presents both control and data flows. However DFD is not showing that control path.

One of the better techniques could be one that supports transforming captured requirements into requirement repositories in form of XML database. Since its inception, XML has been used for defining specific vocabularies to represent different human accomplishments. XML is a platform independent standardized representation and structured way to transfer content over network. XML has become a language of data communication over the web. XML is semi-structured, open standard, language independent and extensible [7].

The implementation of DFD diagrams as XML format will be useful for handling such pictorial data representation. It facilitates better system understanding by removing any ambiguities in the notations.

The paper presents a new approach DFDS that uses the XML Schema Definition Language, which is now a recommendation of W3C and thus effectively a standard. It can be used for exchanging requirements amongst stakeholders, business analysts and developers in internal as well as external environment. It also incorporates a broad set of XML elements that define DFD artefacts.

This paper is organized as follows: Section II consists of related research in the field of requirement analysis, DFD representation, and markup languages pertaining to requirement engineering. Section III includes the design structure of DFDS specification. The Section IV includes the scope and potential application areas of DFDS. The result and test is a part of Section V and last section contains the concluding remarks regarding DFDS.

II. RELATED WORK

The relevant work has been done in so many projects to overcome the problems of DFD modelling.

In [8] authors presented a survey of techniques that represent formal semantics to the DFD by analysing different parameters. Techniques were classified as non-standard formal languages, and standard formal languages. Incorporation of formalism in DFD removes ambiguities and helps in checking the syntactic and semantic inconsistencies. The formalization of DFD reduces the chance of ambiguousness in requirement elicitation phase. At a same time formal representation of DFD can also become beneficial in up-gradation of old legacy systems.

Kolhatkar proposed the development of an XML representation of DFDs to overcome a number of identified weaknesses with the graphical DFDs used. The tool mentioned in this paper is user friendly and based on the object oriented features. The diagrams drawn using these tools can be sent over the network. But majority of the files are in DTD format which is not extensible [9] .

In [10], authors presented a tool based on formalized rules for drawing and defining diagram. They have discussed about how to model a business process flow using DFD and presented a set of syntax and semantic rules of DFD.

The Review focuses only on consistency within UML models. The authors address the UML model consistency gaps by introducing a formal consistency management language. To ensure the correctness of DFD, the human intervened validation of errors can be influential but not have the profound impact [11].

Various researches also stated that no formal language has been presently used for semantic specification of DFD [8] [12]. However, Tao and Kung [13] pointed out few CASE tools which provide automated verification facilities to detect inconsistency and incompleteness in a DFD specification.

Dixit et al. in [14] described that the concept of DFD consistency refers to whether or not the depiction of the system shown at one level of a nested set of DFD is compatible with the depictions of the system revealed at other levels.

There are only some markup languages and approaches available in the literature which covers the project description, requirement specification and DFD representation. Most of the methodologies proposed for the

specification of requirements pay less attention to requirement analysis with data flow diagrams.

The RGML (Requirement Generation Markup Language) has created the formal specification method for characterizing the structure, process flow and activities intrinsic to the requirements generation process. The work focuses on characterization of application instantiation, the use of templates and the productions of artifacts to assist the system analyst and requirement engineer. The language is having set of activity elements, however processes and flow of data is not included. [15]

The SRS template is represented in XML by considering the object oriented environment. The template contributed to the simplification and standardization of the procedure for writing requirements and the validation of the domain against use case models. However, it is only focuses section wise SRS representation [16].

The semantic part of use case descriptions are represented in XML. As modelling requirements with use cases is proven useful, the authors Dimitris et al in [17] presented the structure of use cases with appropriate tags. The work revealed in [18] focuses on the formal and informal classification of requirement and specifying those requirements with the XML Schema. However, only few requirements metadata elements are represented.

Requirements Markup Language (RQML) is a XML dialect for specifying software requirements. The goal of RQML is to overcome the drawback of natural language requirement representation, including the relationship among all requirement items. RQML is implemented as the representation of requirements document. The RQML structure is in DTD and not in XML schema. RQML has rich of element types, but only few of them are uniquely defined [19].

