Framework to represent the software design elements in markup text – Design Markup Language (DGML)

P. K. Suri[1] **and Gurdev Singh**[2] ,

1. Professor, Department of Computer Science & Applications, Kurukshetra University, Kurukshetra, (Haryana) India.
2. Department of Computer Science & Applications, Kurukshetra University, Kurukshetra, (Haryana) India.

Abstract: The software design document bridges the requirement phase activity with the implementation phase activity. Design elements are having pictorial information about the possible solution for requirements. Software modules are written from these documents. If we can represent these design elements in well-defined text format, many experiments are possible on software design for optimizing it. This paper is an initiative towards a new framework, which stores the design elements in the form of a text document with the help of proposed DGML (DesiGn Markup Language). A new syntax is created for DGML based documents. This representation of pictorials design elements in the form of text helps in design optimization, reusing the existing design and early prediction of error prone modules. A fresh new design can be obtained from existing design after parsing it for well-defined project requirements.

Keywords: DGML, DGML syntax, DGML DTD, Text based design representation, Design markups, Design element.

I INTRODUCTION

Software design document is very important and very first activity of the implementation phase. Successful design document leads to successful implementation of the later stages of software development process.

Many techniques are available for making the coding process more optimized, error free and reusable for the future use. All this is possible because of the text-based nature of the program modules. Tools are available for reading source code, identifying the scope improvements. However, this type of possibility is not there on the software design.

When analysis is over, next phase of design is to represents the requirements in a way that the coder can think in term of solutions after reading or looking on the figures, which are there in the design document. Most of the cases the pictorial representations are considered from the design documents. The architecture design and component design are used to get the familiarity about the process flow.

Manuscript received January 5, 2010
Manuscript revised January 20, 2010

Some famous design tools gives the flexibility to create the design the document but further inference to extract new design for reusability or design based intelligent optimization to assist to make it robust is almost not available. There are tools available to store the object oriented design in the form of diagrams and then these diagrams are converted in to the text files with the help of UML[1]. Application software are available which convert/ generate the text file from class diagram or vice-versa. They also provide a framework for minimal generation of the code from UML class document.

The procedures of applications are most of the time represented with the help of conventional old methods like flow charts in the design specification document. When a design document is used for writing the programs, the flowcharts are one of the maim components from that. But there is no tool available which store the diagrams for procedural component (like flow chart) to a text file, which could be later referred for further processing.

One of the other requirements during the project development and later in the maintenance phase is the synchronization between the design document and coding. This activity requires time and efforts. Automation of this activity is also in big demand. [5].

All these observations indicate the need of making the design phase more systematic and to invent the ways of design representation which could lead to better understanding of design document, more productiveness, provide the design level reusability and earlier detection of the error prone module in the design phase. If we make the design phase more robust, error free, up to date and reusable, it will minimize the overall cost of the software development process.

II RELATED WORK

Software development life cycle has been reinvented many times with the introduction of new models and also the phases of the software development life cycle. Researchers are identifying the difficulties in the software development life cycle and proposing the solutions for the same. Large

no of experiment have been accomplished to improve the coding style, code betterment, code optimization etc. The work in the field of software design improvements is also considered as research topics.
Researchers have experimented on the design phase by taking different aspects. A good design helps in achieving the stability of the software. [2] Early identification of error prone module from the design stage is good approach and a seed for stability. Time and effort investment in early stage helps in minimizing efforts and cost in the later stages.

Mr. Joost Noppen and their team worked on topic Dealing with imprecise quality factors in software design. They identify that during the design phase, it's a good idea to measure the effectiveness of the design for the specified requirement. Its good to have the alternative design for given requirement. So that we can choose for better one. [3].

In the publication, Measuring software design quality By David N. Card, Robert L. Glass, the importance of modular design is focused [4] and proposed the details of applying the metrics on the modular design. The paper also discusses the identification of the critical design module and concentrate on them in better way. It also derives the design metrics from existing software metrics.

Microsoft is also identifying the way to provide the synchronization between the code sheet and the external documents. [5] This approach of automatic synchronization between associated document of the project helps in getting the updated information about the software at any point of time.

III Overview of DGML

DGML (tags for design elements) is proposed for the storing of the diagrams for the procedural design components such as flow charts, as a text file. The DGML based design document will be a repository of DGML tags. The design document repository contains the flow chart symbols as a markup and stored as a design-repository.

