Construcción de un Modelo Borroso de Predicción del Tiempo de Mantenimiento de Diagramas de Clases UML

Marcela Genero[1], Francisco Romero[2], José Olivas[1], Mario Piattini[1]
[1]Departmento de Informática , Universidad de Castilla-La Mancha
Paseo de la Universidad, 4, 13071, Ciudad Real (España)
{Marcela.Genero ,JoseAngel.Olivas, Mario.Piattini,}@uclm.es
[2]Soluziona Consultoría y Tecnología
Pedro Muñoz,1, 13004, Ciudad Real (España)
fromero@uf-isf.es

Resumen

En este trabajo se presenta una aplicación de la lógica borrosa en el ámbito del descubrimiento de conocimiento y la predicción en la Ingeniería del Software. Concretamente se utiliza el Descubrimiento de Conocimiento Prototípico Borroso con dos objetivos; en primer lugar para encontrar prototipos para caracterizar los diagramas de clases UML de acuerdo a su facilidad de mantenimiento, partiendo de su complejidad estructural y tamaño expresados mediante métricas; y por otro lado para obtener un modelo de predicción del tiempo de mantenimiento de los diagramas de clases. Los datos objeto de este estudio han sido obtenidos a partir de dos experimento controlados, en el cuales se obtuvieron los valores de las variables independientes, las métricas, y por otro lado, los valores de la variable independiente, el tiempo de mantenimiento. Para lograr los objetivos anteriormente mencionados se han utilizado diversas técnicas como el clustering borroso, funciones de agregación y prototipos deformables borrosos, enmarcadas dentro del proceso denominado Descubrimiento de Conocimiento Prototípico Borroso. La utilización de estas técnicas supone una aportación substancial a la significación del modelo de predicción haciéndolo a la vez más útil y comprensible al usuario. Después de validar el modelo de predicción se puede concluir que el modelo obtenido es en cierta manera válido ya que el 88% de los valores estimados del tiempo de mantenimiento tienen al menos un 75% de exactitud, aunque es prematuro considerarlo como definitivo sin haberlo validado con datos referentes a proyectos reales.

Palabras claves: descubrimiento de conocimiento prototípico borroso, lógica borrosa, modelo de predicción, diagramas de clases UML, métricas, mantenibilidad, complejidad estructural, tamaño.

1. Introducción.

En la Ingeniería del Software es bien sabido que las características que hacen a la calidad de los productos software orientados a objetos (OO), como la mantenibilidad, debe ser garantizada desde las etapas iniciales de su ciclo de vida [6,13,32]. Para ello es necesario evaluar la mantenibilidad de los productos obtenidos en dichas etapas, como por ejemplo los diagramas de clases UML[1]. Pero la mantenibilidad es un atributo externo de la calidad, que sólo puede ser medido una vez que el producto está terminado. Por ello como menciona [14] es necesario contar con indicadores de la mantenibilidad basados en características estructurales de los diagramas de clases (complejidad estructural, tamaño, etc.), que lógicamente se puedan medir en etapas iniciales. Siguiendo esta idea hemos definido en [14,16] un conjunto de métricas para medir la complejidad estructural y el tamaño de los diagramas de clases UML (ver tabla 1) y hemos corroborado mediante varios estudios empíricos (dos de los cuales se presentan en la sección 2) que la complejidad estructural y el tamaño de los diagramas de clases influye en gran medida sobre su tiempo de mantenimiento.

[1] Nos centramos en diagramas de clases UML, ya que actualmente UML es el lenguaje estándar de modelado OO [24].

Tipos de Métricas	Definición de las Métricas
Métricas de Tamaño	Número de Clases (NC). Número total de clases.
	Número de Atributos (NA). Número total de atributos..
	Número de Métodos (NM). Número total de métodos.
Métricas de complejidad estructural	Número de Asociaciones (NAssoc). Número total de asociaciones. .
	Número de Agregaciones (NAgg). El número total de relaciones de agregación existentes en el diagrama de clases (cada par de entidades en una relación de agregación).
	Número de Dependencias (NDep). Número total de relaciones de dependencia.
	Número de Generalizaciones (NGen). Número de generalizaciones existentes en un diagrama de clases (cada relación de padre-hijo en una relación de generalización).
	Número de jerarquías de Generalización (NGenH). Número total de jerarquías de generalización existentes en un diagrama de clases.
	Máximo DIT (MaxDIT). Es el máximo valor de DIT obtenido para cada clase del diagrama de clases. El valor de DIT para una clase dentro de una jerarquía de generalización es la longitud de la ruta más larga desde la raíz de la jerarquía a las hojas.
	Máximo HAgg (MaxHAgg). Es el máximo valor de HAgg obtenido para cada clase del diagrama. El valor de HAgg para una clase dentro de una jerarquía de agregación es la longitud de la ruta más larga desde la raíz de la jerarquía a las hojas.

