Revista Española de Innovación, Calidad e Ingeniería del Software

REICIS

Volumen 6, Número 3 (especial XI JICS), noviembre, 2010

Web de la editorial: www.ati.es
Web de la revista: www.ati.es/reicis
E-mail: calidadsoft@ati.es
ISSN: 1885-4486

Publicado por la Asociación de Técnicos de Informática (ATI), Via Laietana, 46, 08003 Barcelona.

Secretaría de dirección: ATI Madrid, C/Padilla 66, 3º dcha., 28006 Madrid

Contenidos

REICIS

Análisis de métricas básicas y herramientas de código libre para medir la mantenibilidad

Emanuel Irrazábal[1,2], Javier Garzás[1,2]

1. Kybele Consulting S.L.
{emanuel.irrazabal,javier.garzas}@kybeleconsulting.com

2. Kybele Research
{emanuel.irrazabal,javier.garzas}@urjc.es

Resumen

Durante los últimos años, la calidad se ha convertido en uno de los temas principales de la Ingeniería del Software. En ese sentido, el control de calidad debe realizarse desde un punto de vista cuantitativo y cualitativo siendo necesario establecer mediciones sistemáticas en todo el ciclo de vida del producto software. La mayoría de las pequeñas organizaciones no tienen la capacidad de desarrollar o adquirir herramientas para verificar la calidad de sus productos software y a la vez desarrollar el software que comercializan y que constituye su principal negocio. En ese contexto, las herramientas de código abierto emergen como una opción para obtener el soporte técnico con el cuál recolectar los datos. En este trabajo, se estudia cómo las herramientas de código abierto cumplen con las necesidades de medición de la calidad de acuerdo con la norma ISO/IEC 9126. Nos hemos enfocado en la característica de mantenibilidad debido a su relevancia histórica y a su impacto directo en los costes totales.

Palabras clave: ISO 9126, mantenibilidad, métricas del producto, herramientas

Abstract

During the last years, quality is becoming a hot topic in the context of Software Engineering. The quality control should be done from a quantitative and qualitative point of view, for what it is necessary to establish measurement systems throughout the product lifecycle. Most small organizations can not cope with the development of tools to check the quality of their software products plus the development of the software that constitutes their main business. In this context, open-source tools emerge as the answer to provide with the technical support to collect the information needed to assess the quality of software assets. In this work, we review how existing open-source tools fulfill the needs for quality measures raised when you want to assess product quality according to the ISO/IEC-9126 standard. We focus on maintainability, since it has been historically recognised as one the most relevant, given its direct impact over costs.

Key words: ISO 9126, maintainability, product metrics, tools.

Irrazábal, E. y Garzás, J., "Análisis de métricas básicas y herramientas de código libre para medir la mantenibilidad", REICIS, vol. 6, no.3, 2010, pp.56-65. Recibido: 8-11-2010; revisado: 14-11-2010; aceptado: 24-11-2010

1. Introducción

En un mercado competitivo y en evolución como el actual, la calidad del software y la medición del software son cada vez más importantes [1]. La calidad en el software tiene influencia directa sobre los costes finales, considerándoselo también un factor de diferenciación entre las organizaciones que desarrollan [2].

La calidad del software puede ser descrita desde diferentes puntos de vista, cuando se trata del desarrollo software la calidad se relaciona tradicionalmente con el producto software y con el proceso de desarrollo [3]. En este sentido, existe una tendencia global de institucionalizar las prácticas de calidad en los desarrollos software. Esta tendencia se evidencia en las acciones emprendidas por las organizaciones de desarrollo para cumplir con modelos de referencia bien conocidos, como CMMI-DEV [4] o ISO/IEC 15504 [5]. Un caso particular son las 198 organizaciones de desarrollo software que en España se encuentran certificadas en el modelo CMMI hasta septiembre de 2010 [4] y que posicionan al país en el quinto lugar a nivel mundial y el primer lugar en Europa. Sin embargo, la principal crítica al proceso de evaluación de la calidad ha sido la falta de pruebas para demostrar que seguir un modelo de referencia para los procesos software garantiza la calidad del producto software resultante [6]. De hecho, la institucionalización del proceso de desarrollo puede incluso institucionalizar los malos resultados. En 2008 Maibaum y Wassyng señalaron que las evaluaciones de la calidad deben basarse en evidencias extraídas directamente de los atributos del producto software en lugar de evidencias tomadas del proceso de desarrollo [7]. Como respuesta a esto, han surgido diferentes modelos para la evaluación de la calidad del producto software [8][9][10]. Sin embargo, no hay consenso sobre cómo este tipo de modelos han de ser utilizados en las organizaciones.

