Research Journal of Applied Sciences, Engineering and Technology 9(5): 346-352, 2015
DOI:10.19026/rjaset.9.1412
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: September 13, 2014 Accepted: October 11, 2014 Published: February 15, 2015

Research Article

Design Parser For CDM Graph Geneartion in Graph Transformation Based User Interface Modeling

[1]Suneeta H. Angadi, [2]S. Mohan
[1]Department of CSE, RNSIT, Anna University, Chennai, Bangalore, Karnataka, India
[2]Department of ECE, CIET, Coimbatore, Tamilnadu, India

Abstract: In the direction of performance analysis form based web application development is reckoned in our work. User interface development part of web application development is further taken into consideration. Navigation aspect is emphasasised; to model navigation class diagram in Unified Modeling Language (UML) to Conceptual Data Model (CDM) mapping is carried through. For achieving transformation, in this study design document is parsed then CDM graph is drawn from parsed design data. Directed CDM graph generation using either critical use case or algorithm is conversed.

Keywords: Class diagram, coceptual data model, performance analysis, user interface modeling

INTRODUCTION

Software development begins with information from requirement and design specification; performance the non functional property, its verification is achieved by analysis of requirement and design document. Functional software is delivred if it addresses requirements and design specifications. Software with performance property is delivered if requirement and design specifications are accurately mapped. Conceptual data model is the most abstract form of data model. The development of CDM is a design activity influenced by various kinds of knowledge including both application domain knowledge and data modelling representation and process knowledge. Graph transformation is the process of transforming one form of the graph into another fom algorithmically; it is the rule based manipulation of graphs (Ehrig *et al*., 1999). Performance engineering field and graph transformation field have similar notations like petri nets, process algebra; due to this reason performance analysis can be performed with graph transformation appoach. Article by Jaime *et al*. (2000) proposes OO-Hmethod for software production from conceptual models considering conceptual design and conceptual modelling patterns. Navigation access diagram to illustrate navigation model is introduced as part of their work; authors state its required to have at least one navigation access diagram for each user type. Software performance engineering field's critical use case based software execution model might be used for representating navigation in the form of graph. Martin (2009) propose mapping of logical data model to user interface model based on graph transformation for modeling user interface elements. In the initial stage of this work CDM graph generation need to be performed; for CDM graph generation navigation aspect need to be taken into account. Towards this goal, in this study an attempt is made to automate design mapping in software development before code is built, we have mapped class diagram to CDM graph, in addition development needs navigation details; navigation support from the wok of Jaime *et al*. (2000) is referred. Graph transfomation approach by Martin (2009) is related to navigation concept present in work of Jaime *et al*. (2000) as part of our work. Work by Mohan *et al*. (2014) focusses performance analysis and for the process of performance analysis; software performance engineering principles, process and models are used. Another approach is CDM graph generation from UML class diagram as mentioned in work of Suneeta and Mohan (2014) with the help of algorithm. In this study an enhancement in the form of design document parsing has been done. Form based web application development is considered here, its user interface development is modeled; for modeling CDM is used and graph transfomation is used to simulate navigation through the application.

LITARTURE REVIEW

Thomas *et al*. (1998) categorized web pages into two groups as text based and GUI or metaphor based; text based is functional and metaphor based is form

Corresponding Author: Suneeta H. Angadi, Department of CSE, RNSIT, Anna University, Chennai, Bangalore, Karnataka, India

based. Chun-Cheng (2010) used interviews with users like type scale rating and principal component analysis to collect and analyze data, author discuss design criteria and main factors influencing web interface design. Article by Kuo-Wei *et al*. (2011) converses redesigned user interface for a case study by considering users degree of cognitive and information acceptance giving importance to human computer interaction. Importance of user interface along with difficulty in design and implementation of user interface is communicated by Brad (1993). Authors set up an extension of conceptual modeling approach known as OO-H method that supports the conceptual design of web applications (Jaime *et al*., 2000). As part of work carried through authours Hallvard (2002) propose a framework for classifying design representations and task modelling language. Antovic´ *et al*. (2012) have analysed entities, their attributes relationship and their influence on the user interface in addition its indicated that software requirements are the basis of developing good user interfcae. Work of Nora and Andreas (2002) implements semi automatic generation of web applications from design models. Dorina (2009) contends about non functional properties such as performance, scalability, reliability, security and safety; Model driven approach for user interface development has been implemented as part of this work; descibed as given next, Process of design document specification to CDM graph mapping begins by reading the design in the form of UML class diagram, then from this design class names and associations are extracted. Design document in XMI fom is parsed to fetch class names and associations; this output is stored in a file, from this file content graph is drawn. Steps of this process are shown in Fig. 1 in addition steps are implemented in methodology section of the paper, Contribution of this paper: Design document parsing, drawing graph from design document parsed data.