Tabla 1. Métricas para la complejidad estructural de diagramas de clases UML

Nuestro enfoque sobre cómo la complejidad estructural y el tamaño como propiedades internas de los diagramas de clases están potencialmente relacionadas con su mantenibilidad surgió a partir de trabajos similares realizados en el campo de la ingeniería del software empírica, en los cuáles estas propiedades se mostraron como los mayores determinantes de características externas de la calidad, como la mantenibilidad [8,17].

Como es bien sabido en el campo de la medición del software [6,9,13] para que las métricas "early" que definimos sean útiles, es necesario que sirvan para predecir algún atributo externo de la calidad, como la mantenibilidad. Por ello sería necesario construir un modelo de predicción del tiempo de mantenimiento de los diagramas de clases UML basado en sus valores. Esto es lo que nos motivó a realizar el trabajo que presentamos en este artículo.

Para la construcción de dicho modelo de predicción utilizaremos el proceso de Descubrimiento de Conocimiento Prototípico Borroso[2] (DCPB) [26] que se ha utilizado exitosamente en otros dominios como la predicción de incendios forestales [26,27], el diagnóstico médico [21], el análisis financiero [25], etc. Recientemente se han presentado diversos artículos sobre la aplicación del DCPB en la búsqueda en Internet [28] y la organización y gestión del correo electrónico y la monitorización de conversaciones [30]. Dado los resultados de la aplicación del DCPB en diversos ámbitos creemos relevante utilizarlo en otro ámbito como es el de la Ingeniería del Software.

El DCPB es una ampliación del proceso clásico de KDD [11] 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 [26,37]. La utilización de la Lógica Borrosa [36] 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

[2] Todas las actividades del DCPB han sido automatizadas a través de una herramienta [31].

predicciones en cuanto a situaciones reales y también tomar decisiones a partir de dichas predicciones. Además se utilizan en el DCPB otras técnicas como el clustering borroso y las funciones de agregación [10], facilitando la generación de modelos estructurados, significativos y fácilmente actualizables:

- Estructuración: La representación de los prototipos en forma de números borrosos acompañados de un marco de características que definen a cada prototipo determina una forma estructurada y formal de expresión de los resultados obtenidos.
- Significación: la utilización de la noción de Prototipo Deformable Borroso supone la construcción de un modelo de predicción más rico y cercano a la realidad que la que supondría la utilización la noción clásica de prototipo en psicología cognitiva.
- Fácil Actualización: La capacidad de deformación de los prototipos y la definición completa del proceso de construcción define un proceso de actualización sencillo y completo.

En este trabajo se parte de una colección de datos obtenida a partir de un experimento controlado y se lleva a cabo un proceso de DCPB para la obtención de prototipos borrosos que permitan caracterizar de forma completa los diagramas de clases de acuerdo con su facilidad de mantenimiento. A su vez estos prototipos serán la base para la construcción de un modelo de predicción del tiempo de mantenimiento de los diagramas de clases.

Este trabajo está organizado de la siguiente manera: en la sección 2 se presentan dos experimentos controlados de los que se obtuvieron los datos de medición empíricos que se utilizarán posteriormente como fuente de datos en el DCPB. A continuación se describen las etapas del DCPB: en la sección 3 se presenta la transformación de los datos obtenidos de los experimentos. La construcción de los prototipos y el modelo de predicción del tiempo de mantenimiento de los diagramas de clases UML se presenta en la sección 4. La sección 5 presenta la validación del modelo de predicción. Finalmente las conclusiones y las líneas de investigación que surgen a partir de éste trabajo se presentan en la sección 6.

2. Fuentes de datos.

En esta sección describiremos dos experimentos controlados que se realizaron teniendo en cuenta algunos consejos suministrados por expertos en la Ingeniería del Software empírica como en los trabajos de Wholin et al. [35], Perry et al. [29], Briand et al. [7] y Kitchenham et al. [20]. Los datos obtenidos en el primer experimento se utilizarán para definir los prototipos borrosos que permiten agrupar los diagramas de clases UML de acuerdo a su tiempo de mantenimiento y a continuación mediante la deformación de dichos prototipos se obtendrá un modelo de predicción del tiempo de mantenimiento de los diagramas de clases UML (ver sección 5). Los datos del segundo experimento se utilizarán para validar el modelo de predicción obtenido (ver sección 6).