En particular, la Organización Internacional de Normalización (ISO) publicó la norma ISO / IEC 9126:1991 Ingeniería de Software - Calidad del producto [8]. Esta norma especifica un modelo de calidad general, identificando las siguientes características de calidad: funcionalidad, fiabilidad, eficiencia, usabilidad, mantenibilidad y portabilidad, y la forma en que se descomponen en subcaracterísticas. Cabe destacar que existe actualmente una nueva norma asociada con la calidad del producto, la norma ISO/IEC 25010 [9] pero que actualmente se encuentra en borrador (estado 50.20) de acuerdo al catálogo oficial de

ISO[1]. Por todo ello este trabajo se centrará en la característica de mantenibilidad de la norma ISO/IEC 9126, ya que ha sido históricamente reconocida como uno de los más relevantes, teniendo en cuenta su impacto directo sobre el coste de desarrollo y mantenimiento del producto software. Estudios previos señalan al mantenimiento como la fase que más recursos requiere a lo largo del ciclo de vida del producto, dos veces superior a los costes de desarrollo [11][12]. A pesar del hecho de que el modelo ISO 9126 no especifica un conjunto de medidas para evaluar la capacidad de mantenimiento de productos software [13] sí establece las características deseables de dichas mediciones. Tienen que ser objetivas, independientes y reproducible, expresada por medio de escalas válidas y suficientemente precisas para apoyar comparación fiable [14]. Con estas premisas, el enfoque más conveniente parece el uso de herramientas comerciales para la captura de las mediciones. Sin embargo, algunas organizaciones no pueden permitirse estas herramientas. Varios estudios han revelado que la aplicación de modelos de calidad en las pequeñas y medianas empresas (PYME) es una tarea muy difícil [15][16] ya que implica grandes inversiones de dinero, tiempo y recursos humanos. Por consiguiente, en estas organizaciones es necesario adaptar las prácticas de ingeniería de software a su tamaño y a la naturaleza de sus actividades [17]. En particular, las PYMEs han buscado el apoyo de los proyectos de código abierto para automatizar la evaluación de la calidad del software que desarrollan. Las soluciones de código abierto han demostrado ser tan eficaces como las comerciales [18][19] mientras que implican un menor coste de licencia y mantenimiento.

Este trabajo evalúa el apoyo técnico de herramientas de código abierto actuales que recogen métricas básicas para evaluar la característica de mantenibilidad especificada en ISO/IEC 9126. La sección 2 presenta un resumen de las métricas básicas propuestas por los modelos de medición actuales y la sección 3 detalla los criterios de selección y lista las herramientas obtenidas. Por último en la sección 4 se resumen las conclusiones del estudio.

2. Elección de las métricas asociadas a la mantenibilidad

Debido a la imposibilidad de encontrar un único modelo de medición lo suficientemente reconocido y que detalle el conjunto de métricas básicas recomendadas para medir la mantenibilidad, se presenta un resumen de los modelos de medición más destacados. Se

[1] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35733

han recopilado diversos modelos de medición de la mantenibilidad que relacionan un conjunto de métricas de calidad obtenidas a partir del código fuente con las subcaracterísticas de la mantenibilidad. Se han tenido en cuenta los modelos de medición de herramientas libres y de código propietario, modelos de medición reconocidos por entidades de certificación, así como otros modelos de medición [20][21][22][23][24][25].

Métricas de calidad	Analizabilidad	Cambiabilidad	Estabilidad	Capacidad de ser probado
Complejidad ciclomática (CC)	X	X		X
Cantidad de las instrucciones (LOC)	X			
Promedio del tamaño de las instrucciones (PLOC)	X	X		X
Frecuencia de los comentarios (PCOM)	X			
Peso de los métodos por clase (WMC)	X			
Número de clases base (NCB)	X			
Número de saltos incondicionales (NSI)		X	X	X
Número de niveles anidados (NNA)		X		X
Acoplamiento entre objetos (CBO)	X	X	X	X
Ausencia de cohesión (LCOM)	X	X	X	X
Profundidad del árbol de herencia (DIT)	X	X	X	X
Componentes llamados directamente (CCD)			X	
Número de hijos (NOC)	X	X	X	X
Número de salidas de estructuras condicionales (NSE)				X
Respuestas de una clase (RFC)				X
Número de mensajes enviados en un método (NOM)	X	X	X	X
Cobertura de las pruebas unitarias (COB)			X	X
Número de errores de las pruebas unitarias (UTE)			X	X
Violaciones en el código fuente (VCF)	X			
Violaciones de estilo (VST)	X			
Distancia a la secuencia principal (D)		X	X	
Número de dependencias cíclicas (CYC)	X	X		
Acoplamiento aferente y eferente (CAF, CEF)		X	X	
Código duplicado (CDU)	X	X		

