An Ameliorated Methodology for the Design of Project Data Flow Diagram

Shivanand M. Handigund[1], Kavitha H. B[2]

Dept of M. Tech. CS&E Programme, Bangalore Institute of Technology, Bangalore-570004, India

[1]smhandigund@gmail.com; [2]kavitha672@gmail.com

***Abstract*- The success rate of the software development projects is pathetically very low. The main causes of low success rate are attributed to i) Lack of knowledge of the project management activities; ii) Non availability of automated tools for most of the project activities; iii) Ignorance of understanding the differences between software project management and other project management activities. Thus, there is a great need to streamline the software project management activities [1]. In this paper an attempt has been made to abstract the needy components through automated methodology for the design of Project Data Flow Diagram (PJDFD). This avoids the use of arbitrary human skills in the component abstraction process. The Data Flow Diagram (DFD) components are abstracted through the design of Project work break structure (PJWBS). PJWBS is designed through the decomposition of the project work. The entire work of the project is represented as a single node and then a tree structure is generated using the designed node as the root node and then decomposing the entire work of the project randomly in three consecutive levels based on Knowledge areas (KA), Project life cycle (PLC), Software development life cycle (SDLC). The leaf nodes obtained after three tier decomposition contain the activities, both quantitatively and qualitatively. The output of these activities (the attributes defined in these activities) forms the deliverables / milestones. Some of these deliverables / milestones may have been referenced (input) to other activities. Each of these deliverables thus form the data store between the predecessor and ancestor activities. The rest of the deliverables which are not referenced in any of the activities may form intermediate or final products, results or services. Another level of datastores is identified during the pipelining process of activities. The identified activities are to be trimmed as per the semiotics of DFD [2]. During this process if the data flows in a pipelining way, then the activities are merged to form a process. Similarly if the flow of data from an activity is stacked (set of data is processed) and then flowed, we consider the stacking point as the data store and the target activities themselves as the processes. This paper discusses our proposed 22 steps methodologies. In our proposed PJDFD, we considered software requirements specification (SRS) as input and the products, results or services as output and referenced / defined attribute set as data flows.**

Keywords- Software Requirements Specification (SRS); Enterprise Environmental Factors (EEF); Organizational Process Assets (OPA); Knowledge Areas (KA); Project Life Cycle (PLC); Software Development Life Cycle (SDLC)

I. INTRODUCTION

Vision: To develop an automated methodology for the design of project data flow diagram.

Mission: To abstract the needy components of the PJDFD viz., Actor, Data store, Data flow and Process.

Objectives:

1. To design PJWBS from software requirements specification [1].
2. To abstract the components of PJDFD from model elements of PJWBS.

A. Motivation

Though DFD is one of the three major system design diagrams, the lack of existence of any automated methodology makes the design, developer dependent as the design process is to be carried manually. Since the manual procedure uses the arbitrary human skills, the correctness and completeness of the design is always at stake. This has ripple effect on the ensuing software. Since the DFD is the unique tool that transforms, the number of activities into a single activity, it virtually clusters the $\{7\pm2\}$ synecdoche activities in to a single activity.

PJWBS is an intermediate product in our attempt to develop PJDFD. The PJWBS is useful in authenticating correctness & completeness. The manual process the correctness & completeness is authenticated through the manual design of PJDFD and then consulting the client organization for its scope. The automated methodology eliminates the need to authenticate the scope through client's organization. PJWBS is a tree structure initially without interdependencies between the same hierarchical level nodes that represent activities (siblings). Here each hierarchical level node represents activities of Software Development Activities or Project Life Cycle Phases or Knowledge Areas or any combination of the above. The reference and defined noun phrases identifies respectively the input and output. This can be used to interconnect or inter-relate the activities.

According to [10], 80-90 percent of all software and 30-45 percent of all systems projects fail. Moreover, over half of all systems projects overrun their budgets and schedules by up to 200 percent or more. Of the projects that fail, approximately half of that are restarted fail again. The project is success only if it delivers the product or service on time, on budget, to the customers prescribed requirements. This needs delinking of projects from the arbitrary human skills. This is possible only

when the activities involved are automated.

