IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS), ISSN: 2249-9555
Vol. 2, No.2, April 2012

"OCET" Ontology Cost Estimation Tool

Prof. Dr. Abdul Hamid M. Ragab

Dept. of Information Systems, Faculty of Computing and Information Technology, King Abulaziz University, Jeddah, Saudi Arabia

Dr. Abdulfattah Suliman Mashat

Dept. of Information Technology, Faculty of Computing and Information Technology, King Abulaziz University, Jeddah, Saudi Arabia

.

***Abstract*-- In this paper, software design and implementation for project cost estimation tool is presented. The tool is based on the ONTOCOM model. It is implemented using Microsoft visual studio ASP.Net C#. It can run easily on Personal Computers. It has user's friendly interfaces, where project designers can access it through helpful labeled screens and menus. Project estimated size and effort results are obtained in the form of numerical and graphical charts. The tool is tested successfully using several previously data projects. The methodology used is compared with other cost estimation methods. Results show that, the proposed design provides several advantages to software vendors, among these are: speed, accuracy and adaptability, and it can reprogram easily to carry out required tasks.**

Keywords- ONTOCOM; cost estimation tool; project management; ontology engineering; software implementation.

I. INTRODUCTION

Software cost estimation is important for budgeting, risk analysis, project planning and software improvement analysis. Software project cost estimation techniques are required in order to compute project cost expressed in person months. This process is an essential part of software project management life cycle that is usually done by software project providers before implementing the project. Many cost estimation models are used by software project vendors, among these are COCOMO models [1, 2]. The way to measure the size of a software system; in these models; usually expressed in lines of code or function/object points [3]. These models; however; cannot be directly applied to ontologies, due to the fact that the implementation of an ontology is mostly realized using tools such as ontology editors. The main size factor to express the complexity of an ontology is not given by the actual size of an implementation in a specific representation language, but by the number of ontological primitives (concepts, properties relations, functions, constraints and axioms) contained by the conceptual model. For these reasons, COCOMO models are not suitable to estimate software projects cost that are intended to be implemented using ontology engineering.

In this paper, we propose software design and implementation for an automation tool that can be used to estimate software project cost that is based on ontology engineering using ONTOCOM model [5, 6]. The ONTOCOM model is a cost estimation model for the area of ontology engineering, whose goal is to predict the cost; expressed in person month (PM); arising in typical classes of ontology engineering processes such as ontology building, reuse or maintenance. The ONTOCOM cost model can be permanently calibrated and refined with the collection of empiric data on person month efforts spent in developing real-world ontologies. A parametric prediction equation contains product personnel and project management-related effort multipliers (EM) are used to adjust the nominal development effort, reflecting the specialties of the ontology and of the underlying engineering process. The proposed tool has many advantages for software project designers among these are: Speed- since it process projects information much more quickly. Repetition- since same task can be done over again. Accuracy: since detailed work can follow precise instructions without error. The quality of the work can be done of the same standard. Adaptability- the tool can be reprogrammed to do different other needed tasks. Reduce cost – since the tool can operate several continues hours economically. Ease of Use- this is provided using friendly user interfaces. Help and Support- are provided through tutorials and online documentations when required.

II. COST ESTIMATION FOR ONTOLOGIES

A. Common Estimation Techniques

Cost estimation consists of techniques for planning, estimating, and monitoring the cost, budget, or schedule of a project. There are several common approaches to cost estimation. These can

include [4]: 1-Analogy Method: In this method, the cost associated with similar projects should have similar costs. 2-Bottom-Up Method: This method tries to identify specific components and estimates the costs associated with the development of each component. Subsequently it calculates the overall effort as the sum of its parts. 3-Top-Down Method: In this method, a partition is done at a certain phase in the project where such a partition is justifiable. It is basically the opposite of the Bottom-Up approach, applicable in situations where at an early stage of the project the components cannot be identified and only global properties are known. 4-Expert Judgment/Delphi Method: This method involves a structured process of data collection based on expert opinion about the efforts associated with different aspects of the project. 5-Parametric/Algorithmic Method: This approach uses a mathematical formula to calculate the effort based on a statistical analysis of data from previous projects. It tries to improve accuracy and find dependencies between cost factors.