Tabla 1. Resumen de las métricas internas relacionadas con las subcaracterísticas de la mantenibilidad de acuerdo a la norma ISO/IEC 9126

Como resultado se ha obtenido la Tabla 1 que resume el conjunto de métricas internas básicas que se encuentran relacionadas con las subcaracterísticas de analizabilidad,

cambiabilidad, estabilidad y capacidad de ser probado, las cuales definen a la característica de mantenibilidad de acuerdo con la norma ISO/IEC 9126. Se han agregado acrónimos a las métricas para facilitar la construcción de tablas posteriores.

3. Elección de las herramientas

Una vez obtenido el resumen de las métricas básicas se ha pasado a la selección de las herramientas de código libre con las que se puede analizar el código fuente del producto software a ser evaluado. Se ha seguido el enfoque utilizado en [26], revisando los repositorios de proyectos de código abierto existentes y seleccionando el repositorio principal a examinar. La búsqueda se ha complementado con la selección de herramientas realizada en trabajos similares.

De acuerdo con estudios previos [27] el repositorio SourceForge alberga actualmente la mayor cantidad de proyecto de código abierto sobre otros como Codeplex, Google Code o Kenai. En particular, apoya a más de 260.000 proyectos de software con una comunidad de más de 2.7 millones de usuarios registrados. A continuación, se han identificado las palabras típicas con las que llevar a cabo la búsqueda en el repositorio y así obtener las herramientas de medición más valoradas por los usuarios. De acuerdo con la lectura de la norma ISO 9126 se ha concluido una serie de palabras claves utilizadas para realizar la búsqueda en el sitio Web de www.sourceforge.net. En las búsquedas se han tenido en cuenta solo los lenguajes de programación más populares. Se han identificado las herramientas asociadas con los lenguajes .NET (tanto en VB.NET y C #), Java y C/C++, ya que estos son los marcos de desarrollo más populares de acuerdo con el índice TIOBE[2], actualizado en noviembre de 2010.

En la Tabla 2 se enumeran las principales herramientas encontradas de acuerdo con el nivel de actividad de la comunidad en términos de número de visitas en el sitio Web correspondiente, las entradas de los foros relacionados con cada herramienta, descargas, etc. Estas estadísticas han sido obtenidas en enero de 2010.

[2] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Herramientas	% de Actividad	Descargas	Frase de búsqueda	Lenguaje de programación
• CheckStyle	• 99.94%	• 4.473.961	• Source Code Analysis	• JAVA
• FindBugs	• 99.93%	• 606.868	• Static Analysis	• JAVA
• PMD	• 99.38%	• 508.312	• Source Code Analysis	• JAVA
• CCCC	• 92.76%	• 64.523	• Source Code Analysis	• C
• JUnit	• 99.74%	• 2.900.703	• Test	• JAVA
• NUnit	• 99.68%	• 1.908.561	• Test	• .NET
• CPPUnit	• 98.97%	• 545.474	• Test	• C
• EMMA	• 99.95%	• 1.487.428	• Source Code Analysis, Test	• JAVA

Tabla 2. Herramientas encontradas

Herramientas	Métricas calculadas
JavaNCSS	CC, LOC
PMD/CPD	VCF, CDU
CheckStyle	VST
FindBugs	VCF, CDU
JDepend	CYC,D,CAF,CEF
CCCC	CBO,DIT,NOM, NOC ,NOM ,RFC, LOC, PCOM, CC, WMC, D,CEF, CAF
StyleCop	VCF
FxCop	VST
JUnit	UTE
CPPUnit	UTE
NUnit	UTE
EMMA	COB
Analyst4j	CC, LOC, PLOC, PCOM, CBO,DIT,LCOM,NOC,NOM,RFC,WMC
C&K Java Metrics	CBO, DIT, LCOM, NOC, NOM, RFC
Dependency Finder	LOC, NNA, CCD, DIT, NOC, NOM
Eclipse Metrics	CC, LOC, CAF, CEF, D, LCOM, CEF, DIT, NOC, NOM
OOMeter	CBO, DIT, LCOM, NOC
Semmie	DIT, LCOM, NOC, NOM, RFC
Understand for Java	CBO,DIT, LCOM, NOC, NOM
VizzAnalyzer	LOC, CBO, DIT, LCOM, NOC, NOM, RFC, WMC
CodePro AnalytiX	CC, LOC, PLOC, PCOM, NCB, NSI, NNA, CCD, D, CYC, CAF, CEF, CBO, DIT, LCOM, NOC, NOM, RFC, WMC, CDU

