International Journal of Advances in Engineering Research http://www.ijaer.com

(IJAER) 2016, Vol. No. 12, Issue No. V, November e-ISSN: 2231-5152, p-ISSN: 2454-1796

SOFTWARE DATA GRAPHS USING ONTOLOGY-DRIVEN SOFTWARE EVOLUTION AND ITS APPLICATION

P.Kiran Kumar[1], Dr.V.Khanna[2]

Research scholar, Dept of C.S.E, Bharath University, Chennai, Tamil Nadu, India[1].
Dean of Info, Bharath University, Chennai, Tamil Nadu,

ABSTRACT

Software Data graphs encompasses the development and evaluation of methods for graphically representing different aspects of methods of software, including its structure, execution and evolution. Creating data graphs helps the user to better understand complex phenomena. It is also found by the software engineering community that a data graph is essential and important. In order to visualize the evolution of the models in Software Evolution, authors have proposed a framework which consists of areas and key features for the assessment of Software Evolution process and addresses a number of stakeholder concerns. The framework is derived by Ontology the application of the Question Metric Paradigm. This paper aims to describe an application of the framework by considering different data graphs tools/CASE tools which are used to visualize the Features in different views and to capture the information about stakeholders during their evolution. Comparison of such tools is also possible by using the framework.

Keywords: *Ontology-Driven Software Evolution, Software Data graphs, Data graphs tools.*

1. INTRODUCTION

Data graphs are used to enhance information understanding by reducing cognitive overload. Using data graphs methodologies and tools, people are often able to understand the information presented in a shorter period of time or to a greater depth. The term “data graphs” can refer to the activity that people undertake when building an internal picture about real world or abstract entities. Visualizing can also refer to the process of determining the mappings between abstract or real-world objects and their graphical representation. This work uses the term “data graphs” in the later sense: the process of mapping the evolution of models to the stakeholder concerns.

The introduction of Ontology Driven Engineering (ODE) needs a new style of evolution i.e. Ontology - driven Software Evolution. The first fundamental premise [1] for Ontology-driven Software Evolution (ODSE) is that evolution should be a continuous process. The second premise is that reengineering of legacy systems to the Ontology-driven of the paradigm should be done incrementally.. Due to these multitude languages in ODSE, there is a need to have the model (Ontology) interaction, integration, mapping and transformation.. For this purpose multiple views for ODSE have been proposed in [9]. Stakeholder's involvement in ODSE typically has interests in, or concerns relevant to that system. The ability of models to evolve gracefully is becoming a concern for many stakeholders. Due to different and interrelated models used to design an entire system in ODSE, the concerns of stakeholders may differ from one role to another role that a stakeholder play during the life time of a software project. So, data graphs provides better solution to understand the complex information during evolution of the models. This can be done by using the existing data graphs and/or CASE tools. Software Data graphs tools use graphical techniques to make software artifacts visible.

Evaluating a particular data graphs tool for ODSE is essential. Common practice is that some set of guidelines are followed and a qualitative summary is produced. However, these guidelines do not usually allow a comparison of competing techniques or tools. A comparison is important because it identifies possible flaws in the research area or software development. Thus, a framework for describing attributes of tools is needed. Once the tools have been assessed in this common framework, a comparison is possible. However, a framework can be used for comparison, discussion, and Formative evaluation of the tools. Such framework was proposed in [8]. So, the major contribution of this paper is to show how the framework can be applied to compare the Data graphs Tools which is presented in section 4. A Framework for visualizing Ontology-Driven Software Evolution falls into key areas (views): Context View, Inter-model View, City View, Metric View, Transformation View, Evolution View and Evaluation view [9] and 22 Key features are identified for all key areas. The framework is used to evaluate data graphs tools and it is also used to assess tool appropriateness from a variety of stakeholder perspectives.

This paper is structured as follows: Section 2 discusses the related work. Section 3 summarizes the framework. Section 4 discusses an application of the framework by considering different Data graphs tools/CASE tools. Section 5 outlines the conclusions and giving an outlook on future work.

