Herramienta para evaluar atributos de mantenibilidad en aplicaciones PHP

Julio Acosta, Cristina Greiner, Gladys Dapozo

Departamento de Informática. Facultad de Ciencias Exactas y Naturales y Agrimensura
Universidad Nacional del Nordeste, Av. Libertad 5450, 3400, Corrientes, Argentina
julio_acosta_01@hotmail.com, {gndapozo, cgreiner}@exa.unne.edu.ar

Resumen. La medición de software permite expresar de manera cuantitativa atributos como el grado de mantenibilidad de una aplicación a partir del análisis de un conjunto de métricas específ cas. En este trabajo se presenta una herramienta de medición de atributos vinculados con la mantenibilidad en aplicaciones orientadas a objetos, en particular, al código escrito en PHP. Esta herramienta automatiza un método de medición basado en GQM, calcula los valores de un conjunto de métricas, almacena los resultados de las distintas mediciones y permite comparar cómo se comportan los atributos evaluados. Para la validación del método y la herramienta se evaluó el sistema de gestión de contenidos Joomla en cinco versiones. Esto permitió observar el comportamiento de los indicadores que surgen de las métricas OO que se vinculan a determinados atributos de calidad en un producto software específ co que se encuentra en estado de explotación y susceptible de mantenimiento evolutivo.

Keywords: Calidad del Software. Calidad del producto, Mantenibilidad. Gestión cuantitativa de proyectos. Repositorio de conocimiento.

1 Introducción

La medición del software se ha convertido en una actividad importante en los modelos y estándares de calidad. Existe en la actualidad una gran cantidad de herramientas enfocadas en la medición de atributos de calidad de productos software. Sin embargo aún existen lenguajes de programación que no cuentan con herramientas de medición suf cientemente maduras para generar una amplia variedad de métricas y umbrales de comparación que permitan evaluar atributos de calidad, en particular los vinculados con la Programación Orientada a Objetos. PHP es uno de ellos, situación preocupante si se tiene en cuenta que es uno de los lenguajes de programación más ampliamente utilizado tanto en la comunidad de código abierto como en la industria para construir aplicaciones centradas en web y entornos de aplicaciones.

La calidad del software está estrechamente vinculada con la medición del mismo. En [1] se señala que la medición de atributos internos del software es el primer indicador de cumplimiento de atributos externos, como la mantenibilidad, funcionalidad, entre otros atributos. Por las características inherentes al software, sus medidas y métricas son indirectas y, por lo tanto, expuestas al debate [2].

Para las empresas de software, es una necesidad creciente eliminar prácticas def - cientes y reducir la variabilidad en la ejecución de sus procesos de desarrollo. Por lo

tanto, deben abordar planes de mejora de procesos con el objetivo de alcanzar un determinado grado de calidad, en sus procesos y en sus productos software. La mejora de procesos basada en medición promueve la gestión cuantitativa de proyectos de software, mediante el seguimiento continuo de procesos y productos, con el f n de predecir su comportamiento y detectar desviaciones durante su ejecución. Las mediciones, cuando son analizadas, constituyen una base importante para una gestión efectiva por parte del equipo de desarrollo [3], [4].

La gestión cuantitativa de procesos [5] proporciona una visión del grado de cumplimiento de metas así como de las causas que explican desviaciones signif cativas en procesos o productos. El propósito de esta gestión es dirigir un proyecto u organización basado en un conocimiento cuantitativo, es decir medible, determinable, de los aspectos de mayor relevancia, que generalmente son procesos cuyo rendimiento afecta en forma signif cativa al logro de los objetivos del proyecto y la satisfacción de los clientes [4]. La medición permite modif car aquellos factores que aportan una mayor ef cacia en el proceso productivo, haciendo a las organizaciones más ef cientes y permitiendo una ventaja estructural frente a sus competidores [6].

En [7] se señala que atributos como la mantenibilidad y comprensibilidad son evaluados utilizando métricas de software que proveen un modo de representar en números, atributos abstractos como la complejidad y el tamaño. Los mismos autores mencionan que la utilización de una sola métrica es insuf ciente para analizar efectivamente atributos de calidad, por lo que sugieren utilizar un conjunto de métricas para evaluar cada atributo externo de calidad.

Con los datos recogidos en el proceso de medición se genera un repositorio que se mantiene como un recurso organizacional, conservando registros históricos de todos los proyectos aun cuando los datos no se hayan utilizado durante un proyecto particular. Este repositorio permitirá realizar comparaciones entre los proyectos, y las métricas específ cas pueden ser ref nadas de acuerdo con las necesidades organizacionales [8].

Por otra parte, la gestión del conocimiento es un campo que suministra conceptos y herramientas para manejar el conocimiento organizacional. El aprendizaje organizacional está orientado a capturar, almacenar y reutilizar experiencias o conocimiento en una organización.

Enmarcada en la Ingeniería de Software, la gestión del conocimiento es un campo de estudio que busca organizar y representar las experiencias obtenidas en los proyectos de desarrollo, en forma de repositorios de experiencia, de manera que el aprendizaje pueda ser recuperado y reutilizado en la resolución de nuevos problemas [9].

