CIGET - MATANZAS IDICT INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

Herramienta para aplicar métricas al Diagrama de Clases del Diseño Orientado a Objetos.

Tool to apply metrics to Objects Oriented Design Class Diagram.

Ing. Carlos Rafael Rodríguez Rodríguez.
Centro de Gobierno Electrónico, Facultad 3, Universidad de las Ciencias Informáticas. Cuba.
crodriguezr@uci.cu

Resumen

Las métricas de diseño Orientado a Objetos (OO) son un recurso poderoso para validar artefactos generados durante el importante flujo de Diseño; no obstante resulta engorroso recopilar la información necesaria para aplicarlas, además del gasto de tiempo que implica.

El presente trabajo propone automatizar el uso de métricas de diseño OO, con el objetivo de contribuir en la reducción del tiempo de desarrollo y mejorar la calidad final de los productos. Como resultado se obtuvo un sistema que automatiza el cálculo de 13 métricas de diseño OO de las conocidas como CK y LK entre las que se encuentran: acoplamiento entre objetos (CBO), nivel de profundidad del árbol de herencia (DIT) y número de métodos reemplazados (NMO). Para calcularlas se recupera la información necesaria del fichero XML de un diagrama de clases generado por una herramienta CASE. El sistema es extensible, permite la incorporación de plug-ins para aplicar nuevas métricas y para interpretar XML de otras herramientas de modelado.
Palabras clave: calidad, diseño, herramientas CASE, métricas, patrones, plug-in.

Abstract

The object oriented design metrics are a powerful resource to validate artifacts generated during the important flow to design, however is cumbersome to collect the necessary information to applied, besides the time involved.

This document proposes to automate the use of object oriented design metrics, with the objective to help reduce development time and improve final product quality. The result was a system that automates the calculation of 13 metrics of object oriented design known as CK and LK, among which are: coupling between objects (CBO), level of depth of inheritance tree (DIT) and number of overridden methods (NMO).To calculate those metrics obtains the required information from

CIGET - MATANZAS IDICT INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

an XML file of a class diagram generated by a CASE tool. The system is extensible, allows the addition of plug-ins to implement new metrics and interpret XML files from other modeling tools.
Keywords: CASE tools, design, metrics, patterns, plug-in, quality.

Introducción

Desde que el hombre creó sus primeros proyectos, un objetivo común ha estado presente en todos ellos: lograr la mayor calidad posible. Ante los crecientes avances de la ciencia esta aspiración es aún más relevante, a tal punto que, seguir las más estrictas normas de calidad establecidas internacionalmente es un imperativo si se pretende ser competitivo en el mercado internacional. Según la norma 9000 de la Organización Internacional para la Estandarización (ISO), la calidad es el grado en el que un conjunto de características inherentes cumple con los requisitos (ISO, 2005).

El incremento de la producción de software en la actualidad, ha acelerado la necesidad de elaborar productos de mayor calidad con menor costo y en el menor tiempo posible. Esto está condicionado por el hecho de que a los clientes no solo les interesa obtener la solución de software, sino que esta cumpla los más refinados estándares de calidad.

En ese contexto el desarrollo de software dirigido por modelo (MDSD) fue cobrando fuerza en el mundo de la ingeniería de software hasta erigirse en un nuevo paradigma. MDSD promueve el uso de modelos no sólo para la documentación y la comunicación sino como artefactos de primera clase para generar otros productos durante el proceso de desarrollo, tales como otros modelos y código fuente.

Así mismo provee otras características y ventajas entre las que se encuentran (Stahl & Völter, 2006):
- ✓ desarrollar software de mayor calidad más rápido;
- ✓ elevar el nivel de abstracción: modelos;
- ✓ menos detalles “accidentales”, notaciones más cercanas al problema;
- ✓ no sólo documentación: generación de código;
- ✓ dominios específicos;
- ✓ evitar codificar las mismas soluciones una y otra vez.

Este nuevo paradigma parece dejar claro que la actividad más importante dentro del marco de trabajo de la ingeniería de software pasa a ser el modelado en lugar de la construcción (implementación), lo que llevaría invariablemente a que el aseguramiento de la calidad tenga un papel relevante durante esta etapa.