2.1 Primer experimento.

Este experimento se llevó a cabo con el objetivo de averiguar si existe correlación entre la complejidad estructural y el tamaño de los diagramas de clases (variables dependientes) y la mantenibilidad de los mismos (variable independiente).

La variable dependiente se midió a través de un conjunto de métricas definidas por Genero et al. [14,16] (ver tabla 1).

Los sujetos fueron 24 estudiantes del tercer año de Ingeniería Informática de la Universidad de Castilla-La Mancha que tenían una experiencia media en el modelado OO usando UML.

El experimento consistió en 9 diagramas de clases UML sobre distinto dominio de aplicación, pero lo suficientemente general para ser fácilmente comprendido por los sujetos. Las tareas a realizar fueron de dos tipos:

- Contestar el cuestionario adjunto a cada diagrama de clases, que consistía en cinco preguntas, y anotar el tiempo que tardaban en contestarlas, al que llamamos tiempo de entendibilidad (medida de la variable dependiente expresada en minutos y segundos). Este tiempo refleja el tiempo que tardan los sujetos en entender cada diagrama.
- Modificar cada diagrama de clases de acuerdo a cinco nuevos requerimientos y anotar el tiempo utilizado, al que llamamos tiempo de mantenimiento (medida de la variable dependiente expresada en minutos y segundos). Este tiempo refleja el tiempo que tardan los sujetos en modificar cada diagrama.

Como resultado obtuvimos los datos del cálculo de las métricas de la complejidad estructural y el tamaño de cada diagrama (ver tabla 2) y los tiempos de entendibilidad y de mantenimiento. Adicionalmente se han comprobado las tareas realizadas para establecer los siguientes índices de corrección, completitud:

- Coeficiente de completitud en la modificación (modificaciones realizadas / modificaciones requeridas).
- Coeficiente de corrección en la modificación(modificaciones realizadas correctamente / modificaciones requeridas).
- Coeficiente de correción en la entendibilidad: (preguntas contestadas correctamente/preguntas).

Diagrama	NC	NA	NM	NAssocc	NAgg	NDep	NGen	NAggH	NGenH	MaxDIT	MaxHAgg
1	13	30	47	11	5	2	3	2	1	2	1
2	22	43	56	11	6	3	15	2	7	4	3
3	9	24	25	6	1	1	5	1	2	1	2
4	7	17	26	1	4	0	3	2	1	2	1
5	9	24	40	2	3	1	5	1	2	1	1
6	11	26	36	5	0	2	7	0	4	0	3
7	52	76	35	15	23	8	17	3	6	7	4
8	22	62	39	7	12	0	2	4	1	1	1
9	7	23	31	3	1	1	2	1	1	1	1

Tabla 2. Valores de las métricas para cada diagrama de clases del primer experimento

2.2 Segundo experimento.

Ya que este experimento es muy similar al anterior y se presentó detalladamente en [15] sólo presentaremos las características que lo diferencian del primero.

- Los sujetos fueron 16 estudiantes del quinto año de la Ingeniería Informática de la Universidad de Castilla-La Mancha y 7 profesores del área de Ingeniería de Software. Tanto los profesores como los alumnos tenían suficiente experiencia como para realizar las tareas que se pedían en el experimento.
- A los sujetos se les entregó 9 diagramas de clases UML sobre distinto dominio de aplicación, sobre los cuáles debían realizar tareas de mantenimiento teniendo en cuenta tres nuevos requerimientos que se le solicitaban en un documento adjunto a cada diagrama. Además debían anotar el tiempo empleado en dichas tareas, lo que denominamos tiempo de mantenimiento (medida de la variable dependiente).
- Como resultado obtuvimos los datos del cálculo de las métricas de la complejidad estructural y el tamaño de cada diagrama (ver tabla 3) y los tiempos de mantenimiento que cada sujeto a necesitado sobre cada uno de los diagramas.

Diagrama	NC	NA	NM	NAssoc	NAgg	NDep	NGen	NAggH	NGenH	MaxDIT	MaxHAgg
1	7	11	22	1	0	0	5	0	1	2	0
2	8	12	31	1	6	0	1	1	1	1	2
3	3	17	24	2	0	0	0	0	0	0	0
4	10	12	21	15	3	0	0	2	0	0	1
5	9	19	29	3	3	0	3	3	1	2	1
6	7	16	7	6	0	0	0	0	0	0	0
7	23	33	66	4	5	2	16	2	3	3	3
8	20	30	65	6	5	0	14	4	3	3	2
9	23	65	80	20	3	2	3	3	1	2	3

Tabla 3. Valores de las métricas para cada diagrama de clases del segundo experimento

3. Transformación de los datos.

En esta etapa el objetivo es transformar los datos que se tienen en datos que puedan ser analizados. Para ello se han realizado fundamentalmente dos pasos:

- Agrupación de los resultados del experimento.