B. Literature Survey

In [3] Handigund et al have developed a methodology to abstract the components like data store, actor, data flows. But their abstraction is from the Cobol programming system in which the conventional data processing is used. They failed to identify dataflows in a clear cut way. However, the process provides some guidelines for developing the methodology.

In [1] the author has discussed the PJWBS without exemplifying the automated technique. However, the author has identified the metric of activity through the intersection of KA, SDLC, PLC [4]. This paves the way for decomposition of the work in a systematic automated order. We have developed automated methodology for the design of activity quantitatively and qualitatively. The components abstracted from PJWBS may not follow the pragmatics defined in [2]. To incorporate this pragmatics, we have refined the abstracted model elements based on sound logic of data flows pipelining and stacking activities.

In [5, 6, 7] the authors have identified synonyms for the attributes used through three methodologies viz., the logical DFD, state chart diagram, and refinement of DFD. We have used the same methodologies to resolve synonyms and homonyms.

C. Taxonomy

- *Control Flow Graph (CFG) [2]:* The control flow between the statements of SRS in the realization process [1].
- *Correctness and Completeness:* The injection & bijection of functionalities are used to authenticate the result. Below Figure 1 shows the correctness & completeness verification through surjection.
- *Data Flow Diagram (DFD):* It identifies the behavior of the information system identifying the data flows from input actor to output actors through various processes, data stores/ data bases.

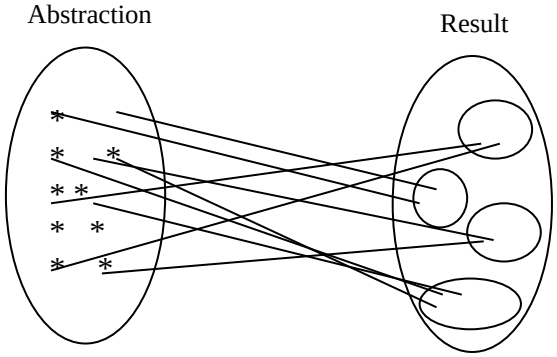


Fig. 1 Injection & Bijection

- *Defined Items:* The nouns or ad-nouns whose values are modified during the realization of the statement.
- *Defined Items/ Attributes:* The nouns or ad-nouns whose values are modified during the presumed implementation of the statement are the defined items.

Domain Knowledge: Knowledge that is specific for the particular information system which is application dependant and it mainly focuses on the application to be realized.

General Knowledge: The knowledge possessed by any software engineer independent of domain knowledge. Automated methodologies needs only general knowledge and do not require any specific domain knowledge.

- *Referenced Items:* The nouns or ad-nouns which are used in a statement that may define other nouns or noun phrases without undergoing any modification of their own values during the presumed implementation of the statement are termed as referenced items.
- *SRS:* It is a document prepared by the client organization involving the detailed requirements of their information system which includes the overview of the system, the functional & nonfunctional requirements of the system, the actor interfaces, the constraints & the prototypes etc [3]. This document is a part of the project charter which leads to the formal commencement of the software development.
- *Model Elements:* The view element that has got a specific syntax in the semiotic of the language. Here, model elements include the object class, object state, interrelationship, visibility, package, activity, action state, actor, event, message, use case, guard condition and signal.
- *Enterprise Environmental Factors:* The environmental constraints that affect the project developments.
- *Organizational Process Assets:* The resources those are available within the organization to carry out the project.

II. PROPOSED METHODOLOGY

The input, output and tools & techniques of this proposed methodology are as follows:

Objective: To develop an automated methodology for the abstraction model, for the abstraction model elements from SRS.

Input: Software requirements specification (SRS) of an information system, Software project management activities as defined by [1], Knowledge area, Software development life cycle stages, Project life cycle phases.

Output: Project data flow diagram

Tools: Software Project Management concepts developed by [1], Synonyms and homonyms tools [3, 5, 6], our proposed Data Process Activity Table (DPAT) technique and our proposed PJWBS technique

Techniques:

Nowadays, PJDFD is designed by manually abstracting the needy model elements viz., process, actor, data store and dataflow from the SRS. There are some constraints on these model elements in the absence of clear cut definitions for these model elements, manual abstraction depends on the arbitrary human skills used. Moreover, there does not exist any standard metrics for these model elements. As a result, the design of PJDFD is designer dependent. This violates the uniqueness criterion, thus making the design an art. Now it is the time to develop an automated methodology (science) and then optimize with respect to model elements (engineering). In [1] author has identified the activities and reticulated them in three dimensional space formed by KA, PLC and SDLC.

