UMBRALES PARA METRICAS ORIENTADAS A OBJETOS

Pablo Negro [1, 2] pnegro@matba.com.ar
Roxana Giandini [1, 3] giandini@lifia.info.unlp.edu.ar

[1] Facultad en Tecnología Informática, Universidad Abierta Interamericana
[2] Mercado a Término de Buenos Aires SA
[3] LIFIA, Facultad de Informática, Universidad Nacional de La Plata

Resumen

Una aplicación practica de las métricas orientadas a objetos (OO), es predecir que clases tienen una alta probabilidad de contener defectos. Esto tiende a ser significativo dado que, se cree que las métricas orientadas a objetos son indicadores de complejidad psicológica y, las clases que son más complejas son más probables que contengan defectos. Recientemente se ha propuesto una teoría cognitiva la cual sugiere que existe un efecto de umbral para varias métricas OO. Esto significa que las clases OO son fáciles de entender, mientras su complejidad este por debajo del valor de umbral. Por encima del valor de umbral, su comprensión decrece, llevando a una probabilidad de fallas incremental. Acorde a esta teoría, esto sucede debido a que la memoria humana a corto plazo colapsa. Si esta teoría se confirma, proveería un mecanismo que podría explicar la introducción de fallas en sistemas OO, y proveería también una guía practica de cómo diseñas sistemas OO. En este artículo se testea empíricamente esta teoría sobre dos sistemas de mercado electrónico. Se testeó el efecto de umbral sobre la suite de métricas de Chidamber & Kemerer (CK). Se utilizo como variable dependiente la incidencia a fallas. Los resultados indican que no existe un efecto de umbral para las métricas estudiadas, donde la propensión a fallas cambie de ser estable a incrementarse rápidamente. Los resultados son consistentes para ambos sistemas. Por lo tanto, no se puede proveer soporte a la teoría cognitiva presentada.

Palabras Clave: Métricas Orientadas a Objetos, calidad de software, modelos de calidad, Complejidad de software.

1 INTRODUCCION

Se ha realizado una gran cantidad de trabajo, con el propósito de investigar la relación entre las métricas orientadas a objetos (OO), y la propensión a fallas en las clases. Una clase propensa a fallas, se define como aquella que tiene una alta probabilidad de tener un defecto, que cause un error una vez que el sistema ha sido puesto en producción [8].

Una vez validadas, estas métricas pueden servir como indicadores que llevan a identificar clases propensas a errores. Tales clases pueden identificarse para someterlas a acciones específicas de administracion de calidad, tales como inspecciones más intensas y testeo o, más aun, pueden rediseñarse.

Un enfoque operacional que apela a la administracion de la calidad utilizando métricas OO, es el desarrollo de umbrales. Los umbrales se definen como "Valores heurísticas usados para fijar rangos de valores deseables y no deseables de métricas, para el software medido" [7]. Estos valores de umbral se utilizan para identificar anomalías. Por ejemplo podríamos decir que una métrica dada que mide acoplamiento, tiene un valor de umbral de 6; Si el valor de la medición es mayor a 6, entonces podríamos identificar a esa clase como de alto riesgo.

Los umbrales tienen un significado práctico, teórico y metodológico. Es mucho más simple para el equipo de SQA, utilizar umbrales para identificar clases con un alto riesgo potencial. Más aun, Hatton [9] presento el caso de umbrales basado en la teoría cognitiva. Específicamente, utiliza un modelo de memoria humana para sugerir que, las clases más complejas, colapsaran la memoria de corto-plazo, llevando a más defectos. Como disciplina, es importante testear empíricamente la verosimilitud de esta teoría dado que, si se verifica, puede mejorar ampliamente nuestro entendimiento del diseño OO. Si la teoría de umbrales se confirma, proveerá un mecanismo para explicar la introducción de errores en un sistema OO. Finalmente, si ciertamente existen umbrales, entonces los modelos empíricos utilizados para

validar métricas OO, abordaran los datos mucho mejor; Esto resultaría en una mejora en la predictibilidad de clases con alto riesgo.

Una forma práctica de ver los umbrales puede ser la de utilizarlos como una alarma que se dispara cada vez que el valor de una métrica interna excede el valor de umbral.

Al día de hoy, ninguno de los estudios realizados para determinar valores de umbral, han sido testeados empíricamente. En este trabajo, se presenta una técnica estadística para estimar y evaluar umbrales, y aplicarla en un subconjunto de métricas de la suite de Chidamber & Kemerer (CK) [3] [11]. Debe notarse que por razones históricas, las métricas CK son las más referenciadas.

EL Resto del artículo se estructura de la siguiente manera:

La sección 2 presenta un Background sobre la teoría y evidencia de umbrales OO. Se introducen además la suite de métricas estudiadas.

En la sección 3 se especifica el planteamiento del problema tratado, el alcance y definición de las dimensiones de las variables medidas. En la sección 4, se detalla el método de investigación, y las fuentes de datos utilizadas para llevar a cabo el análisis. En la sección 5 se presentan los resultados arrojados por el modelo y la evaluación del efecto de umbral. Finalmente se presentan las conclusiones del análisis, contribuciones, y líneas de trabajo futuro.

2 BACKGROUND

En esta sección se presenta la teoría y evidencia de umbrales OO, y se introduce además la suite de métricas OO estudiadas.

2.1 Teoría y evidencia del efecto de los umbrales.

En la figura 1, se resumen las bases teóricas para el desarrollo de modelos cuantitativos, que relacionan métricas de productos y métricas de calidad externas. Aquí se plantea la hipótesis de que las propiedades estructurales de un componente de software, tales como su acoplamiento, tiene impacto en su complejidad cognitiva. La complejidad cognitiva se define como la carga mental de los individuos que tienen que tratar con el componente, por ejemplo, desarrolladores, testers, inspectores, y mantenedores del sistema [3]. Complejidad cognitiva alta, lleva a que un componente exhiba atributos externos de calidad indeseables, tales como aumento de la propensión a fallas y reducción en la mantenibilidad. Acorde a esta teoría, las métricas OO que afecten la complejidad cognitiva, serán por lo tanto, relacionadas con la propensión a fallas.