- Selección de colecciones de datos.

3.1. Agrupación de los resultados del experimento.

Por un lado se tiene la tabla con los datos de las métricas de los diagramas de clases (ver tabla 1). Y por otro lado se tiene los tiempos de entendibilidad y de mantenimiento. A partir de los datos sobre éstos tiempos se hace una agrupación para obtener los tiempos medios, mínimos y máximos de entendibilidad y mantenimiento de cada diagrama, filtrando los registros que no cumplan ciertas condiciones de completitud, corrección (>=0.9) (ver tabla 4).

Diagrama	Tiempo de mantenimiento promedio (segundos)	Tiempo de mantenimiento mínimo (segundos)	Tiempo de mantenimiento máximo (segundos)
1	269.38	50	580
2	353	353	353
3	410	390	430
4	174.615	36	305
5	244	215	273
6	305.9375	120	535
7	523	261	785
8	305.166	165	567
9	234.391	80	498

Tabla 4. Tiempos obtenidos en el proceso de transformación

De esta forma obtenemos una tabla de 9 filas y 4 columnas que se combina con la tabla de métricas (ver tabla 2) mediante la columna que representa a cada diagrama. Estos serán entonces los datos que se utilizarán en la sección para construir el modelo de predicción.

3.2. Selección de colecciones de datos.

Para llevar a cabo el proceso de DCPB es necesario dividir los atributos del conjunto de datos en tres colecciones no disjuntas:

- Datos de Modelado: Para esta colección deben ser elegidos los datos que puedan definir mejor a los prototipos, sin tener en cuenta si se dispone de ellos a priori o no. En este estudio se han elegido las métricas de complejidad estructural y tamaño de los diagramas de clases UML del primer experimento.
- Datos de Prototipaje: Esta colección debe estar formada por los datos de los que se dispone a priori. Se utilizan para realizar la representación formal de los prototipos en forma de números borrosos. En este estudio se han elegido las métricas de complejidad estructural y tamaño de los diagramas de clases UML del primer experimento.
- Datos de Predicción: En este caso serán los datos referentes al tiempo de mantenimiento obtenido en el primer experimento.
- Datos de Validación: Se consideraron los datos relativos a las métricas de complejidad estructural y tamaño, y el tiempo de mantenimiento del segundo experimento.

4. Construcción del modelo de predicción.

Esta es la fase del DCPB en la que se persiguen dos objetivos:

1. La búsqueda y extracción de forma automática de prototipos borrosos para caracterizar los diagramas de clases a través de su facilidad de mantenimiento expresada como el tiempo de mantenimiento.
2. La obtención de un modelo de predicción del tiempo de mantenimiento de los diagramas de clases realizando la deformación de los prototipos borrosos.

4.1. Obtención de los prototipos.

Con el objetivo de detectar las relaciones entre los diagramas de clases para poder obtener después si ellos tienen un tiempo de mantenimiento bajo, medio o alto se lleva a cabo un proceso de clustering jerárquico (basada en la idea de Repertory Grids, introducida por el psicólogo Kelly en 1955[1].

El conjunto de elementos objetos de análisis serán los datos que denominamos "datos de modelado" o sea 9 filas con datos sobre las métricas de complejidad estructural y tamaño de los diagramas de clases UML del primer experimento (ver tabla 2).

La parrilla o repertory grid a construir será evaluada a siete niveles con las construcciones distribuidas de forma automática utilizando intervalos por densidad. Posteriormente se ha construido una matriz de similaridad de 9 x 9 elementos cuyos valores sobre la diagonal representan los grados de similitud entre los diagramas. Convirtiendo estos valores a porcentajes y desencadenando el algoritmo de clustering jerárquico se consiguen los resultados mostrados en el dendograma de la figura 1.

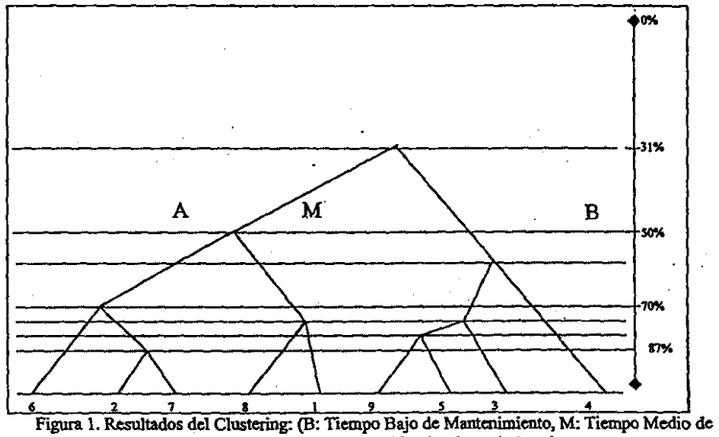

Figura 1. Resultados del Clustering: (B: Tiempo Bajo de Mantenimiento, M: Tiempo Medio de Mantenimiento, A: Tiempo Alto de Mantenimiento).