En [10] los autores señalan que se identif can en general las mismas necesidades en las organizaciones que desarrollan y mantienen software: comprender los procesos y productos, evaluar los éxitos y fracasos, aprender de las experiencias, empaquetar las que resultan exitosas, y reutilizar las mismas.

En el proceso de desarrollo de software, las organizaciones generan conocimiento del producto, del proceso y del proyecto. La calidad del software depende en gran medida de la disponibilidad y uso adecuado de este cúmulo de conocimiento [11].

Los modelos de calidad incorporan una base de conocimiento para sustentar la gestión de los proyectos software, tal como propone Competisoft [12]. Este modelo

propone el proceso de Gestión de Recursos, entre los que destaca el conocimiento de la organización. Los autores consideran que el resguardo de este recurso en una base de conocimiento permite aprender de experiencias pasadas, documentando las lecciones aprendidas, para evitar cometer los mismos errores y disminuir el re-trabajo. Esto se constituye en una ventaja competitiva dentro del mercado para la organización.

Por otra parte, estándares internacionales como CMMI-Dev [13], requieren y destacan la importancia de la gestión cuantitativa de proyectos de software. Dicho estándar sostiene que las organizaciones pueden lograr mejoras progresivas en su madurez utilizando tanto datos cualitativos como cuantitativos para la toma de decisiones. En los niveles de madurez gestionados cuantitativamente, la organización y los proyectos establecen objetivos cuantitativos para la calidad y el rendimiento del proceso, y los utilizan como criterios en la gestión de los proyectos.

Medición de software

Cuando se desea conocer alguna característica del software por medio de la medición, se debe tener en cuenta que la medición del software observa al mismo en tres dimensiones diferentes: proyecto, proceso y producto [1]. Este trabajo se contextualiza en la dimensión Producto. La medición del producto consiste en cuantif car atributos de los entregables del software. Dentro de este tipo de métricas, se puede distinguir dos grandes conjuntos:

Métricas clásicas:

Este conjunto de métricas es fácilmente aplicable a código escrito en cualquier paradigma de programación. Entre ellas se menciona a:

LOC (Line of Code): Esta métrica cuenta la cantidad de líneas de código y se convierte en el primer indicador de tamaño de software. Existen tres variantes de la misma: líneas de código que no están comentadas (NCLOC: no comented line of code), líneas de código comentadas CLOC (comented lines of code), y por último, en muchos lenguajes también es posible observar al número de líneas ejecutables ELOC (Executable line of code).

CYCLO (Cyclomatic complexy): Fue propuesta por McCabe en 1996 [14] y constituye el primer indicador de complejidad del software. Se basa en la teoría de grafos y cuenta la cantidad de caminos linealmente independiente que tiene un algoritmo. Diversos estudios demostraron que el valor obtenido por esta métrica debía ser menor que 10 [15]. Constituye una medida de complejidad debido que un alto valor en esta métrica representa un elevado número de caminos para el algoritmo. Esta métrica tiene una fuerte correlación con el número de errores: a mayor complejidad aumenta la probabilidad de cometer errores. Se asocia también con la mantenibilidad dado que si la complejidad del código aumenta, se vuelve incomprensible y menos versátil a la hora de comprender el software por los desarrolladores.

FAN-IN y FAN-OUT: Se aplica a módulos del código: FAN-IN es el número de f ujos que entran al modulo y FAN-OUT el número de f ujos que salen de él [1].

Métricas Orientadas a Objetos

Con el cambio de paradigma en la programación aparecen nuevos objetos de observación en el software como las características de la Programación Orientada a Objetos (encapsulamiento, herencia, polimorf smo, modularidad, abstracción y ocul-

tación). Por ello, diferentes autores plantean nuevos conjuntos de métricas como los que se describen a continuación:

Métricas CK (Chindamber y Kemerer): Se vinculan con la detección de propensión a errores y el análisis de mantenibiliadad de clases [16]. Estas son:

- WMC (Weighted Methods per Class): Métodos ponderados por clase, mide la complejidad de una clase mediante la suma de las complejidades de sus métodos.
- DIT (Depth of Inheritance Tree): Profundidad del árbol de herencia, mide el máximo nivel de jerarquía en un árbol de herencia considerando que el nivel 0 es la raíz del árbol de clases y va aumentando en cuanto aparecen clases heredadas de este nodo.
- NOC (Number of children). Número de hijos es el número de subclases subordinadas a una clase en una jerarquía de árbol. Es un indicador del nivel de reutilización, probabilidad de haber creado abstracciones erróneas y esfuerzo requerido en pruebas.
- CBO (Coupling betwen objects): Se consideran objetos acoplados cuando uno de estos actúa sobre otro, por ejemplo cuando un método de un objeto utiliza un método de otro objeto para realizar una operación. Se considera una métrica útil para predecir el esfuerzo de mantenimiento y de pruebas.
- RFC (Response for a class): Indica el número de métodos que pueden ser potencialmente ejecutados como respuesta a un mensaje recibido por un objeto de esa clase.
- LCOM (Lack of Cohesion in Methods). Falta de cohesión en los métodos establece en qué medida los métodos hacen referencia a los atributos. Es una métrica de cohesión de una clase en base al número de atributos comunes usados por diferentes métodos. Un alto valor de LCOM implica falta de cohesión, es decir escasa similitud entre los métodos. Siendo siempre deseable un alto grado de cohesión.