Típicamente, propiedades estructurales tales como el acoplamiento y cohesión, se considera que ejercen una influencia significativa sobre la complejidad cognitiva. Por ejemplo, los sistemas compuestos de clases altamente acopladas tienden a ser propensas a error, difíciles de entender y difíciles de mantener. Por otro lado, los sistemas altamente cohesivos y de bajo acoplamiento, tienden a ser menos propensos a error, fáciles de corregir y adaptar a nuevas características.

La teoría expuesta, no plantea ninguna hipótesis respecto de un efecto de umbral específico. Sin embargo, Hatton [9], propone una explicación cognitiva de por que existe un efecto de umbral entre las métricas de complejidad y los errores.

La explicación cognitiva propuesta se basa en el modelo de memoria humana, el cual consiste en una memoria de corto plazo y una memoria de largo plazo. Hatton argumenta que las personas pueden mantener al rededor de 7 +/- 2 piezas de información en un momento dado en la memoria a corto plazo, independientemente del contenido de la información [9] [4]. Luego nota que el contenido en la memoria a largo plazo se encuentra almacenado en forma codificada y que los códigos de recuperación de la información pueden ser confusos bajo determinadas circunstancias. La memoria a corto plazo incorpora además un buffer el cual se refresca a si mismo continuamente. Sugiere que cualquier cosa que entre dentro de la memoria a corto plazo, es mas fácil de entender y menos propenso a error. Las piezas de información que son demasiado grandes o demasiado complejas colapsan la memoria a corto plazo, involucrando el

uso de mecanismos de recuperación codificados más propensos a error, utilizados en la memoria a largo plazo [9] [3] [4].

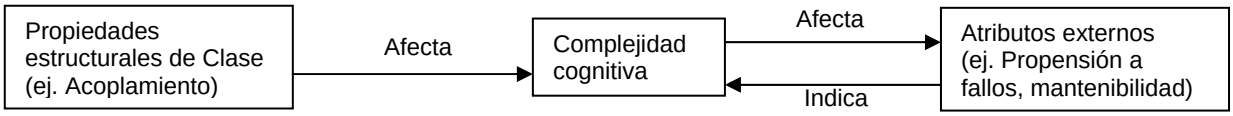

Fig. 1 - Bases teóricas para el desarrollo de métricas para artefactos OO [3]

2.1.1 Umbrales de Tamaño

Hatton [9] argumenta que el concepto de encapsulamiento, central en el paradigma de objetos, nos permite pensar a cerca de un objeto en aislamiento. Si el tamaño de este objeto es lo suficientemente pequeño para que quepa en la memoria a corto plazo, entonces será fácil entender y razonar respecto de este objeto. Los objetos que son muy grandes y colapsan la memoria a corto plazo, tienden a ser más propensos a error. Sin embargo, un estudio reciente ha demostrado que no existe un umbral de tamaño para las clases OO [7]. Por lo tanto, no se consideraran este tipo de umbrales.

2.1.2 Umbrales de Herencia

Se piensa que la herencia dificulta el entendimiento del software OO. Es notorio que la herencia da lugar a descripciones de clases distribuidas. Es decir que, la descripción completa de una clase A, solo puede ensamblarse examinando A tanto como las superclases de A. Dado que diferentes clases se describen en diferentes lugares dentro del código fuente de un programa, no hay un lugar único donde un programador pueda acudir para obtener una descripción completa de una clase. Mientras que este argumento se basa en términos de código fuente, no es difícil generalizar a documentos de diseño. Para entender el comportamiento de un método, se debe rastrear las dependencias de la herencia, lo cual es considerablemente complejo debido al binding dinámico. En un estudio realizado en [7], se propone que si la jerarquía de herencia esta diseñada de manera apropiada, entonces el efecto de distribuir funcionalidad sobre la jerarquía de herencia no va en detrimento del entendimiento. Sin embargo, se ha argumentado que existe una creciente inconsistencia conceptual conforme se viaja por la jerarquía (los niveles inferiores en la jerarquía se caracterizan por ser extensiones inconsistentes o especializaciones de las superclases), por lo tanto las jerarquías de herencia no pueden ser diseñadas adecuadamente en la practica [3].

Acorde a la teoría de Hatton [9], los objetos que son manipulados en la memoria a corto plazo que poseen propiedades heredadas de objetos codificados en la memoria a largo plazo, requieren referenciar a la memoria a largo plazo. Sin embargo, acceder a la memoria a largo plazo rompe el tren de pensamiento y es inherentemente menos exacto. Por lo tanto, acorde a esta teoría, es probable que las clases sean mas propensas a error si utilizan herencia, y la propensión a fallos aumenta conforme la magnitud de la jerarquía de herencia aumenta.

2.1.3 Umbrales de Acoplamiento

Las estrategias OO de limitar las responsabilidades de una clase y reutilizarla en múltiples contextos, resulta en una profusión de pequeñas clases en los sistemas OO. Por ejemplo Chidamber & Kemerer [3] encontraron en dos sistemas estudiados que la mayoría de las clases tienden a tener un pequeño numero de métodos (0-10), lo cual sugiere que la mayoría de las clases son relativamente simples en su construcción, proporcionando abstracción y

funcionalidad especifica. Otro estudio llevado a cabo en Bellcore[1], encontró que la mitad de los métodos tiene menos de 4 líneas SmallTalk o 2 sentencias C++, sugiriendo que las clases están implementadas con métodos pequeños. Muchas clases pequeñas, significa que habrá muchas interacciones entre estas clases [3].

Se cree que esta situación incrementa la complejidad del programa, dado que los programadores tienen que entender el contexto de uso de un método, rastreando a través de la cadena de llamados que lo alcanzan, y rastreando la cadena de métodos que este utiliza. Al haber muchas interacciones, se exacerba el problema del entendimiento.

La teoría de Hatton [9] declara que cuando hay una difusión de la funcionalidad, entonces un objeto en la memoria a corto plazo puede estar haciendo referencia a varios objetos en la memoria a largo plazo. Por lo tanto esto conlleva a dificultades en la comprensión y, acorde a la figura 1, mayor propensión a fallas. Consecuentemente puede argumentarse que, cuando los objetos que interactúan colapsan la memoria a corto plazo, se incrementa la propensión a errores [3] [11].

2.2 Métricas estudiadas

A continuación se presenta un breve sumario de las métricas CK que se han analizado. Se ha excluido explícitamente la métrica de cohesión LCOM, dado que no existe una razón a priori, basada en la teoría presentada, para creer que podría exhibir un valor de umbral.