Notations used in DFD are usually graphical and researchers, analyst, practitioner interpret these notations separately. Therefore well defined semi-structured XML based formalized model is required.

From the above it has been observed that the work mentioned in RQML, RGML cover all the metadata of the requirements without DFD analysis. The RGML approach covers the process description language also but not covering the DFD properties of the requirements. The RGML is using requirement generation process, describing the process structure, flow of control, and individual activities. The RQML is implanted with objective to run on Palm OS. But it forms the requirement representation with basic class in DTD form. The other XML based and consistency checking methodologies of DFD are not having profound impact on setting DFD as common exchangeable format.

III. DESIGN STRUCTURE OF DFDS

The DFD Schema enables the organizations to represent complex and unstructured DFD components in electronic and interoperable form. The captured field wise requirement in the form of XML can be easily validated against DFD schema. DFDS reduces ambiguity, making clear definitional distinctions where diagrammatic representation leaves room for uncertainty while analysing the requirements with traditional approach. The XML and its supporting formats together provide a correct metadata for parsing provision by any tool. The exchange of DFD (in XML) improves the efficiency in transferring data over network, as it is easily transferrable and improves readability of data due to metadata description. It can be merged with use case schema.

This section describes the design structure of DFD Schema. The design of schema and testing is implemented in Oxygen XML editor trial version tool.

A. Main DFD Schema

DFDS provides a broad set of XML elements that define project detail and context level diagram detail. The DFDS schema is integration of 2 different schemas depicted in "Figure. 1".

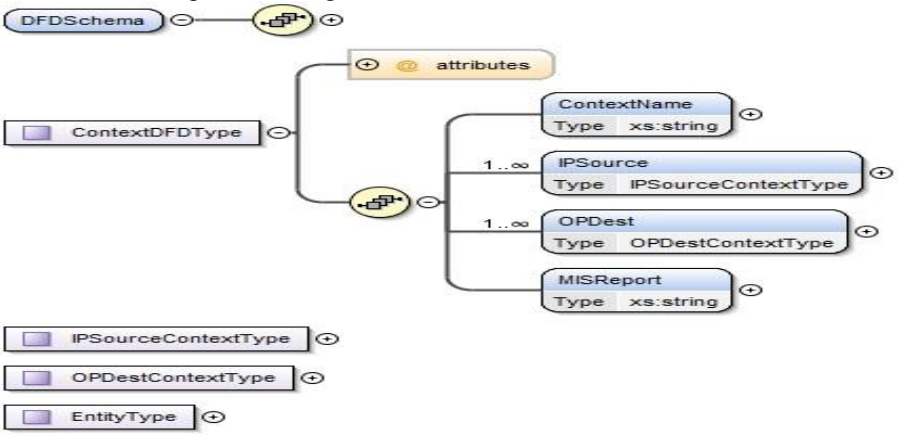

Figure 1 DFD Schema

The ContextDFD Type is having 2 important elements like input source and output source. They are having the same schematic description with parameters, data type which is having set of external entities and which actor has performed that operation. The following code snippet shows the sample elements of the main DFD schema.
Sample Element Tags of DFDS

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
<xs:include schemaLocation="DFD_Leve1_9oct.xsd"/>
<xs:element name="DFDSchema">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="ProjectName" type="xs:string"/>
                        <xs:element name="ContextDFD" type="ContextDFDType"
        maxOccurs="unbounded"/>
                <xs:element name="Level1" type="xs:Level1"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>

