Utilización de Técnicas Basadas en la Lógica Borrosa para Predecir el Tiempo de Entendimiento de los Diagramas de Estados UML

José A. Cruz-Lemus, Marcela Genero, José A. Olivas, Francisco P. Romero y Mario Piattini

Departamento de Informática, Universidad de Castilla-La Mancha
Paseo de la Universidad, 4, 13071, Ciudad Real (España)
{JoseAntonio.Cruz, Marcela.Genero, JoseAngel.Olivas, Mario.Piattini}@uclm.es
fpromero@soluziona.com

Resumen

En este trabajo se presenta una aplicación de la lógica borrosa en el ámbito de la predicción en la Ingeniería del Software. Concretamente se ha utilizado el Descubrimiento de Conocimiento Prototípico Borroso para caracterizar los diagramas de estados UML de acuerdo a su facilidad de entendimiento, partiendo de su complejidad estructural y tamaño, expresadas mediante métricas, y los Prototipos Deformables Borrosos para obtener un modelo de predicción del tiempo de entendimiento de los diagramas de estados UML. El modelo obtenido es válido –en cierta medida- ya que el 75% de los valores estimados tienen al menos un 70% de exactitud, aunque es necesario validarlo con datos referentes a proyectos reales.

1. Introducción.

En la Ingeniería del Software es bien sabido que las características que hacen a la calidad de los sistemas orientados a objetos (OO), como la mantenibilidad, debe ser garantizada desde las etapas iniciales de su ciclo de vida, centrándonos en lo modelos obtenidos en dichas etapas, más aún teniendo en cuenta el gran auge que ha tenido en estos últimos años el desarrollo orientado a modelos (Model-Driven Developement) [1] y la arquitectura basada en modelos (Model-Driven Architecture (MDA)) [19]. Utilizando UML en el desarrollo OO, se realizan una serie de diagramas que cubren tanto aspectos estáticos (diagramas de clases) como dinámicos (diagramas de casos de uso, diagramas de estados, etc.). Para evaluar la calidad de tales diagramas de manera objetiva es necesario contar con medidas cuantitativas que eviten sesgos en el proceso de evaluación

Existen varios trabajos sobre la medición de la calidad de los diagramas de clases UML y de los diagramas de casos de uso [15]. Sin embargo, apenas hay unas pocas referencias en la bibliografía sobre métricas para los diagramas de comportamiento, tales como los diagramas d estados, los diagramas de secuencia, los diagramas de actividad, etc. Una de las primeras aproximaciones hacia la definición de métricas para diagramas de comportamiento aparece en [11], donde se definen y aplican métricas para diagramas desarrollados con OMT (Object Modelling Technique) [22]. En [26] se definen medidas de complejidad para modelos conceptuales OO dirigidos por eventos. Así mismo, [9] y [26] señalan que la definición de métricas para diagramas que capturen los aspectos dinámicos de los sistemas OO es un área relevante, un tanto descuidada en el ámbito de la medición del software. Este hecho nos motivó a definir métricas para los diagramas de comportamiento UML, comenzando con los diagramas de estados [20]. La Tabla 1 muestra brevemente la definición de cada una de dichas métricas.

Tabla 1. Métricas para diagramas de estados UML

	Métrica	Definición
Tamaño	**NEntryA**	El número total de acciones de entrada, i.e., las acciones que se ejecutan cada vez que se entra en un estado.
	NExitA	El número total de acciones de salida, i.e., las acciones que se ejecutan cada vez que se sale de un estado.
	NA	El número total de actividades (*do/activity*) del diagrama.
	NSS	El número total de estados sencillos, considerando también los subestados dentro de un estado compuesto.
	NCS	El número total de estados compuestos, i.e., aquellos que tienen subestados anidados.
	NE	El número total de eventos.
	NG	El número total de condiciones de guarda.
Complejidad estructural	**NT**	El número total de transiciones considerando aquellas en la que el estado origen es distinto al estado destino, las transacciones final e inicial, las auto-transacciones (el estado origen y el destino son el mismo) y las transacciones internas (aquellas que responden a un evento sin dejar el estado).
	CC	*Complejidad Ciclomática de McCabe* [18][1], definida como |NSS-NT|+2

Nuestro enfoque sobre cómo la complejidad estructural y el tamaño como atributos internos de los diagramas de estados está potencialmente relacionada con su facilidad de entendimiento (*understandability*) surgió a partir de trabajos similares realizados en el campo de la Ingeniería del Software Empírica [8][16], en los cuáles esta propiedad se mostró como uno de los mayores determinantes de características externas de la calidad, como la facilidad de entendimiento y de mantenimiento.

