ISSN 0717-9103
ISSN Online 0718-8307

APLICACIÓN DEL MODELO MGRSOFT EN LA REUTILIZACIÓN DE REVISIONES A REQUSITOS DE SOFTWARE.

THE APPLICATION OF THE MODEL MGRSOFT IN REUSE OF REVISIONS TO SOFTWARE REQUIREMENTS.

Martha Dunia Delgado Dapena[1] , Indira Chávez Valiente[2], Yucely López Trujillo[3].
[1]Dra. Profesora Titular, Centro de Estudios de Ingeniería de Sistemas (CEIS), Facultad de Ingeniería Informática, Instituto Superior Politécnico "José Antonio Echevarría" (CUJAE), Habana, Cuba. marta@ceis.cujae.edu.cu
[2]Ing. Profesora Instructora, Centro de Estudios de Ingeniería de Sistemas (CEIS), Facultad de Ingeniería Informática, Instituto Superior Politécnico "José Antonio Echevarría" (CUJAE), Habana, Cuba. ichavez@ceis.cujae.edu.cu
[3]Mg. Profesora Asistente, Centro de Estudios de Ingeniería de Sistemas (CEIS), Facultad de Ingeniería Informática, Instituto Superior Politécnico "José Antonio Echevarría" (CUJAE), Habana, Cuba. ylopez@ceis.cujae.edu.cu

RESUMEN

En este trabajo se presentan las experiencias de la implementación de un Modelo para la Gestión de Revisiones (MGRSoft), haciendo énfasis en la utilización de mecanismos de reutilización para las revisiones de los requisitos a nuevos proyectos de software. Se exponen las características de una herramienta que reutiliza revisiones anteriores y su utilización dentro del modelo MGRSoft. Se presentan, además, los resultados obtenidos en una prueba realizada a proyectos reales, que evidencia el aprovechamiento de la experiencia acumulada utilizando los mecanismos de reutilización incluidos en el modelo.

Palabras clave: Calidad de Software, Ingeniería de Software, Inteligencia Artificial, Razonamiento Basado en Casos, Revisiones.

ABSTRACT

This paper presents the implementation experiences of the Manage Revisions Model (MGRSoft), making emphasis in the use of reutilization mechanisms for revisions to requirements in new software projects. It includes the characteristics of a tool that reuse previous revisions and its use inside the MGRSoft model. The results of the results of a test on real projects, which demonstrates the use of experience using the reuse mechanisms included in the model.

Keywords: Software Quality, Software Engineering, Artificial Intelligence, Case Based Reasoning, Revisions.

Autor para correspondencia: : marta@ceis.cujae.edu.cu

Recibido: 04.01.2010 Aceptado: 17.08.2010

INTRODUCCIÓN

Varios trabajos reflejan la importancia de establecer el Proceso de Revisión en las empresas de software Boehm & Basili, (2001); Biffl, (2000); O'Neill, (2000), sustentado en el argumento de que las dos terceras partes de los defectos de los sistemas son el resultado de errores cometidos en etapas tempranas del desarrollo del proyecto, y sólo una tercera parte son el resultado de errores cometidos en etapas avanzadas, por lo que se hace necesario prevenir los defectos o detectarlos en las primeras etapas (McEwen, 2004), (Usaola, 2006), (Pressman, 2005).

Schulmeyer, en su libro "Handbook of Software Quality Assurance" (Schulmeyer, 1997), plantea que en estudios realizados se ha demostrado que las revisiones al software son un potente método para la detección de defectos, que encuentran de un 60 a un 90% de todos los defectos, así como proveen retroalimentación que permitirá a los desarrolladores de software evitar la inserción de defectos en trabajos futuros.

Con la aparición del Lenguaje Unificado de Modelado (OMG, 2003), (Rumbaugh, 1999), la definición del Proceso Unificado de Rational (Jacobson, 2000) y la amplia utilización de éstos en el desarrollo de proyectos a nivel mundial, que los han convertido en estándares empleados por una gran cantidad de empresas desarrolladoras de software, se abre una vía para trabajar en la detección de defectos de forma automatizada.

Existen en el mercado internacional herramientas automatizadas, entre las que se encuentra la Suite de Rational (Rational, 2001), que detectan defectos relacionados con el balance entre artefactos y otros chequeos de consistencia. Además, existen herramientas dedicadas a la detección automatizada de defectos a partir del código (Markosian, 2003), (Reitzig, 2003), que se centran en detectar fundamentalmente errores sintácticos. Sin embargo, estas herramientas no detectan defectos relacionados con la semántica propia de cada proyecto, como puede ser la omisión de algún requisito o el diseño inadecuado de alguna clase, que no necesariamente producirán una falla en el funcionamiento del software. Sería conveniente, entonces, abordar la detección de defectos relacionados con el modelado de sistemas, analizando las posibles semejanzas entre los proyectos a diseñar y las soluciones que han sido dadas con anterioridad a problemas del mismo tipo. Además, se deben tener en cuenta los defectos que han sido detectados en Revisiones a proyectos con características similares, con el fin de prevenir la ocurrencia de ellos.