2. RELATED WORK

This section reviews the literature related to the fields of Software Data graphs, Software Evolution Data graphs and Model Driven approaches.

Source Viewer 3D [6] is a Software Data graphs framework that builds on the See Soft metaphor. 3D can show large amounts of source code in one view. Object based manipulation methods and simultaneous alternative mappings are available to the user. The types of user tasks and interactions that are supported by sv3D, is not directly related to solving/visualizing specific software engineering tasks and it is a prerequisite for a software data graphs tool.

Architecture to Support Software Data graphs [7], borrows the field of Ontology Driven Engineering (ODE) to assist with the creation of highly customizable interfaces for Software Data graphs. In order to validate the architecture, framework for Eclipse was developed. Model Driven Datagraph is intended to address the customization of information data graphs tools, especially in the program comprehension domain. The architecture describes how to leverage the work done in the Model Driven Engineering community and apply it to the problem of designing data graphs tools.

The Graphical Modeling Framework (GMF)[12] project for eclipse has facilities to allow ontology to define graphical editors for their data. These graphical editors can be used as viewers, however, the views they support are limited to simple graphs with containers. The GMF project currently lacks the ability to specify "Query Result" data graphs.

An Open Framework for [10] visual mining of CVS based software repositories has three major aspects are data extraction, analysis and data graphs. An approach was proposed for CVS data extraction and analysis. CVS data acquisition mediator used to extract the data from CVS repositories. Analysis techniques are used to analyze the raw data retrieved from the CVS repositories from CVS Querying. It also provides the comparison of the open source projects. CVS graph is a software tool used to visualizing project at file level. This open framework does not provide the data graphs of models, it provides for program at file level only.

CVS scan[11] is a tool in which a new approach for data graphs of software evolution was developed. The main audience targeted here is the software maintenance community. The main goal is to provide support for program and process understanding. This approach uses multiple correlated views on the evolution of a software project. The overall evolution of code structure, semantics, and attributes are integrated into an orchestrated environment to offer detail-on- demand. And also provides the code text display that gives a detailed view on both the composition of a fragment of code and its evolution in time. It is focused on the evolution of individual files.

2.1 Motivation for Framework and its Application

There are number of frameworks exists in the literature for comparison and assessment of the various CASE tools. Comparison of these tools is essential to understand their differences, to ease their replication studies, and to discover what tools are lacking. Such a comparison is difficult because there is no well-defined comprehensive and common comparative study for different category of the tools. For design recovery tools a comparative framework [14] was derived for comparison. This framework comprises eight concerns, which were further divided into fifty three criteria and which were applied on ten design recovery tools successfully. Another framework [7] also exists in the literature for comparison and assessment of the software architecture data graphs tools. Software architecture is the gross structure of a system; as such, it presents a different set of problems for data graphs than those of visualizing the software at a lower level of abstraction. Both the frameworks applied against the stakeholders concerns. From this analysis it is easy to know that how a selected tool satisfies the stakeholder concerns. Thus the motivation for this work lies in above mentioned two frameworks. Combining the data graphs approach with ODSE is essential to understand the evolution of models in a better way. Large numbers of data graphs tools are available in the literature. Among them many tools support the evolution at source code level, data level. This work aims to find out the data graphs tools which support the data graphs at model level. As such there is no framework exists in the literature to evaluate tools which are useful for the Data graphs and also to understand the evolution of the models with respect to stakeholder perspectives. Hence this paper aims to evaluate the already proposed framework for data graphs of ODSE.

3. FRAMEWORK SUMMARY

This section provides the summary of the already proposed framework for Ontology-driven Software Evolution data graphs in [8]. The framework has key areas (views) for visualizing ODSE: Texere View, Stable View, Dynamic View, Metric View, Implementation View, QualityView an. These six views are derived based on the viewpoints and were discussed in detail [8]. The dimensions proposed in the framework are not proposed as formal representation of the characteristics of ODSE, but are necessary for discussion about, and evaluation of, such dimensions with respect to stakeholders and tools which they use. The Question/Metric (QM) paradigm [9] is used to identify the questions and then to enable the formation of framework features.