Las métricas presentadas en la Tabla 1 fueron validadas teóricamente utilizando el marco basado en propiedades de Briand et al. [6], obteniendo que las métricas NEntryA, NExitA, NA, NSS, NCS, NE y NG son métricas de tamaño y las métricas NT y CC son métricas de complejidad.

Como es bien sabido en el campo de la medición del software, para que las métricas que miden atributos internos (tamaño, complejidad, etc.) sean útiles, es necesario que sirvan para predecir algún atributo externo de la calidad, como la facilidad de entendimiento.

Mediante un experimento controlado y su réplica [21] (que se detalla en la sección 2), se ha encontrado que de las métricas que se habían propuesto para diagramas de estados UML, parecen estar fuertemente correlacionadas con el tiempo de entendimiento de los mismos, las métricas *Número de Actividades* (NA), *Número de Estados Simples* (NSS), *Número de Guardas* (NG) y *Número de Transiciones* (NT). Esto nos llevó a pensar en la construcción de un modelo de predicción de la facilidad de entendimiento de diagramas de estados UML basado en los valores de estas métricas. Para la construcción del modelo de predicción se han usado todas las métricas, ya que se considera demasiado prematuro descartar alguna de ellas.

Viendo los alentadores resultados obtenidos al aplicar el proceso de Descubrimiento de Conocimiento Prototípico Borroso (DCPB) y Prototipos Deformables Borrosos para la construcción de modelos de predicción aplicados a diferentes dominios [24][13][14], se decidió usarlos para nuestro propósito. Para no extendernos en demasía, no se explicarán en profundidad todos los pasos del proceso de predicción, aunque pueden encontrarse más detalles en [23]

En nuestro caso, los principales objetivos del proceso de predicción son:

1. La búsqueda y extracción automática de prototipos borrosos que caractericen los diagramas de estados UML, a través de su facilidad de entendimiento, expresada a través de la complejidad estructural y el tamaño de dichos diagramas. Esto se llevará a cabo mediante el proceso de Descubrimiento de Conocimiento Prototípico Borroso (DCPB).

[1] A pesar de que el Número Ciclomático de McCabe se definió para calcular la complejidad del código, lo hemos adaptado para medir la complejidad estructural de diagramas de estados en UML.

2. La obtención de un modelo de predicción del tiempo de entendimiento de los diagramas de estados UML, llevando a cabo la deformación de los prototipos borrosos previamente definidos.

El DCPB es una ampliación del proceso clásico de KDD [12] que presenta como novedades la incorporación de conocimiento en diferentes puntos mediante decisiones del usuario o del experto y un resultado preparado para generar unos prototipos conceptuales denominados Prototipos Deformables Borrosos, basados en la idea de Categorías Prototípicas Borrosas [23][28]. La utilización de la Lógica Borrosa permite conseguir estos resultados de forma más comprensible y útil para el posterior uso en la predicción, con ello, se permite evaluar situaciones nuevas a partir de dichos prototipos, establecer predicciones en cuanto a situaciones reales y también tomar decisiones a partir de dichas predicciones. Además se utilizan otras técnicas como el clustering borroso y las funciones de agregación [10], facilitando la generación de modelos estructurados, significativos y fácilmente actualizables.

Los datos utilizados para la obtención del modelo de predicción se obtienen a partir de un experimento controlado realizado con profesores y alumnos de la Ingeniería Informática de la Universidad de Castilla-La Mancha. Para validar dicho modelo se utilizan los datos obtenidos en una réplica de dicho experimento realizado con otro grupo de alumnos.

Este trabajo está organizado de la siguiente manera: en la sección 2 se presentan las fuentes de datos que son los datos obtenidos en un experimento controlado y su réplica. A continuación, en la sección 3 se describen las distintas etapas seguidas para obtener el modelo de predicción, que consisten en el proceso DCPB, el proceso de predicción propiamente dicho y la validación de dicho modelo. Finalmente, en la sección 4, se presentan las conclusiones y las líneas de investigación que surgen a partir de éste trabajo.

2. Fuentes de datos

En esta sección se describen un experimento controlado y su réplica, que se realizaron teniendo en cuenta algunos consejos suministrados por expertos en la Ingeniería del Software Empírica [7][17][25][27]. Los datos obtenidos en el experimento se utilizarán para definir los prototipos borrosos del tiempo de entendimiento de los diagramas de estados UML y a partir de dichos prototipos construir un modelo de predicción del tiempo de entendimiento de los diagramas de estados UML (ver sección 3). Para validar dicho modelo se utilizarán los datos obtenidos en la réplica.