B. The Top Down Breakdown Methodology

The top-down partitioning considered by ONTOCOM is based on a study of several ontology development methodologies [5]. A typical ontology engineering process depicts the following development steps, shown in Fig.1:
1- Requirements analysis: It consists of tasks such as analysis of project settings based on a pre-determine set of requirements, knowledge gathering activities and use or reuse of any information sources. 2- Conceptualization: Where, the application domain is modeled in terms of ontological primitives such as concepts, properties, or axioms. 3- Implementation: Where, the conceptual model is implemented in a language, whose expressiveness is appropriate to the richness of the model. 4- Evaluation: Where, the resulting ontology is evaluated in a manual, semi-automatic or automatic way after which the ontology can undergo changes based on the results of the evaluation.

C. ONTOCOM Estimation

ONTOCOM is an important model used to investigate the economic aspects of knowledge structures for ontology development. It deals with estimating the development effort needed to build an ontology, taking into account all the phases in the ontology life-cycle. It uses the well known parametric approach of the Constructive Cost Model (COCOMO II) [2] to derive a similar cost model for ontologies. ONTOCOM applies a parametric formula to calculate the effort in person months,

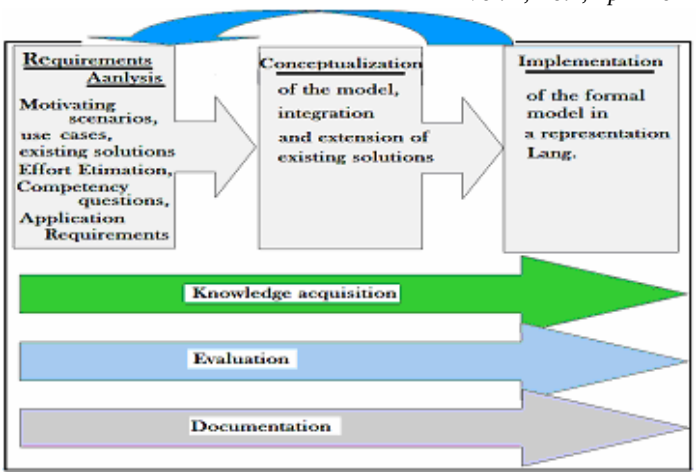

Fig.1 shows the Top down breakdown methodology, Adapted From [4].

statistically calibrating the formula based on expert input and historical data from developers. The ONTOCOM model is realized in three main steps. First, a top-down work breakdown is done along the phases of the ontology engineering process. Second, a set of cost drivers and values associated with pre-defined intervals are proposed and evaluated by experts in the field of ontology development. Third, an a-priori model is proposed based on a mathematical formula after which empirical (historical) data from previous ontology building projects are gathered and used in conjunction with the expert data to statistically calibrate the model and analyze dependencies between cost drivers. This calibration results in a better and a validated a-posteriori model. The ONTOCOM model operates as follows:
1-Ontology Lifecycle Phases: The top-down breakdown of ontology engineering processes is used to reduce complexity by decomposition. It defines the life cycle of the ontology phases: Building, Maintenance and Reuse. Ontology Building- includes the sub-tasks like: specification, conceptualization, implementation, instantiation and evaluation. Ontology Maintenance- involves costs related to getting familiar and updating the ontology. Ontology Reuse - accounts for the efforts related to the re-usage of existing source ontologies for the generation of new target ontologies and involves costs related to finding, evaluating and adapting the former ones to the requirements of the latter.
2-Specify Cost Drivers: As a consequence, estimating the effort (in person months PM) related to ontology engineering is reduced to a sum of the costs arising in the building (with or without reuse) and maintaining ontologies: PM = PMB + PMM + PMR. Where PMB, PMM and PMR represent the effort associated to building, maintaining and reusing ontologies, respectively. The partial costs are calculated in ONTOCOM using formula shown in Fig.2.

3-Refine the ONTOCOM Model: Similar to other parametric models, ONTOCOM relies on statistics based on previous project data to calibrate the model and thus create a-posteriori model which will produce better estimates. ONTOCOM follows the calibration techniques described in [7, 9], which refine the values (weights) on the ratings of the cost drivers by statistically tuning the values to reflect both the input from the experts and those of the historical data.

