Predicting UML Statechart Diagrams Understandability Using Fuzzy Logic-Based Techniques

José A. Cruz-Lemus[1], Marcela Genero[1], José A. Olivas[2], Francisco P. Romero[2] and Mario Piattini[1]
[1]*ALARCOS Research Group,* [2]*ORETO Research Group*
Department of Computer Science,University of Castilla-La Mancha
Paseo de la Universidad, 4, 13071, Ciudad Real (Spain)
{JoseAntonio.Cruz, Marcela.Genero, JoseAngel.Olivas, Mario.Piattini}@uclm.es
fpromero@soluziona.com

Abstract. In this work, we present an application of the Fuzzy Logic in the field of prediction in Software Engineering. We specifically use the Fuzzy Prototypical Knowledge Discovery for characterizing the UML statechart diagrams according to their understandability, starting from the structural complexity and size of the diagrams, expressed by means of metrics, and the Fuzzy Deformable Prototypes, to obtain a prediction model of the understandability time of the UML statechart diagrams. The obtained model, built from data obtained through experimentation, is valid –in a certain way- since the 75% of the estimated values are at least 70% accurate, although it is necessary further validation with data obtained from real projects.

1. Introduction

It is well-known in Software Engineering that the quality characteristics of object-oriented (OO) systems, such as maintainability, must be guaranteed from the initial stages of their lifecycle, focusing on the models obtained in these stages. In the recent years, this fact has been emphasized given the great growth that the Model-Driven Development [1] and the Model-Driven Architecture [19] have experimented. In the OO development, some diagrams are done to cover static (class diagrams) and dynamic (use case diagrams, statechart diagrams...) aspects. For evaluating the quality of these diagrams in an objective way, it is necessary to rely on quantitative measures that avoid bias in the evaluation process.

There are several works published about quality measurement of UML class diagrams and use case diagrams [15]. However, there are only a few bibliographical references about metrics for behavioural diagrams, such as statechart diagrams, sequence diagrams or activity diagrams. Brito e Abreu et al. [9] and Poels and Dedene [24] pointed out that the definition of metrics for diagrams that capture dynamic aspects of OO systems is a relevant are for further research, but it has been disregarded in the software measurement field. This fact motivated us to define metrics for UML behavioural diagrams, starting with statechart diagrams [20] (see Table 1).

Table 1. Metrics for UML statechart diagrams

	Metric Name	Metric Definition
Size	**NEntryA**	The total number of entry actions, i.e., the actions performed each time a state is entered.
	NExitA	The total number of exit actions, i.e., the actions performed each time a state is left.
	NA	The total number of activities (do/activity) in the statechart diagram.
	NSS	The total number of states considering also the simple states within the composite states.
	NCS	The total number of composite states, i.e., the states with nested sub-states.
	NE	The total number of events.
	NG	The total numbers of guard conditions.
Structural Complexity	**NT**	The total number of transitions, considering common transitions (the source and the target states are different), the initial and final transitions, self-transitions (the source and the target states are the same) and internal transitions (transitions inside a state that respond to an event but without leaving the state).

	Metric Name	Metric Definition
	CC (McCabe's [18] Cyclomatic Complexity)[1]	Defined as |NSS-NT|+2

Our approach about how the structural complexity and the size as internal attributes of statechart diagrams are potentially related with their understandability emerged from similar works [8][16] done in the field of Empirical Software Engineering, in which these properties were showed to be some of the greatest determinants of external quality characteristics, such as understandability and maintainability.

The metrics presented in Table 1 were theoretically validated using the Briand et al.'s [6] property-based framework, obtaining that the metrics NEntryA, NExitA, NA, NSS, NCS, NE and NG are size metrics, while NT and CC are complexity metrics.

As it is well-known in the software measurement field, if we want metrics that measure internal attributes (size, complexity...) to be useful, it is necessary that they can be used to predict some external attribute of quality, such as understandability or maintainability.