In this way diagrams for the procedural components could be represented as a text file. DGML is a collection of well-defined unique tags which are named after the components of the diagrams of procedural design, like <Decision> and </Decision>, for diamond box of the flow chart etc. A well-defined DTD is also there for the grammatical verification of any document to be used for further processing.

The text based stored design documents are capable for reproduction of the pictorial design elements. This is possible because of the one-to-one correspondence between the pictorial design element and its associated DGML tag.

IV DTD of DGML

All the DGML tags for design representation are having the DTD entry for syntax check.

```
<!ELEMENT MODULE ( ASSUMPTIONS, CONSTRAINTS,
   OPTIMIZATION, INPUT, OUTPUT,    LOGIC ) >

<!ATTLIST MODULE ID NMTOKEN #REQUIRED >
<!ATTLIST MODULE TYPE NMTOKEN #REQUIRED >
<!ELEMENT ASSUMPTIONS ( #PCDATA ) >
<!ELEMENT CONSTRAINTS ( #PCDATA ) >
<!ELEMENT OPTIMIZATION ( #PCDATA ) >

<!ELEMENT LOGIC ( ALTERNATEPROCESS | DECISION |
DELAY | DOCUMENT | LIBPROCESS | MAGNETICDISC |
MANUALINPUT | MULTIDOCUMENT | PREDEFPROCESS |
PROCESS | SEQUENTIALACCESS | STOREDDATA )* >

<!ELEMENT MAGNETICDISC ( #PCDATA ) >
<!ELEMENT MANUALINPUT ( #PCDATA ) >
<!ELEMENT DECISION ( #PCDATA ) >
<!ELEMENT ALTERNATEPROCESS ( #PCDATA ) >
<!ELEMENT DELAY ( #PCDATA ) >
<!ELEMENT DOCUMENT ( #PCDATA ) >
<!ELEMENT INPUT ( INPUTLIST ) >
<!ELEMENT INPUTLIST ( #PCDATA ) >
<!ELEMENT LIBPROCESS ( #PCDATA ) >
<!ELEMENT MULTIDOCUMENT ( #PCDATA ) >
<!ELEMENT OUTPUT ( OUTPUTLIST ) >
<!ELEMENT OUTPUTLIST ( #PCDATA ) >
<!ELEMENT PREDEFPROCESS ( #PCDATA ) >
<!ELEMENT PROCESS ( #PCDATA ) >
<!ELEMENT SEQUENTIALACCESS ( #PCDATA ) >
<!ELEMENT STOREDDATA ( #PCDATA ) >
```

Figure 1: DTD for DGML used to represent the Flowchart based design

There exists different type of DTDs depending upon the different type of design element.

One DTD is there for textual design representation of procedural design, one is for textual design representation for the object-oriented design, and other one is for the structural English design representation. Following is the DTD for the DGML based representation of the procedural design (pictorial to text based).

V Syntax of DGML

To discuss the syntax of DGML, let us take the example of bubble sort program. Figure2 represent the design in the form of flowchart for bubble sort problem.

This design is having the well-defined symbols for representation of flow chart methodology. As these symbols are well defined and standard, we can represent each symbol of the flow chart in the form of DGML tags.

As defined in figure1 for DTD, each possible element of the pictorial design for procedures is having one DGML

tag. When flowchart is created, its related entry is stored in the text based DGML file.

Figure3 represents the DGML based notations for the design, which is represented in the form of flowchart. Every design element must be having one root element, which is known as module.

Each module must be having two attributes. One is type of module and other is id for module. Type of module is indicating its category. One design module representation in DGML could be for functional design (FN), object oriented design (OOD) or structure English. This paper discusses everything for functional design. When inference or search lookup process is carried out on the stored design in the DGML form for category, it will be compared with module type attribute. The module id is used to give the name to the design module for identification. This should be relevant name like we have given sort. Good name for module id helps in identifying the existing module for reusability. Module name and module type attribute can never be null. The DTD is having strict definition for this.

Each module is consisting of one or all of six elements namely *assumptions*, *constraints*, *optimization*, *input*, *output* and *logic*.