Schulmeyer, 1997, sugiere que la manera más efectiva de realizar revisiones es utilizar solamente personal entrenado en la conducta propia de las revisiones. Por tanto, la experiencia que posean los inspectores determina la efectividad de la Revisión, pues serán capaces de detectar mayor cantidad de defectos (Biffl, 2000). Es importante que el inspector conozca los defectos detectados y la solución que se ha dado a proyectos con características similares, pues esto le dará mayores posibilidades para encontrar los defectos.

En la pequeña y mediana empresa (PYME) no abunda el personal entrenado en la conducta de las Revisiones, pues, en general, son pocos los especialistas dedicados al desarrollo de software y, en la mayoría de los casos, es personal de poca experiencia en la detección de defectos (Febles, 2004), por lo que se hace indispensable incorporar técnicas y herramientas que ayuden a los desarrolladores y revisores a detectar la mayor cantidad de defectos posibles en cada revisión y contar con la experiencia acumulada en el desarrollo de proyectos anteriores y en las revisiones realizadas a ellos.

En estas condiciones puede pensarse en trabajar en la acumulación, en la computadora, de la experiencia que se vaya obteniendo en cada una de las revisiones, de forma tal que los

especialistas dispongan de un banco de casos que puedan ser empleados en la detección de defectos en nuevos proyectos. Para lograrlo parece aconsejable utilizar técnicas de Inteligencia Artificial (IA).

Si se considera además el hecho de que a los posibles expertos les resulta muy difícil establecer cadenas de reglas generalizables, que permitan inferir los defectos en un nuevo proyecto, y les es más fácil expresarlo en términos de casos ya ocurridos, parece aconsejable pensar en emplear el Razonamiento Basado en Casos (RBC) (Althoff, 2001), (Manjares, 2001). El RBC es una técnica de IA, que comienza a introducirse en aplicaciones de la Ingeniería de Software (OOReuse, 2003), (ROSA, 2003), (Ganesan & Allen 2000), (García *et al.* 2001), (Vicinanza, 1991), (Vanmali, 2002), (Last *et al.* 2003), (Last, 2006), (Elbaum, 2004), (Walcott *et al.*2006), (Fraser, 2007), y que permite resolver nuevos problemas partiendo de la experiencia acumulada en la solución de problemas similares resueltos con anterioridad.

En el CEIS se desarrolló un modelo de revisiones que utiliza RBC para asistir las revisiones a los proyectos de software (Delgado, 2006). En este trabajo se presenta la forma en que MGRSoft incorpora la reutilización de las revisiones realizadas con anterioridad por parte de los propios especialistas de la entidad, describiendo una herramienta que da soporte a esta actividad y la aceptación que ha tenido esta propuesta entre los especialistas.

Descripción general de MGRSoft: Modelo de Revisiones para proyectos de software
Descripción general de MGRSoft

El Modelo de Revisiones para Proyectos de Software (MGRSoft) consta de un *Sistema de Procesos* que contempla procedimientos, roles y métricas, y que se basa en la utilización de la experiencia acumulada por la empresa en materia de revisiones, almacenada en una *Base de Conocimientos* que posee toda la información referente al proceso de revisiones. Además, se incluye un conjunto de *Herramientas Automatizadas,* integradas en un paquete que permite gestionar dicho conocimiento en los dos procesos definidos dentro del modelo.

El *Sistema de Procesos* incluye dos procesos: el de nivel estratégico “Gestión de Revisiones” y el clave “Revisiones de Proyectos Específicos” (Delgado, 2006). La definición de los procesos está contenida en fichas de procesos y seis documentos de instrucciones, que se hacen acompañar de 13 plantillas que contienen la información generada durante la ejecución de los procesos.

La Base de Conocimientos contiene la información referente a las revisiones que permite contar con la experiencia acumulada en este proceso para que pueda ser utilizada en el desarrollo y revisión de nuevos proyectos.

Los procesos definidos están soportados por un paquete de herramientas automatizadas integradas (RevisionCASE), que gestionan la Base de Conocimientos. RevisionCASE, que integra la gestión de las actividades descritas en el modelo con un asistente para ayudar en la detección de los defectos; ellas son: Herramienta automatizada para la planificación y seguimiento de las Revisiones (Revision) y Asistente para Revisiones a Proyectos, aprovechando la experiencia anterior (ARPA).

Reutilización de revisiones anteriores en la ejecución de nuevas revisiones

El Modelo MGRSoft utiliza los casos almacenados con anterioridad para ayudar en la detección de defectos. Estas actividades están contenidas en la actividad de “Ejecutar Revisión” en el subproceso “Ejecución de Revisiones Específicas” dentro del Proceso “Revisiones de Proyectos

Específicos". Se parte del análisis que debe realizar el rol encargado de su ejecución y de la experiencia almacenada en la base de casos de la empresa. Para poder reutilizar esta experiencia, el modelo incorpora la técnica de RBC y, para ello, se han definido mecanismos de almacenamiento y recuperación de la información. El diagrama de actividad de UML que describe esta parte del proceso, que permite reutilizar las soluciones anteriores, se muestra en las figuras 1 y 2 y aparece destacado con óvalos negros.