By means of a controlled experiment and its replication [15] that will be detailed in section 2, we have found out that the proposed metrics NA, NSS, NG and NT, seem to be strongly correlated with the understandability time the UML statechart diagrams. This led us to think about the construction of a prediction model of the understandability time of UML statechart diagrams, based on the values of these metrics. For the construction of the prediction model, we have used all the metrics, since we considered too premature to discard some of them.

Considering the encouraging results previously obtained by applying the Fuzzy-Prototypical Knowledge Discovery (FPKD) process and the Fuzzy Deformable Prototypes for the construction of prediction models applied to different domains [22][13][14], we decided to use them for our purpose. For the shake of brevity we will not explain in-depth all the steps of the prediction process, although further details about them can be found in [21].

In our case, the main goals of the prediction process are: firstly, the automatic search and extraction of fuzzy prototypes to characterize the UML statechart diagrams across their understandability, expressed as the structural complexity and size of the diagrams. This will be done using the Fuzzy Prototypical Knowledge Discovery (FPKD) process; and secondly, the obtainance of a prediction model of the understandability time of the UML statechart diagrams, by means of the deformation of the previously discovered fuzzy prototypes.

The FPKD is an extension of the classic KDD [10] process that presents as novelties the incorporation of knowledge in different points by means of the user or the expert decisions and a result prepared to generate some conceptual prototypes called Fuzzy Deformable Prototypes, based on the idea of Fuzzy Prototypical Categories [21][27]. The use of fuzzy logic let us get these results in a more understandable and useful way for their later use in the prediction process. We can evaluate new situations from such prototypes, establish predictions for real situations and also make decisions from these predictions. Some other techniques, such as fuzzy clustering and aggregation functions [10], are also used, making easier the generation of structured, significant and easily updatable models.

This work is organized as follows: in section 2, the controlled experiment and its replication are presented. In section 3 we describe the different steps followed until getting the prediction model, which consist of the FPKD process, the proper prediction process and the validation of the prediction model. Finally, in section 4 we present the main conclusions and the future research lines emerged from this work.

2. Description of the data sources

In this section we will briefly describe a controlled experiment and its replication. They were carried out taking into account some suggestions provided by experts in Empirical Software Engineering [7][17][23][26]. Futher details of the exepriment and its repliaction can be found in [20].

2.1. First experiment

Using the GQM [2] template for goal definition, the goal of the experiment is detailed in Table 2.

[1]Even tough the Cyclomatic Number of McCabe was defined to calculate single module complexity and entire system complexity, we tailored it for measuring the structural complexity of UML statechart diagrams.

Table 2. Goal of the experiment.

Analyze	*Structural complexity and size metrics for UML statechart diagrams*
For the purpose of	*Evaluating*
With respect to	*the capability of being used as indicators of the understandability of UML statechart diagrams*
From the point of view of	*researchers*
In the context of	*Undergraduate students of Computer Science and Software Engineering teachers of the Computer Science Department at the University of Castilla-La Mancha*

The experiment consisted of 20 UML statechart diagrams related to different universes of discourse but easy enough to be understood by each of the subjects (see an example in Appendix A). Each diagram had a test enclosed, which included a questionnaire in order to evaluate whether the subjects had really understood the content of the UML statechart diagrams. Each questionnaire contained four questions, each of these conceptually similar and written in the same order. Each subject had to write down the time he/she started and finished answering the questionnaire. The difference between these two values, expressed in seconds, is what we called 'understandability time'. The subjects were given the material and they have to solve tests alone. We allowed them one week to return the experiment solved.

2.2. Experiment replication

The main differences between the experiment and its replication are:

- The subjects were twenty students enrolled in the third-year of Computer Science. Therefore, the subjects experience was lower than in the first experiment.
- They had to complete the tests alone and in no more than two hours. Any doubt could be solved by the person that coordinated the experiment, what contributed to control the plagiarism.