CIGET - MATANZAS IDICT INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

Desarrollar modelos de calidad es entonces una necesidad cada vez más apremiante para los ingenieros de software, es decir, lograr que estos modelos representen eficientemente todos los detalles del dominio en cuestión y al mismo tiempo se ajusten a las buenas prácticas establecidas. Convirtiéndose en ahorro de recursos, tiempo y sobre todo propiciando la creación de un software robusto desde su base.

El diseño es una etapa fundamental en el desarrollo de sistemas OO, por lo que es de gran importancia asegurar su calidad, evitando la propagación de errores a todo el proceso de desarrollo y el alto costo de su corrección. Una de las técnicas empleadas para evaluar el diseño de software OO es el uso de métricas. Las métricas son la medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado (IEEE, 1993), y tienen como objetivo asegurar la calidad del producto.

Para facilitar su uso estas métricas son implementadas por varias herramientas independientes o integradas como plug-in con otras (Cyvix, 2006) (JDepend, 2005) (Metric1.3.6, 2005) (NDepend, 2004) (SDMetrics, 2002) (Refactorit, 2008) (Design Advisor, 2004) (GenMETRIC, 2007) (SAAT, 2002). Pero en contradicción a lo que se ha fundamentado antes, la mayoría de esas herramientas reciben como entrada el código fuente o binario obtenido después de la implementación, lo que no permite medir la calidad en las etapas iniciales del proceso de desarrollo, específicamente durante el diseño. Solo un número reducido de herramientas aplican métricas a partir de modelos y en su mayoría lo hacen para instituciones y herramientas de modelado específicas.

De la problemática antes descrita se deriva como objetivo de esta investigación desarrollar un sistema que automatice la aplicación de métricas para facilitar la evaluación de los diagramas de clases del diseño OO en los proyectos de la Universidad de Ciencias Informáticas (UCI).

Esta investigación constituye un beneficio para los equipos de desarrollo de la UCI pues contribuye a la reducción del tiempo y esfuerzo requeridos para la aplicación de métricas OO en la evaluación del software durante el modelado, evitando que en el futuro aparezcan debilidades en la mantenibilidad del sistema; por lo que fortalecerá el proceso de desarrollo de software.

En lo adelante el trabajo se estructura de la siguiente forma: a continuación se señalan las herramientas, tecnologías, patrones de diseño y métricas empleados; luego se describe la solución propuesta detallando cada una de sus fases así

como la arquitectura y la seguridad del sistema, además se muestran varios resultados de la validación técnica de la propuesta; finalmente se ofrecen algunas conclusiones y líneas de trabajo futuro.

Materiales y métodos

Herramientas y tecnologías empleadas

Como metodología de desarrollo de software se emplea Agile Unified Process (AUP) por combinar las ventajas de RUP (robusta) con técnicas agiles. Esta metodología permite seleccionar entre todos los artefactos de RUP solo aquellos que se ajusten a las necesidades del equipo de desarrollo y a pesar de ser una metodología ágil presta gran importancia a la arquitectura del sistema (Ambler, 2005).

Como Entorno de Desarrollo Integrado (IDE) se emplea SharpDevelop 4.0 una herramienta gratis y de código abierto que es compatible con las plataformas .NET y Mono, en este caso se emplea con Microsoft.NET 4.0 y C# como lenguaje de programación.

En su primera versión la herramienta es compatible con XML 2.x y acepta ficheros XML generados por Visual Paradigm para UML por ser esta la herramienta de modelado establecida para los proyectos de software de la UCI.

Patrones de diseño empleados

En el modelado de la solución se emplean 4 patrones de diseño del grupo conocido por GoF (Gamma, Helm, Johnson, & Vlissides, 1995).