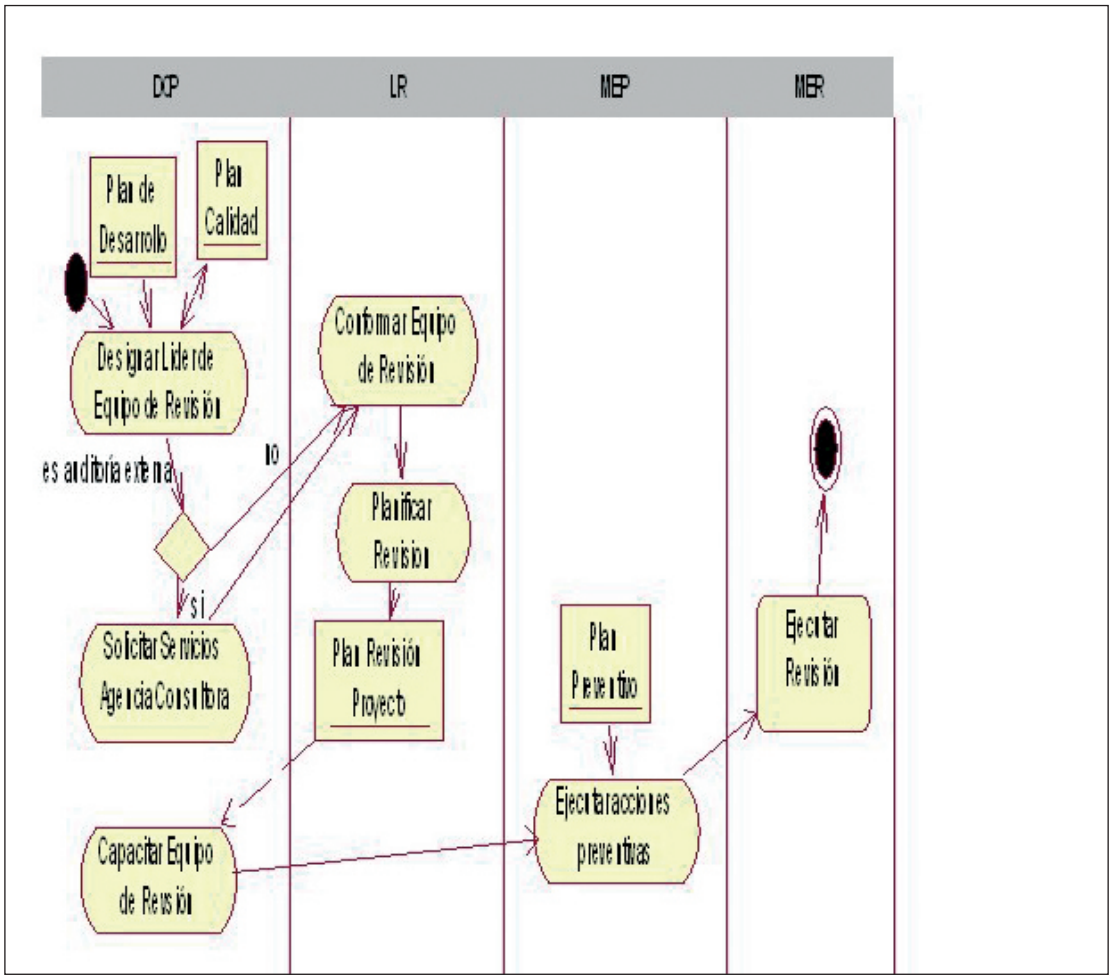

Figura 1. Diagrama de actividades del subproceso Ejecución.

