Software design simulation for quick and qualitative application development

P.K. Suri[1] 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: Quick and qualitative software development is the prime objective of every software development organization. This paper addresses a new approach by storing the design document in the form of (DGML – Design Markup Language) text and applying the search mechanism for specified requirements on the stored design documents for finding a new solution by reusing the existing. The search mechanism in the simulation algorithm uses the design rank approach, which specifies the quality of a design. The proposed simulation framework for design reusability helps in speed up the development process and makes it of a good quality. It provides a new design to start with, from exiting design module.

Keywords-Qualitative software design, Software design reusability, Software design inference, Software design representation, DNSIM, Design Notation Storage.

I. INTRODUCTION

In software development lifecycle, when analysis phase is over, the next phase is to represent the requirements in the form of design solution figures, like flowcharts, DFDs etc. These pictorial representations of the design documents give the familiarity about the process flow .

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. Application software are available which convert/ generate the text file from class diagram or vice-versa. They also provide a framework for synchronization by regeneration of the UML class document from the classes written in the code[1].

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 main components from that. From the study, it has been found that there is no tool available which store the diagrams for procedural component (like flow chart) to a text file[2]. From the experiments and study we found that there is a possibility to store the procedural design document in the form of text.

PROPOSED DNSIM (DESIGN NOTATION STORAGE & INFERENCE MECHANISM)

The diagrams for the procedural components could be represented as a text file. The proposed DNSIM will do the required functionality. DNSIM is a collection of the unique tags which are named after the components of the diagrams of procedural design. We name this notation as DGML, i.e. Design Markup Language. This language is having the tags which gives the possibility to represent the pictorial design as a text document. The key set is some thing like <Diamond> for decision, <Rectangle> for process, <Looplink> for link connector of a loop etc. These key elements give the freedom to store the complete procedural design document in the form of text. The mechanism allows the user to create the design document. When a user selects a design element to create the design and save it, its get saves in the form of a text internally with the help of proposed tags.

II. NEED FOR DNSIM

DNSIM is required for the storing the design of procedural components as a text file. DNSIM is also capable of retrieving the stored design and create the flow diagrams back. This way the design document is having a repository, which contains the flow chart symbols as a markup tags. Following are the key points which identifies the need:

- DNSIM creates the design repository in the form of design text. This is created by placing a tag for each of the component, which is present there in the design figure.
- DNSIM generates the pictorial design document back from the stored text documents having design tags.
- DNSIM is having the inference engine. When a design document is stored, its associated keywords are also stored along with it. The DNSIM inference engine looks into the requirement specification sheet. It collects the design related keyword and search for them in the design repository. As each design document is having the keywords. The total design repository is having a big pool of keywords per design. The inference mechanism of DNSIM gives the best match for the specified requirements after context base searching.
- DNSIM provides the ranking mechanism. This mechanism gives the good quality design solution while reusing existing. Every time a design is reused, its rank gets incrementing by one. When search for a design is carried out, it gives a big result. More is the

Manuscript received August 5, 2009
Manuscript revised August 20, 2009

rank of a design, more it is suitable for the requirement.

III. THE DESIGN REUSABILITY FRAMEWORK USING DNSIM

- The reusability framework for DNSIM is having an application program, which will be responsible for following
- The generation of the flowcharts from reading the design repository.
- The storing of the updated flowchart in to a text file base design repository
- Identifying keywords after reading requirement specification document.
- The lookup mechanism on the stored design and evaluation of the design components as a candidate for the reusable entity.
- Providing a good quality design solution from existing after applying the ranking algorithm.

The application will be a DNSIM tag parser, which read the design-repository keywords and keywords from requirement specification documents. It the checks for similarity. The match found after comparison will be reported as a success. Interface will show the ready to use designed modules cloud to be pick-up as a reusable building blocks of new application.

IV. BENEFITS FROM DNSIM SYNCHRONIZATION FRAMEWORK

The benefits from the synchronization framework tool are many folds. The existing design document represented with DNSIM can be reused for other domain problems having some similarity in functional components. The design metric can also be predicted easily from DNSIM based design document.

The reusable design is a new concept introduced from this new type of design representation. The DNSIM 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.

This approach of using the design components of existing software design for the building blocks of new software design saves the time for new system. This also makes it more robust because the reused component is driven from an already tested, implemented and maintainable system. This reduces a lot of issues, which could arise otherwise if design is new. New design need a lot of rework to finally get is a working design.

Following example shows the representation of a procedural design component of bubble sort procedure with the help of proposed DNSIM framework.