The primary goal of the framework is to assess and understand the evolution of the models in model driven software evolution. The framework is derived from an extensive analysis of the literature in the area of software data graphs with special emphasis on model driven software evolution. Each of the seven views is a conceptual goal which the framework must satisfy. It is this that makes the application of the QM Paradigm [13] straightforward.

Framework summary with its goals, questions are given in Table.1 First column represents the key features (questions) which are abbreviated with view names. Second column represents the key areas (views) . The responses for these questions will be the values used in the Table 2.

Throughout the scenarios, we shall refer to different widgets, or areas, of our visualization tool, as follows (see also Fig. 2)[1]. The `Quality attribute tree' shows the hierarchy of quality attributes according to a particular quality model, in this case the extended ISO-9126 or `Quint' model [17]. The `Quality attributes of interest' area shows the quality attributes of interest, which capture the customer's idea of `quality' and are an input to the remainder of the audit. The `Effect matrix' shows the quality criteria relevant to the cur-rent audit. Relevant criteria are those criteria that have a positive or negative effect on one or more attributes of interest, as well as criteria for which it is determined that they should or should not be present in the product.

Quality Attribute represents a quality attribute that can be further specialized in subattributes. For example, in ISO 9126 `efficiency' is further divided into `time behaviour', `resource utilisation', and `efficiency compliance' [10].

Effect is a reified relation from criterion to quality attribute, having two attributes: effect type (positive or negative) and [0 . . . N] reciprocal relations to other `effect' relationships, which indicate the relative strength (stronger than, weaker than, or comparable) of the `effect' relations.

Audit models a software product audit in which particular quality criteria have been used to assess a prioritized set of quality attributes. The *usedIn* relation captures the relation between criteria and audits. The *priorityIn* relation captures the relation between quality attributes and audits.

Key Features	Key Areas
	Key Area 1 : Texere(TV)
TV1	Does the data graphs provide context of a model?
TV2	Does the data graphs provide the scope of a model
TV3	Does the data graphs express the model completely
including all its surrounding elements?	
	Key Area2:Stable Representaion view
SV1	Types of software Architectures needed
SV2	Accommodate huge information
SV3	Backup of software architecture information
	Key Area 3:Dynamic Representation View
DV1	Live collection
DV2	Replay Data

DV3	Multiple action of data
DV4	Support unstabled data
	Key Area 4 : Metric View (MV)
MV1	Does the data graphs provide the metrics to estimate the impact analysis of the models during evolution?
MV2	Does the data graphs provide the data graphs techniques to know the evolution of the models?
	Key Area 5:Implementation
I1	Platform dependence
I2	Multiple Users
	Key Area 6:Quality
Q1	High completeness
Q2	Dynamic changing arch

4. APPLICATION OF THE FRAMEWORK

This section describes an application of the framework. For this purpose tools which are mainly research oriented and non commercial tools are considered. These tools are also having the features which are necessary for data graphs of models. The expensive commercial tools such IBM rational Rose Suite, Enterprise architect etc. are not considered here.

4.1 Visual Paradigm for UML

Visual Paradigm [16] for UML 6.4 (VP-UML) is a powerful visual UML CASE tool. It is designed for a wide range of users, including software engineers, system analysts, business analysts, and system architects like who are interested in building software systems reliably through the use of Object-Oriented approach. VP- UML can run in different operating systems. It supports more than 20 diagram types including UML 2.1, BPMN, SysML, ERD, DFD and more. Different editions are also available such as Enterprise, Professional, Standard, Modeler, and Personal are commercial editions. Community and Viewer are non-commercial editions. It supports a rich array of tools. One special feature is Resource-Centric interface, which lets the user access modeling tools easily without referring back and forth from the workspace to various toolbars. Users can draw diagrams or models as with a pen and paper, executing complicated modifications with just a click and drag, creating completely visual environment