ISSN 0717-9103
ISSN Online 0718-8307

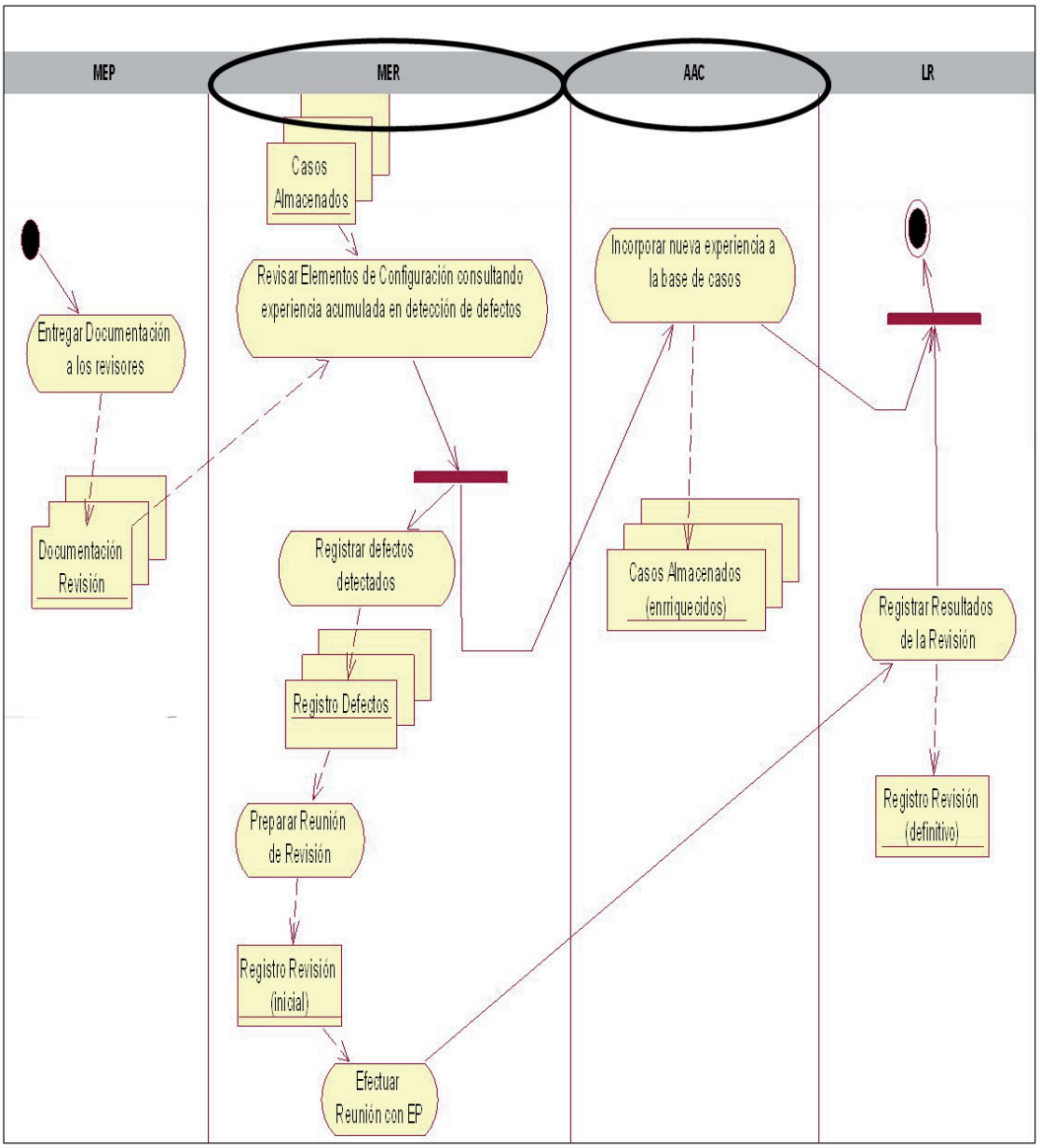

Figura 2. Diagrama del estado Ejecutar Revisión

La Base de Conocimiento de la Organización (BCO) debe contener información relativa a las revisiones, que permita contar con la experiencia acumulada en este proceso para que pueda ser reutilizada en el desarrollo y revisión de nuevos proyectos. A continuación se expone una propuesta de diseño de la BCO, que contiene la experiencia de las revisiones y que permite gestionar la información que se deriva de ellas.

La BCO se compone de tres repositorios que almacenan la información referente a las revisiones, tanto a nivel de procesos como a nivel de proyectos, logrando unificar en una sola propuesta información específica del proyecto e información de tipo gerencial sobre el proceso de revisiones. Para ello se han definido:

1. Repositorio de Proyectos: Contiene la información generada en el proceso de Revisiones de Proyectos Específicos.
2. Repositorio de Procesos: Almacena la información generada en los procesos de la organización y los resultados de las revisiones realizadas a los artefactos generados en cada proceso.
3. Repositorio de Desarrollo y Mantenimiento: Contiene la información generada durante el desarrollo y mantenimiento de los proyectos. Puede ser reutilizada en la solución de nuevos proyectos.

La información acumulada en estos repositorios es empleada en los procesos definidos en MGRSoft y se procesa a través de las herramientas automatizadas que soportan el modelo. Para tener acceso a la experiencia sobre detección de defectos acumulada en esta base, las autoras proponen utilizar la técnica de RBC que se sustenta en la semejanza entre los proyectos de software. Para lograr el empleo de los casos ha sido necesario definir el repositorio de Desarrollo y Mantenimiento como una Base de Casos (Figura 3). Esto se logra con mecanismos propios de recuperación de proyectos, basados en una función de semejanza entre proyectos. Además se han diseñado los correspondientes mecanismos de generación y adaptación de la nueva solución.