Background: Our work whose emphasis is implementation of software performance engineering to achieve non functional property performance, makes use of software performanc engineering field perfomance objectives, software performance engineering process, software execution model along with these even performance analysis activity is considered which is an imporatnt component. Then continues with modeling methodology which includes user interface modeling since user interface modeling is taken as first step of our work, software execution model is incorporated to represent navigation aspect of user interface. Graph transformation approach is used to transform class diagram to CDM graph. Thus navigation aspect is represented with the help of one of the two ways mentioned below:

- Navigation determined by software excution model framed by following critical use case
- Navigation determined with the help of graph tranformation algorithm

Requirement and design analysis, based on performance analysis will be step in achieveing performance attributes like response time, scalability and throughtput in the field of software performance engineering; this design and requirement analysis are part of proactive software performance engineering. Requirement analysis to identify type of software system can be performed by identifying keywords that are part of software requirement specification; then appropriate graph representation to model development is associated. Advantage of modeling is rework can be reduced; to streamline work in this direction we started with software system development which is narrowed to web application development and further cut down to form based user interface development of web application with the help of user interface modelling as given below in Fig. 2.

Conceptual Parameters of our work are tabulated in the Table 1 given next.

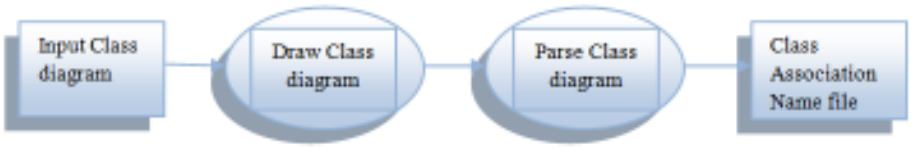

Fig. 1: Design to CDM mapping

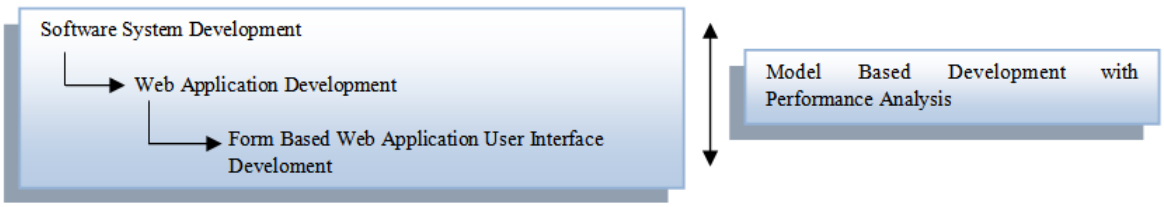

Fig. 2: Performance analysis pursued in software development

Table 1: Conceptual parameters of our work

Field	Approach	Process adopted	Activity	Modelling
Software Performance Engineering (SPE)	Graph transformation	SPE Process, graph transformation based process	Performance analysis	Use interface modelling, software execution model

METHODOLOGY

Process of software design document parsing shown in Fig. 1 begins by considering design document as input; our process focuses on UML class diagram of the design, from the design diagram its required to extract class names and associations, from these CDM graph need to be drawn. UML class diagram part of design is drawn using AgroUML tool available from http://argouml.software.informer.com/ (2009) then design representation in XMI is parsed to extract node names and edges, class names in the UML diagram are node names and association end participant value (determined by association end participant tag) represent source and destination of edge. Algorithm 1 is proposed and implemented for XMI document parsing to map design onto CDM graph its steps are given below, this generated CDM graph represents navigation aspect of the application.