<xs:complexType name="ContextDFDType">
        <xs:sequence>
                <xs:element name="ContextName" type="xs:string"/>
                <xs:element name="IPSource" type="IPSourceContextType"
        maxOccurs="unbounded"/>
        <xs:element name="OPDest" type="OPDestContextType" maxOccurs="unbounded"/>
        <xs:element name="MISReport" type="xs:string"/>
        </xs:sequence>
        <xs:attribute type="xs:integer" id="SystemID"/>
</xs:complexType>
<xs:complexType name="IPSourceContextType">
                <xs:sequence>
                        <xs:element name="Entity" type="EntityType"/>
                        <xs:element name="DataFlow" type="DataFlowType"/>
                        </xs:sequence>
</xs:complexType>
<xs:complexType name="OPDestContextType">
                <xs:sequence>
                <xs:element name="Entity" type="EntityType"/>
                <!-- DataStore can be described with type of primary and secondary -->
                <xs:element name="DataFlow" type="DataFlowType"/>
                </xs:sequence>
</xs:complexType>

<xs:complexType name="EntityType">
        <xs:sequence>
                <xs:element name="EntityName" type="xs:string"/>
                <xs:element name="EntityChoice" type="xs:string"/>
        <!-- Entity choice can have multiple levels like Person, Stakeholder, Customer, Object or any other entity involves in
system -->
        </xs:sequence>
</xs:complexType>
</xs:schema>
```


B. OutputSoucre Type and Entity Type

The Figure 2 shows the sub elements of input and output sources. The sources can be from entity or data store. The Data flow can have extension of multiple data flow in the form of text.

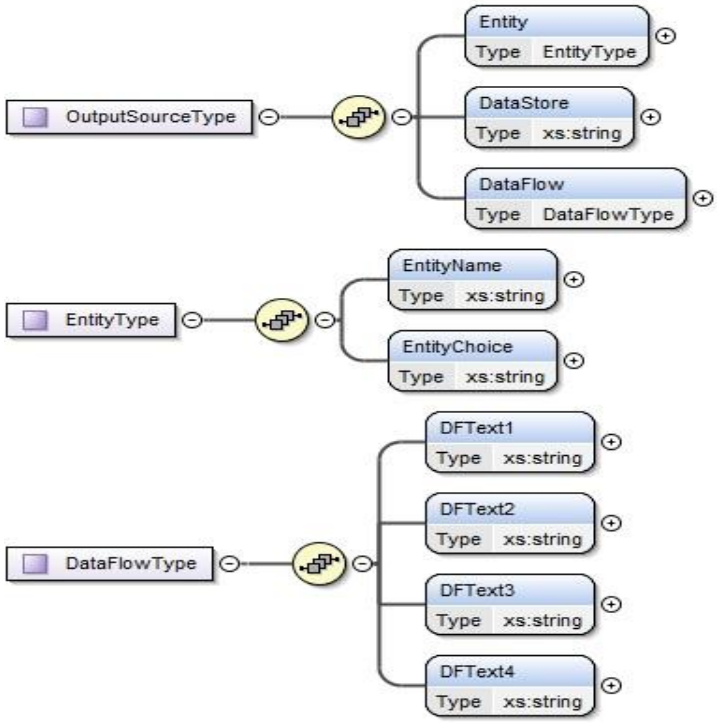

Figure 2 Output and EntityType

C. Element Level1 Schema:

This schema is including the main DFD schema. There are various processes in Level 1 which includes ProcessID, ProcessName elements. The input source and output source is having same schematic elements of entity, data store and data flow like context level DFD. This element is represented in Figure 3. The processes are dependent by the element dependencyPID element.

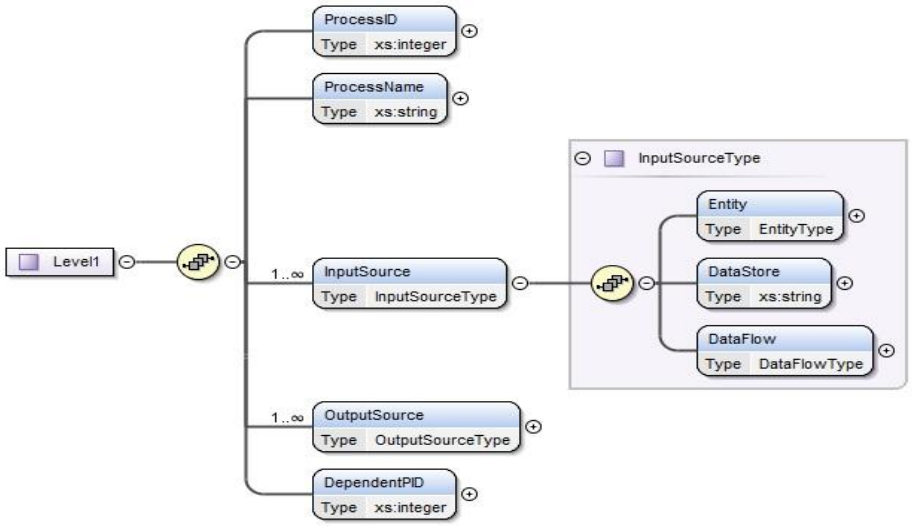

Figure 3 Level1 Schema

IV. SCOPE OF THE DFD SCHEMA

DFDS can be applied to a broad continuum of Requirement Engineering. This can make the requirement elicitation and specification process more rapid. Some of the potential areas and scope are mentioned below.

A. Web-based requirement Support

The XML has developed into the standard platform for structured data exchange on the web. The web uses the XML standard for data exchanges for numerous applications. The DFDS data can be easily interoperable with other standards which supports web based requirement analysis and specification.

B. Use case and actor matching

The processes of level-1 and level-2 DFDs shall be converted into use cases of Use case Diagram. The set of external entities of DFD can be matched to the actor elements of the UML diagram. This mapping can be easily done from raw data of XML files to use cases.

C. Collaboration with Service oriented application