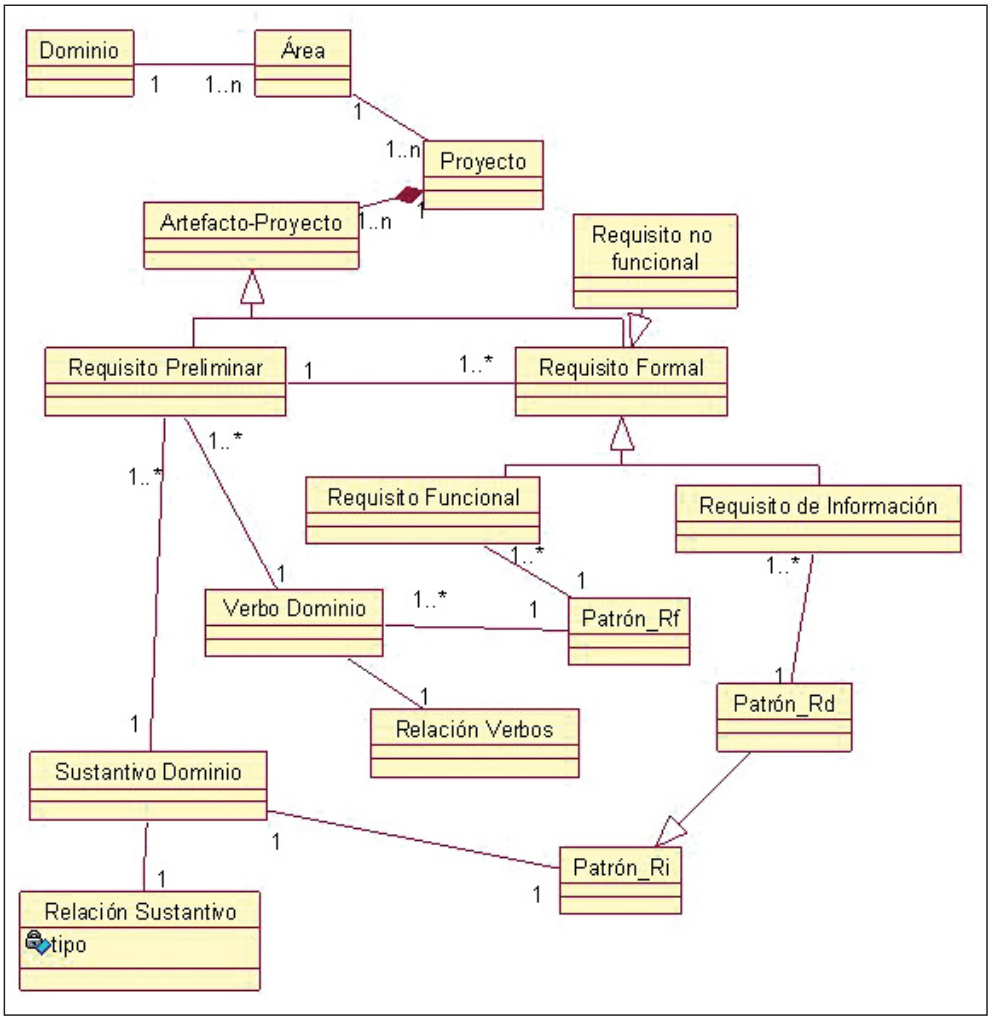

Figura 3. Repositorio de Desarrollo y Mantenimiento

ISSN 0717-9103
ISSN Online 0718-8307

Herramientas automatizadas para MGRSoft

Como soporte al modelo propuesto se ha implementado un paquete de herramientas automatizadas, RevisionCASE, que integra la gestión de las actividades descritas en el modelo con un asistente para ayudar en la detección de los defectos, Asistente para Revisiones a Proyectos, aprovechando la experiencia Anterior (ARPA). Cada una de ellas soporta diferentes actividades dentro del modelo, pero intercambiando información contenida en los repositorios como parte de MGRSoft.

Revisión CASE, paquete de herramientas que soporta los subprocesos de MGRSoft, está compuesto por seis herramientas que cubren los subprocesos del modelo y han sido agrupadas en tres: herramientas que sirven de soporte al proceso de revisiones a alto nivel, herramientas que soportan el proceso de revisiones a nivel de proyecto y herramientas que soportan el seguimiento, mejoramiento de los subprocesos e integración del conocimiento en ambas direcciones.

En particular la herramienta que soporta la reutilización de casos almacenados en la detección de defectos en las nuevas revisiones es ARPA, Asistente para Revisiones a Proyectos aprovechando la experiencia Anterior.

La herramienta ARPA tiene como entradas los requisitos candidatos y la solución que dieron los desarrolladores en la definición de los requisitos. El asistente desarrollado cuenta en la actualidad con algoritmos para asistir la detección de defectos en la etapa de Requisitos, pero puede ser extendido a otras etapas de desarrollo de proyectos, incluyendo los algoritmos correspondientes en la fase de Adaptación del Sistema de Razonamiento Basado en Casos (SRBC). El sistema tiene tres tipos de usuarios:

1. Miembro del equipo de revisión o revisor: En este caso, la información requerida por el sistema es la especificación completa que se quiere Revisar, que incluye el listado de Requisitos del Proyecto y su Modelo de Casos de Uso, representado en un Diagrama de Casos de Uso (Figura 4). La respuesta del sistema para este usuario está compuesta por un listado de defectos potenciales, encontrados en el proyecto que se está revisando, a partir de una comparación con una solución propuesta que considera la experiencia acumulada en la Base de Casos, y una propuesta de solución a estos defectos, materializada en un diagrama de casos de uso y una descripción de los requisitos (Figura 5).