2.1. Primer experimento

2.1.1. Definición del experimento

Utilizando la plantilla para definición de objetivos que presenta el método GQM (Goal-Question-Metric) [2], el objetivo del experimento es el que detalla la Tabla 2.

Tabla 2. Objetivo del experimento según GQM

Analizar	*las métricas de complejidad para diagramas de estados en UML*
Con el propósito de	*evaluar*
Con respecto a	*la capacidad de ser empleadas como indicadores de la facilidad de entendimiento de los diagramas de estados en UML*
Desde el punto de vista de	*los diseñadores de SOO*
En el contexto de	*estudiantes en el último año de la Ingeniería Informática y profesores del área de Ingeniería del Software en el Departamento de Informática en la Universidad de Castilla-La Mancha*

2.1.2. Planificación

La planificación incluye las siguientes actividades:

- *Selección del contexto.* El contexto del experimento es un grupo relacionado con el área de Ingeniería del Software de la universidad, y por tanto el experimento se desarrolla en un ambiente no industrial. Los sujetos son 11 estudiantes y 8 profesores. Los estudiantes están matriculados en último curso de la Ingeniería Informática en la Universidad de Castilla-La Mancha. Todos los profesores pertenecen al área de Ingeniería del Software. El experimento es específico ya que se centra en métricas para capturar la complejidad de los diagramas de estados en UML. El experimento se refiere a un problema real: es decir, qué indicadores pueden utilizarse para evaluar la facilidad de entendimiento de los diagramas de estados en UML.
- *Selección de los sujetos.* Se eligen los sujetos por conveniencia, son estudiantes y profesores con suficiente experiencia en el diseño y desarrollo de sistemas orientados a objetos con UML.
- *Selección de las variables.* La variable independiente es la complejidad de los diagramas de estados en UML. La variable dependiente es la facilidad de entendimiento de los mismos.
- *Instrumentación.* Los objetos usados en el experimento son 20 diagramas de estados UML. La variable independiente, se mide a través de las métricas presentadas en la Tabla 1. La variable dependiente, se evalúa a partir del tiempo que cada sujeto emplea en contestar el cuestionario asociado a cada diagrama (al que se denomina tiempo de entendimiento). Suponiendo que un diagrama de estados es más comprensible cuanto menor sea su tiempo de entendimiento.
- *Formulación de hipótesis.* Con el experimento se pretenden evaluar las siguientes hipótesis:
  - Hipótesis nula, H_0: No existe correlación significativa entre la complejidad estructural, el tamaño de los diagramas de estados en UML y el tiempo de entendimiento.
  - Hipótesis alternativa, H_1: Existe correlación significativa entre la complejidad estructural, el tamaño de los diagramas de estados en UML y el tiempo de entendimiento.
- *Diseño del experimento.* Se elige un diseño intra-sujetos para el experimento, es decir, todos los sujetos tienen que resolver todas las tareas propuestas en todos los cuestionarios. Los cuestionarios se entregaron en distinto orden a cada sujeto y se recaleó que deberían de resolverlos según orden en el que se le entregaba, para evitar efectos de aprendizaje.

2.1.3. Operación

En esta fase se recogen los resultados, e incluye las siguientes actividades:

- *Preparación.* Cuando se hizo el experimento todos los estudiantes habían cursado 2 asignaturas de Ingeniería del Software, en las que aprendieron en profundidad cómo construir sistemas OO con UML. Los profesores tienen como mínimo 3 años de experiencia en el diseño y desarrollo de sistemas OO OO. Además todos los sujetos recibieron una sesión intensiva de entrenamiento, previa a la realización del experimento. Sin embargo ninguno conocía los aspectos que se trataban de estudiar, ni las hipótesis establecidas. Se preparó el material que se entregó a los sujetos, consistente en un guía explicando la notación de los diagramas de estados en UML, y los 20 diagramas. Los diagramas se referían a diversos universos de discurso, pero eran lo suficientemente generales como para ser fácilmente comprendidos por los sujetos. La complejidad de cada diagrama es diferente ya que, como se puede ver en la Tabla 3, se trató de cubrir un amplio rango de los valores de las métricas. Cada diagrama iba acompañado de un test que incluía un cuestionario para evaluar si los sujetos realmente entendían el contenido de cada diagrama de estados UML. Cada cuestionario contenía cuatro preguntas conceptualmente similares y escritas en el mismo orden, y cada sujeto tenía que anotar la hora a la que comenzaba a responder el cuestionario y la hora a la que finalizaba. La diferencia entre esas dos anotaciones es lo que se llama *tiempo de entendimiento* y se expresa en segundos.