Steps to identify class name:

1. Open XMI file that contains class diagram details in the form of tags
2. Identify tag '<' (opening tag) followed by UML: Class element
   Then word name =
   'ClassName' (this value must be extracted, start with opening single quote and end at closing single quote)
   extract class id field value
   Process continues till closing tag.
3. Repeat step 3 till EOF
4. Store these class-id namepair values in an output file.
5. Display class names.

Algorithm to extract class names:

Steps to identify Association start and end

// Node denotes edge's source and destination classes

1. Open XMI file that contains class diagram details in the form of tags
2. Identify tag '< ' (opening tag) followed by UML:Association element
   Repeat till end of association element
   then UML: AssociationEnd.participant
   then UML: Class
   Nodeid = xmi id reference
   if(Nodetid = nameid.classid)
   writefile(nameid.name)
   writefile(' - - ') //write this character sequence for only source
3. From the out put file generated in step 2 copy graph definition and generate graph using GraphViz tool available from http://graphs.grevian.org/

Algorithm to extract associations

Result of the above algorithm implementation i.e., class names and associations are stored in text file, from this text file CDM graph showing navigation aspect of application is drawn. Further this undirected graph is transformed to directed graph using either:

- Sequence diagram steps of critical use case as present in work of Mohan *et al*. (2014)
- Using algorithm given below referred from the wok of Suneeta and Mohan (2014)

Algorithm UML-CDM (G):

```
//MST is minimum spanning tree generated with algorithm //represented as adj matrix a[][]
//Input:UML undirected graph a[][]
//output:directed CDM graph with minimum spanning //tree(MST)
    apply MST(a)
    directed-graph(MST)
Algorithm directed-graph(MST)
//alg generates directed graph from MST
//Input: MST undirected graph t[][]
//Output:Directed Graph t[][]
1. For i = 1 to n do
   For j = 1 to 2 do //since two column MST
       Es[k] = t[i]->t[j]
       Es[k+1] = t[j]->[i]
       k++
2. read edge sequence numbers
   for i = 1 to n do
if(sn==es[i])
       i->j = 1
       j->i = 0
       es[i] = 0
for i = 1 to n
if (es[i] && sn!= es[i])
//remaining edges have to be inserted from j i
j->i = 1
i->j = 0
t[i][0] = j
t[i][1] = i
```

Algorithm UML-CDM //adopted for relationship association: Above algorithm is suitable for class diagrams with relationship associations, for class diagrams with generalization and specialization following algorithm is used:

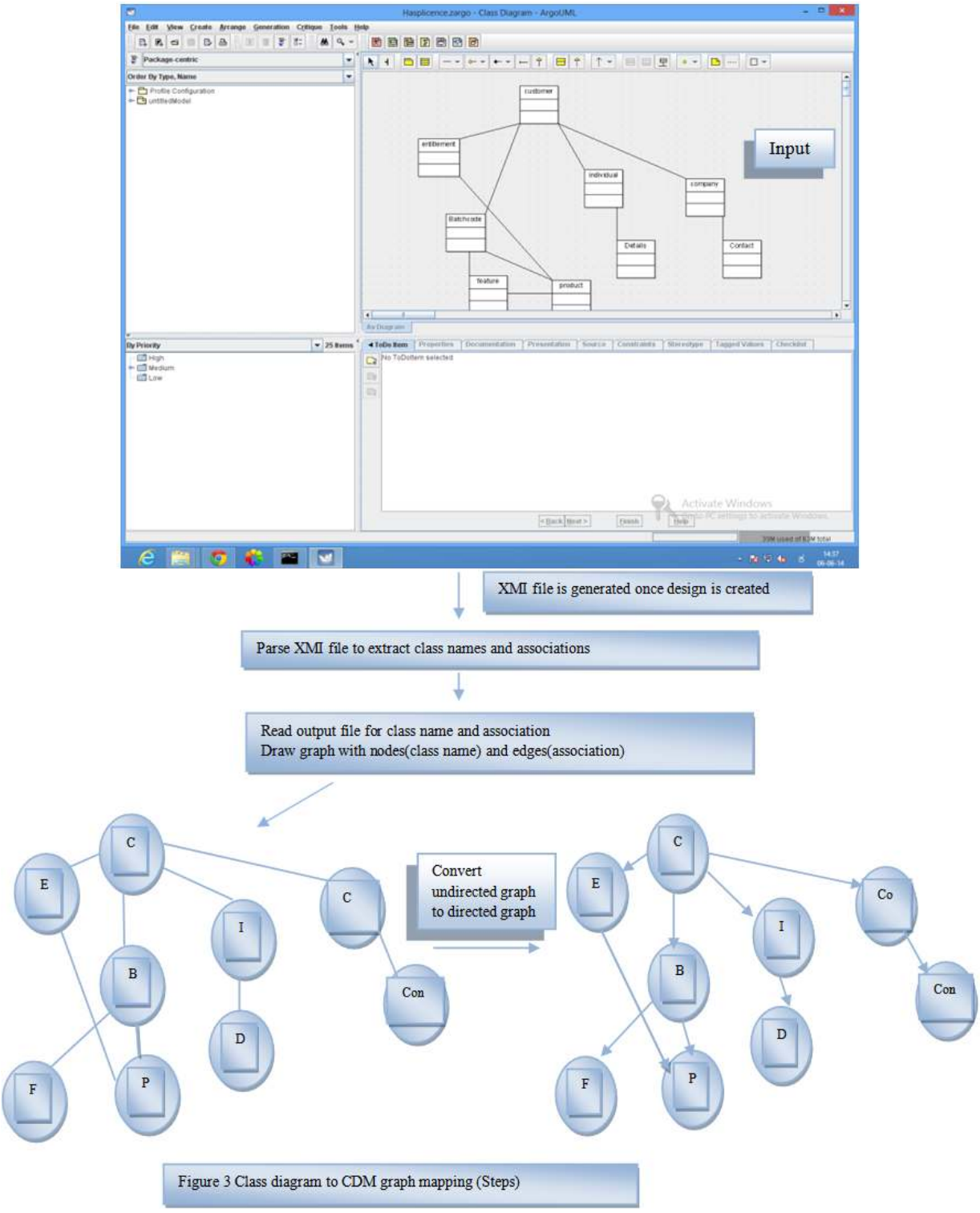

Fig. 3: Class diagram to CDM graph mapping (Steps)

Algorithm CDM_partial_directedGraph (adj[][])
//algorithm for directed edge insertion with partially directed //graph as input
//Input: graph with partial edge directions (Boolean value) from UML class diagram
//Output: Directed CDM graph
For i = 1 to n
For j = 1 to n
If(a[i][j]!=a[j][i])
If(a[i][j] && a[i][j].s==T)

```
a[j][i] = 0
if(a[j][i] && a[j][i].s==T)
a[i][j] = 0
```

Algorithm CDM_partial_directedGraph // Adopted for generalization specialization associations: Steps followed in the process are shown in Fig. 3. Output CDM graph is used for logical data model to user interface modeling mapping which is present in work of Martin (2009) we have referred this graph for navigation aspect to develop user interface, using user interface modeling.

RESULTS AND DISCUSSION

Methodology proposed here as shown in Fig. 1 has been implemented and tested for case studies referred from http://www.uml-diagrams.org/class-diagrams-examples.html, it was observed that class diagram mapping in the form of CDM graph helps in analysis of the process from the navigation viewpoint, implementation has been carried through in Java by considering parser present in javax. Consider case study of Student Attendance Entry software.

Student attendance entry: This software allows faculty to enter attendance of students for different subjects taught by faculties. Description of system in UML class diagram form is given in Fig. 4 (along with algorithm application) results obtained in terms of number of directed edges and number of edges in input graph is depicted in Fig. 5. Class names are shown in screenshot given below.

XMI representation for this example along with UML class diagram element's mapping with XMI id's (marked in bold) is shown below (Fig. 6).

XMI id's generated for classes student, faculty etc and for association's are shown in the Fig. 7, arrows indicate element to which id is associated.