3. Building a prediction model for the understandability time of UML statechart diagrams

In order to build the prediction model, we carried out two main processes. First, a FPKD process, which consists of several steps: data transformation; obtainance of the prototypes using clustering techniques; parametric definition of the prototypes; fuzzy representation of the prototypes (using the data obteined in the fisrt experiment). Then, we carried out a Prediction process, which consists of the 'deformation' of the fuzzy prototypes for predicting the understandability time of UML statechart diagrams and the Validation process (using the data of the replication).

Next, we will describe how we carried out each of these steps.

3.1. Data transformation

Firstly, it was necessary to transform the data so that they were valid for the FPKD process. On one hand, we obtained the table with the metric values for statechart. On the other hand, we obtained the understandability time for each diagram and subject. From these times, we obtained the minimum (MinUT), average (AvgUT) and maximum (MaxUT) understandability time of each diagram (see Table 3).

Table 3. Time obtained (in seconds) in the transformation process.

Diagram	AvgUT	MinUT	MaxUT	Diagram	AvgUT	MinUT	MaxUT
1	110.00	15	420	11	153.16	85	360
2	95.00	30	170	12	86.37	50	180
3	191.94	61	360	13	88.05	35	300
4	163.39	69	405	14	136.05	44	360
5	129.50	30	215	15	152.22	85	420
6	124.56	58	310	16	140.05	50	300
7	154.05	72	300	17	108.63	59	195
8	140.00	50	360	18	154.89	65	265
9	131.79	70	300	19	84.26	40	180
10	85.21	50	180	20	85.84	42	140

3.2. Obtainance of the prototypes using clustering techniques

With the aim of detecting the relationships between the UML statechart diagrams to be able later to ascertain whether they have a low, medium or high

understandability time, we will carry out a hierarchical clustering process, in the way of *Repertory Grids's* technique [3].

The diagrams were grouped in three prototypes according to the values of the metrics that reflect their structural complexity and size (see Table 4).

3.3. Parametric definition of the prototypes

Considering the data prototypes found in the previous section and their values of the understandability time shown in Table 3, we obtained the parametric definition of the prototypes, as Table 5 shows.

3.4. Fuzzy representation of the prototypes

The three prototypes were represented as 'fuzzy numbers', which would allow us to obtain a degree of membership (between 0 and 1) of a new statechart diagram with each of the prototypes. To use triangular fuzzy numbers it is only necessary to know their centre and the size of the base of the triangle (named *Centre, a* and *b* in Table 6).

Table 4. Diagrams grouped in prototypes

Prototypes	Diagrams
Low Understandability Time	10,13,19
Medium Understandability Time	1,2,4,5,8,9,12,14,16
High Understandability Time	3,6,7,11,15,17,18,20

Table 5. Parametric definition of the prototypes

H: High Underst. Time		M: Medium Underst. Time		L: Low Underst. Time	
Avergae	2 min. 15 sec.	Average	2 min. 5 sec.	Average	1 min. 25 sec.
Maximum	7 min.	Maximum	7 min.	Maximum	6 min.
Minimum	42 sec.	Minimum	15 sec.	Minimum	35 sec.

Table 6. Fuzzy definition of the prototypes

Prototypes	Diagrams	a	Centre	B
Low Understandability Time	10,13,19	0	0.08	0.74
Medium Understandability Time	1,2,4,5,8,9,12,14,16	0	0.26	0.92
High Understandability Time	3,6,7,11,15,17,18,20	0	0.34	1

The formal definition of the prototypes as fuzzy numbers is obtained by means of a normalization process, carried out in the following way $x_n^{'} = \frac{x_n - x_{\min}}{x_{\max} - x_{\min}}$, and the aggregation by means of average of the data corresponding to the metric values.

3.5. Deformation of the fuzzy prototypes to predict the understandability time of UML statechart diagrams

In this section we will show how to predict the understandability time for a new statechart diagram. We use as example the diagram 16 used in the experiment (showed in Appendix A)

The process is as follows:

1. Normalization of the values measured by means of the indexes of normalization associated with the obtained prediction model. The same formula is used as in the definition of the fuzzy numbers and with the same coefficients of minimum and maximum. In this way we obtained the values shown in Table 7.
2. Calculate the average of the previously normalized values (this value is called *X*). *X*=0.53.
3. From *X*, we obtain the degrees of membership to the prototypes represented by means of the fuzzy numbers as follows:

$$X > centre_{pi} \Rightarrow m_{pi} = \frac{X - a_{pi}}{centre_{pi} - a_{pi}}$$

$$X >= centre_{pi} \Rightarrow m_{pi} = \frac{c_{pi} - X}{c_{pi} - centre_{pi}}$$

The results for the diagram 16 are shown in Table 8.