2.2.1 WMC (Weighted Methods per Class)

Esta métrica puede clasificarse como una métrica de complejidad tradicional. Básicamente es la cuenta de los métodos en una clase. Los desarrolladores de esta métrica, dejan el esquema de peso, como una decisión de implementación. En el presenta estudio, se ha utilizado complejidad ciclomatica. Sin embargo, otros autores no adoptan un esquema de peso. Los métodos en clases ancestras, no se tienen en cuenta [3]. Basado en sus experiencias con proyectos OO en NASA GSFC, Rosenberg [10] define un umbral para WMC de 100.

2.2.2 DIT (Depth in Inheritance Tree)

Esta métrica se define como la longitud del camino mas largo desde la clase raíz en la jerarquía de herencia. Se ha declarado que cuanto mas abajo se va en la jerarquía de herencia, las clases se vuelven mas complejas y, por lo tanto, mas propensas a error [3] [11].

2.2.3 NOC (Number Of Children)

Esta métrica cuenta el número de clases que heredan de una clase dada, es decir, el número de clases en el árbol de herencia hijos de una clase determinada. No se ha determinado un umbral específico para esta métrica [3] [11].

2.2.4 CBO (Coupling Between Object Classes)

Se dice que una clase esta acoplada con otra, si los métodos de una clase, utilizan métodos o atributos de otra y viceversa. En esta definición, los usos pueden significar tipos miembro, parámetros o variables de métodos locales. CBO es el nro. de clases a las cuales una clase dada esta acoplada [3]. Incluye el acoplamiento basado en herencia (acoplamiento entre clases relacionadas vía herencia). Rosenberg [10] define un umbral para CBO de 5.

2.2.5 RFC (Response For a Class)

---

1 El estudio consistió en analizar sistemas en C++ y SmallTalk y entrevistar a los desarrolladores de ambos sistemas. Para el sistema en C++, la métrica de tamaño fue el número de sentencias ejecutables, mientras que para el sistema en SmallTalk el tamaño fue medido en líneas de código.

El conjunto de respuestas de una clase, consiste del conjunto de N métodos de la clase, y el conjunto de métodos invocados directamente por los métodos en N (el conjunto de métodos que pueden ser potencialmente ejecutados en respuesta a un mensaje recibido por la clase.). RFC es el número de métodos en el conjunto de respuesta de la clase [3] [11]. Rosenberg [10] derivo un umbral de 100 para RFC.

3. PLANTEAMIENTO DEL PROBLEMA

La medición es una actividad fundamental a cualquier disciplina de ingeniería, y la ingeniería de software no es la excepción. Típicamente, las métricas son esenciales en la ingeniería de software dado que proveen mecanismos para la medición de la complejidad y calidad, estimando costos y esfuerzo invertido en un proyecto, solo por mencionar algunos aspectos [11].

En una disciplina de ingeniería no podemos tolerar ambigüedades. Si vamos en busca de rigor, las herramientas de lógica matemática y el razonamiento formal son cruciales, aunque estas no sean cuantitativas [1].

El concepto de calidad, dentro de la Ingeniería de Software, se ha convertido en uno de los objetivos principales durante el ciclo de construcción de un sistema [11]. Pero dado que este puede tener implicancias diferentes según las necesidades del observador, no necesariamente los aspectos de calidad que se quieren cubrir, son los mismos. Aquí es donde se vuelve fundamental el rol de las métricas durante el ciclo de desarrollo, dado que las mismas entregan mediciones sobre diferentes aspectos, tanto del proceso de desarrollo de software, como de la construcción de las clases (entidades) y relaciones, que conforman el sistema resultante [1].

Los números entregados por las métricas nos ayudan a entender y controlar el proceso de ingeniería.

Dado que el Software Orientado a Objetos (OO) es fundamentalmente distinto del software que se desarrolla utilizando métodos convencionales, las métricas para sistemas OO deben ajustarse a las características que lo distinguen del software convencional [1].

Las métricas OO hacen hincapié en conceptos tales como el encapsulamiento, herencia, complejidad de clases y polimorfismo. Por lo tanto las métricas OO se centran en las mediciones que se pueden aplicar a las características de encapsulamiento, ocultamiento de información, herencia y técnicas de abstracción de objetos que hacen única a una clase [11].

Hemos visto varios modelos propuestos de métricas OO y técnicas de medición, tanto orientadas al tamaño de las clases como a la complejidad de las mismas, y al nivel de acoplamiento entre clases de un sistema. Pero una vez determinado lo que se va a medir, como medirlo y que técnica utilizar, debemos contar con algún criterio de evaluación para poder determinar si el aspecto medido se halla dentro de los valores aceptados, o si es necesario el rediseño del modelo.

3.1 Análisis de los diferentes aspectos del problema (dimensiones)

Las propiedades estructurales de un componente de software (como su acoplamiento) tienen un impacto asociado a su complejidad cognitiva. La complejidad cognitiva es definida como la carga mental de los individuos quienes tienen que tratar con dichos componentes, por ejemplo, diseñadores, verificadores, inspectores, y mantenedores [5] [8]. La alta complejidad cognitiva lleva a que un componente exhiba cualidades externas indeseables, como el incremento en la propensión a fallas, y el aumento del costo de mantenimiento [5].

3.2. Formulación del problema concreto

Una vez que el ingeniero de software realiza las mediciones susceptibles a un proyecto dado, y obtiene los valores que arrojan las métricas utilizadas para tal fin, ¿que decisiones puede tomar frente a estos datos?, ¿Cómo puede determinar si estos datos que arrojo la medición, están dentro de valores aceptables?, ¿Requerirán las estructuras medidas, algún tipo de rediseño o

refactoring?, ¿Cómo está afectando la complejidad cognitiva de las clases, a las características externas? [1][8]. En definitiva, una vez obtenidos estos datos, deseamos saber ¿que significan?, para poder tomar acciones correctivas o no, dependiendo de los valores obtenidos.