First three elements are used for the scope and better understanding of the design process.
Assumptions specify the helpful comment about the specified module. Like in our case, assumption is that this module when implemented can sort the array data not data stored on external storage like file. This helps the programmer about before using the design for writing some solution.
Constraint defines the restrictions about the design. Like the current design is having the restriction not to use the control variable as parameter to procedure. This makes a module more cohesive as the one module represent one functionality and its internal architecture is not driven from the control parameter.

Optimization element defines the scope to code the defined module in more effective way. Sorting algorithm is smart if it checks for data on which it is to be applied. If data is already sorted it should stop intelligently and hence save processor time.
Input is the definition of the input parameter to the procedure. *Output* is the result of the procedure. These definitions are to specify the details to be taken care while implementing the procedure.

Logic is the representation for the core process. Like for the bubble sort algorithm, we specify the process inside

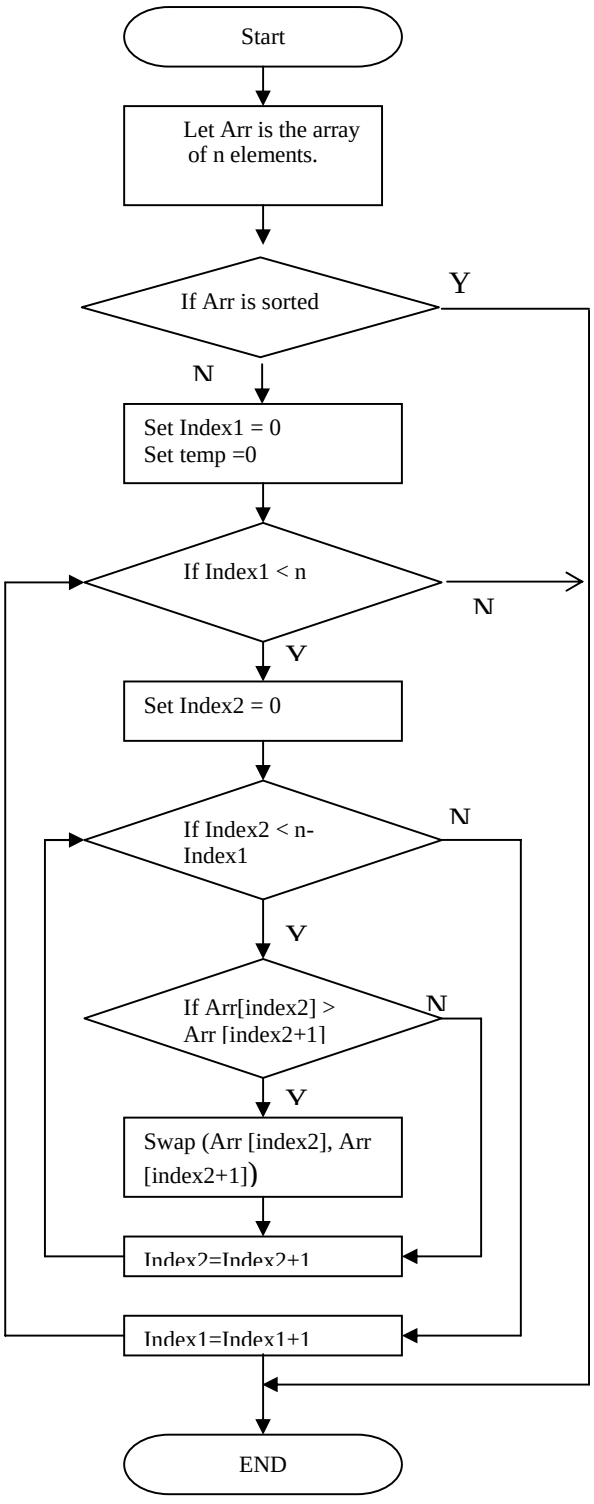

Figure 2: Flowchart based design of bubble sort algorithm

the logic tag to sort the given list of array. According to DTD in figure1, there could be twelve possible elements, which could be used to representation the logic of some procedural problem in the DGML format.

```
- <MODULE TYPE="FN" ID="SORT">
    <ASSUMPTIONS>This design module can be used to implement
      array sorting program. No file based content
      sorting.</ASSUMPTIONS>
    <CONSTRAINTS>This module is assumed to have no control
      parameter as arguments.</CONSTRAINTS>
    <OPTIMIZATION>Take care if input is already
      sorted.</OPTIMIZATION>
  - <INPUT>
      <INPUTLIST>Unsorted array of N elements,
        Elementcount</INPUTLIST>
    </INPUT>
  - <OUTPUT>
      <OUTPUTLIST>Sorted array</OUTPUTLIST>
    </OUTPUT>
  - <LOGIC>
      <PROCESS>LET ARR IS THE ARRAY OF N ELEMENTS.</PROCESS>
      <PREDEFPROCESS>If ARR is unsorted</PREDEFPROCESS>
      <PROCESS>SET INDEX1 = 0, SET TEMP =0</PROCESS>
      <DECISION>INDEX LT N</DECISION>
      <PROCESS>INDEX2 EQ 0</PROCESS>
      <DECISION>IF INDEX2 LT N-INDEX1</DECISION>
      <DECISION>IF ARR [INDEX2] GT ARR [INDEX2+1]</DECISION>
      <PROCESS>SWAP (ARR [INDEX2], ARR [INDEX2+1])</PROCESS>
      <PROCESS>INDEX2=INDEX2+1</PROCESS>
      <PROCESS>INDEX1=INDEX1+1</PROCESS>
    </LOGIC>
  </MODULE>
```

Figure 3: DGML based representation for the flowchart of bubble sort.

VI Working with DGML

DGML is text-based representation of the design document. The design document which could be represented using the notation procedure design which is represented using flowchart, object oriented design which is represented by class diagram and structural English. Other design elements like UI design can also be implemented using proposed approach.