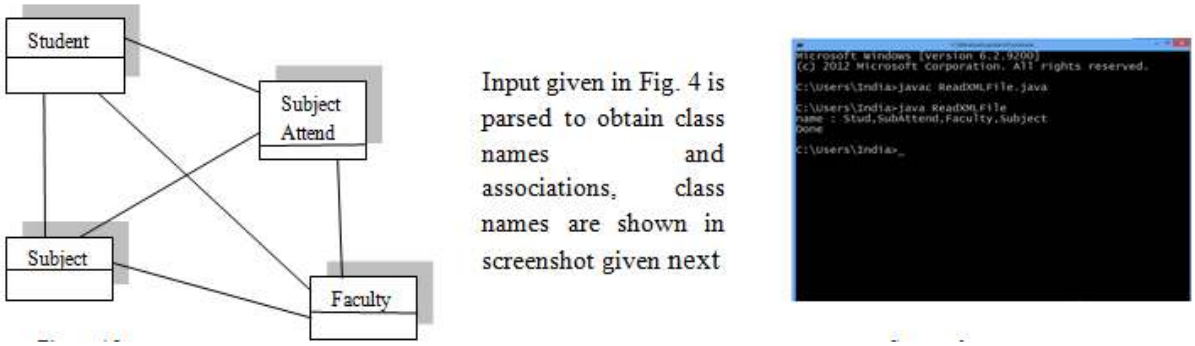

Fig. 4: Input and screenshot

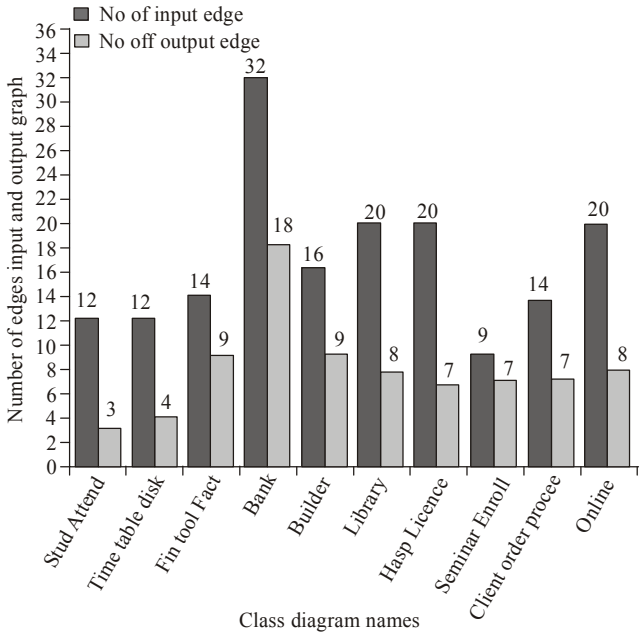

Fig. 5: Number of edges in input UML class diagram and in directed CDM graph

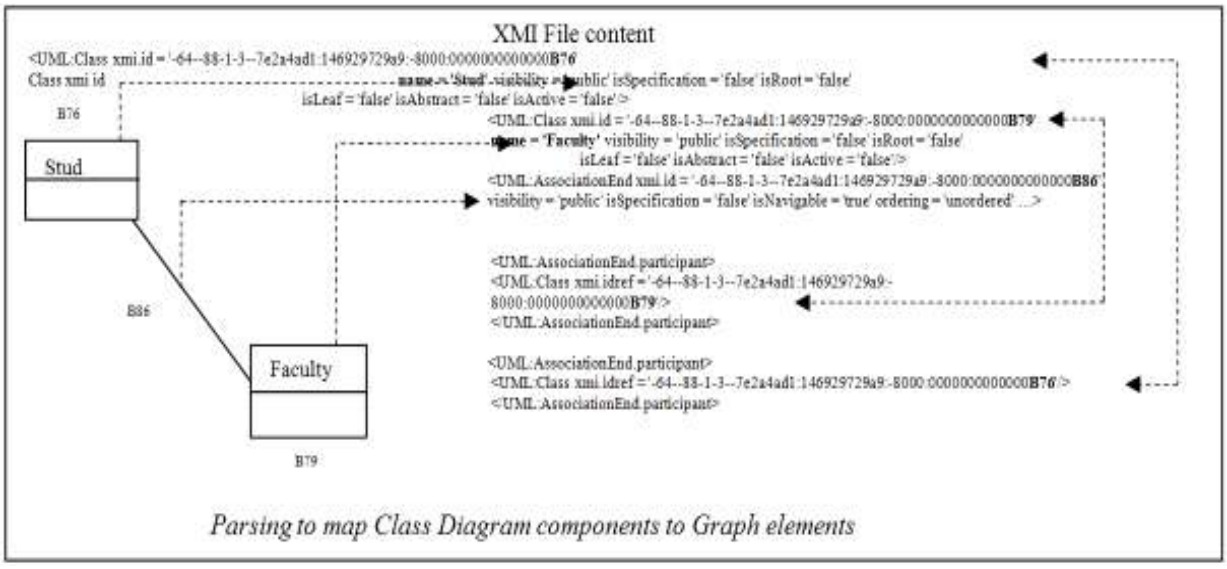

Fig. 6: Parsing to map Class Diagram components to Graph elements

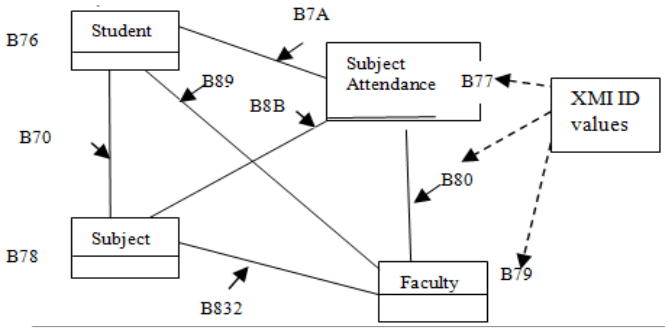

Fig. 7: Class Diagram with XMI ID's generated after designing with AgroUML