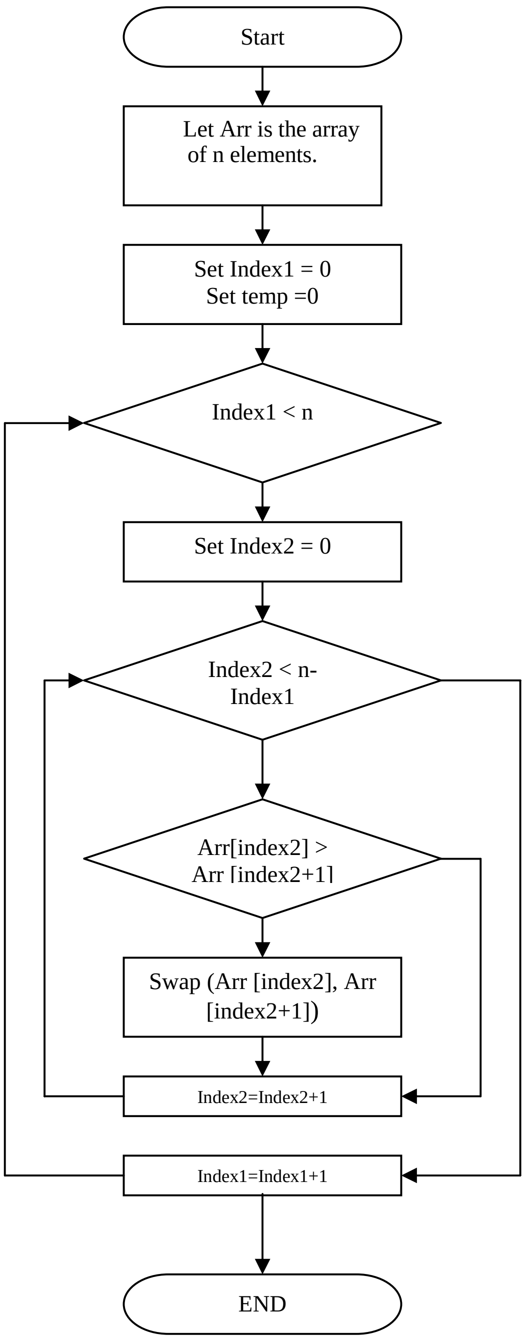
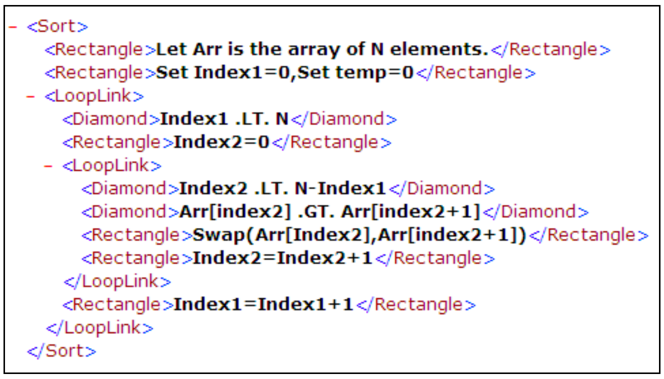


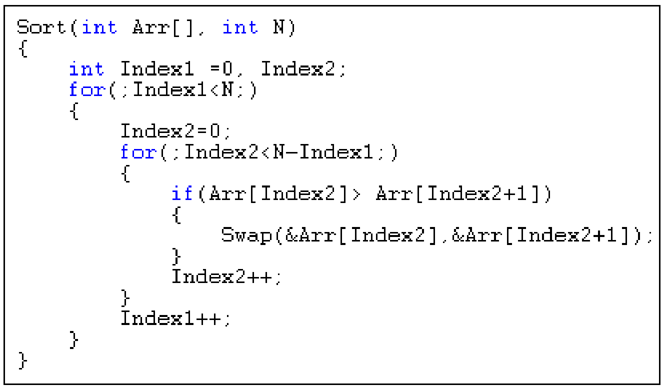
Figure 1. Procedural design of bubble sort algorithm