The design is created with DGML based tool. This tool is having the responsibility to store the design in the form of DGML based tags and to show graphical representation of the design. The complete design can be stored in the form of text by assigning it a type and identification.

Experiments have been carried out to store the procedural design like bubble sort algorithm and results for representation are very encouraging. Also, the reusability of user interface design for a software system could be achieved by storing the graphical design documents in the form of text. The graphics-symbols of the design document can be represented in the form of text. Tools are available which allow the user to draw the user interface with diagrams. These tools however lack one aspect to store the design elements in the form of structured text as backend.

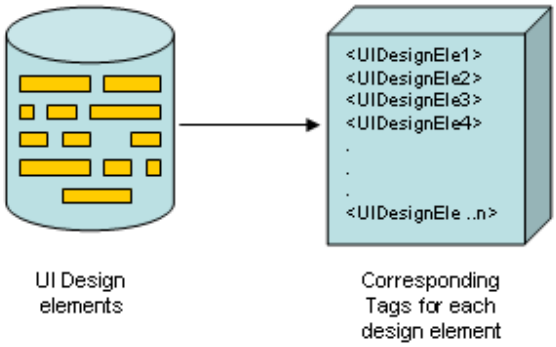

Figure 4: UI element and its associated design tag of DGML.

As we store the functional design of software product using tag based notations of DGML, we can store the UI design of software also. We device the mechanism using which we can store the UI diagrams for user interface in the text format. So that when user creates one UI design and save it, it generates one text file also. The approach that we choose is to represent the text per design element is as a DGML tag. So every design element will have a unique tag. Initially the process start with making a new UI design document, which is DGML based.

The system to create the UI design is having the collection of UI elements. All finite design elements have assigned names. User has to select and place these design elements while creating design document. When one UI design module gets completed, there will be one complete DGML document associated with it. These UI design repository will be a collection of associated DGML document per design as shown in following figure2. There will be a repository of designs after following this approach and every design is well defined in terms of DGML tags. The text format of storing the design is having the predefined tags. Text format will be stored in DGML notations and is having the tags for all knows design elements. When user picks one design element, its respective tag will be placed in its corresponding text file.

When UI design complete for some scenario by following this approach, system will be having a complete representation in textual format in the form of tags. The DGML text file having the details of design figures is the basis of out research. The approach has many benefits. When pictorial representations are saved in text form, lots of possibilities are there which increase the overall development process. Reusability of design, search with in the available UI design, automatic generation of design skeleton on the basis of requirement document and available design text keywords, auto generation of test

case scenarios. Along with the design element tags, the DGML based file is having the information with the help of special tags like keywords, which is important and is a key for further reusability criteria implementation. This additional information is the list of keywords that designer wants to assign to his design.