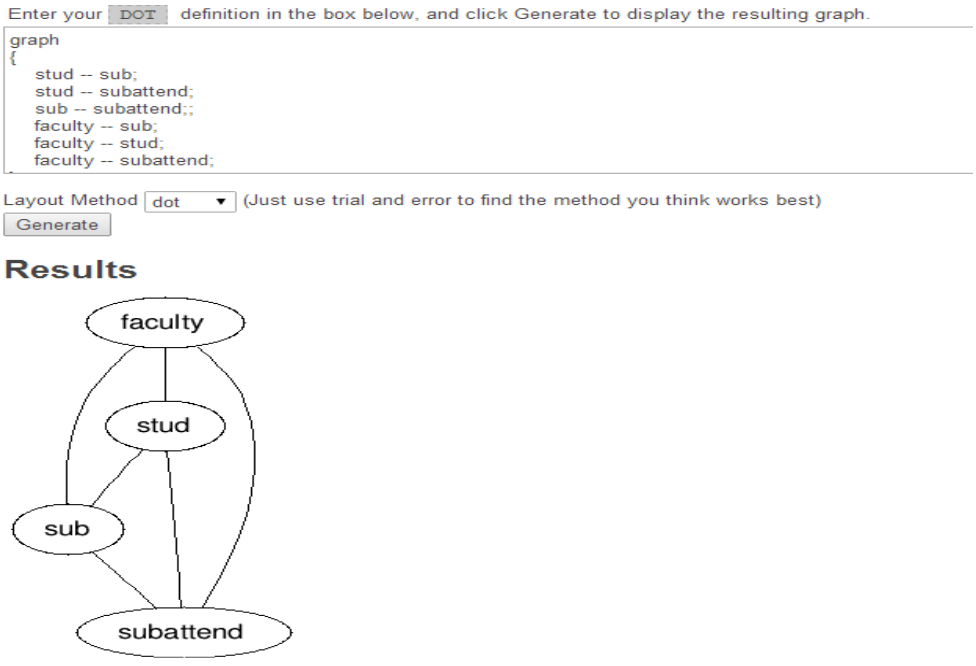

Fig. 8: Graph generation

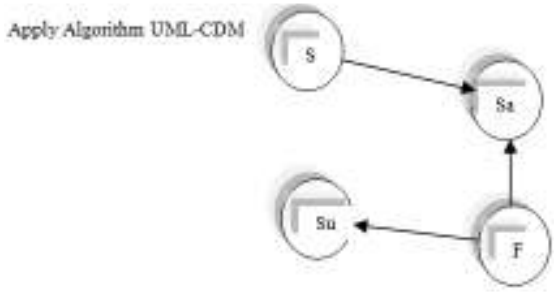

Fig. 9: Output

Definition to draw graph is written by referring output text file which is result of design parsing; graph generated is shown under results header as given below in Fig. 8, to this graph UML-CDM algorithm is applied to generate directed graph given in Fig. 9.