It is observed that the names of the features in VP-UML differ from the features of the framework. But the purpose and intention of the features are same. So, they have full support for those features that labeled as 'Y' in the Table 3. Transformation of the models such as model to model, model to code and code to model available in the tool but transformation rules and languages are not available. Hence, features as SV4, SV5 are not applicable (NA). MeV1,MeV2, MeV3, MeV4 features for metrics of the models and which are not in the VP-UML tool that is shown in the Table 3 as ' NA'. Features such as IV2, EaV3 are not mainly supported in the tool i.e. shown in the Table 3 as 'N?' Data graphs of the models by using different diagrams is possible but the techniques are not available. So, the response is 'N?' for Mev2. Stakeholder's feedback (EaV3- N?) is not mainly provided, but the user can store their opinions/ideas about the evolution of the models design problems and many design issues and rules is also available. The remaining two features like 'transformation rules' and 'transformation languages' are not applicable and not supported by these three tools. By comparing the tools under this common framework a stakeholder can easily understand and asses the tools and can find out the flaws in a particular tool.

From the comparison of various features of the three tools it is observed that still there is a need to consider few more possible data graphs/CASE tools which are exists in the literature. It is possible to check the unsatisfied features of the three tools can be satisfied by the other tools and also possible to know the role of the data graphs tools in MoDSE. From the comparison of number possible tools framework can be strengthen further. Another application of the framework is to evaluate stakeholder concerns considered in the framework against the concerns of the software practitioners (stakeholders) from diverse organizations. These are the subjects of the future work.

4.2 Visual Design

We followed several well-known design principles in information- and software visualization [4, 6]. First and foremost, our visual design is simple. Each of our four linked views supports a user task: the quality attribute hi-erarchy for browsing all available attributes and selecting those of interest; the attributes-of-interest view for selecting attributes for a given analysis; the effect matrix for showing relations between criteria and attributes and criteria proper-ties; and the criteria matrix for showing relations between criteria (Fig. 2). We use 2D matrix *layouts* to show relations, rather than graphs or 3D layouts. 2D matrix layouts are highly scalable and simple to use, as shown by many software visualization examples [1, 8]. Third, we use a small set of contrasting *colors*, which is effective in attracting the user's attention to salient events, *e.g.* large positive or negative ranks (effect matrix) or conflicting relations (criteria matrix). Finally, *interaction* is simple and directly doable on all views: just a sequence of sorts and selects. Overall, the tool's minimal design and classical GUI made it easily usable and accepted by its target group, and effectively lets users perform `what if' scenarios in just a few mouse clicks. As such, our tool and its application fits in the newly emerging Visual Analytics discipline [11]: in-stead of being a static data presentation, our tool guides and supports the user's *decision and reasoning process*. Interaction, linked views, and continuously changing the displayed data based on the decision path are essential elements to this visual analytics design.

The auditors of DNV-CIBIT who assessed our tool re-acted very positively. Especially the easy selection of quality criteria and the way the tool invites the user to `play around' and consider `what if' scenarios were cited as the tool's main benefits [16].

4.3 Technical Design

The tool's technical design relies on the use of semantic technologies. The ontology is implemented using the `Web Ontology Language' (OWL), which is endorsed by the World Wide Web consortium and supported by various ontology editors and reasoning engines. The QuOnt ontology presented in Section 3 can be expressed in OWL quite straightforwardly.

The constraints from Table 1 are expressed using the Se-mantic Web Rule Language (SWRL), an OWL-based rule language. Like OWL, SWRL makes an `open world' assumption. Briefly, this means that the absence of a statement does not necessarily mean the statement is false. Hence, in the OWL implementation of QuOnt, the absence of the statement that a criterion `should be used' in an audit does not automatically imply that the criterion `should not be used'; it only means that it is not known yet whether the criterion should be used. This mimics the way in which auditors reason about quality criteria.