Métricas MOOD

Su principal objetivo es medir encapsulamiento, herencia y polimorf smo [17]:

- MHF (Method Hiding Factor): Mide la proporción entre métodos def nidos como protegidos o privados y el número total de métodos. Se propone como una medida de encapsulamiento o cantidad relativa de información oculta.
- AHF (Atribute Hidding Factor): Se def ne como el cociente entre la suma de atributos ocultos def nidos en todas las clases y el número total de atributos. Constituye una medida de encapsulamiento.
- MIF (Method Inheritance Factor): Se def ne como el cociente entre la suma de los métodos heredados en todas las clases del sistema y el número total de métodos existentes. Se def ne como una medida de herencia, y por lo tanto como una medida de nivel de reutilización.
- AIF (Atribute Inheritance Factor): Se def ne como el cociente entre la suma total de los atributos heredados en todo el sistema y el número total de atributos. Se considera como un medio para expresar capacidad de reutilización del sistema.

- PF (Polimorphism Factor): Se def ne como el cociente entre el número actual de posibles diferentes situaciones de polimorf smo y el número máximo de posibles situaciones distintas de polimorf smo para la clase.

Atendiendo al contexto y problemática descripta anteriormente, en este trabajo se presenta una aplicación, denominada HEMAC, que aprovecha una herramienta de medición, PHPDepend, disponible en el mercado, y la integra para automatizar la implementación de un proceso de medición def nido previamente. Su principal aporte es brindar información desde un repositorio de valores de métricas, permitiendo evaluar atributos de mantenibilidad del software y mejorar estos aspectos.

2 Metodología

Se elaboró una metodología de medición de atributos de mantenibilidad, utilizando la herramienta PHPDepend, cuya implementación se describe en un trabajo previo de los autores [18]. Esta implementación permitió detectar los siguientes inconvenientes:

- Se requiere adicionar procedimientos manuales para obtener los resultados f nales.
- Es difícil ubicar en el código las entidades (clases, métodos) que tienen valores fuera de los rangos establecidos en las mediciones.
- La organización de la información no favorece un acceso rápido y la interfaz para el usuario no es del todo amigable.

Para superar estas dif cultades, se procedió al análisis y diseño de una herramienta para automatizar la implementación de la metodología de medición elaborada, de modo tal de brindar información acerca de los valores de los atributos de mantenibilidad del software que se desarrolla, permitiendo detectar las entidades con valores fuera del rango en las métricas evaluadas. Se diseñó además una interfaz gráf ca sencilla y amigable al usuario f nal para facilitar su utilización por parte de los gestores de proyectos y/o desarrolladores.

La aplicación presenta las siguientes características:

- Análisis estático de código
  Para el análisis estático de código se utilizó la herramienta PHPDepend, la cual debe estar previamente instalada en el servidor donde se ejecutará Hemac.
  En la metodología original [18], la herramienta PHPDepend es invocada desde una interfaz de línea de comando con los datos necesarios para su ejecución. Como propuesta superadora, Hemac crea un asistente para abstraer al usuario del uso de la herramienta PHPDepend, de manera que el usuario solamente debe ingresar la ruta donde se encuentra el código fuente. Luego se procede al análisis estático de código en modo *background*. Esto permite al responsable de la medición realizar los procesos, de forma ágil y sencilla.
- Medición de código
  Debido a que PHPDepend no proporciona todas las métricas necesarias para implementar la metodología de medición, Hemac incorpora el cálculo de las métricas faltantes, ejecutando *scripts* de medición inmediatamente después del procesa-

miento del análisis estático del código. Esto disminuye el tiempo de la medición y la probabilidad de introducir errores en la misma.

- Persistencia en base de datos
  Una vez realizadas las mediciones, los valores obtenidos de las métricas def nidas en la metodología, se almacenan en una base de datos, constituyendo un repositorio de mediciones. El mismo puede ser accedido para realizar consultas y elaborar informes de medición que brindan al usuario información histórica sobre las mediciones realizadas en el proyecto. Permite además realizar comparaciones con mediciones anteriores.
- Detección de entidades con mediciones fuera de rango
  PHPDepend proporciona umbrales por defecto. Hemac permite su modif cación para adaptarlos a las necesidades de cada proyecto. El uso de umbrales y la persistencia de las mediciones, permite el análisis e identif cación de valores fuera del rango establecido.
  Para la detección de entidades con valores fuera de rango se implementan tablas que contienen los resultados obtenidos en las mediciones de cada una de las entidades procesadas. Las tablas se pueden ordenar por diferentes criterios, por ejemplo, valores de métricas de mayor a menor, permitiendo detectar fácilmente las entidades que obtuvieron como resultado valores que se encuentran fuera del rango deseado.
- Administración de proyectos de medición
  Los proyectos de medición deben ser creados antes de empezar las mediciones de software. De esta forma Hemac permite la administración de varios proyectos de software diferentes al mismo tiempo. Esta característica permite al usuario comparar mediciones de distintas versiones de la aplicación a medida que crecen en funcionalidades o debido al mantenimiento correctivo.