Presentado el valor 4, como una media promedio de la cantidad de argumentos para una librería de clases, no se debe necesariamente saber que significa, (bueno, malo, no pertinente). Evaluado contra las medidas publicadas de aceptación, o contra las medidas realizadas en proyectos anteriores, el valor de la medida realizada entregara información más significativa [1] [8]. Particularmente significativos son los puntos periféricos: si el valor medio para una propiedad dada es 5 con una desviación normal de 2, y la medición tomada arroja un valor de 10 para un nuevo desarrollo, probablemente valga la pena realizar una verificación posterior, asumiendo por supuesto que hay alguna teoría para apoyar la asunción de que la medida es relevante al proyecto [1]. Por consiguiente la investigación busca determinar, mediante el análisis y desarrollo estadístico, valores de umbral que sirvan como puntos de referencia respecto de las mediciones realizadas, con el objetivo de proveer al ingeniero de software una herramienta de valoración, que lo asista a la hora de decidir, y dependiendo del aspecto medido, si un componente de software necesita algún tipo de rediseño, o en la indicación de clases con propensión a fallas, o de aquellas que tengan un grado elevado de complejidad cognitiva.

En un estudio realizado en [8], se ha presentado evidencia empírica de que no existe relación (valores de umbral) entre el tamaño de un componente de software (Clase, métodos, etc.), y la propensión a errores. En el presente estudio, se analiza si existen valores de umbral, realizado el análisis con métricas de complejidad; Es decir que se analiza si existe alguna relación ente la complejidad de una clase, y la propensión a fallas.

3.3. Definición de las dimensiones incluidas en el estudio

En el presente caso de estudio la determinación de las variables y dimensiones se vuelve un tanto difusa, debido a la relación que existe entre las variables a medir. Pero en primer lugar, expongamos que entendemos por variables y dimensiones.

> Por dimensión entendemos **un componente significativo de una variable que posee una relativa autonomía**. Nos referimos a componentes porque estamos considerando a la variable como un agregado complejo de elementos que nos dan un producto único, de carácter sintético [2].

3.3.1 Variables, tipo y definición

Las variables que se pretenden encuadrar en el marco de referencia del presente trabajo son:

- El Tamaño de una clase
- La Profundidad y el ancho de una jerarquía de herencia de clases.
- El Grado de Acoplamiento entre clases.

Una vez expuestas las variables, veamos que queremos significar respecto de cada uno de estos aspectos a tratar.

Tamaño de una clase: Por tamaño de una clase, se entiende la cantidad de métodos que la clase implementa, ya sean públicos y/o privados, y por cada uno de estos métodos, la cantidad de líneas de código en cada método [11].

Las métricas asociadas a esta variable podrían ser, SLOC (Source Lines Of Code), RFC (Response For a Class), y WMC (Weighted Method Class).

Profundidad y Ancho de una jerarquía de herencia de clases: La profundidad del árbol de herencia en una estructura de clases, se obtiene mediante la cuenta desde el nodo raíz de la

estructura, hasta el último nodo hoja. Este tipo de cuenta se realiza por niveles, es decir, cuantos niveles jerárquicos hay desde el nodo raíz, hasta el nodo hoja [11].

El ancho del árbol de herencia en una estructura de clases, se obtiene mediante la cuenta desde el nodo en el extremo derecho, hasta el nodo en el extremo izquierdo, contando los nodos intermedios [11], como se muestra en la figura 2.

Las métricas asociadas a esta variable podrían ser, DIT (Deep in Inheritance Tree) y NOC (Number of Children).

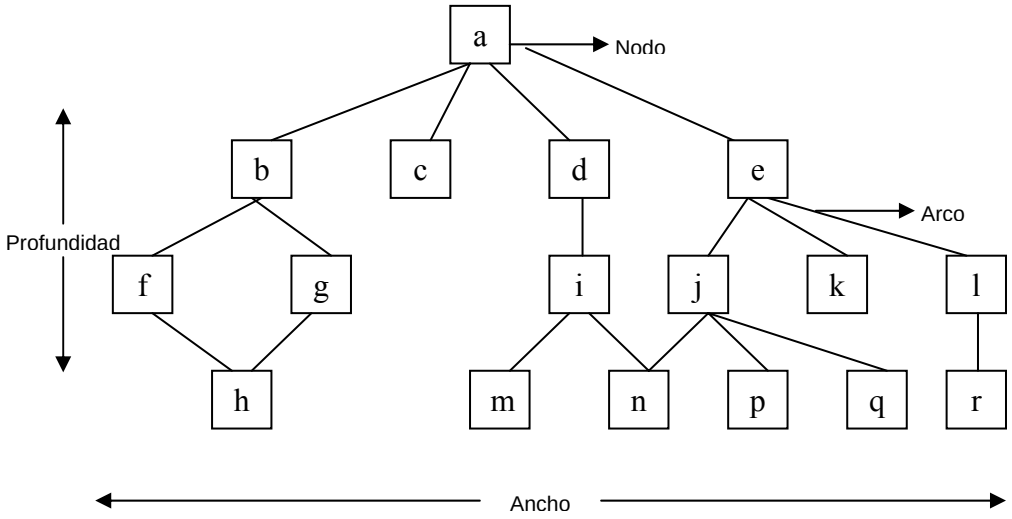

Fig. 2. Profundidad y Ancho en una jerarquía de Clases

El Grado de Acoplamiento de entre clases: El acoplamiento de módulo proporciona una indicación de la "conectividad" de un módulo con otros módulos, datos globales y entorno exterior. Es la cantidad de relaciones que se establecen entre los módulos de un programa [11] [5].

Las métricas asociadas a esta variable podrían ser, CBO (Coupling Between Objects) y todas sus redefiniciones, y LCOM (Lack of Coupling between Objects Methods).

Presentado el conjunto de variables que conformaran el estudio, podemos concluir dado el tipo y naturaleza de las mismas que, tales variables son más bien atómicas, en el sentido que no son susceptibles de ser descompuestas en dimensiones. Al tratarse de conceptos independientes entre sí, los consideramos como elementos autónomos.

3.4. Objetivos de la investigación

Como se ha mencionado anteriormente, la presente investigación busca determinar, mediante el análisis y desarrollo estadístico, valores de umbral que sirvan como puntos de referencia respecto de las mediciones realizadas, con el objetivo de proveer al ingeniero de software una herramienta que lo asista a la hora de decidir, si un componente de software necesita algún tipo de rediseño, o en la indicación de clases con propensión a fallas, o de aquellas que tengan un grado elevado de complejidad cognitiva.