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.

Prototipos	Diagramas
B: Tiempo Bajo de Mantenimiento	3,5,9,4
M: Tiempo Medio de Mantenimiento	1,8
A: Tiempo Alto de Mantenimiento	2,7,6

Tabla 5. Diagramas agrupados según el prototipo al que pertenecen

4.2. Definición paramétrica de los prototipos.

Para la definición paramétrica de los prototipos (mediante frames o marcos) se tiene como datos de entrada la colección de datos que llamamos "datos de predicción" y a qué prototipo corresponde cada diagrama según los resultados obtenidos en la fase de clustering (ver tabla 5). Los algoritmos utilizados para la definición paramétrica de los prototipos son funciones de agregación estándar. Los prototipos se definen mediante la media, el máximo y el mínimo de los valores de la colección de datos de predicción.

En este caso, se obtiene la media, el mínimo y el máximo del tiempo total de mantenimiento de los diagramas sometidos a estudio, quedando los prototipos definidos como muestra la tabla 6.

	Tiempo
A: Tiempo Alto de Mantenimiento	
Media	6 min. 30 seg.
Máximo	10 min. 45 seg.
Mínimo	1 min. 50 seg.
M: Tiempo Medio de Mantenimiento	
Media	4 min. 50 seg.
Máximo	9 min. 30 seg.
Mínimo	1 min. 20 seg.
B: Tiempo Bajo de Mantenimiento	
Media	4 min. 20 seg.
Máximo	8 min. 45 sec.
Mínimo	1 min. 10 seg.

Tabla 6. Definición paramétrica de los prototipos

4.3. Representación de los prototipos borrosos.

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 clases con cada uno de ellos (ver figura 2). La utilización de números borrosos triangulares permite que para la representación sea únicamente necesario conocer su centro (denominado centro en la tabla 7) y el tamaño de la base del triángulo (denominado a y b en la tabla 7).

En el proceso de DCPB la definición formal de los prototipos en forma de números borrosos se obtiene mediante la normalización, realizada de la siguiente manera $x_n' = \frac{x_n - x_{min}}{x_{max} - x_{min}}$, y la agregación mediante la media de los datos de la colección de datos de prototipaje.

Prototipos	Diagramas	a	centro	b
B: Tiempo Bajo de Mantenimiento	3,5,9,4	0.00	0.12	0.54
M: Tiempo Medio de Mantenimiento	1,8	0.00	0.31	0.73
A: Tiempo Alto de Mantenimiento	2,7,6	0.16	0.58	1

Tabla 7. Definición borrosa de los prototipos

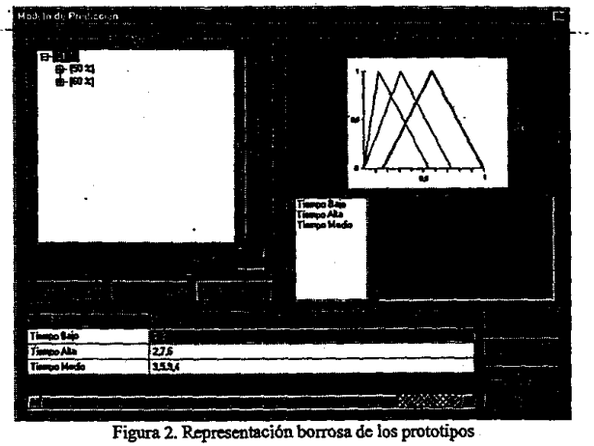

Figura 2. Representación borrosa de los prototipos

El mecanismo básico del modelo de predicción del tiempo de mantenimiento es la deformación de estos prototipos borroso para que describan una situación real nueva. De ahí el nombre de proptotipos "deformables" borrosos.

4.4. Ejemplo de aplicación del modelo de predicción.

A modo ilustrativo mostraremos en esta sección como predecir el tiempo de mantenimiento para un nuevo diagrama de clase.

Dado los valores de las métricas de un nuevo diagrama de clases (datos de prototipaje) se adapta la definición paramétrica de cada prototipo para que se pueda predecir el tiempo de mantenimiento (en forma de intervalo) de este nuevo diagrama.