Tabla 3. Resumen de las herramientas

Para completar la selección anterior, se incluyen herramientas adicionales. En primer lugar, se han considerado las herramientas de análisis estático de código de software libre para. NET, FxCop y StyleCop [28]. También se incluyen herramientas que miden el

acoplamiento y la complejidad ciclomática ya que han sido ampliamente reconocidos como los indicadores más valiosos para evaluar la capacidad de mantenimiento de sistemas orientados a objetos [29]. Se han incluido las herramientas JavaNCSS y JDepend como las más representativas de estas métricas, aunque hayan podido ser incluidas otras similares. Por último se incluyen otras herramientas libres analizadas en otros trabajos [30][31]. En la Tabla 3 se enumeran todas las herramientas seleccionadas y se especifican las mediciones que cada una realiza.

4. Conclusión

Una vez analizados los resultados obtenidos se ha concluye que las herramientas actuales de código abierto sirven como un buen punto de partida hacia la obtención de los indicadores necesarios para apoyar el modelo de medición descrito en la norma ISO 9126. Con el fin de simplificar el uso de estas herramientas, la gran mayoría de ellas pueden ser incorporadas directamente en IDEs de desarrollo. No todas las herramientas obtienen los mismos resultados [30] ni tienen la misma fiabilidad en cuanto a falsos positivos y falsos negativos a la hora de buscar violaciones en el código fuente [31], por lo que es indispensable una tarea de selección de herramientas. Se destaca a la herramienta Code Pro AnalytiX, que evalúa código fuente escrito en lenguaje Java, como una de las más completas en cuanto a cantidad de métricas básicas que obtiene. Aún así, es conveniente comprobar antes la precisión de todas estas herramientas en casos reales, especialmente a las que realizan cálculos más complejos.

Para trabajos futuros se considera importante continuar con este tipo de análisis, estudiando la relación entre las herramientas de código libre y las necesidades de los modelos de procesos existentes, como el Proceso de Medición y Análisis (PMA) de CMMI o el proceso de medición descrito en la norma ISO / IEC 12207:2008 [14]. En este caso será necesario también incluir otro tipo de herramientas, para derivar métricas y relacionar los resultados con las necesidades de negocio de la organización.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Centro para el Desarrollo Tecnológico Industrial (CDTI) (IDI-20090175), dependiente del Ministerio de Ciencia e Innovación y el

proyecto MODELO-CAOS (TIN2008-03582/TIN) financiado por el Ministerio de Ciencia y Educación (TIN2005-00010/).

Referencias

[1] Rifkin, S., "Guest editor's introduction: software measurement", *IEEE Software*, vol. 26 nº 3, pp. 70, 2009.

[2] Piattini, M., García, F., Garzás, J. and Genero, M., *Medición y estimación del software: técnicas y métodos para mejorar la calidad y productividad del software*, Ra-ma, 2008.

[3] Garvin, D.A., "What does "product quality" really mean?", *Sloan management review*, vol. 26, p.25, 1984.

[4] ISO, *ISO/IEC 15504-5 Information technology — process assessment — part 5: an exemplar process assessment model*, ISO, 2006.

[5] Ebert C., "Guest editor's introduction: how open source tools can benefit industry", *IEEE Software*, vol. 26, pp. 50-51, 2009.

[6] Kitchenham, B. and Plfeeger, S.L., "Software quality: the elusive target", *IEEE Journal*, vol. 13 nº 1, pp. 12-21,1996.

[7] Maibaum T, Wassyng, A.,"A product-focused approach to software certification", Computer, vol. 41 nº 2, pp. 91-93, 2008.

[8] ISO, *ISO/IEC Std. 9126 Software product evaluation–quality characteristics and guidelines for their use*, ISO, 2001.

[9] ISO, *ISO/IEC Std. 25010 Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models*, ISO, 2010.