El patrón Compuesto (Composite) se utiliza en el diseño de las clases del modelo, debido a que todas las estructuras que se encuentran en un diagrama de clases constituyen elementos de dicho diagrama con la particularidad de que estos elementos pueden ser simples como el caso de los atributos y los parámetros de una función o compuestos como por ejemplo una clase que constituye un elemento en sí y a su vez está conformada por otro elementos simples o compuestos, como son los atributos y funciones respectivamente.

El patrón Estrategia (Strategy) es utilizado para la creación de las métricas de diseño y de los traductores de XML que serán incorporados al sistema en forma de plug-in. Garantiza que las estrategias (diferentes implementaciones de la función aplicar métrica o traducir modelo) puedan ser manejadas de una forma única.

CIGET - MATANZAS IDICT INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

Además se emplean los patrones Instancia Única (Singleton) y Fábrica Abstracta (Abstract Factory). El primero es utilizado en las clases controladoras del diseño, debido a que su función es controlar todos los eventos del sistema, y sería beneficioso tener una instancia única de dichas clases manteniendo así su estado durante todo el período de ejecución. El segundo se emplea para encapsular la creación de los objetos del negocio, abstrayendo a la vista de la forma en que son instanciadas las entidades lo que permite que el manejo de dichos objetos sea más flexible y des-acopla la vista del modelo.

Métricas empleadas

Las métricas soportadas en la primera versión del sistema corresponden a los grupos propuestos por Chidamber y Kemerer (Chidamber & Kemerer, 1994) y Lorenz y Kidd (Lorenz & Kidd, 1994) y se describen a continuación.

Conjunto de métricas CK

Chidamber y Kemerer proponen 6 métricas basadas en clases para medir cinco atributos básicos en el diseño orientado a objetos: acoplamiento, complejidad de una clase, reutilización, cohesión y herencia. De estas solo se implementan 3 debido a que la información necesaria para aplicar las otras no es posible obtenerla de un modelo estático como es el diagrama de clases del diseño.

- ✓ Profundidad del árbol de herencia (Depth of Inheritance Tree -DIT): Es la distancia desde una clase, a la clase raíz del árbol de herencia. Si la clase se encuentra en situación de herencia múltiple, el DIT será la longitud máxima hasta la raíz. Se utiliza como medida de: la complejidad de una clase, la complejidad del diseño y el potencial de rehúso. Es deseado obtener un número bajo de DIT.

- ✓ Número de hijos (Number of children-NOC): Es el número de subclases inmediatas a una clase en el árbol de herencia y mide la anchura de una jerarquía de clases. Un alto valor del NOC aumenta la reutilización, pero podría disminuir la abstracción por usar incorrectamente la herencia. Aumentaría la dificultad para modificar una clase base y el nivel de pruebas requerido. Es favorable obtener un número bajo de NOC.

- ✓ Acoplamiento entre objetos (Coupling Between Object classes-CBO): El acoplamiento es la dependencia que tiene una clase de las demás clases del sistema. El CBO de una clase es el número de clases a las que ella está relacionada, sin tener en cuenta las relaciones por herencia. Un alto valor de

CIGET - MATANZAS IDICT INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

CBO reduce el encapsulamiento y la posibilidad de reutilización, además aumenta la complejidad de las pruebas.

Conjunto de métricas LK

Lorenz y Kidd concentran las métricas basadas en clases en tres grupos: a) Métricas de tamaño de la clase, b) Métricas de herencia y c) Métricas de las características internas de las clases. Estas métricas están enfocadas a las características internas del diseño orientado a objeto y de esta manera, contribuyen a asegurar la mantenibilidad de los productos de software.

Métricas de tamaño:

- ✓ Número de Métodos de Instancia Públicos (PIM): Es el número total de métodos públicos de instancias, es decir los métodos que están disponibles como servicios para otras clases. Esta métrica mide la cantidad de responsabilidad que tiene una clase. Se sugiere utilizarla en la estimación de la cantidad de trabajo necesario para desarrollar una clase. Es deseado obtener un número bajo de PIM.

- ✓ Número de Métodos de Instancia (NIM): Se define como la suma de todos los métodos definidos para las instancias de una clase ya sean públicos, protegidos o privados. Constituye una medida del tamaño de la clase. Las clases más grandes son más complejas y difíciles de mantener, mientras que las más pequeñas tienden a ser más reutilizable. Es favorable obtener un valor bajo de NIM.