En un estudio realizado en [8], se ha presentado evidencia empírica de que no existe relación (valores de umbral) entre el tamaño de un componente de software (Clase, métodos, etc.), y la propensión a errores. En el presente estudio, se analiza si existen valores de umbral, realizado el análisis con métricas de complejidad; Es decir que se analiza si existe alguna relación ente la complejidad de una clase, y la propensión a fallas.

El objetivo principal de la investigación es, determinar si existen umbrales y valores de umbral para la suite de métricas de Chidamber & Kemerer (CK) [3] [11], referidas a la propensión a errores, respecto de la complejidad cognitiva de los individuos involucrados en el proceso del software; Es decir, si hay alguna relación entre la propensión a errores y la complejidad de una clase. El método utilizado para llevar a cabo el análisis es la regresión

logística (LR). La regresión logística se utiliza para construir modelos cuando la variable dependiente es binaria, como en nuestro caso [13].

Se dice que un proceso es binomial cuando sólo tiene dos posibles resultados: "éxito" y "fracaso", siendo la probabilidad de cada uno de ellos constante en una serie de repeticiones. Un proceso binomial está caracterizado por la probabilidad de éxito, representada por p (es el único parámetro de su función de probabilidad), la probabilidad de fracaso se representa por q y, evidentemente, ambas probabilidades están relacionadas por p+q=1. En ocasiones, se usa el cociente p/q, denominado "odds", y que indica cuánto más probable es el éxito que el fracaso, como parámetro característico de la distribución binomial aunque, evidentemente, ambas representaciones son totalmente equivalentes [13] [6].

Los modelos de regresión logística son modelos de regresión que permiten estudiar si una variable binomial depende, o no, de otra u otras variables (no necesariamente binomiales): Si una variable binomial de parámetro p es independiente de otra variable X, se cumple p= p|X, por consiguiente, un modelo de regresión es una función de p en X que, a través del coeficiente de X permite investigar la relación anterior. La regresión logística es útil cuando se trata de predecir el valor de una variable respuesta dicotómica Y, esto es, una respuesta binaria del tipo 0/1, ausente/presente, sano/enfermo, etc., que presumiblemente depende de otras m variables explicativas (Xj, j= 1, ..., m) a través del modelo de probabilidad [13] [6].

Consecuentemente, seria factible brindar al ingeniero de software una herramienta de comparación y evaluación una vez que haya tomado las medidas pertinentes al dominio del desarrollo que este llevando a cabo. Es decir, toda medida es útil siempre y cuando se la pueda evaluar en comparación con algo. En consecuencia, la determinación de los valores de umbral, estarían aportando el marco de referencia matemático, contra el cual podrían compararse las mediciones tomadas y así, tomar acciones correctivas o no.

La exposición anterior ha presentado las bases teóricas existentes para los efectos de umbral en las métricas OO. También se presentan las métricas CK que se evalúan en el presente estudio, como así también el umbral que se ha derivado en la literatura para cada una.

4 METODO DE INVESTIGACION

En esta sección, se detalla el método de investigación y las fuentes de datos utilizadas para llevar a cabo el análisis

4.1 Medidas

Las métricas OO estudiadas descriptas anteriormente, fueron recolectadas utilizando un analizador de código comercial. Con el objetivo de probar el efecto de umbral, se necesita controlar el efecto potencial del tamaño [7]. Para tal fin la métrica utilizada fue SLOC.

4.1.2 Medidas de Fallas

Dentro del contexto de construir modelos cuantitativos referidos a fallas de software, se ha argumentado que es más importante considerar a aquellas fallas que suceden cuando un sistema entra en producción, que aquellas fallas encontradas durante el estadio de prueba [7]. De hecho, se ha argumentado que el objetivo principal del modelado de calidad es identificar la propensión a fallos post-release [8]. En al menos un estudio se encontró que la propensión a fallas pre-release no es una buena medida substituta para la propensión a fallas post-release. La razón que fundamenta esta afirmación es que la propensión a fallas pre-release es una función del esfuerzo de testeo [8].

Como consecuencia, las fallas contadas para los sistemas bajo estudio fueron debido a fallas ocurridas durante su uso actual en producción. A cada clase se las ha clasificado como “con falla” o “sin falla”. Una clase con errores tiene al menos una falla detectada durante su uso en producción. Los distintos defectos que son originadas por la misma falla, son contados como uno.

4.2 Fuentes de Datos

El estudio se llevo a cabo sobre dos aplicaciones java, los cuales se describen a continuación.

4.2.1 Aplicación java Nº 1

Este es un sistema de mercado electrónico, desarrollado en java. El mismo se encuentra operacional desde hace aproximadamente 10 años. Este sistema se ha puesto en producción en distintos países de Latinoamérica y en distintos entornos de negociación electrónica. En total han trabajado seis desarrolladores en la construcción y mantenimiento del mismo. Esta compuesto de 85 clases que se han analizado.

Lógicamente y dado que el sistema ha evolucionado en funcionalidad durante los años, este trabajo se enfoca en una versión del mismo, donde se pudieron obtener datos confiables referidos a las fallas.

Los datos sobre fallas fueron recolectados del sistema de administracion de configuración. En este sistema se documenta la razón para cada cambio realizado al código fuente y, por lo tanto, facilita la tarea de identificar que cambios fueron realizados debido a fallas. Se ha puesto el foco en las fallas reportadas por errores en producción.
En total, se encontró que 31 clases tenían una o mas fallas que se atribuían a errores de campo.

4.2.2 Aplicación java Nº 2

Este conjunto de datos viene de un framework de mercado electrónico desarrollado para realizar la firma digital de contratos, desarrollado en java. El sistema implementa varios patrones de diseño en los diferentes niveles de la arquitectura. Entre las funcionalidades de este sistema de firma digital se pueden mencionar, generación del par de claves, firma digital de contratos, liquidación de contratos, delegación de contratos, firma de contratos por cta. y orden de terceros, etc.

Este sistema esta siendo utilizado por el mercado líder en negociación de cereales y oleaginosas de Latinoamérica. Un total de 174 clases componen este framework bajo análisis. Un total de 4 programadores han sido involucrados en el proceso de desarrollo y mantenimiento de este conjunto de clases.