The project managers are not aware of the standard PMBOK [4] taxonomy. Moreover, their does not exist any unique taxonomy for the project activities. Thus different project managers named the activities with different transitive verbs. Different verbs for the same activity form a synonymy so, in using our automated methodology the users (project managers) need to refer the synonymy for each activity. Here, as in Figure 2 the entire activities of the Software Project Management (SPM) are classified into number of realms, based on their proximity to the specific management perspective.

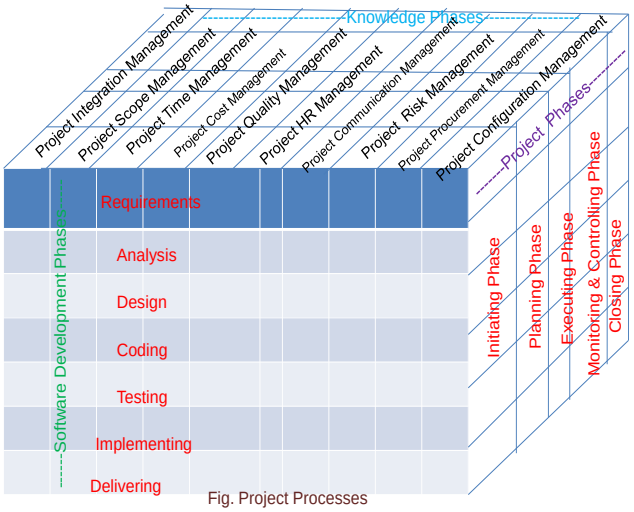


Fig. 2 Project Processes

We have classified the activities in 10 different realms called Knowledge Areas viz.:

1) Project Integration Management,
2) Project Scope Management,
3) Project Time Management,
4) Project Cost Management,
5) Project Quality Management,
6) Project Human Resource Management,
7) Project Communication Management,
8) Project Risk Management,
9) Project Procurement Management,
10) Project Configuration Management.

The PMBOK [4] has categorized the project related activities into five different phases PLC on the progress ladder of the project. The Project Management Institute (PMI) has considered the following PLC's in their PMBOK:

1) Initiating Phase,
2) Planning Phase,
3) Executing Phase,
4) Monitoring & Controlling Phase,
5) Closing Phase.

Since the software project is invisible, complex, conform to logical laws, in need highly of skilled human resource for comparatively smaller amount of time, *with* poor budgetary & schedule estimate and flexible [8, 9], it needs different tools & techniques than other projects. To overcome these lacunae, the activities need to incorporate certain additional operations like strengthening the documentation & design diagrams, deleting & correcting errors in activates and authenticating the correctness & completeness. These have been well defined in software engineering as SDLC stages. Therefore, the project needs to consider *defacto* standard life cycle stages:

1) Requirements,
2) Analysis,
3) Design,
4) Coding,
5) Testing,
6) Implementation,
7) Delivering.

Incorporating these activities, we propose to consider the activities as a common factor for all the three realms.

Activity = (((Project work ÷{KA}) ÷{PLC}) ÷{SDLC})

Here we have used the computer science meta language. The project work contains the functional and non functional requirements referenced in SRS.

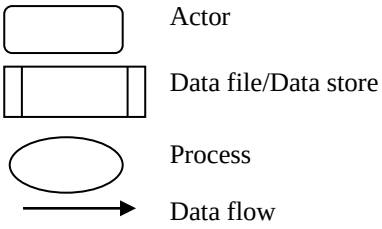
Therefore, in our proposed methodology, we attempt to identify the project activities from the project work formed by functional & non functional requirements of SRS. The software project comprises of number of activities, if the activity is defined as commonality between KA, PLC, and SDLC. It is wise to decompose the activity at a tier based on once each of the following KA, PLC, and SDLC. Therefore, we first consider the design of project work break down structure where decomposition is made on the three types of realms.