Service-Oriented Architecture (SOA) is a way of designing, developing, deploying systems that are considered by coarse-grained, loosely coupled web services. The web services are identified using a business process to identify functional capabilities needed to accomplish the objectives of system. The role of DFDS is to map the business process and requirements which are converted from Functional requirements specification into web service. The process presented in XML format can be easily converted into web service. So after requirement analysis phase, designer will have the list of web services mapped from DFD processes.

D. Incorporation of DFDS from into Elicitation GUI tool

The requirement elicitation tool can be designed to collect DFD detail through various forms. This will save time and reduces the efforts to support semi-automatic requirement analysis. The GUI based tool can be used to better represent and map filed wise data collection in XML format which is to be validated with DFDS Schema.

E. DFD Process to web service mapping

The use of XQuery from the sample DFDS XML file searches the processes of the module which in turn gives the relevant result of matching relevant web services. So, the DFDS process element can help in the process of identifying web services

V. RESULTS AND TEST CASES FOR CONFORMANCE REQUIREMENTS

The XML repositories are evaluated for their well formed and valid criteria against different schema generated from DFDS which is shown in Table 1, Table2.

Table 1 Sample Test Result for Valid Main DFDS instance document

	Description
a) Test purpose	Verify the validity of the DFDS instance document against the XML Schema definition of each DFDS module that is part of the DFDS profile. This may be any combination of DFDS extension modules in conjunction with the DFDS core module.
b) Test method	Validate the DFDS XML instance document against the XML Schema definitions of all employed DFDS modules. The process has used appropriate software tool for validation process that checks all relevant definitions from the respective XML Schema specification of the employed DFDS modules.
c) Reference	DFD Main Schema
d) Test type	Basic Test

Table2 Sample Test Result for Valid DFDS Level 1 Schema instance document

	Description
a)Test purpose	Verify the validity of the Level 1 instance document against the XML Schema definition of Level1 DFD module. This may be any combination of DFDS extension modules in conjunction with the DFDS core module.
b)Test method	Validate the Level1 schema instance document against the XML Schema definitions of Level1 of DFDS modules. The process has used appropriate software tool for validation process that checks all relevant definitions from the respective XML Schema specification of the employed DFDS modules.
c) Reference	DFD Level1 Schema
d) Test type	Basic Test

VI. CONCLUSION

This paper reveals the initial design of the DFD schema specification is suitable for the use by system analyst, requirement engineer, business analyst. Some of the sub schema of DFDS convention facilitates translation and mapping to requirement representation systems. The semi automatic process of eliciting and specifying requirements can be developed by implementation of this schema. The schema is tested with instance document as conformance of requirements. The advanced architecture with DFDS can

extend the application of RE for managing requirement data exchange and integration, and web service mapping with processing of module element tag.

REFERENCES