3 Resultados

En esta sección, a modo de ejemplo, se realiza una descripción de las funcionalidades de Hemac y a continuación se muestran los resultados de su aplicación en la evaluación de las distintas versiones del gestor de contenidos Joomla.

Funcionalidades

- **Inicio**: En la Fig. 1 se observa la pantalla de inicio, llamada Dashboard. Contiene un mensaje de bienvenida, un menú en la parte superior y botones de acceso rápido en la parte inferior.

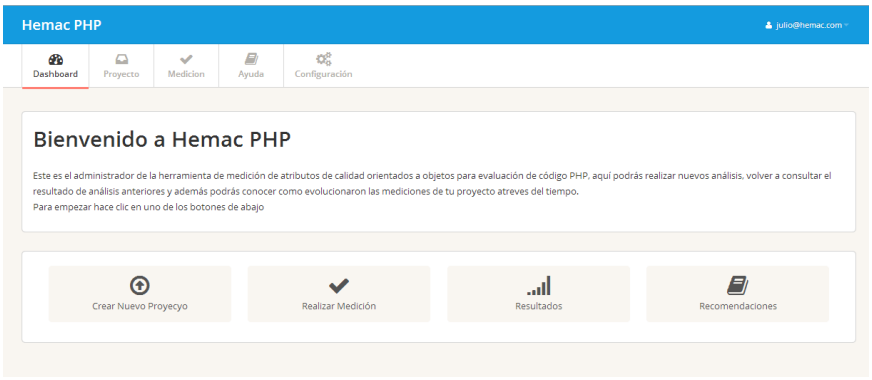

Fig. 1. Pantalla de inicio

- Creación de un proyecto: Para iniciar el proceso de medición es necesario crear un proyecto para permitir el seguimiento del mismo, y agregar y eliminar mediciones.

- Explorador de proyectos: La opción Proyectos/Información brinda una lista de todos los proyectos activos, que permite acceder mediante un menú desplegable a las mediciones de cada uno de los proyectos y una breve descripción de los mismos. La pantalla de esta funcionalidad se muestra en la Fig. 2.

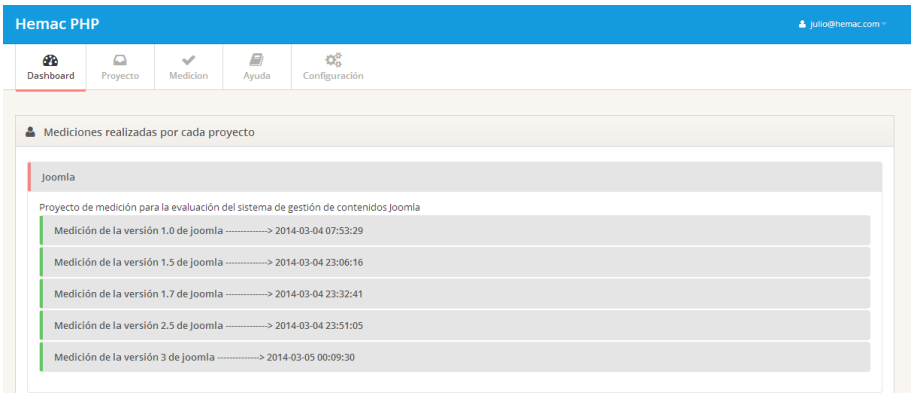

Fig. 2. Vista de explorador de proyectos

- Realizar medición: Esta funcionalidad permite dar inicio al asistente de medición. En la vista se muestra una lista de los proyectos disponibles para la realización de medición. El usuario deberá seleccionar un proyecto para iniciar la medición. Una vez seleccionado, se visualiza el formulario de ingreso de datos para incorporar los datos necesarios.

Una vez incorporados los datos requeridos se inicia el proceso de medición. Finalizado el mismo se muestran los resultados. En la Fig. 3 se puede ver el gráf co Overview Pyramid y en la Fig. 4, el gráf co Abstraction Inestability Chart, correspondiente a una medición.