The National Aeronautics and Space Administration (NASA) has defined a systems architecture [9] as: "….How functions are grouped together and interact with each other. The architecture applies to the mission and to inter- and intra-system, segment, element and subsystem". The PJWBS can also be used for designing software architecture for the entire project. The PJWBS is designed by considering the entire project task at the root node. This root node is decomposed into sub systems in subsequent levels based on KA, PLC and SDLC. In each hierarchy of the subtree of PJWBS the decomposition through each of KA, PLC and SDLC should be only once. The decomposition may continue maximum upto level 4. The leaf nodes may not be in the same level. Here PJWBS levels refer to successive tiers of activities with lowest tier representing activity qualitatively and quantitatively.

The interdependencies between different activities can be identified through referenced and defined attributes. These attributes may represent attributes of a record or the entire file (document) or any of the design diagrams or any technique.

These activities are clustered based on the pipelining of defined attributes of an activity with the referenced attributes of the subsequent activity. In such case the activities are clubbed together to design a process for the DFD. If the defined attributes of an activity are stacked and then referenced to the next adjacent activity indicating the presence of data store or database. If the defined attributes of an activity are not further referenced in any other activities such activities form the output and are clustered together based on the functional dependencies to form the output actor in the DFD. Of course the input actor is SRS. Splitting is possible if input for different output of the activity otherwise it may not be possible.

In DFD, we normally consider the following notations as de facto standards.



Actor: File or person which is outside the scope of the system and which interacts with the system either providing or consuming the information from the system. The information system does not have any control over the system i.e. the actor's interface attributes are not modifies within the behavioral transformation of the information system.

Dataflow: It is denoted in the form of the "arrow mark" that represents the information flow between the components.

Data store: It may contain data file/database these are the storage area that are modified within the behavioral transformation of the system. Here, the stack of data that has to be stored between processes and/or actors

Process: A process may comprise single or multiple activities with pipelining of dataflows between them.

We use siblings technique to cluster the sibling activities using the following norms based on sound logic:

1) Consecutive activities with common data flow, as outflow of the first activity and inflow of the subsequent activity can be merged into a single activity which later on can participate as a process in the data flow diagram.

2) Siblings can be combined when one file can give data to two different activities.

3) Input and output for two or more activities emanate from same actor and/or the output flow is also joining same actor or data store then there is possibility of combining all three. At least one can be combined.

The proposed automated methodology for Project DFD:

1) Treat the entire project work as a single node and design PJWBS.

2) Decompose the node according to any of the three project space components such as KA, PLC and SDLC [1, 9]. Each level node needs to be divided successively with respect to one of the above three KA, PLC, SDLC at a time to ensure the unique use of each one of them. The order of decomposition is immaterial and also all nodes of the same level need not be decomposed with respect to same project space components.

3) Each leaf node is the intersection of KA, PLC and SDLC.

4) Identify the reference and defined attribute of each activity.

5) Let 'R_i' denote the ref attributes and 'D_i' denote the defined attribute of the process 'P_i'. Here the referenced attribute set R_i can be obtained by the input or the values of the input of the standard process of SPM that matches with P_i. Similarly the defined attribute set D_i can be obtained by the output or the values of the output of that process of software project management that matches with P_i. In the worst case, if any of the software project management does not match, then the input/output of the activity is obtained by the organization process assets.

6) Now from 'Abstract view elements activity' we obtain two data dictionaries containing:

a) Actor A_i, its corresponding reference interface attribute A^r_i and defined interface attribute A^d_i;

b) Class name C_i, its reference attribute C^r_i and defined attribute C^d_i.

7) For each P_i, compute $R_i \cap A^d_j$ for j= 1 to n. Compute $R_i \cap C^d_j$ for j = 1 to n and $R_i \cap C^d_j$ for j≠1 varies from 1 to n.

For each P_i, compute $D_i \cap A^r_j$ for j = 1 to n. Compute $D_i \cap C^r_j$ for j = 1 to n and $D_i \cap R_j$ where j≠i varies from 1 to n.

8) Store the above computation in separate data dictories in the descending order of the number of elements.

9) Read the data dictionary entries form k = 1 to n. R_i - $R_i \cap A^d_k$, $R_i = R_i - R_i \cap A^d_k$. Actor=$A_k$, Inflow = A^d_k. Repeat the process till $R_i \neq \varphi$.

10)Read the data dictionary entries form k=1 to n

$$R_i – R_i \cap C^d_k, R_i = R_i - R_i \cap C^d_k$$

Class =C_k, Inflow= C^d_k. Repeat the process till $Ri \neq \varphi$.