D. Comparison with Other Methodologies

TABLE 1. illustrates comparisons between several software development tools and methodologies that can be used to estimate project costs for ontology based software projects. The proposed tool is based on ONTOCOM method, since it is cost effective and it is widely used and can be implemented using friendly user interfaces as described in the paper.

TABLE 1. COMPARISONS BETWEEN SEVERAL ONTOLOGY BASED SOFTWARE PROJECT DEVELOPMENT APPROACHES.

Authors	Approach Used	Methodology
Khaled Hamdan[10]	Estimation of software project effort with case-based reasoning (CBR).	This system is expected to enable project managers to elicit software project factors, features and terms that are semantically equivalent to those used in a new project and for which effort and cost estimate are required.
Mei-Hui Wang[11], Chang-Shing Leea[12]	Fuzzy ontology for project planning based on Capability Maturity Model Integration (CMMI).	Combined the fuzzy inference, CMMI, and the ontology, the fuzzy agent infers the total project cost based on the predefined ontology and then stores the results to the project estimation repository. The simulation results show that the proposed method is feasible to estimate the total project cost.
Markus Schacher [13]	CASSANDRA: An Automated Software Engineering Coach.	It analyzes project information held in one of many familiar UML-based CASE tools and derives issues to be clarified, creates new models or suggests the next steps to be done in a project. It provides a very simple, human-like user interface.
Active Knowledge Powered Enterprise [14]	ONTOCOM Lite	The cost drivers is based on both the ONTOCOM and its extension for lightweight ontologies, to ensure maximum compatibility with the level of knowledge representations required by ACTIVE.
This Paper	"OCET": Ontology Cost Estimation Tool	It is a technique for reliably estimating development efforts. It is a parametric approach to cost estimation for ontology software projects development. It is based on ONTOCOM.

III. ONTOCOM METHODOLOGY

Fig. 2 shows the ONTOCOM effort estimation formula. Each of the three development ontology phases is associated with specific cost factors. The most significant one is the *Size* of the ontology involved in a project. The *Size* parameter is expressed in kilo entities of ontological primitives – (the sum of all concepts, relations, axioms and instances). The total *Size* is computed as: *Size = Size_b + Size_M + Size_r*. Where, *Size_b* corresponds to the size of the newly built ontology i.e. the number of primitives which are expected to result from the conceptualization phase. *Size_m*, in case of ontology maintenance, depends on the expected number of modified items. *Size_r*, for reuse purpose, is the size of the original source after being tailored to the present application setting. In particular this involves the parts of the source ontologies which have to be translated to the final representation language, the ones whose content has to be adapted to the target scope and the fragments directly integrated. The possibility of a non-linear behavior in effort of the model w.r.t. the size of the ontology is covered by the exponential factor *B*. Further on, start-up costs, which are not proportional to the size of a project, are intended to be counterbalanced by a baseline multiplicative constant *A* in person months. For example, for an ontology with 800 concepts, 100 relations and 50 axioms, the *Size_b* will have the value, *Size_b = (800 + 100 + 50) / 1000 = 0.95* kilo entities .

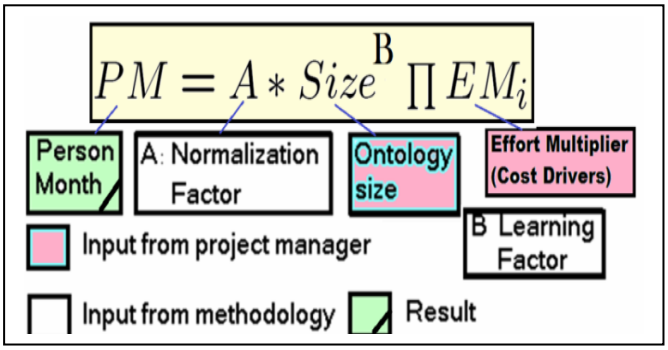

Fig. 2 shows ONTOCOM effort estimation formula, Adapted From [7].

