World Applied Sciences Journal 3 (4): 559-564, 2008
ISSN 1818-4952
© IDOSI Publications, 2008

Designing a Software Tool for Evaluating Qualitative Parameters

[1]H. Motameni, [2]A. Movaghar, [3]M. Ebrahimi, [3]S. Peirovi and [4]A. Khosrozadeh Ghomi

[1]Department of Computer Engineering, Islamic Azad University, Sari Branch, Iran
[2]Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
[3]Department of Computer Engineering, University of Science and Technology of Mazandaran, Babol, Iran
[4]Department of Computer Engineering, Islamic Azad University, Amol Branch, Iran

Abstract: UML is known as one of the most common methods in software engineering. Since this language is semi-formal, many researches and effort have been performed to transform this language into formal methods including Petri nets. The quality of an architectural design of a software system has a great influence on achieving non-functional requirements to the system. Thus, the operation of verification and validation of the qualitative and nonfunctional parameters could be achieved with more ability. In this paper, a case tool named AriaPN which is presented for calculating performance parameters from Generalized Stochastic Petri Net (GSPN) to be able to analyze the stochastic behavior of the system. We discuss about them in this paper in addition to a case study.

Key words: Software engineering . UML . Generalized Stochastic Petri Net (GSPN) . Continuous Time Markov Chain (CTMC) . Non-functional parameters . markov reward models . case tool

INTRODUCTION

Nowadays, UML diagrams are extensively used in software design. However, the semi-formal characteristic of this method is a limitation for verification operations and predicting non functional parameters is more critical control, critical reactive and real time systems. Several researches have been performed to tackle with the semi formal problem of UML [1-12]. Some of these researches have only used a transformation algorithm, which transforms the created UML model into a Petri net which transforms model into a Petri net as a mathematical and formal model that, in turn, contains the visual aspect of modeling and pursues the verification operation which further ability [1-6]. Some of the researches in this field besides representing a transformation algorithm (or with out presenting an algorithm and only by using the available Algorithm) evaluates the capability of the non operational parameters and commonly qualitative parameters on the obtained nets of the UML model created [7-12]. IN our previous researches besides of studying and presenting transformational patterns for some kinds of usual UML diagrams, especially state diagrams and Activity diagrams, we presented methods for evaluating some qualitative parameters [13-17].

In this research, we present at first about evaluating qualitative parameters. Then we discuss about AriaPN to introduce its fundamentals, at last a case study is introduced which is explained by AriaPN that we designed it.

EVALUATING QUALITATIVE PARAMETERS WITH MARKOV CHAIN THEORIES

For example, a metric for comparing the security of different architectures can be gained by using the equation [17]:

$$\mathrm{Security}_{\mathrm{Net}} = \frac{\left(\dfrac{\left(\sum_{t\in T} S_t * f_t\right)}{\sum_{t\in T} f_t} + \dfrac{\left(\sum_{p\in P} S_p * T_p\right)}{\sum_{p\in P} SJ_p} \right)}{2}$$

Where S_t is the data security factor associated to the transition t, f_t is the firing rate of t, S_p is the data security factor associated to the place p, T_p is the expected time in which there is a token in place p. This is similar to the authors' previous work using simulation. Identically the reliability can be computed, but because the reliability is usually related to the processes of system, the reliability factor is just usually associated to the transitions than the places [17]:

$$\mathrm{Reliability}_{\mathrm{Net}} = \frac{\left(\sum_{t\in T} RL_t * f_t\right)}{\sum_{t\in T} f_t}$$

Where RL_t, stands for the reliability of process t.

Corresponding Author: Dr. H. Motameni, Department of Computer Engineering, Islamic Azad University, Sari Branch, Iran

We can compute some other parameters like those computed above for example we can gain a metric for comparing the availability of different architectures but because availability is usually related to the places of the system. Thus, the availability factor is usually associated to the places than transitions. We can gain it by the equation [13]:

$$\text{Avalibility}_{\text{Net}} = \frac{(\sum_{p\in P} a_p * T_p)}{\sum_{p\in P} SJ_p}$$

Which A_p is the availability associated to the transition t, t_p is the expected time in which there is a token in place P.