- ✓ Número de Variables de Instancia (NIV): Se determina por el número total de variables a nivel de instancia que tiene una clase. Un alto valor de NIV no es deseado debido a que un gran número de variables de instancia puede indicar demasiado acoplamiento con otras clases. Lorenz y Kidd sugieren que las clases son más reutilizables cuando tienen menos variables de instancia.

- ✓ Número de Métodos de Clase (NCM): Es el número total de métodos a nivel de clase. Un método de clase es un método que es global para sus instancias. El número de métodos de la clase puede indicar la cantidad de elementos comunes que se manejan para todas las instancias. Este número generalmente debe ser relativamente pequeño en comparación con el número de métodos de instancia.

- ✓ Número de Variables de Clase (NVV): Es el total de variables de clases que

CIGET - MATANZAS IDICT INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

tiene una clase. Los valores de las variables de clase son constantes y son compartidos por todos los objetos de la clase (variables estáticas-"static"). El número medio de variables de clase debe ser bajo. En general debe haber menos variables de clase que variables de instancia.

Métricas de herencia:

- ✓ Número de Métodos Reemplazados (NMO): Es el número total de métodos que redefine una subclase de los que haya heredado de una superclase, esto se conoce como reemplazar el método. Mide la calidad del uso de la herencia. Los métodos reemplazados, especialmente en niveles muy profundos de la jerarquía de herencia, pueden indicar un problema en el diseño. Es deseado obtener un valor bajo de NMO.

- ✓ Número de Métodos Heredados (NMI): Es el número de métodos que hereda una subclase. Es deseado un número alto de NMI ya que indica la fuerza de la subclase en la especialización. También mide la calidad del uso de la herencia.

- ✓ Número de Métodos Añadidos (NMA): Se define como el número total de métodos que se definen en una subclase. Las subclases deben definir nuevos métodos, que extiende el comportamiento de las superclases. El número de nuevos métodos por lo general debería disminuir a medida que se mueven a través de las capas de la jerarquía. Igual que las anteriores mide la calidad de uso de la herencia.

- ✓ Índice de Especialización para una clase (SIX): Mide el grado en que una subclase redefine el comportamiento de una superclase. Indica cuándo hay demasiados métodos redefinidos. Una subclase debe extender el comportamiento de la superclase con métodos nuevos más que redefinir comportamiento. Propone un valor del 15% para identificar superclases que no tienen mucho en común con sus subclases.

  *SIX = Nº de métodos redefinidos * Anidamiento en la jerarquía / Nº total de métodos.*

Métrica de características internas de una clase:

- ✓ Promedio de Parámetros por Método (APPM): Se determina como el cociente entre el número total de parámetros por método y el número total de métodos.
  APMM = Nºtotal de parámetros por métodos/ Nºtotal de métodos.

Resultado y discusión

CIGET - MATANZAS IDICT INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

Herramienta para aplicar métricas al Diagrama de Clases del Diseño Orientado a Objeto

El sistema propuesto (Rodríguez, Almenares, & Pérez, 2011) representa las actividades que en la práctica se llevan a cabo para aplicar las métricas. A grandes rasgos esas actividades podrían definirse como:

(1) Recopilación de la información, (2) Selección de las métricas a aplicar, (3) Definición de los intervalos de aceptación para cada métrica, (4) Cálculo de las métricas, (5) Análisis de resultados [comparar contra intervalos].

En el sistema estas actividades se encuentran agrupadas en tres fases: inicio, cálculo y análisis (figura 1).

Figura 1 Representación de los procesos que ocurren en el sistema.

Fase de inicio

El principal objetivo de esta fase es recopilar la información necesaria para poder realizar el cálculo de las métricas. Para lograrlo, el sistema recibe como entrada un fichero XML generado por la herramienta de modelado empleada y que es cargado por el usuario. Dicho XML representa la estructura de un Diagrama de Clases del Diseño e incluye detalles sobre las clases (atributos y operaciones) y sus relaciones. Este XML es sometido a un proceso de interpretación en el que se extraen los datos necesarios y se obvian los otros. Finalmente con los datos extraídos se confecciona un modelo de clases sobre el que se realizará el proceso de cálculo.