2. Miembro del equipo de desarrollo o desarrollador: La información a suministrar al sistema por el usuario es únicamente la lista de requisitos preliminares, y el sistema de RBC devolverá al usuario un listado completo de Requisitos con su Modelo de Casos de Uso del sistema. Esta información se obtiene a partir de los casos más parecidos almacenados en la base de casos, es decir, se reutilizan los casos parecidos.

3. Administrador de la Base de Casos: Las actividades asociadas a este usuario son las de mantenimiento de la Base de Casos. La inserción de los casos a la Base sólo puede ser realizada desde el módulo de administración, puesto que el administrador es el único con permiso para hacer variaciones en los datos que maneja la herramienta. No obstante, el resto de los usuarios pueden hacer propuestas que serán revisadas por el administrador para su inclusión o no en la Base de Casos.

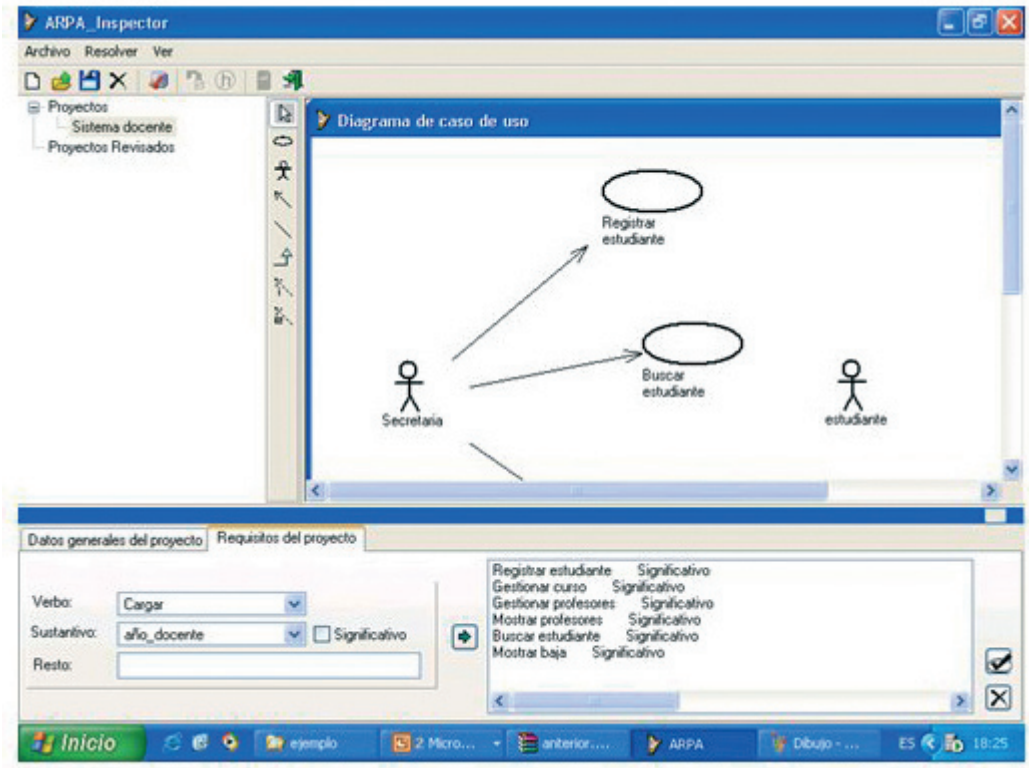

Figura 4. Información suministrada por el revisor en ARPA.