4. To obtain the predicted value of the understandability time for a new statechart diagram, the fuzzy prototypes are 'deformed' to consider the affinity degree with all the prototypes. Applying the concept of Fuzzy Deformable Prototypes defined in [21], the characterization of the proposed new statechart diagram can be described by the following linear combination:

$$C_{real}(w_1 ... w_n) = \left| \sum mp_i (v_1 ... v_n) \right|$$

Where:

C_{real} Real case proposed.

$(w_1 ... w_n)$ Parameters that describe the real case proposed.

μ_{pi} Degree of membership with the non-zero Fuzzy Deformable Prototypes.

$(v_1 ... v_n)$ Parameters of these Fuzzy Deformable Prototypes.

For the diagram 16 the predicted value is shown in Table 9.

The result of applying the prototype deformation to every diagram is shown in Table 10.

Table 7. Metric values for the diagram 16.

	NEntryA	NExitA	NA	NSS	NCS	NT	NE	NG	CC
Values	0	0	5	9	0	21	22	1	16
Normalized	0	0	1	0.7	0	1	0.95	0.3	1

Table 8. Value of the affinities of the diagram 16 with the prototypes.

Prototypes	Affinities
Low Understandability Time	0
Medium Understandability Time	0.591
High Understandability Time	0.712

Table 9. Predicted value for the diagram 16[2].

Average	**0.591 / 2**	2 min. 5 sec.	+	**0.712**	2 min. 15 sec.	=	2 min. 2 sec.
Maximum		7 min.			7 min.		7 min. 5 sec.
Minimum		15 sec.			42 sec.		34 sec.

Table 10. Predicted values for each UML statechart diagrams.

DIAGRAM	X	Aff(B)	Aff (M)	Aff (A)	Estimated value	Real value	MRE
1	0.15	0.894	0.577	0.441	116.38	110.00	0.058
2	0.14	0.909	0.538	0.412	114.925	95.00	0.210
3	0.18	0.848	0.692	0.529	120.52	191.94	0.372
4	0.16	0.879	0.615	0.471	117.765	163.39	0.279
5	0.24	0.758	0.923	0.706	162.75	129.50	0.257
6	0.41	0.5	0.773	0.894	156.41	124.56	0.256
7	0.23	0.773	0.885	0.676	158.9375	154.05	0.032
8	0.34	0.606	0.879	1	177.875	140.00	0.271
9	0.23	0.773	0.885	0.676	158.9375	131.79	0.206
10	0.11	0.955	0.423	0.324	110.785	85.21	0.300
11	0.34	0.606	0.879	1	177.875	153.16	0.161
12	0.12	0.939	0.462	0.353	112.155	86.37	0.299
13	0.06	0.75	0.231	0.176	79.92	88.05	0.092
14	0.1	0.97	0.385	0.294	109.4	136.05	0.196
15	0.39	0.53	0.803	0.924	162.485	152.22	0.067
16	0.53	0.318	0.591	0.712	122.205	140.05	0.127
17	0.18	0.848	0.692	0.529	120.52	108.63	0.109
18	0.51	0.348	0.621	0.742	125.555	154.89	0.189
19	0.05	0.625	0.192	0.147	66.565	84.26	0.210
20	0.16	0.879	0.615	0.471	117.765	85.84	0.372

3.6. Validation of the prediction model

We based on the most commonly used techniques [11] to evaluate the accuracy of our prediction model, MMRE, MdMRE and Pred(25%).