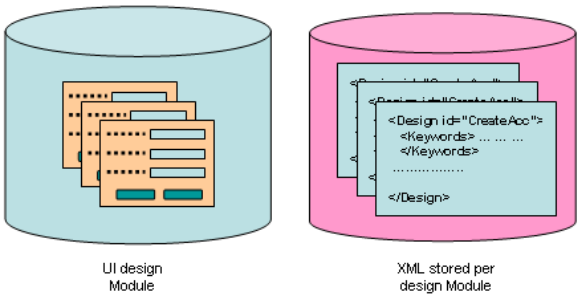

Figure 5: Each Design module is stored in the form of DGML-XML document.

Each UI design is having one name, keywords and some attributes. Name identifies the design module, keywords are the handles for reusability, and attributes are factors, which design document gain after reusability. Experts review the UI design and assign scores. A good score by expert make design module a good candidate for reusability.

VII Benefits from DGML based storage notation

The reusable design is a new concept introduced from this new type of design representation. The DGML is capable of producing not only the structure design but UI design also. When user produces a design sheet and its design repository, the reusable components can be selected from the design repository. Like in case of UI design, the basic UI for authenticating the user name and password is almost same for most of the application. The repository will be having a component for authenticating the user name and password. That user can select from a component list of repository and add to its own new design.

Other benefits of using this notation for design representation are that we can calculate the design metrics on the design document repository very easily and efficiently. Various design checks and initial design performance related checks could be enforced at earlier stages. Software developers should be able to find out software quality attribute during the design process.

Project	Module Type	Module No	Sub Design Elements	DGML Design size	Evaluated local declaration count	Expected local declaration count
Web plug-in	Procedural	Module1	Browser interaction	283	18	6
		Module2	Native file interaction	235	3	5
		Module3	Web links resolutions	567	6	8
		Module4	User interaction	900	15	12
		Module5	Reporting	160	5	4

Table 1: DGML based design evaluation results for LDC

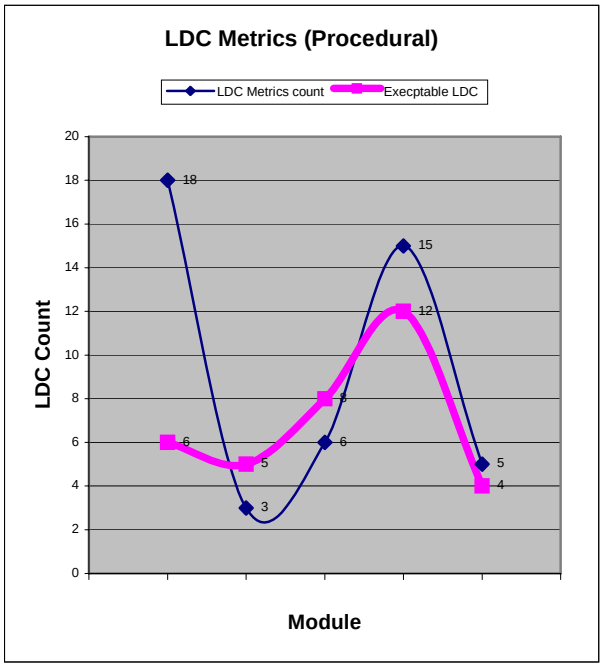

Figure 6: DGML based design evaluation results for LDC

VIII Discussion

We have conducted a small experiment to store the design of web plug-in software using the notations of DGML. This is DGML text base file. We can evaluate the size of this file and evaluate the LOC for this on the same pattern as we evaluate the LOC (line of code) for code [6][7]. From this LOC size we can calculate an approximate value for local declaration count for design document. When DNSIM parser executed on this DGML based stored

design, it gives calculated local declaration count for each module. Table1 shows the results for LDC. Here we have the expected LDC and evaluated LDC. We have compared this for five different modules as shown in table1. The graph in figure6 depicts that for module4 of user interaction is having high value of the LDC then expected. So this module requires further improvements to minimize the LDC. Such observation and corrections in the design on the bases of observations improve the design quality and minimize the efforts of later stages of software development process.