A. ONTOCOM Cost Drivers

The core parts of the ONTOCOM-formula are the cost drivers (CD), which have a rating level that impact on the development effort (*EM*). The total amount of cost driver's equals 20. TABLES 2 and 3 show descriptions of the cost drivers. Identification of these cost drivers are estimated through literatures survey, analysis of empirical data and expert interviews. They are also subject of further calibration on the basis of the statistical analysis of real-world projects data. These cost drivers are classified into three main categories: *Product*

drivers, *Project drivers*, and *Personnel drivers*. *Product drivers* account for the influence ontology characteristics have on project costs: E.g. Complexity of the Domain Analysis (*DCPLX*), Required Reusability (*REUSE*), and Documentation Needs (*DOCU*). *Project drivers* account for the influence of project setting characteristics on the overall development which looks at the environment settings that supports or hinders progress in the engineering process. E.g. Support Tools (*TOOL*), multi-site development (*SITE*). *Personnel drivers* emphasize the role of team experience, ability and continuity w.r.t. the effort invested in the project. E.g. Ontologist / Domain Expert Experience (*DEEXP*), Language/Tool Experience (*LEXP*).

TABLE 2. DESCRIPTIONS OF THE COST DRIVERS.

No	Cost Drivers	Rating Scale and Initial Values				
		Very Low	*Low*	*Nominal*	*High*	*Very High*
1	*DCPLX* (DOMAIN Complexity)	narrow scope, common-sense knowledge, low connectivity	narrow to moderate scope, common-sense or expert knowledge, low connectivity	moderate to wide scope, common-sense or expert knowledge, moderate connectivity	moderate to wide scope, common-sense or expert knowledge, high connectivity	wide scope, expert knowledge, high connectivity
		0.70	0.85	1	1.30	1.60
	DCPLX (REQUIREMENTS complexity)	few, simple req.	small number of non-conflicting req.	moderate number of req., with few conflicts, few usability req.	high number of usability req., few conflicting req.	very high number of req. with a high conflicting degree, high number of usability req.
	DCPLX (INFORMATION SOURCES availability)	high number of sources in various forms	competency questions and text documents available	some text documents available	some unstructured information sources available	none
2	*CCPLX* (Conceptualization Complexity)	----------	The semantics of the conceptualization compatible to the one of the impl. lang.	Minor differences between the two	Major differences between the two	----------
		0.70	0.85	1	1.30	1.60
3	*ICPLX* (Implementation Complexity)	----------	The semantics conceptualization. compatible to the one of the impl. lang.	Minor differences between the two	Major differences between the two	----------
		0.85	0.85	1	1.30	1.30
4	*DATA* (Instantiation Complexity)	structured data, same repr. language	structured data with formal semantics	semi-structured data e.g. databases, XML	semi-structured data in natural language, e. g. similar web pages	unstructured data in natural language, free form
		0.80	0.90	1	1.30	1.60
5	M Increment for DATA	direct mapping	concept mapping	taxonomy mapping	relation mapping	axiom mapping
		0.2	0.4	0.6	0.8	1
6	*REUSE* (Required Reusability)	for this application	for this application type	application independent domain ontology	core ontology	upper level ontology
		0.70	0.85	1	1.15	1.30
7	*DOCU* (Documentation Needs)	many lifecycle needs uncovered	some lifecycle needs uncovered	right-sized to lifecycle needs uncovered	excessive for lifecycle needs	very excessive for lifecycle needs
		0.70	0.85	1	1.15	1.30
8	*OE* (Onto-Evaluation) *Building*	small number of tests, easily generated and reviewed	moderate number of tests	high number of tests	considerable tests, easy to moderate to generate and review	extensive testing, difficult to generate and review
		0.80	0.90	1	1.30	1.60
9	OI (Ontology Integration)	1-1 mappings, approx. 50% precision and recall required, barely overlapping, 2 ontologies	1-1 mappings, approx. 60% precision and recall required, barely overlapping, 2 ontologies	1-n mappings, approx. 70% precision and recall required, some overlapping, 2 ontologies	1-n mappings, approx. 80% precision and recall required, high overlapping, more than 2 ontologies	n-m mappings, approx. 95% precision and recall required, high overlapping, more than 2 ontologies
		0.80	0.90	1	1.30	1.60
10-1	OU (Ontology Understanding) for Complexity	complex dependency graph large domain complex representation language no concept names	taxonomic dependency graph large domain complex representation language concept names	taxonomic dependency graph ,middle domain, moderate representation language concept names	no imports middle domain simple representation language concept names	no imports small domain simple representation language concept names
10-2	OU (Ontology Understanding) for Clarity	representation language know-how, no comments in natural language, no metadata	representation language know-how no comments in natural language no metadata	representation language tool 30% comments in natural language, no metadata	representation language tool 60% comments in natural language,no metadata	representation language tool 90% comments in natural language metadata
		1.80	1.40	1	0.90	0.80