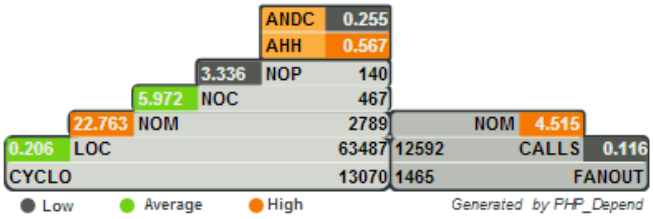

Fig. 3. Gráf co Overview Pyramid

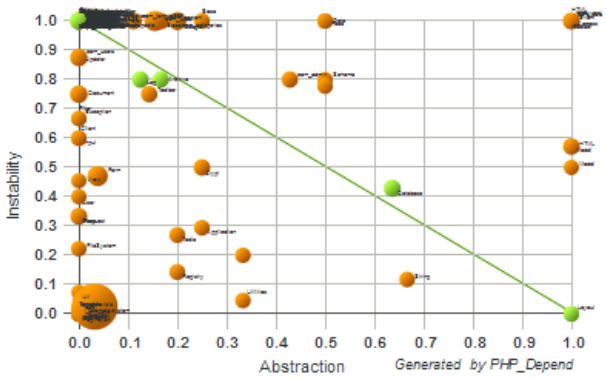

Fig. 4. Gráf co Abstraction Inestability Chart

En la Fig. 5 se puede ver los resultados de las métricas vinculadas con atributos de mantenibilidad, brindadas por la herramienta Hemac en una medición.

MHF	MIF	AIF	CBO	DIT	WMC
0.18645948945616	0.56936736958935	0.6570586308549	3.0856531049251	1.2740899357602	27.259100642398

Alto Promedio Bajo

Fig. 5. Valores de métricas vinculadas con la mantenibilidad

En las Fig. 6, Fig. 7 y Fig. 8 se puede observar el comportamiento de las métricas MHF, MIF y AIF, respectivamente, a lo largo de un proyecto de medición.

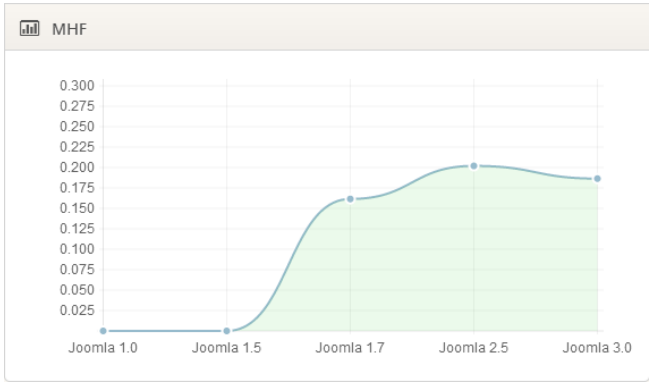

Fig. 6. Gráf co MHF

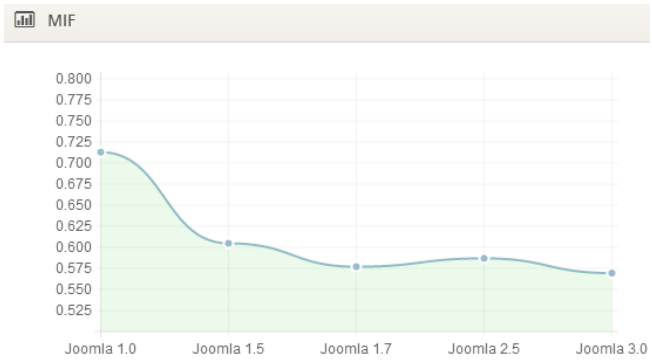

Fig. 7. Gráf co MIF

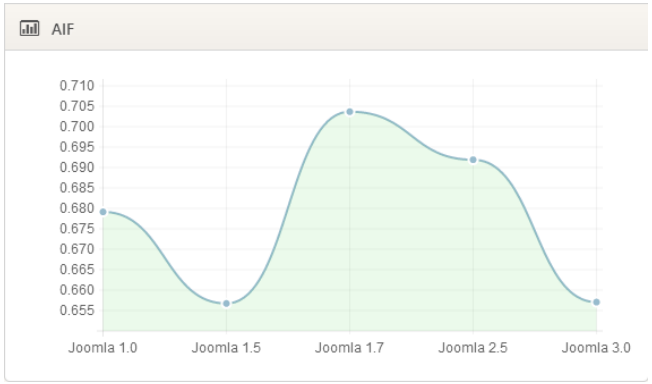

Fig. 8. Gráf co AIF

- Ver resultados: Esta funcionalidad permite ver todas las mediciones realizadas por Hemac con sus respectivas clasif caciones según los umbrales establecidos. Es posible ordenar los valores en las tablas de acuerdo a las métricas que se desean observar mediante los botones del lateral derecho, como se muestra en la Fig. 9.

Fig. 9. Pantalla ver resultados

- Eliminar medición: Para la eliminación de una medición se debe ingresar al menú Medición\Eliminar medición, donde se listan los proyectos con sus respectivas mediciones.
- Conf gurar umbrales: Con esta funcionalidad se pueden modif car los umbrales que Hemac presenta por defecto para su adaptación a cada equipo de desarrollo, tal como se muestra en la Fig. 10.