[1] I. Sommerville and P. Sawyer, "Requirements engineering: A good practice guide", John Wiley & Sons, 1997.

[2] H. Meth, M. Brhel, and A. Maedche, "The state of the art in automated requirements elicitation," Information and Software Technology, vol. 55, no. 10, pp. 1695–1709, 2013.

[3] D. Firesmith, "Modern Requirements Specification", Journal Of Object Technology, vol. 2, no. 1, pp. 53–64, 2003.

[4] P. D. Bruza and T. P. Van Der Weide, "The Semantics of Data Flow Diagram", 1989.

[5] I. Sommerville and J. Ransom, "An empirical study of industrial requirements engineering process assessment and improvement", ACM Transactions on Software Engineering and Methodology, vol. 14, no. 1, pp. 85–117, Jan. 2005.

[6] C. Patidar, "A Report on Latest Software Testing Techniques and Tools", International Journal of Scientific Research in Computer Science and Engineering, vol. 1, no. 4, 2013.

[7] F. Yergeau and J. Cowan, "Extensible Markup Language (XML) 1.1 (Second Edition)."

[8] A. A. A. Jilani, A. Nadeem, T. Kim, and E. Cho, "Formal Representations of the Data Flow Diagram: A Survey", in 2008 Advanced Software Engineering and Its Applications, pp. 153–158, 2008

[9] S. Salil Kolhatkar, "XML Based Representation of DFD Removal of Diagramming Ambiguity", IJACSA) International Journal of Advanced Computer Science and Applications, vol. 2, no. 8, 2011.

[10] R. Ibrahim and S. Y. Yen, "Formalization Of The Data Flow Diagram Rules For Consistency Check", International Journal of Software Engineering & Applications (IJSEA), vol. 1, no. 4, 2010.

[11] F. J. Lucas, F. Molina, and A. Toval, "A systematic review of UML model consistency management ", 2009.

[12] T. Liu and C. S. Tang, "Semantic specification and verification of data flow diagrams", Journal of Computer Science and Technology, vol. 6, no. 1, pp. 21–31, Jan. 1991.

[13] Y. Tao and C. Kung, "Formal definition and verification of data flow diagrams" Journal of Systems and Software, vol. 16, no. 1, pp. 29–36, Sep. 1991.

[14] J. B. Dixit and R. Kumar, "Structured system analysis and design", Laxmi Publications Pvt. Ltd, 2007.

[15] A. S. Sidky, J. D. Arthur, O. Balci, and S. Mccrickard, "RGML: A Specification Language that Supports the Characterization of Requirements Generation Processes", 2003.

[16] K. Meridji, "Documentation and validation of the requirements specifications : an XML approach", 2003.

[17] T. K. Dranidis D., "Writing Use Cases in XML", 9th Panhellenic Conference in Informatics Thessaloniki, 2003.

[18] S. Chavda and S. Nayak, "Modern Technique To Build Software Requirements Specification", IJSRD-International Journal for Scientific Research & Development|, vol. 2, pp. 2321-0613, 2014.

[19] S. Adibowo, "Rambutan Requirements Management Tool for Busy System Analysts Technical Report", 2003.

Authors Profile

Mr.T R Shah pursed Bachelor of Science in IT from Gujarat University, Ahmedabad in 2002 Master of Science in IT from Gujarat University in year 2004. He is currently pursuing Ph.D. and currently working as Assistant Professor in Department of ICT, Veer Narmad South Gujarat University, Surat since 2006. He has published more than 5 research papers in reputed national and international journals. His main research work focuses on Requirement Engineering, Service Oriented Architecture. He has 12 years of teaching experience.