Fase de cálculo

Esta fase recibe como entrada el modelo de clases obtenido de la fase anterior y tiene como principal objetivo realizar el cálculo de las métricas. En ella se muestran las métricas disponibles para aplicar permitiendo seleccionar las deseadas y configurar sus intervalos de aceptación.

CIGET - MATANZAS
IDICT
INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

Fase de Análisis

Esta fase recibe como entrada el resultado del cálculo, lo compara con los intervalos de aceptación definidos anteriormente e informa los resultados especificando si son positivos o negativos como se observa en la figura # 2.

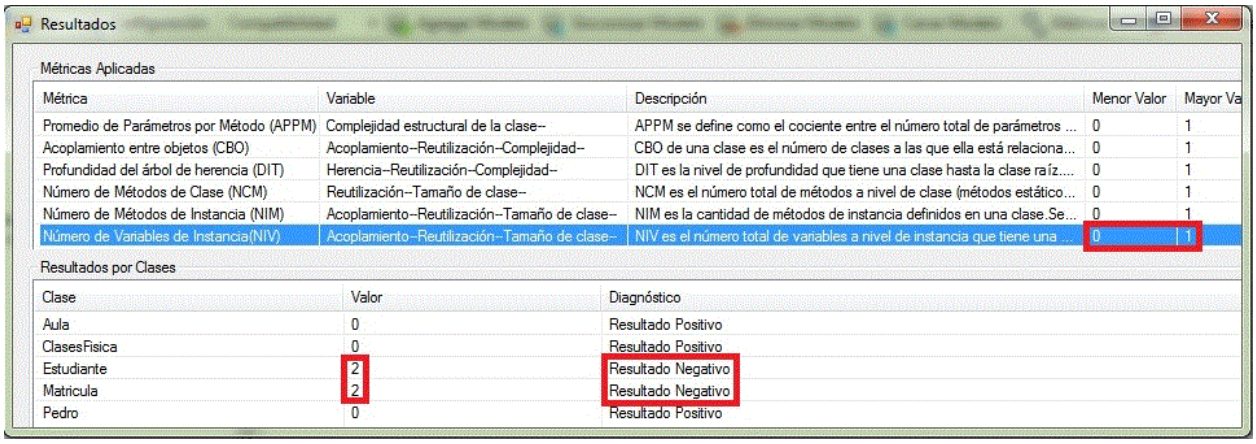


Figura 2 Resultados de una evaluación en el sistema.

Arquitectura del sistema. Funcionamiento basado en plug-in.

El sistema emplea una arquitectura Modelo Vista Controlador (MVC) como se observa en la figura 3 y basa su funcionamiento en el uso de plug-ins tanto para crear nuevos intérpretes de XML como para implementar nuevas métricas; lo que permite extender su comportamiento de forma dinámica creando un objeto de instancia de una interfaz en tiempo de ejecución.

Tiene como requisito que la clase que se instancia, implemente una determinada interfaz para poder tratar las distintas clases plug-in por igual.

Con esto el plug-in se conecta a una clase abstracta parcial, que a su vez, se conecta a la clase principal. El plug-in utiliza esta interfaz para aplicar métodos llamados por la clase principal.
Tiene como ventaja que permite conectar nuevas clases a la aplicación sin necesidad de modificar el código fuente original, facilitando una mayor modularidad en el programa y la posibilidad de que un tercero añada nuevas funcionalidades.