Each cost driver is assigned with five ratings from Very Low to Very High. The initial input values for the ratings of the product factors cost drivers; the so-called "a-priori cost model" are indicated in the TABLE 2. For example, a High or Very High rating for the *DCPLX* means that the domain modeled was complex and that this had a high or very high impact on the development effort. Conversely, if the domain modeled by the ontology is simple in nature the *DCPLX* rating for that ontology should be Low or Very Low. For the a-priori model each of these ratings corresponds to a numeric value i.e. a weight which is derived based on interviews with experts and is calculated as an average of their proposed values. Each rating level of each cost driver is associated to a weight for the effort multiplier (EM). The average EM assigned to a cost driver is 1.0 (nominal weight). If a rating level causes more development effort, its corresponding EM is above 1.0. If the rating level reduces the effort then the corresponding EM is less than the nominal value. For each cost driver, a decision criteria is specified in detail which are relevant when assigning the corresponding effort multipliers. For example, in the a-priori cost model a team of 3 ontology engineering experts (OEXP) assigned start values between 0.1 and 2 to the effort multipliers, depending on the contribution of the corresponding cost driver to the overall development costs.

TABLE 3. RELATED DESCRIPTIONS OF THE COST DRIVERS.

	Cost Drivers	Rating Scale and Initial Values				
		Very Low	*Low*	*Nominal*	*High*	*Very High*
10-3	OU Increment for *UNFM*	self built	team built	every day usage	occasional usage	little experience, completely unfamiliar
		0.0	0.2	0.4	0.6	0.8, 1.0 respectively
11	*OE* (Onto-Evaluation) Reuse, Maintenance	small number of tests, easily generated and reviewed	moderate number of tests	high number of tests	considerable tests, easy to moderate to generate and review	extensive testing, difficult to generate and review
		0.70	0.85	1	1.30	1.60
12	*OM* (Ontology Modification)	few, simple modifications	some, simple modifications	some, moderate modifications	considerable modifications	excessive modifications
		0.80	0.90	1	1.20	1.40
13	*OT* (Ontology Translation)	direct	low manual effort	some manual effort	some manual effort	manual effort
		0.70	0.85	1	1.30	1.60
14	*OCAP / DECAP* (Capability for the Engineering Team)	15%	35%	55%	75%	95%
		1.30	1.15	1	0.85	0.70
15-1	*OEXP* (Ontologists Experience)	2 months	6 months	1 year	1.5 years	3 years
		1.30	1.15	1	0.85	0.70
15-2	*DEEXP* (Domain Experts experience)	6 months	1 year	3 years	5 years	7 years
		1.30	1.15	1	0.85	0.70
16-1	*LEXP* (Language Experience)	2 months	6 months	1 year	3 years	6 years
		1.60	1.30	1	0.90	0.80
16-2	*TEXP* (Tool Experience)	2 months	6 months	1 year	1,5 years	3 years
		1.50	1.25	1	0.90	0.80
17	*PCON* (Personnel Continuity)	6 years	3 years	1 year	6 months	2 months
		1.30	1.15	1	0.85	0.70
18	*TOOL* (Tool Support)	High quality tool support, no manual intervention needed	Few manual processing required	Basic manual intervention needed	Some tool support	Minimal tool support, mostly manual processing
		1.60	1.30	1	0.90	0.80
19	*SITE* (Multisite Ontology Development)	mail	phone, fax	email	teleconference, occasional meetings	frequent F2F meetings
		1.30	1.15	1	0.85	0.70
20	*SCED* (Required Development Schedule)	75%	85%	100%	130%	160%
		1.30	1.15	1	0.85	0.70

For a numerical example assume: ontology with 800 concepts, 100 relations and 50 axioms. Cost drivers: *DCPLX* is high; Evaluation of the results (*OE*) has a high influence on the effort. Implementation complexity (*ICPLX*) has a low impact on the effort. Remaining cost drivers: nominal effort. Constants *A* and *B*: values 2.58 and 0.15 as resulting from the calibration. Hence; from TABLE 2. ; the cost driver's ratings are: *DCPLX = 1.26 (High), OE = 1.09 (High), ICPLX = 1.05 (Low). Hence: Size = (800 + 100 + 50) / 1000 = 0.95 kilo Entities, and PM = 2. 58 * 0.95^0.15 * (1. 26* 1. 09* 1. 05) = 3.68 PMs.*