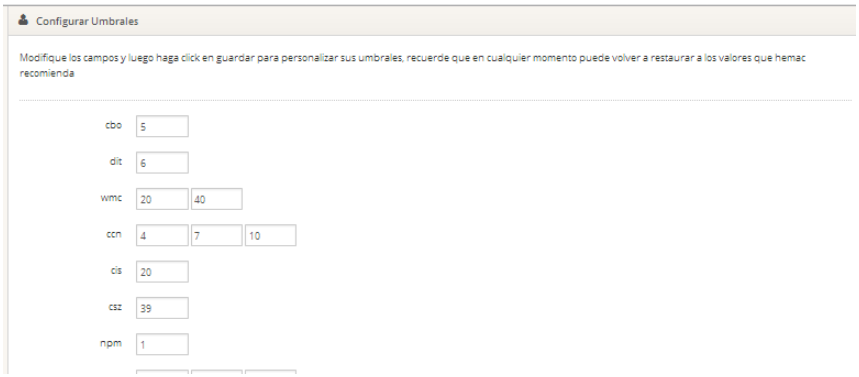

Fig. 10. Conf guración de umbrales

Aplicación de la metodología y herramienta a un software de gestión de contenidos web.

Para precisar la medición orientada a los aspectos de mantenibilidad se utilizó el método GQM (Goal-Question-Metric) definiendo el objetivo, las preguntas y las métricas, según la estructura de la Tabla 1.

Tabla 1: Método GQM

Objetivo	Preguntas	Métricas
Conocer las características de mantenibilidad del software de gestión de contenidos **Joomla!**	**Q1**: ¿Cuál es la densidad de defectos del software?	**M1: MHF** (Proporción de métodos ocultos). Cuando incrementa MHF la densidad de defectos y el esfuerzo necesario para corregirlos disminuye.
	Q2: ¿Cuál es el grado de comprensibilidad del sistema?	**M2: MIF** (Proporción de métodos heredados). El uso de la herencia es visto como un compromiso entre la reusabilidad y la comprensibilidad.
		M3: AIF (Proporción de atributos heredados).
		M4: DIT (Profundidad en árbol de herencia).
	Q3: ¿Cuál es el grado de complejidad del software?	**M5: CBO** (Acoplamiento entre Objetos). Es un indicador del esfuerzo en el mantenimiento y testeo.
		M6: DIT (Profundidad en árbol de herencia). Es una medida de la complejidad de una clase y su potencial de reuso.
		M7: WMC (Métodos ponderados por clase). Es una medida de complejidad de una clase.

Procedimiento para el cálculo de las métricas

Para obtener los valores de las métricas requeridas, se utilizaron los valores resultantes de la herramienta PHPDepend, y se realizó una serie de cálculos adicionales dado que no todas las métricas están consideradas en esta herramienta. Para ello se procedió a la elaboración de algoritmos codif cados en el lenguaje PHP que leen el archivo XML generado por la herramienta PHPDepend y realizan los cálculos necesarios para cada una de las métricas que se explican a continuación:

- **MHF**: El factor de ocultamiento de métodos no es una métrica implementada directamente por PHPDepend por lo que se obtuvo a través de NOM (número de métodos) y NPM (número de métodos no privados). Mediante la resta de NOM-NPM se obtiene el número de métodos privados de una clase. Luego se sumó el total de métodos privados y se dividió por NOM para obtener la proporción de métodos privados con respecto al total, logrando así el valor de MHF, tal como se indica en la siguiente fórmula, donde i es la cantidad de clases y j la cantidad de paquetes.

$$\frac{\sum_{j=1}^{m}\sum_{i=1}^{n}(NOM_i - NPM_i)_j}{\sum_{j=1}^{m}\sum_{i=1}^{n}NOM_{i\,j}} \; ; \quad \sum_{j=1}^{m}\sum_{i=1}^{n}NOM_{i\,j} > 0$$

Fórmula 1. Cálculo de MHF

- **MIF:** De la misma forma, para obtener la proporción de métodos heredados, se obtuvo la cantidad de métodos heredados utilizando las métricas NOAM (número de métodos añadidos) y NOM, realizando la resta NOM - NOAM. Luego se obtuvo la proporción de métodos heredados, obteniendo el valor de MIF.

$$\frac{\sum_{j=1}^{m}\sum_{i=1}^{n}(NOM_i - NOAM_i)_j}{\sum_{j=1}^{m}\sum_{i=1}^{n}NOM_{ij}} ; \sum_{j=1}^{m}\sum_{i=1}^{n}NOM_{ij} > 0$$

Fórmula 2. Cálculo de MIF

- **AIF:** Para el cálculo de la proporción de atributos heredados se utilizaron las métricas VARS (cantidad de atributos de una clase) y VARSI (Cantidad de atributos heredados). VARS solo contabiliza la cantidad de atributos agregados en una clase, por lo que para obtener la cantidad total de atributos fue necesario sumar VARS+VARSI. Para obtener la proporción de atributos heredados, se divide VARSI/(VARS+VARSI).

$$\frac{\sum_{j=1}^{m}\sum_{i=1}^{n}(VARSI_i)_j}{\sum_{j=1}^{m}\sum_{i=1}^{n}VARS_i + VARSI_{ij}} ; \sum_{j=1}^{m}\sum_{i=1}^{n}NOM_{ij} > 0$$

Fórmula 3. Cálculo de AIF

- **CBO, DIT y WMC** son métricas implementadas por la herramienta utilizada, por lo que se sumó el valor total arrojado por la medición para cada una de las clases y luego se dividió por la cantidad de clases para establecer la media.
  En el caso de encontrarse con denominadores nulos, el resultado de la medición es igual a 0 [1].