The values of MRE obtained for each diagram are shown in Table 10. In this experiment, the value for MMRE and MdMRE for is 0.20. The value obtained for Pred(25%) is a 70%, what indicates that a 75% of the obtained values are at least 70% accurate.

In Figure 1 are shown the predicted and real average values for the understandability time of each statechart diagram of the experiment.

[2] In this case, following some recommendations given by the experts, as the sum of the membership degrees is greater than 1, we divided the membership degree of the second most similar prototype by two.

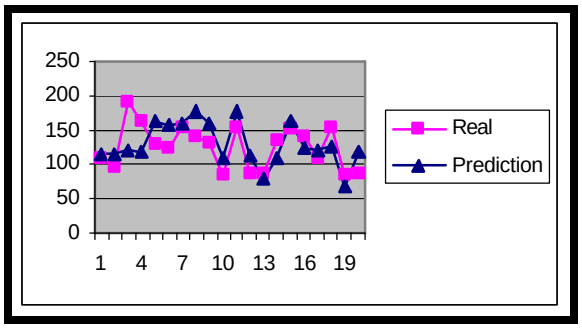


Figure 1. Predicted values vs. real values

4. Conclusions

The main contribution of this work is a prediction model for he Understandability Time of UML statecharts diagrams. This model was built from some metrics for the structural complexity and size of UML statechart diagrams using two fuzzy logic-based techniques: the Fuzzy Prototypical Knowledge Discovery (FKPD) process and the Fuzzy Deformable Prototypes. The data used to build the model was obtained through a controlled experiment. Moreover, the model was validated usin data obtained in a replication of the experiment. Through the validation, we reached the conclusion that – in a certain way- it is a good model, since a 75% of the understandability time estimated values are at least 70% accurate.

Although the results are encouraging, we are aware that we must improve our study in two ways: with respect to the data used for obtaining the prediction model and with respect to the technique applied for building the prediction model.

For that, on one hand we have to replicate the experiment with professionals and examine the usefulness of the metrics in real projects. Related to the prediction model, there are also some aspects to improve. Using algorithms such as Fuzzy C-Means [4], Fuzzy Kohonen Networks [5] or soft clustering algorithms in general, would allow us to raise the power of problems resolution. These algorithms can make the clustering process and the model construction to be done at once, deciding the number of prototypes before being carried out. Moreover, these algorithms allow a better manipulation of great volume of data.

Acknowledgements

This research is part of the MESSENGER project (PCC-03-003-1) financed by "Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha (Spain)" and the CALIPO project supported by "Dirección General de Investigación del Ministerio de Ciencia y Tecnologia (Spain)" (TIC2003-07804-C05-03).

References

[1] Atkinson C. and Kühne T. (2003). "Model-Driven Development: A Metamodeling Foundation". *IEEE Software* 20(5), 36- 41.

[2] Basili, V. R., Caldiera, G. y Rombach, H. D. (1994). *Goal Question Metric Paradigm*. Encyclopedia of Software Engineering, vol. 1. John Wiley & Sons, 528-532.

[3] Bell R. (1990). "Analytic Issues in the Use of Repertory Grid Technique". *Advances in Personal Construct Psychology* 1, pp. 25-48.

[4] Bezdek J., Hathaway R., Sabin M., Tucker W. (1987). "Convergence Theory for Fuzzy c-Means Counterexamples and Repairs". *IEEE Trans Syst., Man and Cybern.* SMC-17 (5), pp. 873 - 877.

[5] Bezdek J., Tsao E., Pal N. (1992). "Fuzzy Kohonen Clustering Net-works". *IEEE International Conference on Fuzzy Systems.* San Diego, pp. 1035-1043.

[6] Briand, L., Morasca, S., Basili, V. (1996) "Property-based software engineering measurement". *IEEE Transactions on Software Engineering*, 22 (1) pp. 68-85

[7] Briand L., Arisholm S., Counsell F., Houdek F., Thévenod-Fosse P. (1999b). "Empirical Studies of Object-Oriented Artifacts, Methods, and Processes: State of the Art and Future Directions". *Empirical Software Engineering*, 4(4), pp. 387-404.