This is similar to our previous work using simulation. Another parameter that we can gain a metric is performance efficiency because performance efficiency is usually related to the transitions of the system it is associated to the transitions than places and we can gain it by the equation [13]:

$$\text{PerformanceEfficiency}_{\text{Net}} = \frac{(\sum_{t\in T} P_t * f_t)}{\sum_{t\in T} f_t}$$

Where P_t is the performance efficiency that associated to the transition t and f_t is the firing rate of transition t.

TOOL

We have designed a case tool, AriaPN, which is implemented by java programming language, so there will be the possibility of GSPN drawing, edition and calculating of qualitative parameters related to drawing GSPN in it. One of the most important and significant characteristics of AriaPN is the platform independency which provides causes of its execution and using in different hardware and operation systems [18-25].

For designing AriaPN, we have used Modeling-View-Control pattern or MVC summary. MVC is a pattern which has been encouraged in designing GUI programs and in object-oriented languages very much. One MVC program in java is made based on swing and listener elements. When a GUI program works with an object, it can show it using a model. Here, a model means a complete representation of the used object by the use of program. This model can be a graphic picture. Program includes some views on models. Each view of the model has its own method in contact with it but all of them relate to general model and ultimately, a view which relates to user interface performs it as object controller. For instance, this object may be a bottom or menu

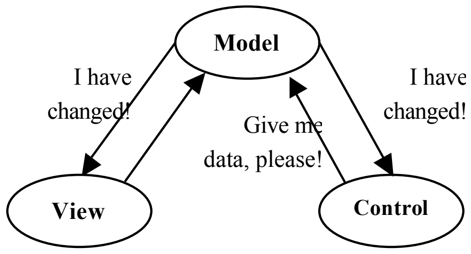

Fig. 1: Model-view control

When a controller receives an order from the user, it uses suitable information from a definite point of view for model adjustment. Everywhere the model changes, all views are informed and will update it. This is considered as an excellent pattern (Fig. 1). Reasonably every view is followed by a controller. For example there will be only a model class but different views by their control classes relate to the model. Controllers will respond to done measures from the user and where is necessary, it takes it needed information and sends a message to the model to create some changes in itself. In the part of case tool designing, for each part of GSPN component, a separate class has been considered-one class for place, one class for transition, one class for arc and one class for inhibitor-and in fact every place or transition which is designed in the program, is kept in one vector from the same class type. This program has been designed automatically graph available from GSPN and has obtained derived CTMC of it and by the use of Markov theories and noted relations in the previous part, it calculates the related amounts to qualitative parameters

Familiarity with major used parts of user: AriaPN has three main parts for edition of a GSPN and calculation of qualitative parameters related to it: GSPN editor part for drawing GSPN, a property panel for information edition related to the existed components in the designed GSPN and result panels for representation of the resulted messages from compile and the values of qualitative parameters which are calculated by AriaPN

The method of working with case tool, AriaPN: In AriaPN, there is the possibility of drawing new GSPN, saving the designed GSPN in the program and also retrieval saved files by this program. In this program, component of GSPN including inhibitor, arc, immediate transition, timed transition, place can be easily designed by the user and show the values and primary information needed for each of the components

Also, after drawing GSPN and giving suitable values for information needed for all objects existing in the designed GSPN, there is the possibility of error-finding in the given GSPN which this error-finding is conducted automatically by the program and the error cases are declared to the user if existing and in the part of GSPN editor also, parts in which error has been occurred are distinguished automatically from other parts

The most important possibility of AriaPN which in fact is the designing goal of AriaPN is performing the designed GSPN in the program and calculating the qualitative parameters by the use of Continuous Time Markov Chain (CTMC) derived from it and the related theories. After the required processing bye the program, the result of parameters, evaluations are represented in parameters part

Storage and retrieval project in AriaPN: Files caused by AriaPN having DAT suffix and in the form of saved object and just by the same program are renewable

For renewing of the saved files, the program reads the file in the form of object and puts each object with regard to its class type in the vector of that class until it is used and represented in the program

CASE STUDY

This case study proposes a selection framework of multiple navigation primitives for a service robot using *Generalized Stochastic Petri Nets* (GSPN's). A guide robot *'Jinny'* was developed by using a Petri net (PN) based control architecture, which was designed for multifunctional service robots.

Through their experiences they concluded it is important for the robot adaptively to select its navigation primitives according to the conditions of environments. For example, in general cases, it is advisable that the robot uses a map-based navigation [20].