El proceso es el siguiente:

1. Normalización de los valores medidos mediante los índices de normalización asociados al modelo de predicción obtenido. Para ello se utiliza la misma fórmula que para la definición de los números borrosos y con los mismos coeficientes de mínimo y máximo.
2. Agregar mediante la obtención de la media de los valores normalizados anteriormente (valor borroso).
3. Obtener los grados de pertenencia a los prototipos representados mediante los números borrosos construidos en el modelo de predicción. La obtención de un grado de afinidad con los prototipos preestablecidos proporciona información relevante para evaluar la situación presentada.
4. Asimilar la definición del caso actual al de sus Prototipos Deformables Borrosos afines. Con esta finalidad se modifica la Definición que se ha obtenido mediante DCPB de los Prototipos Deformables Borrosos en base al grado de compatibilidad del caso real con ellos. Aplicando la definición introducida en el concepto de Prototipo Deformable Borroso [26], la caracterización del caso real propuesto será la siguiente:

$$C_{real}(w_1 \ldots w_n) = \{ \quad \mu p_i(v_1 \ldots v_n) \}$$

Donde:

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

Por ejemplo, dados los valores de las métricas correspondientes a un nuevo diagrama de clases UML:

NC	NA	NM	NAssoc	NAgg	NDep	NGen	NAggH	NGenH	HaggMax	MaxDIT
21	30	50	10	6	3	16	2	3	4	2

se obtienen sus valores normalizados:

NC	NA	NM	NAssoc	NAgg	NDep	NGen	NAggH	NGenH	HAggMax	MaxDIT
0.4	0.39	0.89	0.67	0.26	0.38	0.94	0.5	0.43	0.57	0.5

Determinando la media de estos valores (0.54) se pueden obtener los grados de afinidad existentes con los prototipos que conforman el modelo de predicción (ver tabla 8):

Prototipos	Afinidades
B: Tiempo Bajo de Mantenimiento	0
M: Tiempo Medio de Mantenimiento	0.45
A: Tiempo Alto de Mantenimiento	0.9

Tabla 8. Afinidad del ejemplo con los prototipos

El prototipo más similar es "Tiempo Alto de Mantenimiento" con un grado de afinidad de 0.9, también existe una compatibilidad de 0.45 con el prototipo "Tiempo Medio de Mantenimiento". Para aprovechar la afinidad con ambos prototipos calcularemos los valores del tiempo de mantenimiento medio, mínimo y máximo (ver tabla 9) utilizando un proceso de combinación lineal . Así se obtendrá información sobre el intervalo en el que se moverá el tiempo de mantenimiento para el nuevo diagrama.

	Tiempo de Mantenimiento
Media	5 min. 50 seg.
Máximo	9 min. 40 seg.
Mínimo	1 min. 40 seg.

Tabla 9. Valores obtenidos utilizando el modelo de predicción

5. Validación del modelo de predicción.

Las técnicas más utilizadas para la evaluar el poder o la exactitud de predicción de los modelos de predicción son las siguientes [22].

- Magnitud del Error Relativo (MRE) [19]:

$$MRE_i = \frac{|TiempoReal DeMantenimiento - TiempoDeMantenimientoPredecido|}{TiempoReal DeMantenimiento}$$

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

- Media de la Magnitud del Error Relativo (*MMRE*) [33]: se calcula como la media de todos los MREs

$$MMRE = \frac{1}{n}\sum_{i=1}^{i=n} \frac{|TiempoReal DeMantenimiento - TiempoDeMantenimientoPredecido|}{TiempoReal DeMantenimiento}$$

Predicción a nivel n (Pred(n)) [34]: mide el porcentaje de estimaciones que están dentro de un n% de los valores reales. Según algunas sugerencias [22] el 25% es un valor recomendable para n. Lo que significa que un buen modelo de predicción tendrá una precisión del 75%.

Siguiendo los pasos mostrados en la sección 4.5 para validar el modelo de predicción realizaremos las siguientes tareas (ver tabla 9):

1. Normalización y agregación de los valores de la métricas. Se obtiene el Valor X.
2. Calcular el valor de las afinidades a los prototipos. Af A (afinidad al Tiempo Bajo de Mantenimiento), Af B (afinidad al Tiempo Medio de Mantenimiento), Af C (afinidad al Tiempo Alto de Mantenimiento).