V. DNSIM BASED DESCRIPTION OF ABOVE PROCEDURAL DESIGN OF BUBBLE SORT ALGORITHM



```
- <Sort>
    <Rectangle>Let Arr is the array of N elements.</Rectangle>
    <Rectangle>Set Index1=0,Set temp=0</Rectangle>
  - <LoopLink>
      <Diamond>Index1 .LT. N</Diamond>
      <Rectangle>Index2=0</Rectangle>
    - <LoopLink>
        <Diamond>Index2 .LT. N-Index1</Diamond>
        <Diamond>Arr[index2] .GT. Arr[index2+1]</Diamond>
        <Rectangle>Swap(Arr[Index2],Arr[index2+1])</Rectangle>
        <Rectangle>Index2=Index2+1</Rectangle>
      </LoopLink>
      <Rectangle>Index1=Index1+1</Rectangle>
    </LoopLink>
  </Sort>
```

Figure 2. DNSIM representation of bubble sort algorithm



```
Sort(int Arr[], int N)
{
    int Index1 =0, Index2;
    for(;Index1<N;)
    {
        Index2=0;
        for(;Index2<N-Index1;)
        {
            if(Arr[Index2]> Arr[Index2+1])
            {
                Swap(&Arr[Index2],&Arr[Index2+1]);
            }
            Index2++;
        }
        Index1++;
    }
}
```

Figure 3. Procedural representation of the algorithm

Following are the details of the above-mentioned DNSIM representation of bubble sort algorithm:

<Sort>
This represents the design entity name. This is the name of design component of and is a part of whole software design. This name will appear in the design repository list and could be used a reference for the design component reusability.

<Algorithm id = "Sort">
Other way of representing this could be as writing the algorithm and algorithm name. This notation is known as the attribute naming convention where algorithm name is attribute of algorithm. That could be written as <Algorithm id = "Sort">. This is could be more elaborative and will be helpful in producing a list of algorithms in the design repository.

<Rectangle >
This is a part of the flowchart component and used to represent the task or processing.

<LoopLink>
This is the part of flow chart component and used to represent the looping implementation or iterations.

<**Diamond**>
This is a part of the flowchart component and used to represent the decision-making.

The other flow chart component could be used as by their representation image style name and as their notations are known for flowchart making.

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. By evaluating metrics at an early stage of design, managers and software developers can make better design choices and identify stress points that may lead to costly difficulties during coding and maintenance.

VI. SIMULATION RESULTS AND CONCLUSION

The simulation is carried out on a set of requirement specification documents of different projects. The key words are collected from the requirement specification documents.

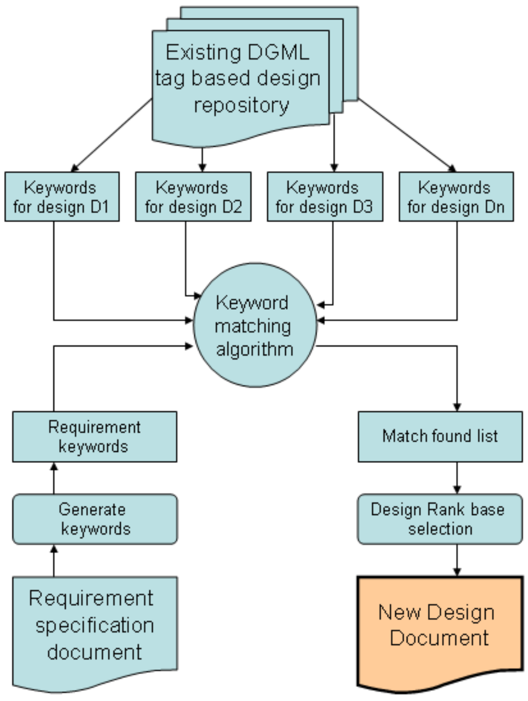
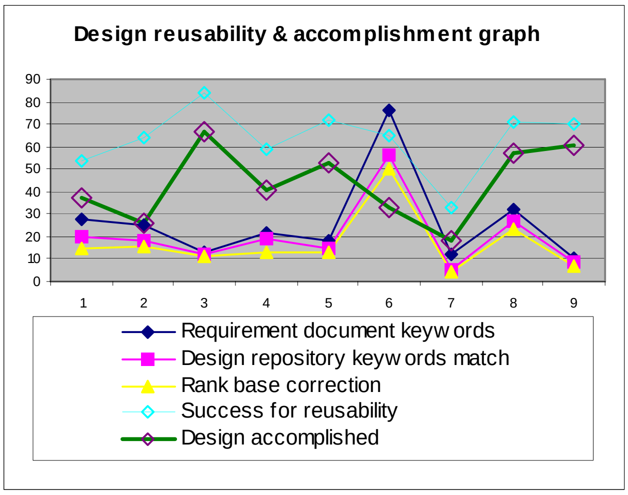


Figure 4. Framework for design notation storage and inference mechanism

These keywords are then compared to the keywords of the existing design documents. The inference mechanism finds the match for the suitable occurrence. The ranking mechanism again assigns a quality check on the outcome of inference result before reusability. The matches found with good rank are used and a new design is proposed. Following are the figures used during simulation.