IV. TOOL SOFTWARE DESIGN AND IMPLEMENTATION

The proposed tool is implemented of three main components, Fig. 3. The 1st component determines the life cycle which include the ontology building scenario. The 2nd component includes specification of the size of ontology to be build, expressed in thousands of ontological primitives (concepts, relations, axioms, and instances). The 3rd component includes specification of cost drivers rating for a project, corresponding to the information available. The tool is implemented using Microsoft visual studio ASP.Net C# that can be run easily on PC. When running the tool, the data of the project will be entered using the helpful labeled menus. Then the Tool developer will be able to see the results in the form of graphical chart and also in the form of tabulated information that include all project type components and project cost related to person month. This is explained in the following Figures (4-8).

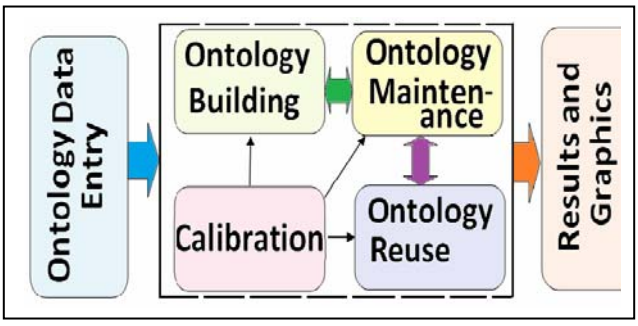

Fig. 3 proposed tool components.

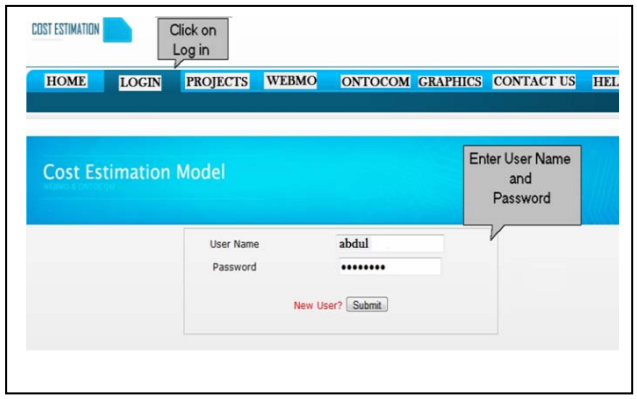

Fig.4 main screen.

V. RESULTS

Fig.4 shows tool main screen. A user can click on Login and enter name and password. Fig.5 shows a screen, where project developer can enter project name, project code, and its category. Fig. 8 shows a screen used to determine the project life cycle phases (Building, Reuse, and Maintenance). Also size primitives are entered manually or estimate. Where: Concepts represent the set of entities within a project domain. Relations specify the interaction among these concepts. Instances indicate the concrete example of concepts within the domain. Axioms are the explicit rules to constrain the use of concepts. The screen shown in Fig.6 help a user for selecting project cost drivers rating values, by clicking on required icon label. Project expert developer selects the suitable of cost drivers values related to project scope specifications. For an example, in this screen, we selected the nominal values. When clicking on the icon named GRAPHICS shown in Fig. 7, we get output graphical chart indicating projects name, code, *Size* in kilo entities and *Effort* in PM, as shown in Fig.8.

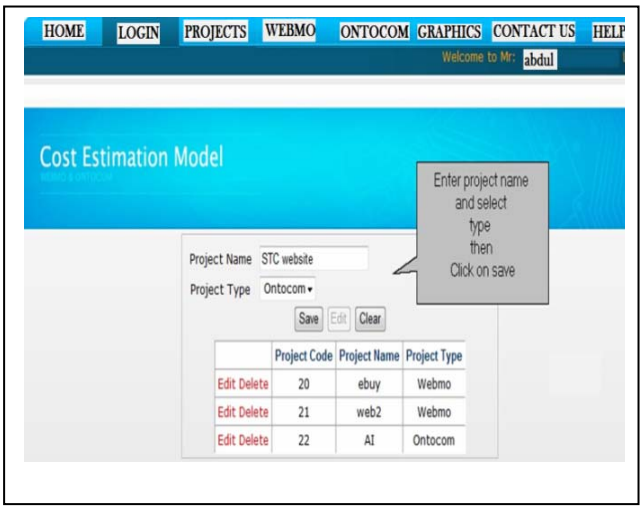

Fig.5 shows a screen, where project developer can enter project name, code, and its type.