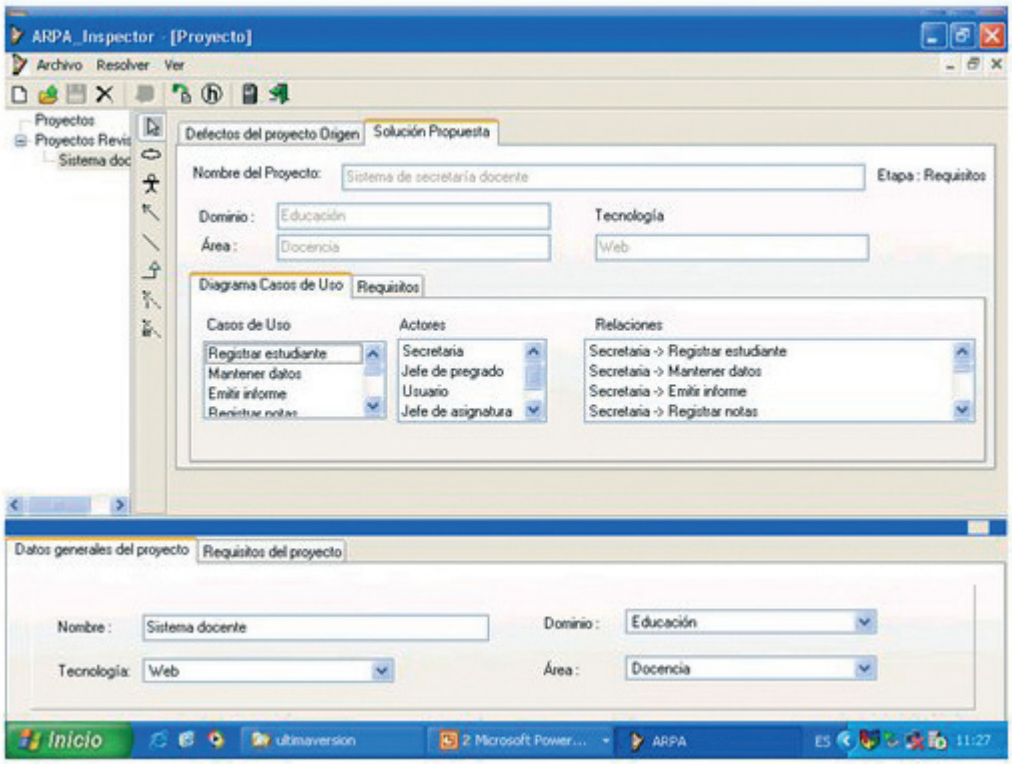

Figura 5. Propuesta de solución ofrecida por ARPA al revisor.

ISSN 0717-9103
ISSN Online 0718-8307

Metodología para la aplicación de MGRSoft en la reutilización

El Grupo de Ingeniería de Software, (CRIS) de la Facultad de Ingeniería Informática del Instituto Superior Politécnico ¨José Antonio Echeverría¨ (CUJAE), tiene a su cargo el desarrollo de investigaciones en el área de la Ingeniería de Software y su incorporación en la docencia de pregrado y postgrado a partir de la labor de sus miembros como profesores a tiempo completo de la Facultad. En el grupo se desarrolla un conjunto de proyectos reales, y en éstos se realizó un grupo de acciones para la validación del modelo propuesto, así como de los mecanismos de reutilización presentes en él. Para ello se efectuó un diagnóstico inicial a investigadores y profesores del grupo sobre el desarrollo de software en el propio grupo. Además, se aplicaron encuestas y se realizaron entrevistas con directivos y miembros del mismo.

A partir de los resultados del diagnóstico y de las reuniones de trabajo con los miembros del grupo, se decidió comenzar la implantación del modelo, pero hacerlo en una etapa inicial con el uso de un proyecto piloto y la definición de algunas de las actividades para validar con proyectos reales desarrollados por especialistas y estudiantes de la CUJAE. Para ello se definió un primer nivel de aplicación del modelo, el Nivel Piloto.

Actividades a desarrollar durante la implantación:

1. Definir Plan de Ejecución preliminar tal y como se indica en el subproceso de Planeación del proceso Gestión de Revisiones.

2. Ejecutar el subproceso de Preparación de la Implantación del proceso Gestión de revisiones, enfatizando en las actividades de capacitación y entrenamiento del personal.

3. Determinar un proyecto piloto, preferiblemente de nueva elaboración.

4. Definir un plan de ejecución para el proyecto piloto, donde se ejecuten las actividades del proceso Revisiones de Proyectos Específicos. Incluir sólo el control de las métricas cantidad de defectos detectados y reutilización de soluciones en la construcción de la nueva solución.

En el trabajo de preparación para la primera etapa (Piloto) se desarrollaron talleres de trabajo en equipo, en los que se realizaron las siguientes tareas:

1. Se definió el Plan de Ejecución piloto y se fijaron las directivas de calidad a tener en cuenta durante el desarrollo de las revisiones.

2. Se adecuaron las plantillas propuestas en el modelo a las necesidades del grupo.

3. Se seleccionó el proyecto piloto y se le definió la planificación de las revisiones correspondientes.

4. Se ejecutó la primera revisión al proyecto piloto. En este caso se trabajó de forma paralela en la conformación de dos grupos:

Grupo A: Se conformó un grupo de expertos a partir de especialistas de otras entidades, con experiencia en el trabajo de revisiones de proyectos de software, y especialistas de la propia Dirección, con al menos 3 años de experiencia profesional.

Grupo B: Se conformó un grupo de especialistas con 1 y 2 años de trabajo en la Dirección para hacer una revisión asistida por la herramienta ARPA.

Para evaluar la validez y aplicabilidad de los mecanismos de reutilización incorporados al modelo, se realizaron las siguientes acciones:

1. Análisis del comportamiento de los mecanismos de reutilización de la herramienta, para una muestra de proyectos reales desarrollados por los estudiantes de la CUJAE
2. Validación de resultados obtenidos por la herramienta, partiendo de la comparación de los sugeridos por ésta y los encontrados por los expertos del Grupo A, a partir de la aplicación del proyecto piloto:
3. Estimación de la aceptación de los mecanismos de reutilización incluidos en el modelo por parte de los especialistas, a partir de la aplicación de encuestas después de la ejecución del proyecto piloto.