Tabla 3. Valores de las métricas para cada diagrama de estados UML usado en el experimento

Diagrama	NEntryA	NExitA	NA	NSS	NCS	NT	NE	NG	McCabe
1	1	1	0	3	0	7	6	2	5
2	1	0	3	1	0	7	6	0	4
3	2	0	2	4	1	7	4	3	1
4	0	0	2	4	0	11	11	2	7
5	3	2	2	4	0	13	11	0	9
6	6	6	0	6	1	13	12	1	5
7	1	0	1	5	2	11	6	3	2
8	1	0	3	5	0	13	12	4	9
9	0	1	4	5	0	10	7	1	7
10	2	1	0	4	0	6	6	0	4
11	1	2	1	6	3	17	12	0	3
12	1	1	1	3	0	5	5	2	3
13	2	1	0	2	0	4	4	0	2

Diagrama	NEntryA	NExitA	NA	NSS	NCS	NT	NE	NG	McCabe
11	1	1	5	3	0	8	8	0	5
15	1	0	1	0	1	13	11	1	3
16	0	0	5	0	0	21	22	1	16
1	2	0	1	5	1	8	6	2	2
18	2	0	1	12	0	21	23	2	13
19	0	1	0	2	0	6	5	0	1
20	0	0	0	5	1	12	11	0	7

- *Ejecución.* Se entregó a los sujetos el material descrito en el párrafo anterior y se les explicó la manera de proceder para realizar el experimento, esto es: cada sujeto tenía que resolver el cuestionario por sí mismo y disponía de tiempo ilimitado. Se indicó que disponían de un plazo de una semana, tras el cual deberían entregar los cuestionarios resueltos. Pasados los 7 días se recogieron todos los cuestionarios con las respuestas a las preguntas y los tiempos de entendimiento de cada diagrama. Somos conscientes de que esto puede haber sesgado los resultados porque los sujetos no fueron supervisados, hecho que se trató de mejorar en la réplica.
- *Validación de datos.* Se comprobaron todos los cuestionarios para ver si eran válidos, y se descartaron 6 cuestionarios ya que su nivel de corrección (número de preguntas contestadas bien/número de preguntas contestadas) y completitud (número de preguntas contestadas bien/número de preguntas totales) era menor que 0,9.

2.2. Réplica del experimento

Las diferencias más importantes entre el experimento y su réplica son:

- Los sujetos fueron 24 estudiantes del tercer año de la Ingeniería Informática, que sólo habían cursado una asignatura de Ingeniería del Software, en la que aprendieron a diseñar software OO utilizando UML. Esto significa que la experiencia de los sujetos es menor.
- Los sujetos tenían que completar los tests solos y en no más de dos horas. Cualquier duda podría ser resuelta por la persona que coordinaba el experimento, lo que contribuyó a controlar el plagio entre los sujetos.
- Se descartaron 55 diagramas porque su nivel de corrección y completitud eran menores que 0,9.

Como se mencionó anteriormente los datos obtenidos en esta réplica se utilizarán para evaluar la precisión del modelo de predicción obtenido (ver sección 3.5).

3. Construcción del modelo de predicción del tiempo de entendimiento de los diagramas de estados UML

Para construir el modelo de predicción (utilizando los datos obtenidos en el experimento que se describe en la sección 2.1) se llevaron a cabo dos procesos principales:

1. Proceso DCPB:
   - Transformación de los datos
   - Obtención de prototipos utilizando técnicas de clustering
   - Definición paramétrica de los prototipos
   - Representación borrosa de los prototipos
2. Proceso de predicción:
   - Deformación de los prototipos borrosos para predecir el tiempo de entendimiento de los diagramas de estados UML
   - Validación del modelo de predicción

A continuación se describe cómo se han llevado a cabo cada una de las etapas mencionadas previamente.

3.1. Transformación de los datos

Primeramente, fue necesario transformar los datos para que fueran válidos para el proceso DCPB. Por un lado, se obtuvo la tabla con los valores de las métricas para diagramas de estados (ver Tabla 3). Por otro, se obtuvo el tiempo de entendimiento para cada diagrama y cada sujeto. A partir de estos tiempos, se obtuvieron los tiempos mínimo (TEMin), promedio (TEAvg) y máximo (TEMax) para cada diagrama (ver Tabla 4).