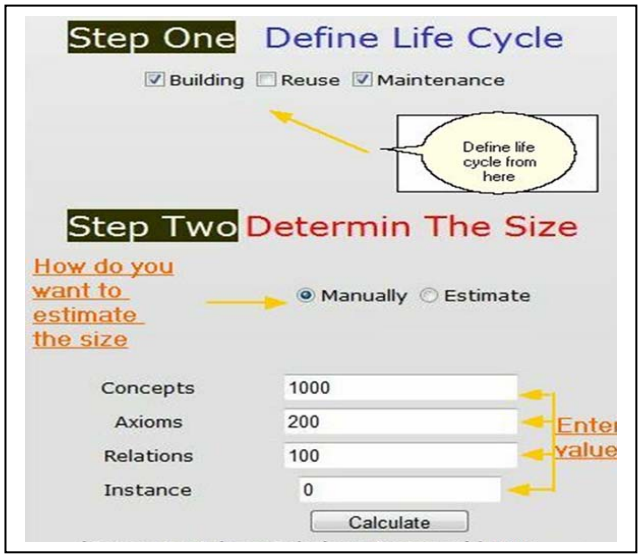

Fig.6 screen shows project life cycle, and project ontology components.

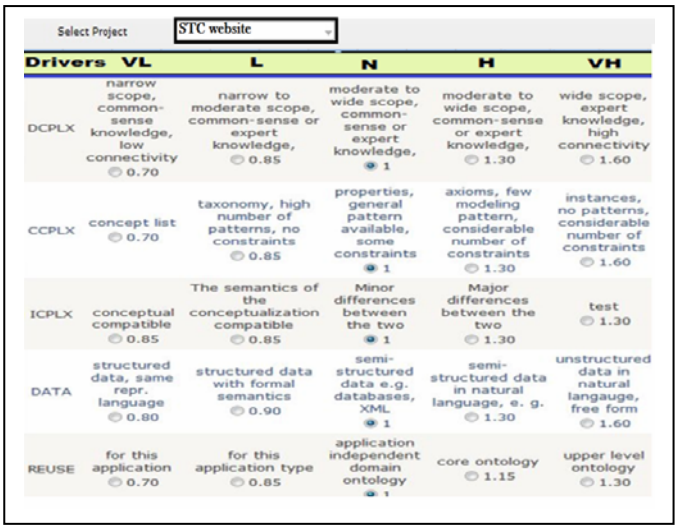

Fig.7 the screen shows types of cost driver's description and their initial rating values.

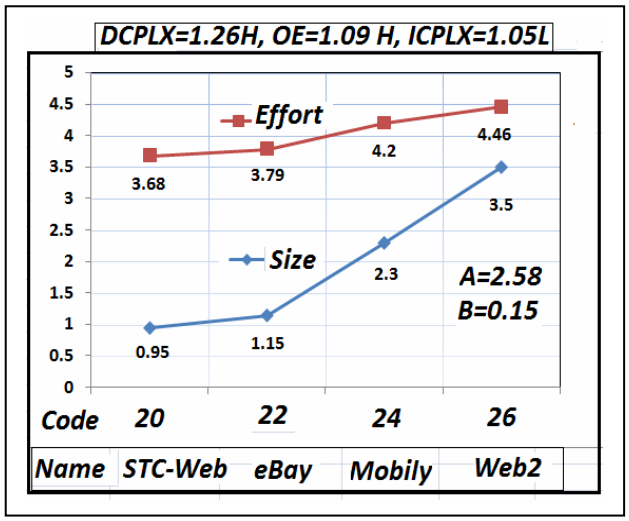

Fig.8 shows a graph chart indicating projects name, code, *Size* in kilo entities and *Effort* in PM.