Figure 5 shows number of edges in input UML class diagram and number of edges in directed CDM graph.

Number of edges in input UML class diagram shows associations present in scenario description; number of edges in directed output graph shows navigation aspect in user interface modelling. Number of edges in output graph is derived first by parsing design document then applying directed CDM graph generation algorithm. Number of edges in the input UML class diagram is more than number of output edges in the resultant graph. Edges in output graph focus on analyst's expectation of navigation through the application by following approach mentioned herein, in absence of this kind of approach, developer has to design forms addressing all the possible navigations.

CONCLUSION

Proactive performance engineering can be realized by incorporating performance objectives or principles in early development stages of software development. Specifically in requirements gathering and design phases, analysis performed in these stages will result in performance valued software development. Modeling approach with graph transformation can be effective by relating these parameters, as part of our research work we have performed design document parsing. This task will ensure mapping of analyst's specification to implementation. Class diagram specification for user interface development has been transformed to CDM directed graph, directed CDM represents navigation aspect; advantage of this approach is requirement achievement with modeling by reducing rework. Future work in this direction can be measuring amount of work reduced and generation of all possible graphs showing all navigation paths allowing the designer to choose appropriate one, integrating all modules of our work in the form of tool.

REFERENCES

Angadi, S.H. and S. Mohan, 2014. Graph transformation based conceptual data model graph generation for user interface development. Int. Rev. Comput. Softw., 9(6).

Antović, I., S. Vlajić, M. Milić, D. Savić and V. Stanojević, 2012. Model and software tool for automatic generation of user interface based on use case and data model. IET Softw., 6(6): 559-573.

Chun-Cheng, H., 2011. Factors affecting webpage's visual interface design and style. Proc. Comput. Sci., 3(2011): 1315-1320.

Ehrig, H., G. Engles, H.J. Kreowski and G. Rozenberg, 1999. Handbook of Graph Grammars and Computing by Graph Transformation. Vol. 2: Application, Languages and Tools. World Scientific, Singapore, pp: 105-180.

Hallvard, T., 2002. Model-based user interface design. Ph.D. Thesis. Department of Mathematics and Electrical Engineering, Faculty of Information Technology, Norwegian University of Science and Technology.

Jaime, G., C. Cristina and P. Oscar, 2000. Extending a conceptual modelling approach to web application design. In: Wangler, B. and L. Bergman (Eds.), CAISE, 2000. LNCS 1789, Springer-Verlag, Berlin, Heidelberg, pp: 79-93.

Koch, N. and A. Kraus, 2002. The expressive power of UML-based web engineering. Proceeding of the 2nd International Workshop on Web-Oriented Software Technology (IWWOST'02).

Martin, M., 2009. A contribution to user interface modelling based on graph transformations approach. Proceedings of the International Workshop on Enterprises and Organizational Modeling and Simulation (EOMAS, 2009).

Mohan, S., S.H. Angadi and G.T. Raju, 2014. Performance analysis of graph transformation based user interface modeling from conceptual data model. Proceedings of the 2nd International Conference on Applied Information and Communications Technology (ICAICT'2014).

Myers, B.A., 1993. Why are human-computer interfaces difficult to design and implement? Technical Report, CMU-CS-93-183, Carnegie Mellon University Pittsburgh, PA, USA.

Petriu, D.C., 2009. Software model-based performance analysis. Proceeding of the MDD4DRES, pp: 1-19.

Powell, T.A., D.L. Jones and D.C. Cutts, 1998. Web Site Engineering: Beyond Web Page Design. Prentice-Hall Inc., NY.

Su, K.W., H.Y. Chang and K.C. Wang, 2011. A practical approach for user interface design of a G2B based official document exchange system in Taiwan. Int. J. Innov. Comput. I., 7(11): 6423-6436.