3. Se calculan el tiempo medio correspondiente (Media Estimada) a la predicción tomando como referencia los prototipos más afines. Es decir realizando la combinación lineal de varios prototipos explicada en el apartado anterior.
4. Se comparan con los tiempos reales calculando MRE.
5. Se calcula MMRE y MdMRE.
6. Se calcula Pred(25%).

valor X	Af A	Af B	Af C	Media Estimada	Media Real	MRE
0.15	0.93	0.48	0	4	2.696	0.48368
0.2	0.81	0.64	0.01	3.564	3.043	0.17121
0.08	0.66	0.26	0	2.904	2.913	0.00309
0.23	0.73	0.74	0.16	3.589	4.087	0.12185
0.27	0.64	0.87	0.26	4.2195	3.345	0.26143
0.08	0.66	0.26	0	2.904	2.739	0.06024
0.53	0.01	0.47	0.88	5.2	4.87	0.06776
0.53	0.01	0.57	0.88	5.2	4.826	0.0775
0.59	0	0.33	0.98	6.681	7.087	0.05729

Tabla 10. Datos de la validación del modelo de predicción

La magnitud media del error en este experimento (MMRE) es 0.145 y el valor de la mediana (MdMRE) es 0.0775. El valor obtenido para Pred(25%) es un 88%. De esto podemos decir que el modelo de predicción obtenido es válido ya que el 88% de lo valores estimados tiene al menos un 75% de precisión.

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

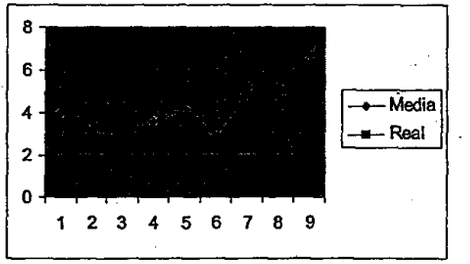

Figura 3. Valores de la predicción vs. valores reales

Además de validar el modelo con los valores medios de mantenimiento obtenidos del segundo experimento, se ha realizado la validación con los valores individuales obtenidos para cada diagrama y para cada sujeto, obteniendo un valor de MMRE de 0.6081 y de Pred(25%) de 71%. Lo cual corrobora en cierta manera los resultados previos, y nos lleva a concluir que los valores obtenidos a partir del modelo de predicción obtenido pueden considerarse precisos y fiables.

6. Conclusiones.

En este trabajo se ha llevado a cabo el proceso de Descubrimiento de Conocimiento Prototípico Borroso [26] sobre los datos de un experimento controlado con el objetivo de poder construir un modelo de predicción del tiempo de mantenimiento de un diagrama de clases UML a partir de los valores de métricas calculadas a priori sobre la complejidad estructural y el tamaño de éstos. Se ha explicado todo el proceso de forma detallada y además se ha validado el modelo de predicción construido, obteniendo que el 88% de los valores estimados del tiempo de mantenimiento tienen al menos un 75% de precisión. Los resultados obtenidos son prometedores aunque somos conscientes que es necesario validar el modelo de predicción obteniendo datos

de proyectos reales realizados en entornos industriales. De dicha validación y utilizando el conocimiento de expertos puede que sea necesario adaptar el modelo y así obtener un modelo preciso y estable. Una vez llegado a este punto se podrá asegurar la validez empírica de las métricas definidas en [15,16] y se podrá utilizar el modelo de predicción para tomar mejores decisiones en las etapas iniciales del ciclo de vida de un producto software que, como mencionan [13], debe ser el principal objetivo de cualquier proyecto de medición que se precie de útil.

Técnicamente, existen ciertos aspectos que mejorar. La utilización de algoritmos como el Fuzzy C-Means [2], las Redes de Kohonen Borrosas [3] o algoritmos de soft-clustering en general, permitirían incrementar la potencia en la resolución de problemas de predicción. 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 posibilitan una mejor manipulación de grandes volúmenes de datos.

Agradecimientos.

Este trabajo de investigación es parte de los proyectos DOLMEN (TIC 2000-1673-C06-06) y CIPRESES (TIC 2000-1362-C02-02) financiados por la Subdirección General de Proyectos de Investigación - Ministerio de Ciencia y Tecnología.

Referencias.

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

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

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

[4] H. Bremermann, "Pattern Recognition", *H. Bossel: Systems Theory in the Social Sciences*, Birkhäuser Verlag, 1976, pp. 116 - 159..

[5] L. Briand y J. Wüst, "Empirical studies of quality models", *Advances in Computers*, 59, 2002, pp. 97-166.

[6] L. Briand, S. Morasca S. and Basili V., "Defining and Validating Measures for Object-Based high-level design", *IEEE Transactions on Software Engineering*, 25(5), 1999, pp. 722-743.

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

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

[9] L. Briand, K. El Emam y S. Morasca, "Theoretical and empirical validation of software product measures", Technical Report ISERN-95-03, *International Software Engineering Research Network*, 1995.

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

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

[12] N. Fenton y M. Neil, "Software Metrics: a Roadmap", *Future of Software Engineering*, Ed. Anthony Finkelstein, ACM, 2000, pp. 359-370.