11) Read the data dictionary entries from k=1 to n

$$R_i - R_{i \cap} D_k , R_i = R_i - R_{i \cap} D_k, \text{Process}= P_k , \text{Inflow} = D_k .$$ Repeat the process till $R_{i =} \varphi$.

12) Read the data dictionary entries from k=1 to n

$$D_i - D_{i \cap} A_k^r , D_i = D_i - D_{i \cap} A_k^r$$

$$\text{Actor} = A_k , \text{Outflow}= A_k^r$$

Repeat the process till $R_{i =} \varphi$

13) Read the data dictionary entries from k=1 to n

$D_i - D_i \cap C_k$, $D_i = D_i - D_i \cap C_k^r$, Class= C_k, Inflow= C_k^r . Repeat the process till $R_i = \varphi$

14) Read the data dictionary entries from k=1 to n

$D_i - D_i \cap R_k$, $D_i = D_i - D_i \cap R_k$, Process= P_k , Inflow= P_k . Repeat the process till $D_i = \varphi$.

15) Design first cut DFD with respect to P_i.

16) Repeat steps 1 to 15 for i=1 to n.

17) (Represent each first cut DFD as an entry in the following table). Combine all the first cut DFD's using the following table.

A	A_d	C	C_d	P_r	R	P	P_d	D	C	C_r	A	$\mid A_r$

A- Input from actor

A_d - Input data flow from actor

C- Input from data store to process

C_d - Input data flow from data store to process

P_r - Input from Process to current process

R- Input dataflow from process to current process

P- current process name

P_d - Connection from current process to other

D- Output dataflow from current process to other process

C- Output from current process to data store

C_r - Output data flow from current process to data store

A- Output from current process to other

A_r - Output data flow from current process to actor

18) Consider the entry P = p_i . If $R \neq \varphi$ for $r = r_1$, then search for entry P = p_{r1}, in P = p_{r1} if R and D are null. Then update $P_i = p_i \cup p_{r1}$; $p_r = p_r - p_{r1}$.

19) Consider the entry P= p_i . If $D \neq \varphi$ for $d = d_1$ then search for entry $P = p_{d1}$, in $P = p_{d1}$ if R and D are null. Then update $P_i = p_i \cup p_{d1}$; $p_d = p_d - p_{d1}$.

20) Consider the entry P= p_i . If $C = c_i c_j$ for $i \neq j$ then $C_1^d \leftarrow C_2^d$ then $c_i = c_i \cup c_j$.

21) Consider the entry P = p_i. If $C = c_k c_l$ for $k \neq l$ then $C_1^r \leftarrow C_2^r$ then $c_k = c_k \cup c_l$.

22) Draw DFD for the above updated table.

A	B	C	D	E	F	G	H	I	J	K	L	M

A- Input from actor

B- Input data flow from actor

C- Input from data store to process

D- Input data flow from data store to process

E- Input from Process to current process

F- Input dataflow from process to current process

G- Current process name

H- Connection from current process to other process

I- Output dataflow from current process to other process

J- Output from current process to data store

K- Output data flow from current process to data store

L- Output from current process to other

M- Output data flow from current process to actor

III. CONCLUSION

In this paper we made an attempt to abstract the needy components for the design of the project data flow diagram from the concepts of KA, PLC & SDLC. An intermediate PJWBS is developed which is another powerful tool to develop other design

diagrams like software architecture and PJDFD.

An attempt is also made to incorporate synonymies of activities. However due to the limitation we yet to develop methodology for the abstraction of type of data store.

Project process has not been standardized, but prototype has been developed by [1]. The correctness & completeness of abstraction depends on the correctness & completeness of Software Project Management.

Author of the SPM named the activities on par with Project Management Institute [9], which is still not standardized therefore giving the scope for individual developers, we have to identify some of the Synonyms and Homonyms.

We have achieved working dataflow model but still challenges like accessing data to and from data store has not be identified.

Project Data Flow Diagram is limited to first cut components, we have not automated for real time, indexed and hash file organization.

ACKNOWLEDGEMENT

Words are insufficient to express our deep sense of gratitude to Dr. D. B. Phatak, Professor Dept. of Computer Science & Engineering IIT Bombay, for his inspiration and encouragement in carrying out this research project.