Tabla 4. Tiempos obtenidos (en segundos) en el proceso de transformación

Diagrama	TEAvg	TEMin	TEMax	Diagrama	TEAvg	TEMin	TEMax
1	110,00	15	420	11	153,16	85	360
2	95,00	30	170	12	86,37	50	180
3	191,91	61	360	13	88,05	35	300
4	163,39	69	405	14	136,05	41	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	240	19	84,26	40	180
10	85,21	50	180	20	85,84	42	140

De esta manera se obtuvo una tabla con 20 filas y 4 columnas, asociada a la tabla que contiene los valores de las métricas para cada diagrama (ver Tabla 3).

3.2. Obtención de los prototipos usando técnicas de clustering

Con el objetivo de detectar las relaciones entre los diagramas de estados para ser capaces posteriormente de determinar si tienen un tiempo de entendimiento bajo, medio o alto, se realiza un proceso de clustering jerárquico (según la técnica de Emparrillados (*Repertory Grids*) [3]).

Se construyó una matriz de similitud de 20 x 20 elementos cuyos valores en la diagonal representan los grados de similitud entre los diagramas. Convirtiendo estos valores en porcentajes y desencadenando el algoritmo de clustering jerárquico, se obtuvieron los resultados que se muestran en el dendrograma de la Figura 1.

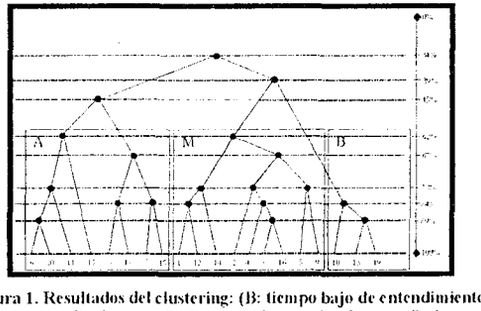

Figura 1. Resultados del clustering: (B: tiempo bajo de entendimiento, M: tiempo medio de entendimiento, A: tiempo alto de entendimiento)

Una vez obtenidos los resultados del clustering, y con el conocimiento heurístico previo de tener tres prototipos, se realiza un corte a una similitud inferior a un 55%. Con lo cual los diagramas se agrupan en tres prototipos de acuerdo a los valores de las métricas que reflejan su complejidad estructural y tamaño, como muestra la Tabla 5.

Tabla 5. Diagramas agrupados según el prototipo al que pertenecen

Prototipos	Diagramas
B: Tiempo Bajo de Entendimiento	10,13,19
M: Tiempo Medio de Entendimiento	1,2,4,5,8,9, 12,14,16
A: Tiempo Alto de Entendimiento	3,6,7,11,15,17,18,20

3.2. Definición paramétrica de los prototipos

Considerando los prototipos de datos encontrados en la sección anterior y sus valores de tiempos de entendimiento mostrados en la Tabla 5, se obtuvo la definición paramétrica de los prototipos, como muestra la Tabla 6

Tabla 6.Definición paramétrica de los prototipos

A: T° Alto de Entendimiento		M: T° Medio de Entendimiento		B: T° Bajo de Entendimiento	
Medio	2 min. 15 seg.	Medio	2 min. 5 seg.	Medio	1 min. 25 seg.
Maximo	7 min.	Maximo	7 min.	Maximo	6 min.
Minimo	42 seg.	Minimo	15 seg.	Minimo	35 seg.

3.3. Representación borrosa de los prototipos

Los tres prototipos han sido representados como "números borrosos", los cuales permitirán obtener un grado de pertenencia (entre 0 y 1) de un nuevo diagrama de estados con cada uno de ellos (ver Figura 2).