[13] Fenton N. y Pfleeger S., *Software Metrics: A Rigorous Approach*, 2nd. edition. London, Chapman & Hall, 1997.

[14] M. Genero, "Defining and Validating Metrics for Conceptual Models", *Tesis Doctoral*, Universidad de Castilla-La Mancha -Ciudad Real., España, 2002.

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

[16] M. Genero, M. Piattini y C. Calero, "Early Measures For UML class diagrams", *L'Objet*, 6(4), Hermes Science Publications, 2000, pp. 489-515.

[17] R. Harrison, S. Counsell y R. Nithi,, "Experimental Assessment of the Effect of Inheritance on the Maintainability of Object-Oriented Systems", *The Journal of Systems and Software*, 52, 2000, pp. 173-179.

[18] ISO/IEC 9126-1.2.,*Information technology- Software product quality* - Part 1: Quality model, 2001.

[19] C. Kemerer, "An empirical validation of software cost estimation models", *Communications of the ACM*, 30(5), 1987, pp. 416-429.

[20] B. Kitchenham, S. Pflegger, L. Pickard, P. Jones, D. Hoaglin, K. El-Emam y J. Rosenberg, "Preliminary Guidelines for Empirical Research in Software Engineering", *IEEE Transactions of Software Engineering*, 28(8), 2002, pp. 721-734.

[21] R. Lago, J. Olivas, J. Megido, C. Grávalos y J. Pérez, "INFEDEC 2.1: Sistema de ayuda en la decisión de diagnóstico y tratamiento de Enfermedades Infecciosas", Actas de ESTYLF'96: VI Congreso Español sobre Tecnologías y Lógica Fuzzy, 1996, pp. 271 - 275.

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

[23] Myrtveit y E. Stensrud, "A controlled experiment to assess benefits of estimating with analogy and regression models", *IEEE Transactions on Software Engineering*, 25(4), 1999, pp. 510-525.

[24] Object Management Group, *UML Revision Task Force*, OMG Unified Modeling Language Specification, v. 1.3, document ad/99-06-08, 1999.

[25] J. Olivas, J. Ordax y P. Ocaña, "Lógica Borrosa en Internet: Implementación en JAVA de Sistemas Expertos de Formación", 3º Simpósio de Investigação e Desenvolvimento de Software Educativo, 1998. pp. 19.

[26] J. Olivas, "Contribución al Estudio Experimental de la Predicción basada en Categorías Deformables Borrosas", *Tesis Doctoral*, Universidad de Castilla La Mancha, Ciudad Real, España, 2000.

[27] J. Olivas y F. Romero F., "FPKD. Fuzzy Prototypical Knowledge Discovery. Application to Forest Fire Prediction". *SEKE'2000*, Knowledge Systems Institute, Chicago, Ill. USA, 2000, pp. 47 - 54.

[28] J. Olivas, P. Garcés y F. Romero, "FISS: Application of Fuzzy Technologies to an Internet Meta-Searcher", *Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS)* Proceedings, IEEE, 2002, pp. 140 - 145.

[29] D. Perry, A. Porter A. y L. Votta, "Empirical Studies of Software Engineering: A Roadmap", *Future of Software Engineering*. Ed:Anthony Finkelstein, ACM, 2000, pp. 345-355.

[30] F. Romero, J. Olivas y P. Garcés P., "Utilización de la Lógica Borrosa para la Mejora de la Calidad de Servicio de los Agentes Inteligentes en Internet", *Actas Congreso Español de Tecnología y Lógica Fuzzy (ESTYLF 2002)*, 2002, pp. 105-110.

[31] F. Romero,"Herramienta para la Extracción de Conocimiento Prototípico Borroso", *Proyecto Fin de Carrera*, Escuela Superior de Informática de Ciudad Real, Universidad de Castilla – La Mancha, 2001.

[32] N. Schneidewind, "Body of Knowledge for Software Quality Measurement", *IEEE Computer*, 35(2), 2002, pp. 77-83.

[33] M. Shepeprd, C. Shofield y B. Kitchenham B, "Effort estimation using analogy", *18th ICSE*, IEEE Computer Society Press, 1997.

[34] M. Shepperd y C. Schofield, "Estimating Software project efforts using analogies", *IEEE Transactions on Software Engineering*, 23(11), 1997, pp. 736-743.

[35] Wohlin C., Runeson P., Höst M., Ohlson M., Regnell B., Wesslén A., *"Experimentation in Software Engineering: An Introduction"*, 2000, Kluwer Academic Publishers.

[36] L. Zadeh, "Fuzzy sets. Information and control", 1965, pp. 338-353.

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