CIGET - MATANZAS IDICT INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

La seguridad del sistema

La seguridad es un aspecto a tener presente durante el desarrollo de todo sistema informático. Más aún cuando se desea construir un software que podrá ser extendido con tan solo colocar un ensamblado en uno de sus directorios. La facilidad brindada por el sistema para ejecutar códigos desarrollados por terceros es una amenaza potencial a la integridad y confidencialidad de los datos del cliente. Por este motivo se toman una serie de medidas entre las que se encuentran:

Figura 3 Arquitectura del sistema

- ✓ Firma de los ensamblados de la herramienta: Con la firma de los ensamblados se garantiza que cada uno de estos ensamblados tengan un nombre único (nombre seguro). De esta forma se evitaría la suplantación o modificación de los ensamblados por parte de un tercero. Certificándose así la autenticidad de los mismos.
- ✓ Verificación de la autenticidad de los plug-in: Se verifica antes de cargar un plug-in que esté firmado con la misma clave que se ha firmado el resto del sistema. Evitándose la carga de un plug-in desconocido que pudiera comprometer la integridad del sistema.
- ✓ Creación de un nuevo Dominio de Aplicación para la ejecución de los plug-ins: Siempre que un sistema admite código de terceros está expuesto a afectaciones ya sea por errores de programación o secuencias de código mal intencionadas. Para minimizar estos riesgos se crea un nuevo Dominio de Aplicación en el cual serán ejecutados los plug-ins. Esto garantiza que la ejecución de los plug-ins no se realice en el mismo entorno en que se ejecuta la aplicación base, permite establecer nuevas políticas de seguridad que definirán un entorno controlado donde se ejecutarán los plug-ins y garantiza que los errores que puedan producirse en ellos no afecten la aplicación base.

Validación de la propuesta

Valorar si el diseño obtenido se ajusta al nivel de calidad requerido es importante para poder conocer la efectividad de los procesos que han sido modelados y si requieren o no de gran esfuerzo para su implementación.

A continuación se muestran los resultados de la aplicación de algunas métricas OO al diseño de la herramienta propuesta con el objetivo de determinar su grado de calidad y fiabilidad. Se empleará como ejemplo el caso de uso Gestionar XML por ser uno de los más complejos en el sistema.

Resultado de la métrica CBO:

Para aplicar esta métrica es necesario conocer la cantidad de clases con las que se relaciona una clase excluyendo las relaciones por herencia. En la Tabla 1 se muestran los datos recopilados.

CIGET - MATANZAS IDICT INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

Como se muestra en la Figura 4 el mayor por ciento de las clases no están acopladas y el por ciento restante presentan una acoplamiento relativamente bajo, permitiendo que aumente el grado de reutilización de las clases. Este resultado también ayuda a que las pruebas o modificaciones que sean necesarias, resulten fáciles de ejecutar.

Tabla 1 Colaboraciones por clases. CU Gestionar XML.

Nombre de la Clases	No. Colaboraciones
AccGestionarXML	0
AccPrincipal	2
Clase	1
CModelo	1
ContextParser	2
ElementoModelo	0
Funcion	1
Atributo	0
Parametro	0
Relacion	0
Nodo	1
ConcreteStrategyParserVP	0

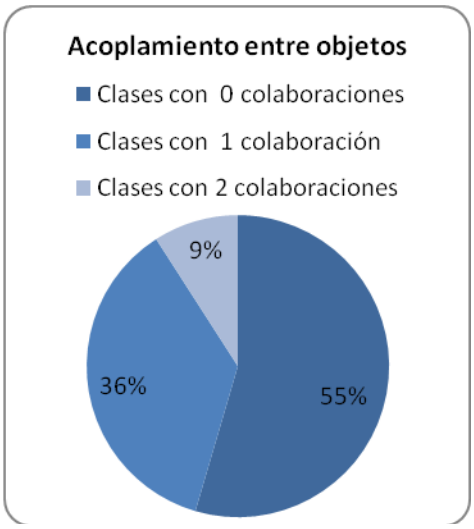


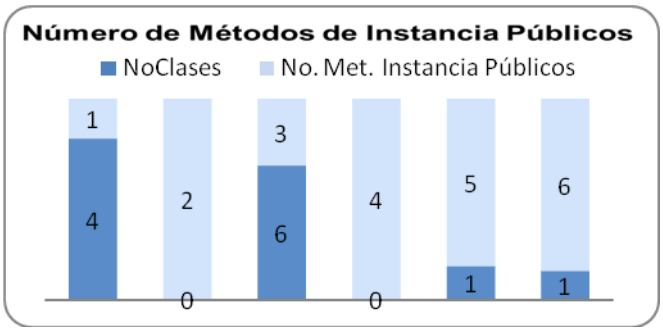
Figura 4 Acoplamiento entre objetos. CU Gestionar XML.