In general, navigation task is accomplished by the cooperation of several components such as a localizer and a path planner. As the related components and navigation primitives increase, it becomes troublesome to manage the relationships between them. A major scope of this paper is to propose a selection framework of multiple navigation primitives for a service robot.

In this approach, modeling, analysis and performance evaluation are carried out based on the *Generalized Stochastic Petri Nets* (GSPN's). Owing to the formalism, the strategy has following three major advantages. First, the framework is developed on firm mathematical foundation. This advantage makes it possible to set up state equations and other mathematical models governing the behaviors of a system. Second, the method supports modular and incremental designs of navigation framework since GSPN's have powerful modeling ability. It can model concurrency, asynchronous events, logical priority relations and structural interactions. Also, the transformation from GSPN model to the mathematical representation can be automated by several free or commercialized tools. Third, as a graphical tool, GSPN's can represent both static and dynamic aspects of a system.

In this case study, It is considered two types of navigation primitives, *AutoMove* and *Contour tracking*. The detailed description of these motions is summarized in Table 1.

From this observation, one rule is made for the primitive selection. It is that if the localizer falls into the *Warning* state, *Contour tracking* is unconditionally selected. The criterion of this selection problem is "which primitive leads the robot to a goal faster than the other with guaranteeing localization safety."

The modeling method goes through following procedure. First, based on a given system description, navigation primitives and required components are identified. Primitives are designed as places and the changes between them are modeled as transitions. Each component is represented as an independent GSPN's model. You can see the resultant GSPN model which is drown in our AriaPN environment in Fig. 2.

Table 2 describes the physical meaning of places and transitions of the model. The GSPN model has six places, seven timed transitions (drawn as white bars) and three immediate transitions (drawn as black bars).

The initial marking is M_0= (1, 0, 0, 0, 0, 1, 0, 1, 0), which is denoted as $P_0P_5P_7$ in the reachability graph in Fig. 3 by specifying the places having tokens.

Table 1: Description of two navigation

Type	AutoMove	Contour tracking
Algorithm	Shortest path planning with obstacle avoidance	A (left, right, center) wall-following technique using only laser scan data
Merits	Optimality (shortest path to anypoint on the maps) Generally (applicable in any situations)	Reactive Rise localization ralability Less affected by localization accuracy
Desirable environment	Generally applicable, but the perdormance drops in a nerrow or crowded region	An area where there are amny static feature like salls or exhibits

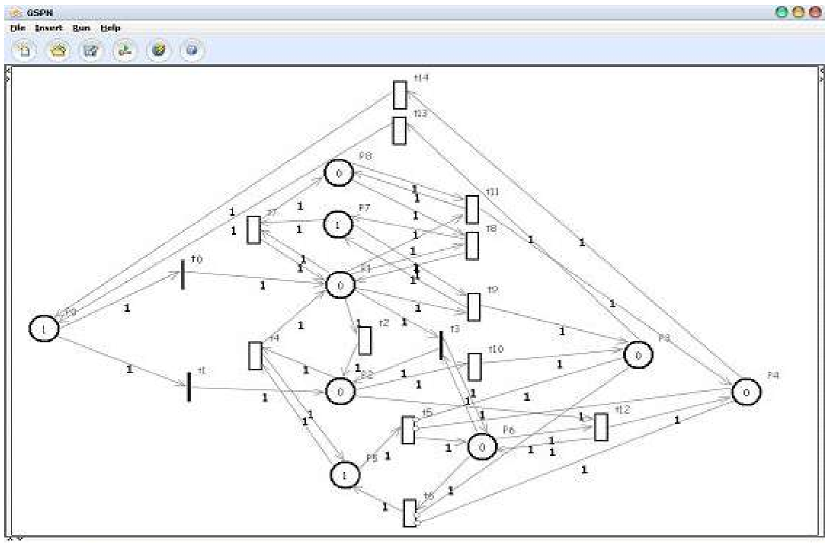

Fig. 2:

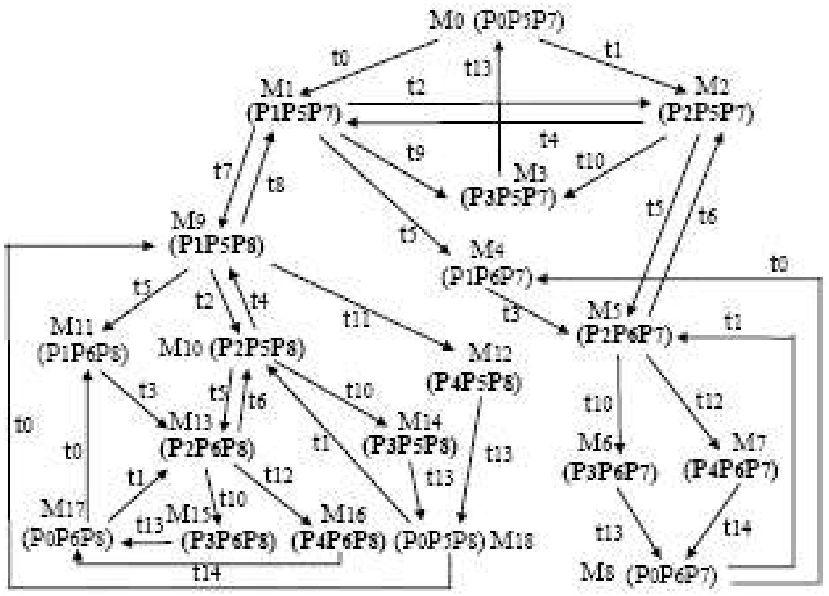

Fig. 3: The reachability graph of the system

Table 2: Description of the transition

Transition	Description	Firing rate
t0	Start AutoMove (Prob. p)	-
t1	Start contour tracking (prob. 1-p)	-
t2 (t4)	Convert to contour tracking (AutoMove) due to performance estimation	λ1 (λ2)
t3	convert to contour tracking due to localization warming	-
t5 (t6)	localization warming (Success) event fired	λ3 (λ4)
t7 (t8)	Path planner normal (Abnormal) event fired	λ5 (λ6)
t9 (t10)	AutoMove (Contour tracking) completed	λ7 (λ8)
t11 (t12)	task failed to due to no path to the goal (failure of contour tacking)	λ9 (λ10)
t13, t14	initialization	λ11

The localizer has two internal states, *Success* and *Warning*. In the initial marking, a token is assigned to P_3, i.e., it is assumed that the localizer initially knows its position. The *Warning* event t_5 fires when the localizer fails in estimating robot's accurate position for several steps. Two navigation primitives, *AutoMove* and *Contour tracking*, are modeled as P_1, P_2, respectively. Initially, the robot selects its motion by a random switch comprising the transitions t_0 and t_1 with corresponding probabilities p and 1-p, respectively. The transition between them takes place according to the change of localizer states. The immediate transition t_3 means that the robot takes *Contour tracking* as soon as the localizer *Warning* event fires. The other transition between two primitives, t_2 and t_4, are modeled as timed

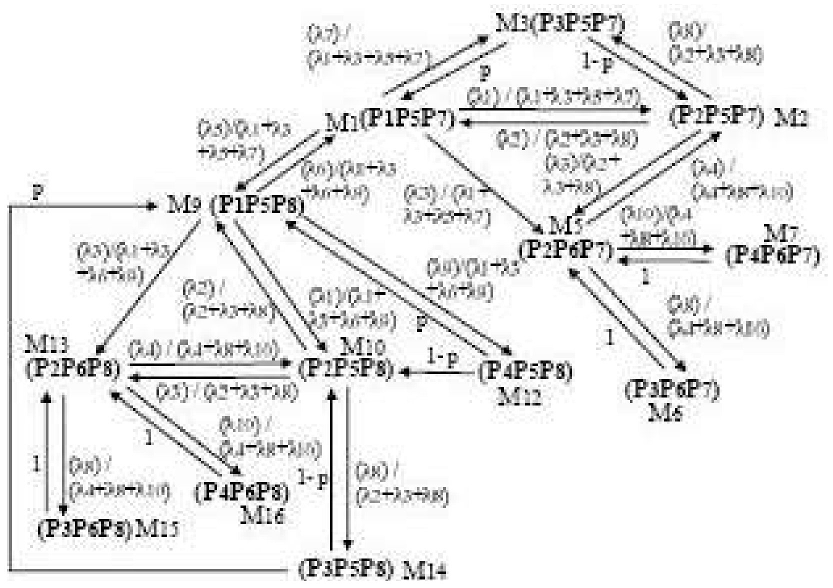

Fig. 4: Reduced embedded Markov chain

Table 3: Description of the places

Place	Description
P0	Navigation available
P1 (P2)	Running AutoMove (Contour tracking)
P3 (P4)	Compilation success (Failure)
P5 (P6)	Localization success (Warning)
P7 (P8)	State: path planner Normal (Abnormal)