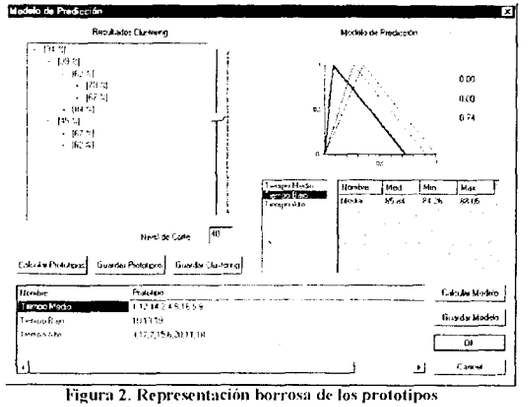

Figura 2. Representación borrosa de los prototipos

La utilización de números borrosos triangulares permite que para la representación de los mismos sea unicamente necesario conocer su centro (ver Tabla 7) y el tamaño de la base del triángulo (denominado a y b en la Tabla 7)

Tabla 7. Definición borrosa de los prototipos

Prototipos	Diagramas	a	Centro	b
B: Tiempo Bajo de Mantenimiento	10,13,19	0	0.08	0.74
M: Tiempo Medio de Mantenimiento	1,2,4,5,8,9,12,14,16	0	0.26	0.92
A: Tiempo Alto de Mantenimiento	3,6,7,11,15,17,18,20	0	0.34	1

La definición formal de los prototipos en forma de números borrosos se obtiene mediante la normalización, realizada de la siguiente manera $X'_{ij} = \frac{X_{ij} - X_{min}}{X_{max} - X_{min}}$, y la agregación mediante la media de los datos de los valores de las métricas

3.4. Deformación de los prototipos borrosos para predecir el tiempo de entendimiento de los diagrama de estados UML

En esta sección se muestra cómo predecir el tiempo de entendimiento para un nuevo diagrama de estados. La idea general es que dados los valores de las métricas de un nuevo diagrama de estados, las definiciones paramétricas de los prototipos que tienen un grado de pertenencia distinto de cero, se adaptan o deforman para predecir el tiempo de entendimiento de este nuevo diagrama de estados.

El proceso a llevar a cabo es el que sigue:

1. Normalizar los valores medidos por medio del indice de normalización asociado al modelo de predicción obtenido. Se usa la misma fórmula que en la definición de los números borrosos y con los mismos coeficientes de mínimo y máximo.
2. Calcular la media de los valores previamente normalizados (a este valor se le llama X).
3. A partir del valor X se obtienes los grados de pertenencia a los prototipos representados por medio de números borrosos, de la siguiente manera:

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

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

4. Para obtener el valor estimado del tiempo de entendimiento de un nuevo diagrama de estados, los prototipos borrosos se "deforman" para considerar el grado de afinidad con todos los prototipos. Aplicando el concepto de Prototipos Deformables Borrosos, definido en [23], la caracterización del nuevo diagrama de estados propuesto puede describirse mediante la siguiente combinación lineal:

$$C_{nuevo}(w_1 \ldots w_n) = \sum \mu p_i (v_1 \ldots v_n)$$

Donde:

C_{real}	Caso real propuesto.
$(w_1 \ldots w_n)$	Parámetros que describen el caso real propuesto
μ_{pi}	Grados de pertenencia con los Prototipos Deformables Borrosos distintos de 0
$(v_1 \ldots v_n)$	Parámetros de estos Prototipos Deformables Borrosos

Utilizando los datos obtenidos en la réplica (ver sección 2.2) se "deforma" el modelo de predicción para predecir el tiempo de entendimiento

Se toma como caso de partida el diagrama nº 16. Los valores de las métricas para este diagrama se muestran en la Tabla 8

Tabla 8. Valores de las métricas del diagrama 16

NEntryA	NExitA	NA	NSS	NCS	NT	NE	NG	McCABE
0	0	5	9	0	21	22	1	16

1. Primero se normalizan los valores de las métricas, obteniendo los valores mostrados en la Tabla 9.

Tabla 9. Valores normalizados de las métricas del diagrama 16

NEntryA	NExitA	NA	NSS	NCS	NT	NE	NG	McCABE
0	0	1	0,7	0	1	0,95	0,3	1

2. La media de los valores normalizados es X = 0.53
3. Después se calcula el valor de las afinidades a los prototipos, como se muestra en la Tabla 10

Tabla 10. Valor de las afinidades de los prototipos

Prototipos	Afinidades
Tiempo Bajo de Entendimiento	0
Tiempo Medio de Entendimiento	0.591
Tiempo Alto de Entendimiento	0.712

4. Y finalmente se calcula el tiempo medio correspondiente (Media Estimada) a la predicción realizando la combinación lineal modificada de los dos prototipos más afines explicada anteriormente (ver Tabla 11).

Tabla 11. Combinación lineal para el cálculo de tiempos medios[2]

Media	0,591 / 2	2 min. 5 seg.	+	0,712	2 min. 15 seg.	−	2 min. 2 seg.
Máximo		7 min.			7 min.		7 min. 5 seg.
Mínimo		15 seg.			42 seg.		34 seg.