Validación de la herramienta

Para la validación de la metodología y la herramienta desarrollada, se eligió el sistema de gestión de contenidos Joomla, dado que es un software de notable crecimiento en los últimos años y es posible acceder a su código fuente. Para este trabajo, se consideraron las versiones 1.0, 1.5, 1.7, 2.5 y 3.

Con los resultados obtenidos es posible inferir el grado de mantenibilidad, anticipando el esfuerzo de actualización y corrección de errores que podrían tener las futuras versiones. Los valores obtenidos para las distintas métricas en cada una de las versiones, se muestran en la tabla 2.

Tabla 2. Resumen de mediciones de versiones de Joomla

Métrica	Joomla! 1.0	Joomla! 1.5	Joomla! 1.7	Joomla! 2.5	Joomla! 3
M1:MHF	0	0	0.161535675	0.2020169160	0.1864594894
M2:MIF	0.7129431866	0.6047008547	0.5769648678	0.586857514	0.5693673695
M3:AIF	0.6791426743	0.656723963	0.7036745406	0.691907364	0.657058630
M4:CBO	1.302670623	2.090146750	3.409090909	3.553879310	3.089935760
M5:DIT	0.7477744807	1.197064989	1.417112299	1.875	1.274089935
M6:WMC	27.64688427	34.27253668	35.83689839	31.39008620	27.25910064

Los valores obtenidos permiten responder a las tres preguntas que aportan información para lograr el cumplimiento del objetivo planteado por el método Hemac.

Q1: ¿Cuál es la densidad de defectos del software?

M1) MHF: Se observa que la proporción de métodos ocultos aumenta de 0% en la versión 1.0 a casi 18% en la versión 3. Como se señaló, se demostró empíricamente que cuando incrementa el valor de esta métrica, la densidad de defectos y el esfuerzo necesario para corregirlos disminuye.

Respuesta: Debido a que en todas las versiones, el valor medido aumenta y entre las mediciones 2.5 y 3 solo disminuye menos del 2%, se puede concluir que disminuye la densidad de defectos del proyecto Joomla y, consecuentemente, el esfuerzo necesario para corregirlos.

En las versiones 1.0 y 1.5, la herramienta Hemac permite comprobar que los valores de MHF dan 0 porque no existen métodos privados declarados. De acuerdo a la Formula 1, si los valores NOM y NPM son iguales, el numerador es 0 y por lo tanto el resultado de la sumatoria dará el mismo número. Algunos valores de medición se pueden observar en la Fig. 11, en la cual se aprecia que los valores de las columnas NOM y NPM son iguales.

NOAM	NOCC	NOM	NOOM	NPM	RCR	VARS
0	0	15	0	15	0.15	0
0	0	1	0	1	0.15	0
0	0	10	0	10	0.15	7
0	0	2	0	2	0.15	0
6	0	6	0	6	0.181875	2
5	0	5	0	5	0.181875	2
0	0	1	0	1	0.15	0
0	0	14	0	14	0.15	6
0	0	4	0	4	0.15	0
0	0	3	0	3	0.15	0
0	0	9	0	9	0.15	0
0	0	7	0	7	0.15	0

Fig. 11. Valores de las métricas en la medición de Joomla 1.0

Q2: ¿Cuál es el grado de comprensibilidad del sistema?

M1) MIF: Si bien en la versión 1.0, se observa un valor del 70%, en las versiones siguientes, el valor se mantiene estable entre 50% y 60%.Altos niveles de MIF implican un buen nivel de reuso.

M2) AIF: Se observa una alta proporción en la herencia de atributos superior al 60% lo que indica también buen nivel de reuso.

M3) DIT: La profundidad en el árbol de herencia tiene valores que no superan el umbral, lo que indica un adecuado nivel de complejidad.

Respuesta: Si bien existe un cambio entre las versiones 1.0 y las siguientes en MIF, luego se observa que se mantiene estable en valores altos, al igual que en herencia de atributos. Valores altos en métricas de herencia podrían signif car posibles costes en comprensibilidad del sistema. Sin embargo, si se considera la métrica que señala profundidad en el árbol de herencia (DIT), se observa que esta se mantiene dentro de los valores recomendados, por lo cual se puede sostener que se mantiene el equilibrio entre reuso y complejidad. Se puede inferir que la conjunción de estos valores, señala un buen grado de comprensibilidad del sistema.

Q3: ¿Cuál es el grado de complejidad del software?

M1) CBO: Se observa que su valor no sobrepasa el umbral, indicando un nivel de acoplamiento conveniente, lo cual se traduce en un adecuado nivel de complejidad en las relaciones de los objetos.

M2) DIT: Niveles muy altos de profundidad en el árbol de herencia pueden indicar complejidad en las clases. Sin embargo, se observa que los valores obtenidos están dentro del umbral recomendado.