transitions in order to express that the robot can change its current navigation primitive during the localizer *Success* state, if necessary. One of the most important modeling issues is how to set the firing rates $\Lambda=\{\lambda_1,\ldots,\lambda_7\}$. In order to perform the evaluation of GSPN designs, it is necessary to obtain an embedded Markov chain (EMC).

Figure 4 shows the EMC induced from the reachability graph of Fig. 3, which is derived from GSPN model of Fig. 2. And f_t, P_t, RL_t, S_t, A_p, t_p, S_p are as follow:

f_t = f_{t0} … $f_{t14}\}$ = {0, 0, 0.013, 0,800, 0.011, 800, 0.019, 789, 0.0062, 0.0053, 0.001, 0.001, 900, 900}

P_t = $\{P_{t0}$ … $P_{t14}\}$ = {0.019, 0.058, 0.512, 0.323, 0.512, 0.032, 0.432, 0.212, 0.56, 0.003, 0.009, 0.232, 0.221, 0.531, 0.531}

RL_t = $\{RL_{t0}$ … $RL_{t14}\}$ = {0.487, 0.47, 0.602, 0.535, 0.441, 0.382, 0.423, 0.593, 0.409, 0.325, 0.352, 0.292, 0.432, 0.573, 0.573}

S_t = $\{S_{t0}$ … $S_{t14}\}$ = {0.503, 0.508, 0.723, 0.603, 0.516, 0.512, 0.523, 0.738, 0.601, 0.473, 0.427, 0.471, 0.582, 0.529, 0.529}

A_p = $\{A_{p0}$ … $A_{p8}\}$ = {0.198, 0.553, 0.57, 0.482, 0.089, 0.432, 0.301, 0.742, 0.462}

T_p = $\{T_{p0}$ … $T_{p8}\}$ = {0.057, 0.193, 0.215, 0.101, 0.003, 0.112, 0.062, 0.352, 0.178}

S_p = $\{S_{p0}$ … $S_{p8}\}$ = {0.331, 0.573, 0.527, 0.631, 0.216, 0.443, 0.405, 0.613, 0.302}

Using Equations in section 2, it is obtained: Availability = 0.0045018, Performance Efficiency = 0.5500028, Security = 0.1776238 and Reliability = 0.3999999. This is shown in Fig. 5.

CONCLUSION

In this paper, we presented a case tool named AriaPN for evaluating non-functional parameters. Using the amounts of the qualitative parameters related to a UML scheme which is calculated by this case tool, we can asses the amount of suitability and efficiency of our desired UML model and we can examine it. Also, this case tool can be a suitable tool in order to education, investigations and engineering works in respect of designing the model. The future working field is about the AriaPN development in such a way that it receives UML scheme from the user and performs automatically the conversion of UML into GSPN and the remained works related to parameters calculation which have been explained in this paper [25-28]

REFERENCES

1. Faul M.B., 2004. Verifiable Modeling Techniques Using a Colored Petri Net Graphical Language. Technology Review Journal, spring/summer.