[8] Briand L., Bunse C., Daly J. (2001). "A Controlled Experiment for evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs". *IEEE Transactions on Software Engineering*, 27(6), pp. 513-530.

[9] Brito e Abreu F., Zuse H., Sahraoui H., Melo W. (1999). "Quantitative Approaches in Object-Oriented Software Engineering". *ECOOP'99 Workshops*, LNCS 1743, A. Moreira and S. Demeyer (eds). Springer-Verlag. pp. 326-337.

[10] Castro J., Trillas E., Zurita J. (1998). "Non-monotonic Fuzzy Reasoning". Fuzzy Sets and Systems 94, North Holland, pp. 217 - 225.

[11] Conte S., Dunsmore H., Shen V. (1986). Software Engineering Metrics. Benjamin-Cummings Publishing Co., Inc., USA.

[12] Fayyad U., Piatetsky-Shapiro G., Smyth P. (1996). "The KDD Process for Extracting Useful Knowledge from Volumes of Data". *Communications of the ACM*, 39(11), pp. 27 - 34.

[13] Genero M., Olivas J., Piattini M., Romero F. (2001). "Using metrics to predict OO information systems maintainability", *CAISE 2001*, Lecture Notes in Computer Science, 2068, Interlaken, Switzerland, 388-401.

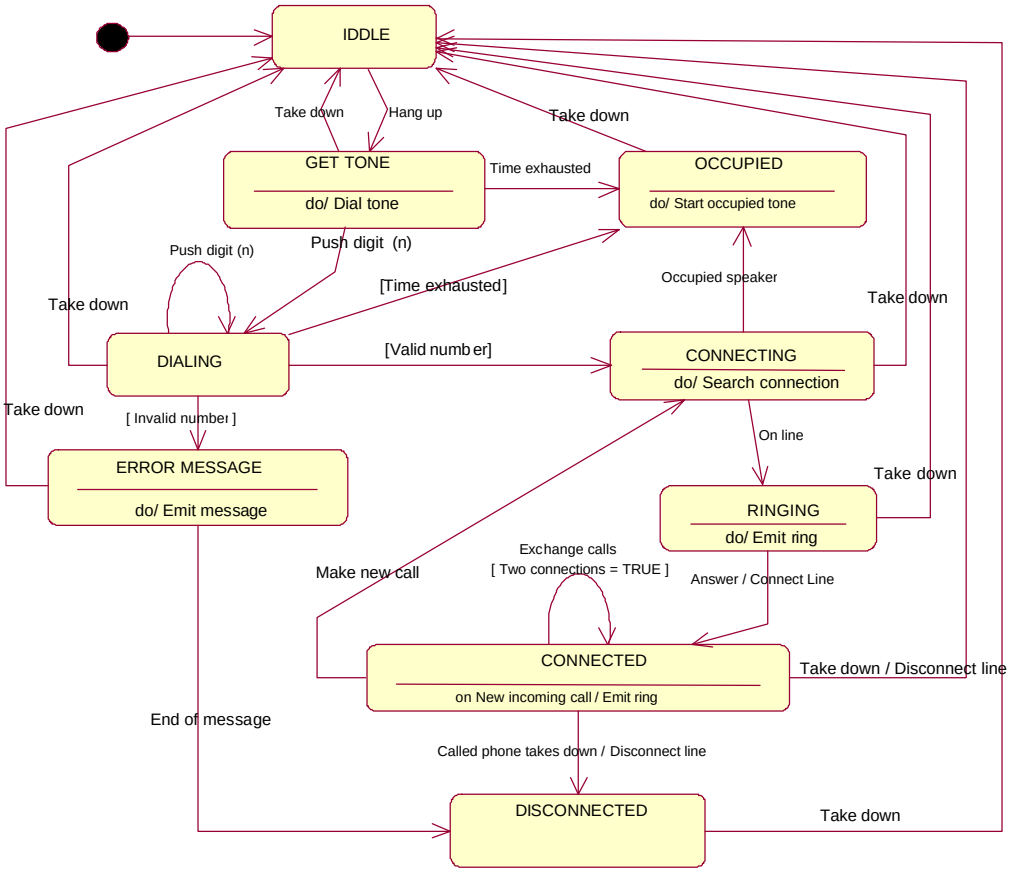
[14] Genero M., Piattini M., Calero C. (2002). "An study to validate metrics for class diagrams". *Jornadas Iberoamericanas de Ingeniería de Requisitos y Ambientes de Software (IDEAS´2002)*, La Habana (Cuba), pp. 226-235.