M3) WMC: Este valor señala la complejidad de los métodos y por ende la de las clases, se encuentra dentro del rango recomendado.

Respuesta: Las tres mediciones observadas para esta pregunta tienen valores dentro de los umbrales deseados, por lo que se concluye que la aplicación no posee un alto grado de complejidad.

4 Conclusiones

El método y la herramienta propuestos en este trabajo permiten evaluar productos software y obtener un producto de medición que aporta indicadores sobre atributos de mantenibilidad de una manera sencilla y efectiva, facilitando su utilización por parte de gestores de proyecto y desarrolladores. Además, el método GQM brinda una estructura adaptable para la agregación de nuevas métricas o nuevas dimensiones de observación.

La posibilidad de consultar tablas y gráf cos que Hemac proporciona, conf gura un valor agregado importante a la hora de elaborar informes de medición, dado que permite una rápida comprensión de las relaciones y detección precisa de las entidades del código que presentan valores fuera de los esperados.

Como trabajo futuro se plantea implementar con esta herramienta procesos de medición en ambientes reales de desarrollo.

5 Referencias

1. Piattini, M.; Garzas, J.; García, F.; Género , M. “Medición y estimación del software”. ISBN: 9789701514139 - Editorial ALFA OMEGA. 2008.

2. Pressman, R. S. “Ingeniería de Software. Un enfoque práctico”. Editorial MCGRAW -HILL -2005

3. Baldassarre, T., Boffoli, N., Caivano, D., &Visaggio, G. (2004). Managing Software Process Improvement (SPI) through Statistical Process Control (SPC).*Lecture Notes in Computer Science*, 3009, 30-46

4. Ardila, C. & Pino, F. (2013). Panorama de gestión cuantitativa de procesos de desarrollo de software en pequeñas organizaciones. Revista S&T, 11(26), 29-46.

5. Gou, L., Wang, Q., Yuan, J., Yang, Y., Li, M., & Jiang, N. (2009). Quantitative defects management in iterative development with Bi Defect. Software Process Improvement and Practice, 14(4), 227-241

6. Hernández Ballesteros, J.F., Minguet Melián, J. M. "La Medida de la Calidad del Software como Necesidad y Exigencia en Modelos Internacionales (CMMI, ISO 15504, ISO 9001). www.issi.uned.es/CalidadSoftware/Noticias/PonIng2005.rtf

7. Herbold, S.; Grabowski, J.; Waack, W. "Calculation and optimization of thresholds for sets of software metrics". Empir Software Eng. 2011.

8. Ian Sommerville. "Ingeniería del Software". 7º Ed. Pearson Educacion S.A., Madrid 2005.

9. Anaya, R.; Cechich, A.; Henao, M.; Oktaba, H. "Enfoque Integrado de la Gestión del Conocimiento en el Modelo de Procesos de COMPETISOFT". Informe IT.11. CYTED. 2006.

10. Basili, V.; Costa, P.; Lindvall, M.; Mendonca, M. Seaman, C.; Tesoriero, R.; Marvin Zelkowitz, M. "An Experience Management System for a Software Engineering Research Organization". 26 th Annual NASA Goddard Software Engineering Workshop. 20001: pp. 26 -35.

11. Lindvall, M. y Rus, I. "Lessons Learned from Building Experience Factories for Software Organizations". Wissensmanagement 2003: pp. 59-63.

12. Oktaba , H; Piattini, M.; Pino, F.; Orozco, M.; Alquicira, C. "COMPETISOFT: Mejora de Procesos Software para Pequeñas y Medianas Empresas y Proyectos". Alfaomega Ra -Ma. 2009. (pp. 33).

13. CMMI para Desarrollo, Versión 1.3 (CMMI-DEV, V1.3). Mejora de los procesos para el desarrollo de mejores productos y servicios. TECHNICAL REPORT. CMU/SEI-2010-TR-033. Noviembre 2010.

14. McCabe, T.; Waston, A. "Structured Testing: A Testing Methodology," in National Institute of Standards and Technology, Septiembre 1996.

15. Grabowski, J.; Waack, S.; Herbold, S. "Calculation and optimization of tresholds for sets of software metrics". Empir Software Eng, 2011.

16. Kemerer, C.F.; Chindamber, S.R. "A metric suite for object oriented design.," in IEEE Transactions on Software Engenieering, 1194, pp. 467-493.

17. Abreu, F.; Melo Brito, W. "Evaluating the impact of Object-Oriented Design on Software Quality," in Proceeedings of 3rd International Software Metric Symp, Berlin, 1996.

18. Acosta, J; Dapozo, G.; Greiner, C.; Estayno, M . "Evaluación de mant enibilidad de un gestor de contenidos open source utilizando métricas de orientación a objetos". An ales de las 42JAIIO Jornadas Argentinas de Informática. 10º Jornadas Argentinas de Software Libre. ISSN 1850-2857 Pp. 15-29. Facultad de Matemática, Astronomía y Física de la Universidad de Córdoba (UNC). Córdoba. 16 al 20 de Septiembre de 2013.