De esta manera se obtuvo la predicción del tiempo de entendimiento para el diagrama 16. El resultado de aplicar la deformación de los prototipos a todos los diagramas se muestra en la Tabla 12.

Tabla 12. Datos de la validación del modelo de predicción

DIAGRAMA	X	Af (B)	Af (M)	Af (A)	Valor Estimado	Valor Real	MRE
1	0,15	0,894	0,577	0,441	116,38	110,00	0,058
2	0,11	0,909	0,538	0,412	114,925	95,00	0,210
3	0,18	0,848	0,692	0,529	120,52	191,91	0,372
4	0,16	0,879	0,615	0,471	117,765	163,39	0,279
5	0,21	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	109,41	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

3.5 Validación del modelo de predicción

Las técnicas más utilizadas para la evaluar la exactitud de los modelos de predicción son las siguientes [20]

Magnitud del Error Relativo (MRE).
Media de la Magnitud del Error Relativo (MMRE).
Mediana de la Magnitud del Error Relativo (MdMRE)
Predicción a nivel n (Pred(n)).

[2] En este caso dado que la suma de los grados de pertenencia es mayor que 1, utilizando heurísticas provistas por el experto, dividimos el grado de pertenencia del segundo prototipo más afín por 2.

MRE se define como:

$$MRE_i = \frac{|TiempoRealDeEntendimiento - TiempoDeEntendimientoPredecido|}{TiempoRealDeEntendimiento}$$

Donde i representa cada observación para la cual se predice el tiempo de mantenimiento.

MMRE se calcula como la media de todos los MREs:

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} \frac{|TiempoRealDeEntendimiento - TiempoDeEntendimientoPredecido|}{TiempoRealDeEntendimiento}$$

La media tiene en cuenta el valor numérico de cada observación en la distribución de los datos, y es sensitiva a valores individuales de predicción con valor de MRE muy alto. Por ello, otra elección es usar la mediana, que también es una medida de la tendencia central, pero menos sensible a valores extremos. La mediana de los valores de MRE para i observaciones se denomina MdMRE.

Otro indicador comúnmente usado es la Predicción a nivel *n*, también conocido como *Pred(n)*. Mide el porcentaje de estimaciones que están dentro de un n% de los valores reales. Según algunas sugerencias [20], 25% es un valor recomendable para n. Lo que significa que un buen modelo de predicción tendrá una precisión del 75%.

Los valores de MRE obtenidos para cada diagrama se muestran en la Tabla 12. La magnitud media del error en este experimento (MMRE) es 0.20. El valor de la mediana (MdMRE) es 0.20. El valor obtenido para Pred(25%) es un 70%, lo que nos indica que un 75% de los valores obtenidos, posee una exactitud de al menos el 70%.

En el gráfico mostrado en la Figura 3 se pueden observar los valores medios procedentes de la predicción y los valores medios reales.

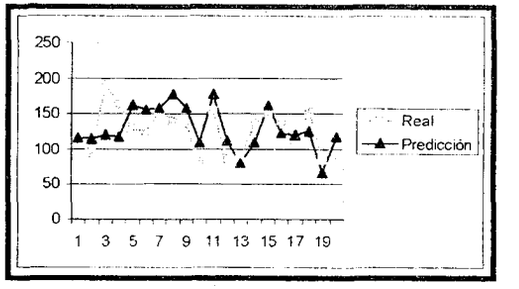

Figura 3. Valores de la predicción vs. valores reales

4. Conclusiones

En la Ingeniería del Software es ampliamente reconocido que las propiedades estructurales de los diagramas estáticos y dinámicos de UML pueden tener una gran influencia en la calidad del producto software que finalmente se entrega. Por esa razón, la existencia de métricas es crucial, ya que permiten evaluar las propiedades estructurales de una manera cuantitativa y objetiva.

Con la hipótesis de que el tamaño y la complejidad estructural de los diagramas de estados UML pueden influenciar la facilidad de entendimiento (y por ende la facilidad de mantenimiento) de los mismos, se definió un conjunto de métricas [21].

La realización de un experimento controlado y su posterior réplica nos llevó a la conclusión de que las métricas que cuentan el número de actividades (NA), el número de estados simples (NSS), el número de guardas (NG) y el número de transiciones (NT) están altamente correlacionadas con la facilidad de entendimiento de los diagramas de estados UML.