Resultados de la métrica PIM:

Para calcular esta métrica se necesita la información de la cantidad de métodos de instancia públicos que tiene cada clase. La Tabla 2 y la Figura 5 muestran y representan gráficamente los resultados.

Tabla 2 PIM por clases. CU Gestionar XML

Número Clases	No. Met. Instancia Públicos
4	1
0	2
6	3
0	4
1	5
1	6



CIGET - MATANZAS IDICT INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

Figura 5 Número de Método de Instancia Públicos. CU Gestionar XML

Los resultados obtenidos son satisfactorios. Diez de las doce clases modeladas tienen tres o menos métodos de instancia públicos lo que es sin dudas un resultado bajo. Las otras dos clases tienen valores que aunque son más elevados que los demás, también resultan bajos. Estas dos últimas clases fueron revisadas y se comprobó que sus métodos son todos necesarios para el correcto funcionamiento del sistema. Estas clases son *AccPrincipal* que regula todo el flujo de eventos de la aplicación y la *Clase* que es el principal elemento creado dentro del modelo.

Resultados de la métrica NIV:

Se necesita conocer el número de variables de instancia de una clase para calcular esta métrica. En la

Tabla 3 se exponen los resultados obtenidos luego de aplicar dicha métrica y se representan en la Figura 6.

Tabla 3 NIV por clases. CU Gestionar XML

Número Clases	No. Var. Instancia
3	0
2	1
2	2
2	3
1	4
2	5

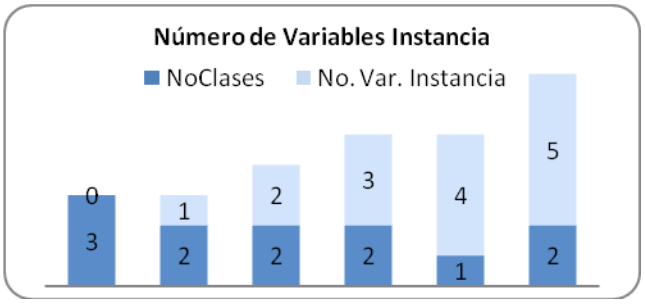


Figura 6 Número de Variables Instancia. CU Gestionar XML

El resultado evidencia que los valores de esta métrica para cada una de las clases son permisibles. Solo dos de las doce clases modeladas tienen cinco variables de instancia que aunque es el mayor, es bajo en comparación con la cantidad que suelen tener las clases. Estas dos clases tienen 5 variables instancia que resultan imprescindibles para la creación de los elementos del modelo, dos de las cuales son heredadas de su superclase.

Resultados de la métrica DIT:

CIGET - MATANZAS IDICT INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

Para aplicar esta métrica es necesario conocer la cantidad de relaciones de herencia que tiene una clase. Seguidamente se muestran los datos recopilados y su representación gráfica.

La Figura 7 muestra en color rojo el nivel que tiene cada una de las clases en el árbol de herencia. Las que tienen valor cero indican que no tienen relaciones de herencia o que están en el nivel cero del árbol. Las restantes poseen valor 1 que es inferior al 5 sugerido por la bibliografía consultada. Estos resultados indican que las clases serán fáciles de desarrollar y de mantener.

Tabla 4 Herencias por clases. CU Gestionar XML.