Para este producto, se han obtenido los datos sobre fallas del framework en consecuencia al uso actual. Cada error se debió a un único campo de fallas, el cual representa un defecto en el programa que causo la falla. Los errores fueron reportados por los usuarios del sistema. Los desarrolladores del framework documentaron las razones de cada una, en un sistema de control de versiones y, fue de aquí de donde se extrajo información sobre que clases contenían fallas. Se detectaron un total de 192 fallas. Estas fallas se encontraron en 70 de las clases.

4.3 Método de análisis

El método utilizado para llevar a cabo el análisis es la regresión logística (LR) o análisis logit. Este método es utilizado para construir modelos cuando la variable dependiente es binaria, como en este caso de estudio. El enfoque general utilizado es construir un modelo LR sin umbral, y un modelo LR con umbral, y luego comparar ambos modelos [13].

4.3.1 Regresión Logística

La forma general de un modelo LR es la siguiente

$$\pi = \frac{1}{1 + e^{-(\beta_0 + \beta_1 size + \beta_2 M)}}$$

eq. N° 1 [8] [13]

Donde π es la probabilidad de que una clase contenga fallas. La variable de tamaño es SLOC, y M es la métrica especifica que se esta evaluando. Los parámetros β se estiman a través de la maximización del log de probabilidad (incondicional)[2] [8] [13].

4.3.2 Modelo con umbral

Un modelo LR con umbral puede definirse como

$$\pi = \frac{1}{1 + e^{-(\beta_0 + \beta_1 size + \beta_2 (M - \tau) I_+ (M - \tau))}}$$

eq. N° 2 [8] [13]

Donde

$$I + (z) \begin{cases} \textbf{0 si z <= 0} \\ \textbf{1 si z > 0} \end{cases}$$

Y ***t*** es el valor de umbral para la métrica. En este modelo se mantiene al tamaño como variable continua, dado que un estudio realizado en [7], indico que no hay valor de umbral para el tamaño de las clases. La diferencia entre los modelos con y sin umbral, se ilustran en la figura 3 [13] [6].

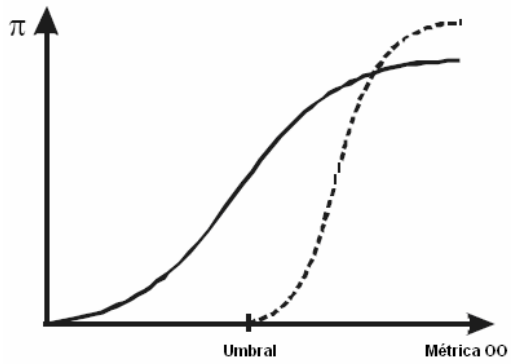

Fig 3. Relación entre la métrica OO M y la probabilidad de fallas para los modelos con umbral y sin umbral. En esta relación vi variable se asume que el tamaño es constante [8] [13] [6].

Para el modelo con umbral, la probabilidad de fallas solo comienza a incrementarse cuando la métrica OO es mayor al valor de umbral t. Para estimar el valor de umbral t se debe maximizar el log de probabilidad para el modelo en la ecuación 2. Una vez que el umbral se ha estimado, se lo debería evaluar. Esto se lleva a cabo comparando el modelo con umbral y sin umbral. Esta comparación se realiza utilizando un ratio de probabilidad estadística, donde la hipótesis nula que se testea es:

$$Ho : t <= M^* = \min M$$ **eq N° 3** [13][6]

2 La regresión logística condicional se utiliza cuando existen concordancias en el diseño del estudio y a cada conjunto concordante se lo trata como un estrato en el análisis.

Donde M* es el valor mas pequeño para la métrica M dentro del conjunto de datos. Si la hipótesis nula no se rechaza, significa que el umbral es igual o esta por debajo del valor mínimo. En el último caso, es exactamente decir que el modelo con umbral es el mismo que el modelo sin umbral. En el caso anterior el modelo con umbral será muy similar al modelo sin umbral, dado que solo una pequeña porción de las observaciones tendrán el valor mínimo. Por lo tanto se podría concluir que el umbral no existe. El ratio de probabilidad estadística se calcula de la siguiente forma: 2(ll(H1) – ll(H0)), donde ll(.) es el log de probabilidad para el modelo dado [8].

Debería notarse que si el valor estimado de ***t*** es igual o cercano a M(n) (el valor más grande dentro del conjunto de datos), significaría que la mayoría de las observaciones en el conjunto de datos tendrían un valor de cero, haciendo inestables a los parámetros estimados para el modelo. En tal caso se concluye que no se encuentra un efecto de umbral para este conjunto de datos. Idealmente, si existe un umbral, entonces no estaría demasiado cerca del valor máximo y mínimo de la métrica M dentro del conjunto de datos.

5. RESULTADOS

En esta sección se presentan los resultados arrojados por el modelo y la evaluación del efecto de umbral.

5.1 Estadísticas descriptivas

Las estadísticas descriptivas para las métricas de complejidad OO y la métrica SLOC se presentan en la tabla 1 y tabla 2. Estas tablas incluyen un resumen tradicional de datos tales como media y desviación estándar. La última columna presenta el número de observaciones que no tienen valor cero. Una de las cosas más notables en los datos presentados en las tablas, es que la herencia tiende a ser baja en ambos sistemas. La métrica NOC para es sistema 2, solo presenta cinco observaciones que no son cero. Por lo tanto esta variable no se considera en lo consecuente, dado que no es posible construir modelos cuando la variación es tan pequeña.

Tabla 1. Estadísticas descriptivas para las métricas extraídas del sistema 1.

	Mean	**Median**	**Std. Dev.**	**IQR**	**N>0**
WMC	15,27	8	19,75	17	159
DIT	0,45	0	0,52	1	76
NOC	0,034	0	0,212	0	5
CBO	0,667	0	1,169	1	64
RFC	12,87	8	15,91	11	159
SLOC	64,34	41,5	61,33	56,75	174

Tabla 2. Estadísticas descriptivas para las métricas extraídas del sistema 2.