[15] Genero M., Piattini M. and Calero M. (Eds.) *Metrics For Software Conceptual Models*. Imperial College Press, UK, 2004.
[16] Harrison R., Counsell S., Nithi R. (2000). "Experimental Assessment of the Effect of Inheritance on the Maintainability of Object-Oriented Systems", *The Journal of Systems and Software*, 52, 173-179.
[17] Kitchenham B., Pflegger S., Pickard L., Jones P., Hoaglin D., El-Emam K. y Rosenberg J. (2002). "Preliminary Guidelines for Empirical Research in Software Engineering". *IEEE Transactions of Software Engineering* 28(8), pp. 721-734.
[18] McCabe, T. (1976). "A Complexity Measure". *IEEE Transactions on Software Engineering*. Vol. 2. Nº4, pp. 308-320.
[19] MDA- The OMG Model Driven Architecture (2002). Available: http://www.omg.org./mda/, August 1st, 2002.
[20] Miranda D., Genero M., Piattini M. (2003). "Empirical validation of metrics for UML statechart diagrams". *Fifth International Conference on Enterprise Information Systems (ICEIS 03)*, 1, pp. 87-95.
[21] Olivas J. (2000). *Contribución al Estudio Experimental de la Predicción basada en Categorías Deformables Borrosas*, Tesis Doctoral, Universidad de Castilla La Mancha, España.
[22] Olivas J., Romero F. (2000). "FPKD. Fuzzy Prototypical Knowledge Discovery. Application to Forest Fire Prediction". *Proceedings of the SEKE'2000*, Knowledge Systems Institute, Chicago, Ill. USA, pp. 47 - 54.
[23] Perry D., Porter A., Votta L. (2000). "Empirical Studies of Software Engineering: A Roadmap". *Future of Software Engineering*. Ed:Anthony Finkelstein, ACM, pp. 345-355.
[24] Poels, G. and Dedene, G. (2000). "Measures for Assessing Dynamic Complexity Aspects of Object-Oriented Conceptual Schemes". *Proceedings of 19th International Conference on Conceptual Modelling (ER 2000)*, pp. 499-512.
[25] Schneidewind N. (2002). "Body of Knowledge for Software Quality Measurement". *IEEE Computer* 35(2), pp. 77-83.
[26] Wohlin C., Runeson P., Höst M., Ohlson M., Regnell B., Wesslén A. (2000). *Experimentation in Software Engineering: An Introduction*, Kluwer Academic Publishers.
[27] Zadeh, L. A. (1982). "A note on prototype set theory and fuzzy sets". *Cognition* 12, pp. 291- 297.

Appendix A

In this appendix we will show, as example, one of the test used in the experiment, corresponding to the diagram 16.

DIAGRAM 16: MAKING A TELEPHONE CALL 2



TIME NOW: _____

Answer the following questions:

1. If you get from CONNECTED to IDDLE, which event has occurred previously? **TAKE DOWN**
2. If you are CONNECTED and the event *Occupied speaker* occurs, which state do you get to? **OCCUPIED**
3. Which events and/or conditions will have occurred at least and in which order for getting form IDDLE to RINGING? **(1) Hang up (2) Push digit(n) (3) [Valid number] (4) On line**

4. Starting from ERROR MESSAGE, which state will you get if the following sequence of events and conditions occurs? (1) Take down (2) Hang up (3) Push digit(n) and (4) [Time exhausted]. **OCCUPIED**

TIME NOW: _____