CONCLUSION

The work explained in this paper, proposed an ONTOCOM based automated tool which is implemented using VS-ASP.Net C#. The tool can be used effectively by software vendors to estimate project costs that are based on ontology engineering. The method used is compared with other exit approaches. The tool proposed is cost effective and contains friendly user interfaces, where a user can access the tool through helpful easy labeled menus. The tool is tested using several previously data projects which are used successfully to calibrate the tool. Project estimated size and effort results can be obtained in graphical charts. The tool also offers several benefits to software project designers, among these are: speed, accuracy and adaptability.

ACKNOWLEDGMENT

Authors would like to thank Mr. Ahmed Alsakaf and Mr. Abdul Meged Zeban who helped in the practical experiments introduced in this work.

REFERENCES

[1] A. H.M. Ragab, etal" Cost Estimation Models for Ontology Engineering Based Projects ", Bsc Thesis Project, KAU, Jun. 2010.

[2] Z. Zia, A. Rashid, K. Zaman, " Software Cost Estimation for Component based fourth-generation-language software applications", IEEE Xplore, Vol.5, No1,PP.103-110, Feb., 2011.

[3] A.A. Issa, "An Algorithmic Software Cost Estimation Model for Early Stages of Software Development", Int. Journal of Academic Research, Vol.3 No.2. PP.336-341, March, 2011.

[4] E. Simperl, I. Popov, and T. Bürger, "ONTOCOM Revisited: Towards Accurate Cost Predictions for Ontology Development Projects" In: Proceedings of the 6th European Semantic Web Conference, pp.248-262, May 31 - June 04 2009.

[5] E. P. Bontas , M. Mochol, "Towards a Cost Estimation Model for Ontology Engineering", In Proceedings of the Berliner XML Days Conference , pp.153-160, 2005.

[6] E. P. Bontas, M. Mochol, R. Tolksdorf, "Case Studies in Ontology Reuse", In Proceedings of the 5th International Conference on Knowledge Management IKNOW05, July 2005.

[7] I. O. Popov, "A cost model for lightweight ontologies: adapting the ONTOCOM model", STI technical report. Aug. 2008.

[8] R. S. York Sure," Cost estimation in ontology engineering", institute AIFB university of Karlsruhe IST, Helisink, Nov.2006.

[9] I. O. Popov, Preliminary Data Analysis for the ONTOCOM Data Set, Research Seminar 2008/09, www.sti-innsbruck.at/fileadmin/.../ ResearchSeminarNov08_01.pdf

[10] Khaled Hamdan1, etal, "A Software Cost Ontology System for Assisting Estimation of Software Project Effort for Use with Case-Based Reasoning", http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4085457.

[11] Mei-Hui Wang, etal,"A novel fuzzy CMMI ontology and its application to project estimation"

http://www.mendeley.com/research/novel-fuzzy-cmmi-ontology-application-project-estimation/

[12] Markus Schacher, etal, "CASSANDRA: An Automated Software Engineering Coach", http://www.knowgravity.com/pdf-e/CASSANDRA Overview 2001.pdf

[13] Chang-Shing Leea, etal," Intelligent Estimation Agent Based on CMMI Ontology for Project Planning", http://ieeexplore.ieee.org/iel5/4803719/4811240/04811279.pdf?arnumber=4811279

[14] ACTIVE Deliverable 4.2.1 Use cases for cost benefit information in collaborative knowledge creation, http://ontocom.sti-innsbruck.at/.

AUTHORS PROFILE

Prof. Dr. Abdul Hamid M. Ragab

is Professor at the Dept. of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia. He is also consultant research at the admission deanship. Permanent Job is Prof. at Computer Science and Engineering Dept., Faculty of Electronic Engineering, Menoufial University, Egypt. He is a university staff member for 40 years, and published many books, and many researches. Fields of interest**:** Next Generation Computer Networks, Intelligent Systems Application, Wireless Sensor Network Applications, Adaptive E-learning Systems, Multi-levels Networks and Information Systems Security, Virtual Reality Lab Tools and Systems Applications.

***Dr.* Abdulfatah S. Mashat**

is Associate Prof and staff member at Dept. of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia. He is the dean of Faculty of Computing and Information Technology. He published many papers in the field of computer applications, and work as computer consultant. He is interested in computer applications and multimedia, and data mining applications. He is know the dean of the College.