	Mean	**Median**	**Std. Dev.**	**IQR**	**N>0**
WMC	16,27	12	17,44	7	85
DIT	0,81	1	0,85	1	49
NOC	0,56	0	1,33	0	17
CBO	13,2	9	9,19	12	85
RFC	35,2	25	35,18	22	85
SLOC	436	280	492	244	85

5.2 Evaluación del efecto de umbral

Los resultados para los modelos de umbral de ambos sistemas, y los resultados de la comparación de los modelos con y sin umbral, se muestran en las tablas 3 y 4.

Todos los modelos con umbral, tienen un numero condicional bajo (por debajo del umbral tradicional de 30), por lo tanto no se considera a la colinearidad como una amenaza [12]. Como se esperaba los valores R^2 son bajos. Contrario de lo que se podría esperar, algunos de los coeficientes de regresión son negativos (DIT para el sistema 1, y DIT, WMC, RFC para el sistema 2). Para el sistema 1, los resultados dejan claro que el modelo con umbral para la métrica CBO, tiene un parámetro estadísticamente significativo. Sin embargo, el modelo con

umbral no es diferente del modelo sin umbral (la ultima columna en las tablas muestran el valor-p, para la comparación de los modelos).

De hecho, ninguna de estas métricas presenta diferencias para los modelos con y sin umbral. Las mismas conclusiones pueden obtenerse del sistema 2.

Para la métrica DIT, se identificaron dos umbrales diferentes para ambos sistemas, aunque cuando se testeo la hipótesis nula, esta no pudo rechazarse. El valor mínimo de DIT es cero, es decir, sin herencia. Por lo tanto, este resultado es de alguna manera consistente con la literatura mencionada anteriormente en que el umbral esta en un DIT igual a cero, en lugar de un DIT mayor a cero.

Basado en estos resultados, se concluye que la incorporación de modelos con umbral, no aporta ninguna información nueva, con lo cual los modelos sin umbral, los cuales son más simples, son preferentes ante los modelos con umbral.

Tabla 3. Resultados para el modelo con umbral del sistema 1, y la comparación de los modelos

Métrica	G (valor-p)	R^2	η	β2 (valor-p)	Umbral	Comparación Valor-p
WMC	9,17 (0,0102)	0,085	4,946	0,0946 (0,173)	11	0,72
DIT	12,21 (0,0022)	0,109	2,87	-7,498 (0,0918)	2	0,22
NOC	12,39 (0,002)	0,111	3,08	0,61 (0,0821)	2	0,602
CBO	33,64 (<0,0001)	0,301	4,71	0,1848 (<0,0001)	1	--
RFC	11,98 (0,0025)	0,107	4,5	0,0249 (0,105)	27	0,835

Tabla 4. Resultados para el modelo con umbral del sistema 2, y la comparación de los modelos

Métrica	G (valor-p)	R^2	η	β2 (valor-p)	Umbral	Comparación Valor-p
WMC	15,19 (0,0005)	0,0647	4,521	-0,03114 (0,2042)	31	0,244
DIT	15,28 (0,0005)	0,065	2,87	-6,636 (0,1919)	1	0,27
CBO	14,32 (0,0008)	0,061	2,87	5,5069 (0,3886)	8	0,393
RFC	16,15 (0,0003)	0,0697	4,43	-0,0532 (0,167)	25	0,2288

5.3 Discusión

Los resultados presentados anteriormente, indican que no existe el efecto de umbrales para el subconjunto de métricas orientadas a objetos CK. Esto significa que, si hay alguna relación entre estas métricas y la propensión a errores, entonces es una relación continua; Esto significa que, conforme crece la complejidad cognitiva de las clases, la propensión a error se desplaza en forma continua, sin presentar discontinuidades en la función que puedan hacer que la curva presente algún tipo de salto a partir del valor de umbral. Esta verificación también puede llevarse a cabo, calculando los límites acercándose por derecha y por izquierda al punto del valor de umbral.

Esto también significa que la teoría cognitiva que se ha postulado en [9] no recibe ningún tipo de soporte del presente estudio, al menos para el software OO. No hay ningún pasaje en las métricas CK estudiadas donde la probabilidad a fallas cambie de ir en aumento progresivo continuo, a incrementarse fuertemente, como se muestra en la fig.3.

Con esto, no se desea significar que los umbrales OO existentes, derivados del conocimiento experimental, no son de utilidad práctica a la luz de estos resultados. Aun si

existiera una relación continúa (sin umbral), entre estas métricas y la propensión a fallas como se propone, si se trazara una línea en el valor mas alto de la métrica y se tomara a este valor como umbral, las clases que estén por debajo de este valor de umbral, continuaran siendo las mas propensas a error. Esta situación se ilustra en el panel izquierdo de la figura 4.

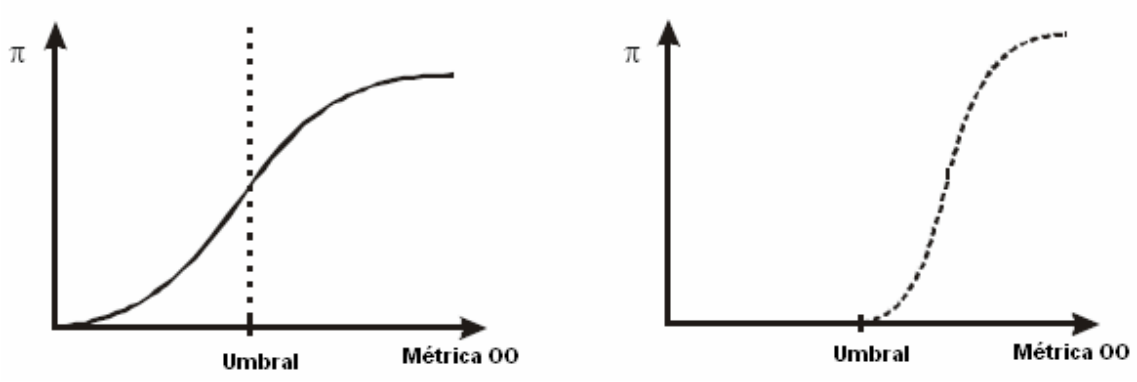

Fig. 4. Diferentes tipos de umbral [8]

Por lo tanto, con el propósito de identificar las clases más propensas a error, estos umbrales probablemente funcionen. Pero debería notarse que las clases por debajo del umbral, aun pueden contener una alta propensión a fallas, aunque seguramente no el más alto.