Another implication of the open world assumption is that OWL and SWRL only support monotonic reasoning; only new facts can be introduced and existing facts cannot be changed. Consequently, SWRL does not support negation (¬) nor disjunction (∨). Since many of the constraints in Table 1 use negation and/or disjunction, this poses some problems when modeling QuOnt constraints as SWRL rules. Fortunately, many of the problems of constraint implementation can be solved. The lack of negation can be largely overcome by introducing a `notUsedIn' relation for quality criteria that should not be used in an audit. With this new relation, some constraints can be rewritten to eliminate the open world assumption where appropriate. For instance, in SWRL the relation constrains $_{X,Y}$ can be implemented as two rules: $\text{notUsedIn}_X \Rightarrow \text{notUsedIn}_Y$ and $\text{usedIn}_Y \Rightarrow \text{usedIn}_X$.

Still, two constraints cannot be implemented in SWRL: the `enables' relation that implies the negation of the `con-strains' relation, which violates monotonic reasoning; and the `overrides' relation that is a disjunction of the `forbids' relation. Although this is unfortunate, we found that the `enables' relation (which is simply defined as `a weak form of constrains' [12]) has limited practical value. Also, given the open world assumption, the tool still allows to define criteria that at the same time `should be' and `should not be' used in an audit, which is in essence the goal of the `overrides' relation in a closed world assumption.

5. CONCLUSIONS AND FUTURE WORK

An application of the framework for visualizing Ontology-driven Software Evolution has been presented. The research oriented, non commercial tools such as ArgoUML, MetricView Evolution, and Visual Paradigm for UML are considered for the framework's application. These three tools have compared successfully under this common framework. From this comparison it is observed that a single tool does not consist of all the features of the framework and each tool has its own intensions and purposes. But, by using these three tools all the features are satisfied except four features. Among these two features such as 'multiple dimensions of evolution', 'stake holder's feedback' are partially supported by the two tools. But, ArgoUML has provided the feature such as 'Cognitive Psychology' which provides freedom for a stakeholder (designer) to make design decisions, to resolve

REFERENCES

[1] J. Abello and F. van Ham. Matrix zoom: A visual interface to semi-external graphs. In *Proc. InfoVis*, pages 183–190. IEEE, 2004.

[2] A. Akerman and J. Tyree. Using Ontology to Support De-velopment of Software Architectures. *IBM Systems Journal*, 45(4):813–825, 2006.

[3] L. Babu T., M. Seetha Ramaiah, T. Prabhakar, and D. Ram-babu. ArchVoc–Towards an Ontology for Software Archi-tecture . In *Second Workshop on SHAring and Reusing ar-chitectural Knowledge / Architecture, Rationale, and Design Intent (SHARK/ADI)*, Minneapolis, MN, USA, 2007.

[4] S. Card, J. Mackinlay, and B. Shneiderman, editors. *Read-ings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, 1999.

[5] L. Carlsen. Hierarchical Partial Order Ranking. *Environ-mental Pollution*, 155(2):247–253, 2008.

[6] K.Madhavi, A.Anand Rao, 'A Framework for Visualizing model-driven Software Evolution', IEEE International Advanced Computing Conference (IACC'09), March 2009, Patiala, Punjab, India, pp 1785-1790. Published in IEEE Xplore.

[7] K.Madhavi, A.Anand Rao, 'Ontology-driven Software Evolution- The Multiple Views', International MultiConference of Engineers and Computer Scientists (IMECS 2009), March 2009, Hong Kong, pp 1089-1094.

[8] Arie van Deursen, Eelco Visser, and Jos Warmer.: 'Ontology-driven Software Evolution: A Research Agenda', In Dalila Tamzalit (Eds.), Proceedings 1st International Workshop on Ontology-driven Software Evolution, University of Nantes, 2007. pp. 41-49.

[9]Christian F.J. Lange, Martijn A.M. Wijins, Michel R.V. Chaudron.: 'Metric View Evolution: UML-based Views for Monitoring Model Evolution and Quality', IEEE 11th European Conference on Software maintenance and Reengineering (CSMR'07), Amsterdam, the Netherlands, March 2007, pp 327-32