IX Conclusion

This research work is carried out in order to device a syntax and notation for a new language which could be capable of storing the design elements in the form of DGML tag based text. This paper shows the feasibility of storing the design in the form of text. We have presented the notation of storing the design elements in the form of text. This text base representation opens many ways to optimize the design. Well-defined syntax for storing the procedural design in the form of text helps in error free representation of the design modules. Stored design can be reused for a new requirement. The approach reusing the design components of existing software design for the building blocks of new software saves the time while creating new system and makes it more robust. This is because the reused component is driven from already tested, implemented and maintainable system. The DGML based representation can be easily evaluated against metrics. New metrics can be invented, which helps in identifying the error prone modules in early stages of software development process.

References

[1] Rudolf K. Keller, “Design and Implementation of a UML-Based Design Repository”, Proceedings of the 13th International Conference on Advanced Information Systems Engineering, January 01, 2001, pp 448-464

[2] Diane Kelly, “A Study of Design Characteristics in Evolving Software Using Stability as a Criterion”, Software Engineering, IEEE Transactions on Volume 32, Issue 5, May 2006 pp. 315 – 329

[3] Joost Noppen, “Dealing with imprecise quality factors in software design”, ACM – 2005, International Conference on Software Engineering archive, Proceedings of the third workshop on Software quality. pp. 1 - 6

[4] David N. Card, Robert L. Glass, “Measuring software design quality”, Prentice-Hall, Inc, ISBN:0-13-568593-1, pp 126

[5] Microsoft Patents, “Synchronizing external documentation with coding” LINK: [http://www.faqs.org/patents/app/20080250394]

[6] A. Al-Janabi and E. Aspinwall, “An evaluation of software design using the DEMETER tool”, IEEE software engineering journal, Nov 1993, pp 319-324.

[7] Barbara A. Kitchenham, Lesley M. Pickard and Susan J Linkman, “An evaluation of some design metrics”, IEEE software engineering journal, Jan 1990, pp 50-58

[8] Tian Zhang1 , Frédéric Jouault2 , Jean Bézivin2 and Xuandong Li1, “An MDE-based method for bridging different design notations”, Innovations in Systems and Software Engineering, Springer London, Volume 4, Number 3 / October, 2008 pp 203-213

[9] Jan Van den Bergh, Karin Coninx, "Model-based design of context-sensitive interactive applications: a discussion of notations", ACM International Conference Proceeding Series; Vol. 86, Proceedings of the 3rd annual conference on Task models and diagrams, 2005, pp. 43 - 50

[10] Antonio Navarro , José Luis Sierra , Baltasar Fernández-Manjó and Alfredo Fernández-Valmayor, May 08, 2007, "XML-based Integration of Hypermedia Design and Component-Based Techniques in the Production of Educational Applications", Computers and Education in the 21st Century, ISBN:978-0-7923-6577-8, pp 229-239

Author Information

Dr. P.K. Suri received his Ph.D. degree from Faculty of Engineering, Kurukshetra University, Kurukshetra, India and master’s degree from Indian Institute of Technology, Roorkee (formerly known as Roorkee University), India. He is working as Professor in the Department of Computer Science and Applications, Kurukshetra University, Kurukshetra– 136119 (Haryana), India since Oct. 1993. He has earlier worked as Reader, Computer Sc. & Applications, at Bhopal University, Bhopal from 1985-90. He has supervised twelve Ph.D.’s in Computer Science and eleven students are working under his supervision. He has more than 125 publications in International/National Journals and Conferences. He is recipient of 'THE GEORGE OOMAN MEMORIAL PRIZE' for the year 1991-92 and a RESEARCH AWARD –“The Certificate of Merit – 2000”for the paper entitled ESMD – An Expert System for Medical Diagnosis from INSTITUTION OF ENGINEERS,

INDIA. His teaching and research activities include Simulation and Modeling, Software Risk Management, Software Reliability, Software testing & Software Engineering processes, Temporal Databases, Ad hoc Networks, Grid Computing, and Biomechanics.

Gurdev Singh received his Masters degree in Computer Science from Department of Computer Science & Applications, Kurukshetra University, Kurukshetra, Haryana, India. Since 2002 he is working as Software Development Professional and had experience of working with MediaTek, and Siemens Information System, India. Currently he is working as senior software engineer for Samsung Electronics in Noida, India. He has completed projects in the field of software development for mobile devices. He loves to transfer user requirements in to piece of software. is interest includes work in the domain of software engineering, effort minimization in software development, qualitative software design and synchronization, software design representation methodologies and reusable software design techniques. He has written many papers in the related domain. He is fond of studying about the digital electronics and experimenting the same.