[10] Milicic, D. y otros, *Software quality attributes and trade-offs*, Blekinge Institute of Technology, 2005.

[11] Frazer, A., "Reverse engineering- hype, hope or here?", Software Reuse and Reverse Engineering in Practice, vol. 12, pp. 209-243. 1992.

[12] Pressman, R. *Ingeniería del software: un enfoque práctico*, McGraw-Hill, 2002

[13] Boehm, B., Characteristics of software quality, North Holland Press, 1978.

[14] ISO, *ISO/IEC 12207 Systems and software engineering - software life cycle processes*, ISO, 2008.

[15] Saiedian, H. and Carr, N., "Characterizing a software process maturity model for small organizations" , *ACM SIGICE Bulletin*, vol. 23 nº 1, pp. 2-11, 1997.

[16] Staples, M., Niazi, M., Jeffery, R., Abrahams, A., Byatt, P. and Murphy, R., "An exploratory study of why organizations do not adopt CMMI", *Journal of systems and software*, vol. 80 nº 6, pp. 883-895, 2007.

[17] Dybå, T, "Factors of software process improvement success in small and large organizations: an empirical study in the scandinavian context". En *European Software Engineering Conference (esec) / Foundations of Software Engineering (sigsoft fse), Helsinki (Finlandia), 1-5 de septiembre de 2003*, pp.148-157, 2003.

[18] SEI, *Process maturity profile - cmmi for development scampi class a appraisal results 2010 mid-year update - septiembre 2010*, SEI, 2010.

[19] Hawkins, R.E., "The economics of open source software for a competitive firm", *Netnomics*, vol. 6 nº 2, pp. 103-117, 2004.

[20] Luijten, B., Visser, J. and Zaidman, A., "Faster defect resolution with higher technical quality of software", *Proceeding of the 4th International Workshop on Software Quality and Maintainability (SQM'10), Madrid (España), 15-18 de marzo de 2010*, pp. 11-20 ,2010.

[21] Alshayeb, M., "Empirical investigation of refactoring effect on software quality" , *Information and software technology*, vol. 51, pp.1319-1326, 2009.

[22] Heitlager, I., Kuipers, T. and Visser, J., "A practical model for measuring maintainability". En *Quality of Information And Communications Technology (Quatic 2007), Lisboa (Portugal), 12-14 de septiembre de 2007*, pp. 30 - 39, 2007.

[23] Kataoka, Y., Imai, T., Andou, H. and Fukaya, T. 2002. "A quantitative evaluation of maintainability enhancement by refactoring", En *18th IEEE Iinternational Conference On Software Maintenance (ICSM'02),Montreal (Canadá), 3-6 de octubre de 2002*, pp.576-585, 2002.

[24] Mouchawrab, S. and Briand, L.C. "A measurement framework for object-oriented software testability", *Information and software technology*, vol. 47 nº 15, pp. 979-997, 2005.

[25] Samoladas, I., Gousios, G., Spinellis, D. and Stamelos, I., "The SQO-OSS quality model: measurement based open source software evaluation". *Open source development, communities and quality*, vol. 275, pp. 237-248. 2008.

[26] von Wangenheim C.G, Hauck, J., von Wangenheim, A., "Enhancing open source software in alignment with CMMI-Dev", *IEEE Software*, vol. 26 nº 2, pp. 59-67, 2009.

[27] Dror, G.F., Gillian, Z.H. and Stephen, R.S., "An empirically-based criterion for determining the success of an open-source project", IEEE Computer Society, Proceedings of the Australian Software Engineering Conference, Sidney (Australia), 18-21 de abril 2006, pp.363-368, 2006.

[28] Mitchell, S., "Static analysis tools for .net", MSDN Magazine, vol. 94, 2008.

[29] Li, W. and Henry S., "Object-oriented metrics that predict maintainability". Journal of systems and software, vol. 23 nº 2, pp. 111-122. 1993.

[30] Lincke, R., Lundberg, J. and Löwe, W., "Comparing software metrics tools". En ACM, Proceedings of the 2008 International Symposium on Software Testing and Analysis, Seattle (USA), 20-24 de julio de 2008, pp. 131-142, 2008.

[31] Plösch, R., Mayr, A., Pomberger, G. and Saft, M., "An approach for a method and a tool supporting the evaluation of the quality of static code analysis tools", *In proceedings of SQMB 2009 Workshop, held in conjunction with SE 2009 conference*, Kaiserslautern (Alemania), 2 de marzo de 2009, pp. ,2009.