Requirement document keywords	Design repository keywords match	Rank base correction	Success for reusability (%)	Achievements in finished design after reusability (%)
28	20	15	54	37
25	18	16	64	26
13	12	11	84	67
22	19	13	59	41
18	15	13	72	53
76	56	50	65	33
12	5	4	33	18
32	27	23	71	57
10	9	7	70	61

Figure 5. Figures for applying simulation and success results



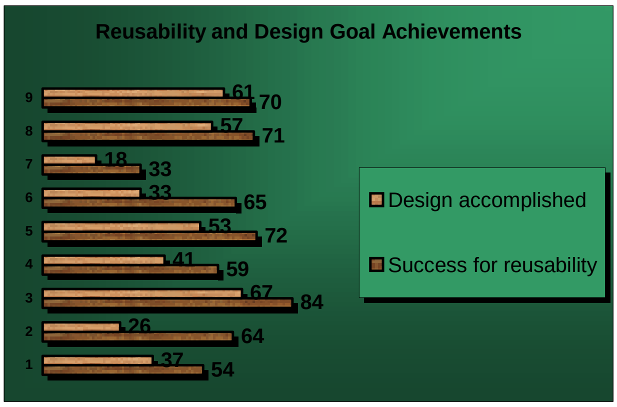


Figure 6. Outcome of the simulation. Design reusability and new design accomplished graphs for give requirements..

VII. CONCLUSION

The simulation is carried out on the requirement specification document of ten different projects. The keywords evaluated from these requirement documents when compared against the stored design document gives more than 33% reusability. That means when creating a new project, at least 33% of the design efforts could be saved which is a good figure to consider design reusability as a new approach. Reusability is counted for the not of design elements used from available design repository. This figure appears after the application of page rank algorithms, which assure the available reusable module of good quality although it decrease the reusable figure that appears after search process.

The design accomplishment figures are the percentage of the requirement fulfillment of the required design. This is evaluated by counting the total design keywords for requirements again the keywords present in the design completed after reusing the existing. This figure further updated because final design after reusing will go through a process of review by human involvement or heuristic. This is important because few of the matches found after inference and page rank algorithm on available designs may not be required.

Also, the evaluated design is of good quality too. This is because the keyword matching is not blind match but is having the power of rank base selection of the outcome of the search process. A rank is the weightage of the design. More the design is used, more is its quality rank. Reusing a particular design again increment the quality factor by one.

VIII. DISCUSSION

The DGML text base storage of the design document gives a new direction in the field of reusability. This text based design representation is having a keyword associated with each design entity. When user requires a new design, he has not to start work from the scratch. The proposed mechanism will read the user requirements and perform the quality base search in the existing design repository and gives the user a first cut of the design document which satisfy more than 33% and save the time and efforts. Further, the reusable design is created from the existing design which already having the good quality and gone through various checks before storing in the proposed notation. This factor gives the satisfaction that the newly created design by proposed mechanism is also having the good quality.

References

[1] Lehman, M. M., and L. Belady, Program Evolution: Processes of Software Change, Academic Press, New York, 1985

[2] Emerging Technologies that Support a Software Process Life Cycle. IBM Systems, 1994

[3] H. D. Rombach, "Design Measurement: Some Lessons Learned," IEEE Software, March 1990.

[4] M. Shepperd, "Design Metrics: An Empirical Analysis," Software Engineering Journal, January 1990.

[5] R. Selby and V. Basili, "Analyzing Error-Prone System Structure," IEEE Trans. Software Eng., 17 (2), February, 1991.

[6] H. D. Rombach, "A Controlled Experiment on the Impact of Software Structure and Maintainability:," IEEE Trans. Software Eng., 13 (5), May, 1987.

[7] L. Constantine, E. Yourdon, "Structured Design," Prentice Hall, 1979

[8] Heineman, G., J.E. Botsford, G. Caldiera, G.E. Kaiser, M.I. Kellner, and N.H. Madhavji.,

[9] Hekmatpour, S., Experience with Evolutionary Prototyping in a Large Software Project, ACM Software Engineering Notes, 12,1, 38-41 1987

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 eleven Ph.D.'s in Computer Science and thirteen students are working under his supervision. He has around 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. His 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.