Así pues, el siguiente paso consistía en la obtención de un modelo de predicción capaz de determinar a priori el esfuerzo de entendimiento que supone cualquier diagrama de estados UML, en función de los valores asociados a las métricas previamente definidas.

En este trabajo se ha detallado la obtención de dicho modelo de predicción, basado en la utilización del proceso denominado Descubrimiento de Conocimiento Prototípico Borroso (DCPB) y de los Prototipos Deformables Borrosos. Validando el modelo de predicción se llegó a la conclusión de que -en cierta medida- es un "buen" modelo, ya que ya que el 75% de los valores estimados del tiempo de entendimiento tienen al menos un 70% de exactitud. Somos conscientes de que, como ocurre con la mayoría de modelos de predicción, el modelo obtenido no es generalizable para cualquier diagrama, ya que el modelo es dependiente de los datos que se utilizan para su construcción. De todas maneras, si una empresa guarda en una base de datos los valores de las métricas de los diagramas que va construyendo en distintos proyectos, podría construir como demostramos en este artículo, un modelo "ad hoc" y de utilidad para su empresa.

Los resultados obtenidos nos hacen pensar que las métricas propuestas podrían servir como indicadores de la facilidad de entendimiento de los diagramas de estados UML, lo que será de gran utilidad a la hora de mantener tales diagramas. También dichas métricas pueden servir a los diseñadores para comparar distintas alternativas de diseño.

Aunque los resultados son alentadores, somos conscientes de que se debe mejorar nuestro es estudio en dos sentidos:

1) Con respecto a los datos utilizados
2) Con respecto a la técnica utilizada para construir el modelo de predicción

Por ello, por un lado debemos replicar el experimento con profesionales y además ver la utilidad de las métricas en proyectos reales. También considerar, en la experimentación futura, no sólo la facilidad de entendimiento, sino otras características que influyen en el mantenimiento, como la facilidad de análisis y modificación. En lo que se refiere al modelo de predicción, existen ciertos aspectos que mejorar. La utilización de algoritmos como el Fuzzy C-Means [4], las Redes de Kohonen Borrosas [5] o algoritmos de soft-clustering en general, permitirían incrementar la potencia en la resolución de problemas. Estos algoritmos podrían hacer que el clustering y la construcción del modelo de predicción se pudieran realizar de una sola vez, tomando las decisiones sobre el número de prototipos antes de su realización. Además la potencia que aportan estos algoritmos posibilita una mejor manipulación de grandes volúmenes de datos.

Agradecimientos

Este trabajo de investigación es parte del proyecto MESSENGER (PCC-03-003-1) financiado por la Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla - La Mancha y del proyecto CALIPO, financiado por la Dirección General de Investigación del Ministerio de Ciencia y Tecnología (TIC2003-07804-C05-03).

Referencias

[1] Atkinson C. and Kuhne 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., Thevenod-Fosse P. (1999). "Empirical Studies of Object-Oriented Artifacts, Methods, and Processes: State of the Art and Future Directions". *Empirical Software Engineering*, 4(1), 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. and 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] Derr, K. (1995). *Applying OMT*. SIGS Books, Prentice Hall. New York.

[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., and 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., Pfleeger 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, N°4, pp. 308-320.

[19] MDA- The OMG Model Driven Architecture (2002). Available: http://www.omg.org/mda/. August 1st, 2002.

[20] Mendes E., Watson I., Triggs C., Mosley N. y Counsell S. (2002). "A comparison of development effort estimation techniques for web hypermedia applications". *Eight IEEE Symposium of Software Metrics (METRICS'02)*. IEEE Computer Society Press.

[21] Miranda D., Genero M. y Piattini M. (2003). "Empirical validation of metrics for UML statechart diagrams". *Fifth International Conference on Enterprise Information Systems (ICEIS 03)*, 1, pp. 87-95.

[22] Object Management Group (2001). *UML Revision Task Force*. OMG Unified Modeling Language Specification, v. 1.4, document formal/01-09-67.

[23] Olivas J. (2000). *Contribución al Estudio Experimental de la Predicción basada en Categorías Deformables Borrosas*. Tesis Doctoral, Universidad de Castilla La Mancha, España.

[24] 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.

[25] 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.

[26] 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.

[27] Wohlin C., Runeson P., Höst M., Ohlsson M., Regnell B., Wesslén A. (2000). *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers.

[28] Zadeh, L. A. (1982). "A note on prototype set theory and fuzzy sets". *Cognition* 12, pp. 291- 297.