Habiendo identificado un umbral, las clases con valores por debajo del mismo representan una región segura donde los diseñadores deliberadamente pueden restringir sus clases [8]. Dentro de esta región se puede tener alguna certeza de que las clases, manteniéndose todo lo demás igual, poseen una propensión a fallas mínima. Esta situación se ilustra en el panel derecho de la figura 4.

6 CONCLUSIONES

En [9], Hatton postula una teórica cognitiva en la cual sugiere que existe un efecto de umbral para varias métricas de software. Esta teoría también se ha extendido al software OO. Independientemente de esta teoría, otros investigadores han derivado sus propios umbrales de métricas OO, basados en sus experiencias.

El propósito principal del estudio presentado en este documento es testear esta teoría empíricamente. El estudio se llevo a cabo en dos sistemas de mercado electrónico, desarrollados en java, utilizando un subconjunto de métricas CK [3][11].

La variable dependiente en el análisis fue la incidencia a fallos, la cual lleva a campos de fallas (propensión a errores). Los resultados presentados anteriormente, indican que no existe el efecto de umbrales para el subconjunto de métricas orientadas a objetos CK. Esto significa que, si hay alguna relación entre estas métricas y la propensión a errores, entonces es una relación continua. Esto también significa que la teoría cognitiva que se ha postulado no recibe ningún tipo de soporte del presente estudio, al menos para el software OO. No hay ningún pasaje en las métricas CK estudiadas donde la probabilidad a fallas cambie de ir en aumento progresivo continuo, a incrementarse fuertemente. Estos resultados son consistentes en ambos sistemas y para todas las métricas utilizadas.

Sin bien no se garantiza que no existe un efecto de umbral, la evidencia de los resultados demuestra lo contrario.

Las implicancias de estos resultados son:

- Aquellos usuarios que derivan o utilizan umbrales de las métricas CK, deberían notar que las clases por debajo del valor de umbral, aun son probables que posean una alta propensión a fallas, aunque quizás no la propensión a fallas mas alta.

- Los investigadores que validen las métricas OO, deberían continuar modelando la relación entre las métricas OO, al menos el conjunto de métricas CK, y la propensión a fallas utilizando asunciones de continuidad en lugar de asunciones de umbrales.

- Con esto, no se desea significar que los umbrales OO existentes, derivados del conocimiento experimental, no son de utilidad práctica a la luz de estos resultados. Aun si existiera una relación continúa (sin umbral), entre estas métricas y la propensión a fallas como se propone, si se trazara una línea en el valor mas alto de la métrica y se tomara a este valor como umbral, las clases que estén por debajo de este valor de umbral, continuaran siendo las mas propensas a error.

- Para finalizar, es fundamental reconocer que la formulación de teorías es importante para una disciplina. Las teorías explican los fenómenos que observamos (proveen de un mecanismo). El conocimiento del mecanismo puede potencialmente guiar a avances de campo. Dada la escasez de teorías previas en la ingeniería de software, es un acto de coraje formular una teoría y exponerla a críticas y testeo empírico. Por lo tanto necesitamos constantemente proponer teorías, explicaciones y validarlas empíricamente.

6.1 Contribución del estudio

Es claro dada la evidencia presentada que no existe un efecto de umbral entre la complejidad de una clase y la propensión a errores. Por lo tanto la teoría que declara que la propensión a errores en una clase permanece estable hasta un cierto nivel de complejidad y una vez que este nivel de complejidad es excedido la propensión a fallas se incrementa, debido a limitaciones en la memoria a corto plazo, queda sin soporte. La única evidencia que podemos presentar es que existe una relación continua entre la complejidad cognitiva de una clase y la propensión a errores. Esto quiere decir que, conforme la complejidad cognitiva de la clase aumente, también lo hará su propensión a fallas.

6.2 Trabajo Futuro

Los resultados presentados no deberían implicar que la complejidad de una clase, es la única variable que se puede utilizar para predecir propensión a errores para el software OO. Por el contrario, mientras la complejidad de una clase pareciera ser una variable importante, otros factores indudablemente tendrán su influencia. Por lo tanto, y con el propósito de construir modelos comprensivos que predigan la propensión a fallas, otras variables deberían ser analizadas.

REFERENCIAS

1. B. Meyer - The role of object-oriented metrics - (November 1998). http://archive.eiffel.com/doc/manuals/page.html

2. C. Sabino – El proceso de investigación – Cap. 5 - http://paginas.ufm.edu/sabino/PI-cap-5.htm

3. Chidamber SR, Kemerer CF. Towards a metric suite for object-oriented design. Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications. ACM Press: New York NY, 1991; 197–211.

4. G. Miller: "The Magical Number 7 Plus or Minus Two: Some Limits on Our Capacity for Processing Information". In Psychological Review, 63:81-97, 1957.

5. Grady Booch, Análisis y Diseño Orientado a Objetos. 2° Edición Addison Wesley Longman – 1996. ISBN: 968-444-352-8

6. Hair, Anderson, Tathan, Black, "Analisis Multivarante" 5ta. Edicion - Prentice Hall, 1999

7. K. El Emam, S. Benlarbi, N. Goel, and S. Rai: "The Confounding Effect of Class Size on the Validity of Object-Oriented Metrics". To appear in IEEE Transactions on Software Engineering, 2000.

8. K. El Emam; Benlarbi, Saida; Goel, Nishith; Melo, Walcelio; Lounis, Hakim; Rai, Shesh N; The optimal class size for object-oriented software; National Research Council of Canada Institute for Information Technology Building M-50, Ottawa, Ont., K1A OR6, Canada; IEEE Transactions on Software Engineering. Vol. 28, no. 5, pp. 494-509. May 2002

9. L. Hatton: "Does OO Sync with How We Think ?" In IEEE Software, pages 46-54, May/June 1998.

10. L. Rosenberg, R. Stapko, and A. Gallo: "Object-Oriented Metrics for Reliability". Presentation at IEEE International Symposium on Software Metrics, 1999.

11. R. Presuman, "Ingeniería del Software, un enfoque práctico"; McGraw-Hill; Cuarta Edición.1998

12. S. Simon and J. Lesage: "The Impact of Collinearity Involving the Intercept Term on the Numerical Accuracy of Regression". *In Computer Science in Economics and Management*, 1:137- 152, 1988.

13. SPSS - Estadística avanzada – AMA 2003