2. Shin, M., A. Levis and L. Wagenhals, 2003. Transformation of UML-Based System Model into CPN Model for Validating System Behavior. In Proc. of Compositional Verification of UML Models, Workshop of the UML'03 Conference, California USA, Oct. 21
3. Bernardi, S. S. Donatelli and J. Merseguer, 2002. From UML Sequence Diagrams and Statecharts to Analysable Petri Net Models. ACM Proc. Int'l Workshop Software and Performance, pp: 35-45.
4. Eshuis, R., 2002. Semantics and Verification of UML Activity Diagrams for Workflow Modelling. Ph.D Thesis, University of Twente.
5. Pettit, R.G. and H. Gomaa, 2002. Validation of dynamic behavior in UML using colored Petri nets' UML. (2000 , Zaragoza, Spain, pp: 295-302.
6. Saldhana, J. and S.m. Shatz, 2000. UML Diagrams to Object Petri Net Models: An Approach for Modeling and Analysis. Proc. of the Int. Conf. on Software Eng. and Knowledge Eng. (SEKE), Chicago10-103.
7. Elkoutbi, M. and Rodulf K. Keller, 1998. Modeling Interactive Systems with Hierarchical Colored Petri Nets. 1998 Advanced Simulation Technologies Conf., Boston, MA, pp: 432-437.
8. Bernardinello, L. and F. De Cindio, 1992. A Survey of Basic Net Models and Modular Net Classes. LNCS, Springer-Verlag, 609: 609.
9. Balsamo, S. *et al*., 2004. Model-Based Performance Prediction in Software Development: A Survey. IEEE Transactions on Software Engineering, 30 (5): 295.
10. Merseguer, J., J.P. L´opezGrao and J. Campos, 2004. From UML Activity Diagrams To Stochastic Petri Nets:Application To Software Performance Engineering. ACM, WOSP 04 January 1416, 2004.
11. Fukuzawa, K. *et al*., 2002. Evaluating Software Architecture by Colored Petri Net. Dept. of Computer Sience, Tokyo Institute of Technology Ookayama 2-12-1, Meguro, UK, Tokyo 152-8552 Japan 2002
12. Merseguer, J., S. Bernardi, J. Campos and S. Donatelli, 2002. A Compositional Semantics for UML State Machines Aimed at Performance Evaluation. Silva, M., A. Giua and J.M Colom (Eds.). Proc. of the th Intl. Workshop on Discrete Event Systems) WODES'02), Zaragoza, Spain, pp: 295-302.
13. Motameni, H *et al*., 2006. Mapping State Diagram to Petri Net: An Approach To use markov Theory For Analyzing non-Functional Parameters. IEEE, International Conferenceon Computer, Information and System Science, December 4_14 2006, University of Bridgport, USA (presented).
14. Motameni, H. *et al*., 2006. Using Markov Theory For Deriving Non-Functional Parameters On Transformed Petri Net From Activity Diagram. Proc of software engineering conference (Russia), 16-17 November 2006, Moscow, Russi, (Presented).
15. Motameni, H., M. Zandakbari and Movaghar, 2006. Deriving performance parameters from the activity diagram using gspn and markov chain. ICCSA 2006 Proceeding of 4th International Conference on Computer Science and Its Aapplications, San Ddiego,California.
16. Motameni, H. *et al*., Evaluating UML State Diagrams Using Colored Petri Net" SYNASC'05.
17. Motameni, H. *et al*., 2005. Verifying and Evaluating UML Activity Diagram by Converting to CPN. Proc. of SYNASC'05, Romania, Sep 2005, (presented).
18. Object Management Group, UMLTM Profile for Schedulability, Performance and Time Specification, OMG Document, Version 1.1, January 2005.
19. Rumbuaugh, J., M. Blaha, W. Premerlani, F. Eddy and W. Lorensen, 1991. Object-Oreinted Modeling and Design. Prentice hall, Englewood Cliffs, NJ, USA.
20. Wang, lu, 2005. Fuzzy UML. Seminararbeit, Sommersemester.
21. Zongmin, Ma., 2005. Fuzzy Information Modeling With the UML. Idea.
22. Ma, Z.M., 2004. Extending UML For Fuzzy Information Modeling In Object_Oriented Database. Theories and Practices, Idea Group Publishing.
23. Murata, T., 1989. Petri Nets: Properties, Analysis and Applications. Proceedings of IEEE, 77: 540-541.
24. Bernardinello, L., F. De Cindio, (Ordinary) Petri Nets (PN), <http://www.daimi.au.dk/PetriNets/classification>(accessed July 2005).
25. Jensen, K., 2005. Colored Petri nets (CPN), http://www.daimi.au.dk/PetriNets/classification/level3/CPN.html> (accessed July 2005).
26. Burcin Bostan-Korpeoglu and Adnan Yazici, 2006. A Fuzzy Petri Net Model For Intelligent Database, Data and Knowledge Engineering (2006), Elsevier.
27. Nihal, Y.Ö., 2007. On the Numbers of the Form n = x2 + Ny2, World Applied Sciences Journal, 2(1): 45-48.
28. Erçetin, S.S., B. Çetin and N. Potas, 2007. Multi-Dimensional Organizational Intelligence Scale (Muldimorins), World Applied Sciences Journal, 2(3): 151-157.