Clases	Profundidad árbol herencia
AccGestionarXML	0
AccPrincipal	0
Clase	1
CModelo	0
ContextParser	0
ElementoModelo	0
Funcion	1
Atributo	1
Parametro	1
Relacion	1
Nodo	1
ConcreteStrategyParserVP	0

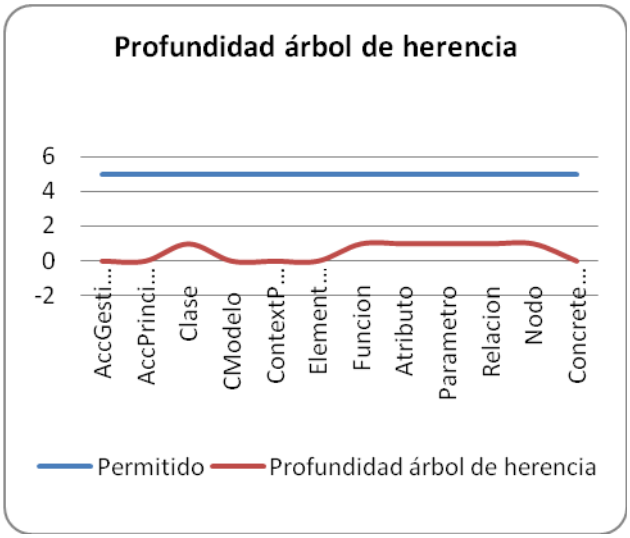


Figura 7 Profundidad árbol de herencia. CU Gestionar XML

Conclusiones

En el presente trabajo se propuso un sistema para medir la calidad del diseño OO basado en métricas, con el objetivo de contribuir a reducir los tiempos de desarrollo y mejorar la calidad final del producto. Se cumplieron los objetivos propuestos y se arribó a las siguientes conclusiones:

- ✓ El uso de métricas es una técnica muy útil para evaluar productos de software, con mayor relevancia aún dentro del paradigma DSDM.
- ✓ Evaluar los modelos de clases del diseño aplicando métricas permite detectar insuficiencias y mejoras potenciales desde etapas tempranas del desarrollo del producto, evitando que se propaguen a fases siguientes y propiciando la creación de un sistema robusto desde su misma concepción.

CIGET - MATANZAS IDICT INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

- ✓ Se obtuvo un sistema para medir la calidad de los diagramas de clases del diseño OO basado en métricas, que permite aplicar 13 métricas a la información extraída de un fichero XML generado por una herramienta CASE.
- ✓ El sistema creado garantiza la extensibilidad, permitiendo la incorporación de plug-ins tanto para aplicar nuevas métricas como para interpretar XML de otras herramientas de modelado.
- ✓ El sistema creado tiene un aporte significativo pues la mayoría de los proyectos de software que se desarrollan en la UCI no aplican métricas a sus productos.

Como trabajo futuro, se hace necesario promover la creación de plug-ins que permitan la compatibilidad del sistema con nuevas herramientas CASE e incrementar el número de métricas soportadas, así como continuar el desarrollo del sistema de forma tal que permita una gestión más amplia y precisa de los proyectos que en él son analizados.

Referencias bibliográficas

Ambler, S. W. (2005). Recuperado el 15 de Enero de 2011, de The Agile Unified Process (AUP): http://www.ambysoft.com/unifiedprocess/agileUP.html

Chidamber, S. R., & Kemerer, C. F. (1994). *A Metrics Suite for Object Oriented Design.* IEEE Transactions on Software Engineering.

Gamma, R., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns. Elements of Reusable Object-Oriented Software.* Addison Wesley Professional Computing Series.

IEEE. (1993). *IEEE Software EngineeringStandards. Standard 610.12-1990, 1993.*

ISO. (2005). *ISO 9000:2005 Sistemas de gestión de la calidad — Fundamentos y vocabulario.* Ginebra: Secretaría Central de ISO.

Lorenz, M., & Kidd, J. (1994). *Object Oriented Metrics.* Englewood, New Jersey: Prentice Hall.

Rodríguez, C., Almenares, Y., & Pérez, J. (2011). *Sistema Automatizado para medir la calidad del diseño OO basado en métricas.* Universidad de las Ciencias Informáticas, La Habana.

Stahl, T., & Völter, M. (2006). *Model-Driven Software Development.* Wiley.

Fecha de recepción: 4/09/2012 Fecha de aprobación: 24/10/2012