REFERENCES

[1] S. M. Handigund: A tutorial presented at "International conference on Advance conference on networks and security-2011" ADCONS 2011 held at National Institute of Techonolgy, Surathkal, Mangalore, India during16th – 18th Dec, 2011.

[2] Roger S. **Pressman**, "**Software Engineering** A Practitioner's. Approach", 6th Edition, McGraw-Hill, 2005.

[3] S. M. Handigund, Reverse engineering of legacy cobol systems, Ph.D. Thesis, Indian Institute of Technology Bombay, 2001.

[4] A paper entitled "Project management body of knowledge" PMBOK 4th edition 2008, published by Project Management Institute.

[5] A paper entitled "Automated Methodologies for the Design of Flow Diagrams for Development and Maintenance Activities" authored by Dr. Shivanand M. Handigund and Swetha has been accepted for the international Conference on E-Business, Technology and Strategy to be held at Ottowa, Canada during 29, 30th of September' 2010.

[6] A paper entitled "Automated Methodologies for the Design of Flow Diagram for Development and Maintenance Activities" (paper no. ICETS-SI-2010-46) has been published in E-business Technology and Strategy Communications in Computer and Information Science, 2010, Volume 113, 93-104, DOI: 10.1007/978-3-642-16397-5_8, Dr. SM Handigund and Swetha Bhat.

[7] A paper entitled "Automated Methodologies for the Design of Flow Diagrams for Development and Maintenance Activities" (paper no. ICETS-SI-2010-46) in proceedings of Joint Conference of International Conference on e-business Strategy and Technology, Ottawa, Ontario, Canada along with Canada-China e-business forum to be held at Ottawa, Ontario, Canada during Sept. 29th & 30th, 2010, Dr. SM Handigund and Swetha Bhat.

[8] A paper entitled "Software Project Management" by Bob Hughes and Mike Cotterell, 5th edition, 2012.

[9] "A paper entitled "Managing information technology project" by James Taylor in Eastern Economy Edition, PHI publication 2011.

Dr. Shivanand M. Handigund, is working as professor of computer science & engineering in Bangalore Institute of Technology, Bangalore, India. He has obtained M. Tech CSE and Ph. D degrees respectively from IIT Delhi and IIT Bombay. He has over 38 years of teaching and research experience. He has published 47 papers in international journals and conferences and delivered several tutorials and key note addresses at various international conferences.

Few of his prominent publications are as follows:

1. In proceedings of "Reengineering Methodology for Legacy COBOL Systems" has been published in proceedings of International conference on Information Technology held during August 01-04, 2000 at Bangkok, Thailand.

2. A paper entitled "Automated Methodologies for the Design of Flow Diagrams for Development and Maintenance Activities" authored by Dr. Shivanand M. Handigund and Swetha has been accepted for the international Conference on E-Business, Technology and Strategy to be held at Ottowa, Canada during 29, 30th of September' 2010.

3. On an invitation by the organizing committee, a tutorial entitled "Software Project Management" has been presented in the Advanced International Conference on Networking and Security, National Institute of Technology, Surathkal, Mangalore, India, December,16-18,2011.

4. S.M.Handigund and shweta, An Ameliorated Methodology for the submitted of object class structure for an Information System, Communicated to the 2010 International conference on Computer Applications and Industrial electronics to be held at Kuala Lumpur, Malaysia December 2010.

Kum. Kavitha H. B. is pursuing her M. Tech degree in CSE, at Bangalore Institute of Technology, Bangalore, India. She has published 3 papers in national and international conferences under the guidance of Dr. Shivanand M. Handigund, Prof. & Head, Dept. M. Tech, CSE. Bangalore Institute of Technology, Bangalore, India.

Few of her publications are as follows:

1. A paper entitled "An Ameliorated Methodology for the design on Project Data Flow Diagram" authored by Dr. Shivanand M. Handigund and Kavitha H. B**.** has been published at the National Conference on "Advance in Information Technology" held at Indo Asian Academy group of institutions, Bangalore on 2nd April 2012.

2. Participated in "IBM Technology Web Contest 2011" conducted all over India, by submitting the paper on "Hidden Treasures of UML diagrams" and got selected within 100 participants.