RESULTADOS

Para realizar el análisis de comportamiento de la herramienta, se realizó una prueba de los mecanismos de reutilización como parte de la validación de sus resultados. Para ello se considero un total de 23 proyectos reales desarrollados por estudiantes de la CUJAE. Del total de defectos relacionados con omisiones de artefactos y alcance del proyecto que la herramienta propone como posibles defectos, obtenidos a partir de la reutilización de proyectos anteriores, aproximadamente el 90% fueron considerados como válidos (o procedentes) para los proyectos particulares, en el chequeo posterior que se realizó el Grupo A. Alrededor de un 16% de los defectos propuestos por ARPA, todos recuperados a partir de la reutilización de proyectos anteriores, no fueron detectados por los especialistas del Grupo B, y de ellos alrededor de un 10% fueron considerados como procedentes.

Como parte de la Validación de los resultados en el proyecto piloto, se solicitó a los expertos del Grupo A que ejecutaran una revisión de aquel, partiendo de una lista de chequeo elaborada en los talleres de trabajo y al Grupo B se les pidió que ejecutaran la revisión utilizando la herramienta. De los 30 tipos de defectos detectados por el Grupo A, la herramienta detectó 14 tipos, que son los 14 posibles según los mecanismos implementados hasta el momento en ARPA.

Una vez obtenidos los primeros resultados con la implementación del proyecto piloto, se decidió realizar un conjunto de reuniones de trabajo para verificar la aceptación del modelo, en su primera fase, y efectuar un grupo de encuestas. Los resultados que se exponen a continuación justificaron la decisión del grupo de trabajo y la dirección de la entidad de pasar al segundo nivel de implantación, Extensión, que es en el que se encuentra actualmente.

Como resultado de las reuniones de trabajo con los equipos involucrados en la ejecución del proyecto piloto y las acciones de capacitación, hay coincidencia entre los especialistas de la dirección en que la introducción de MGRSoft en los tres niveles permite a la empresa:

- Capacitar al personal y crear disciplina en materia de revisiones a los proyectos de software, a partir del estudio y aplicación en un proyecto piloto de las buenas prácticas adoptadas por la empresa.

- Nutrir la Base de Casos a partir de la experiencia que irá acumulando la propia empresa en el desarrollo de revisiones sistemáticas a los proyectos.

- Emplear la experiencia acumulada, en el desarrollo de revisiones a nuevos proyectos.

De esta manera se podrá formar a los nuevos desarrolladores e inspectores en las políticas que en materia de revisiones adopte la empresa.

Para evaluar la aceptación por parte de los especialistas de los mecanismos de reutilización dentro del modelo, se les realizaron las siguientes preguntas: ¿Considera que la combinación de la Gestión de las Revisiones con mecanismos basados en la semejanza entre proyectos que acumulen la experiencia en la detección de defectos, contribuye a elevar la calidad de los productos de software?, ¿Le fueron útiles, para la revisión de los nuevos proyectos, las sugerencias hechas por la herramienta ARPA, basadas en revisiones anteriores?. Todos los especialistas encuestados contestaron afirmativamente.

Al culminar las revisiones al primer proyecto piloto utilizando el Modelo de Gestión de Revisiones (MGRSoft), se hizo circular entre los especialistas un cuestionario para validar su aceptación por parte de los usuarios. Todos los usuarios encuestados contestaron afirmativamente a las siguientes preguntas:

- ¿Es posible implantar MGRSoft en su empresa?
- ¿Considera un aporte importante MGRSoft para el establecimiento de una disciplina en materia de revisiones, que redunde en una mayor calidad de los productos de software?

- ¿Considera que la combinación de la Gestión de las Revisiones con mecanismos basados en la semejanza entre proyectos que acumulen y reutilicen la experiencia en la detección de defectos, contribuye a elevar la calidad de los productos de software?

CONCLUSIONES

Se ha presentado una propuesta de inclusión de mecanismos de reutilización de proyectos anteriores en la detección de defectos en nuevas revisiones. La propuesta contenida en MGRSoft ha sido validada por un conjunto de especialistas de trabajo de desarrollo de software.

Reutilizar la experiencia anterior en materia de revisiones puede contribuir a mejorar la aplicación del modelo, porque puede ayudar a entrenar a los nuevos revisores dentro de la organización.

El trabajo presentado continúa en perfeccionamiento, tanto de los mecanismos de adaptación de los casos propuestos, como para la extensión de estos mecanismos de reutilización a otras actividades de aseguramiento de calidad.

REFERENCIAS

Althoff, K. (2001), Case-Based Reasoning. Handbook of Software Engineering and Knowledge Engineering, kaiserslautern, Alemania. Fraunhofer Institute for Experimental Software Engineering (IESE).

Boehm, B. & Basili, V. (2001). Software Defect Reduction Top 10 List, *IEEE Computer*, 34(1), 135-137, January 2001.

Biffl, S. (2000). Analysis of the impact of reading technique and inspector capability on individual inspection performance, Seventh Asia-Pacific Software Engineering Conference (APSEC'00), IEEE Inc., Diciembre 2000.

Delgado, M., Lorenzo, I., Carralero, J., Travieso, J., Rosete, A. (2006). Una propuesta de apoyo a las Revisiones de Proyectos de Software utilizando Razonamiento Basado en Casos, *Revista Iberoamericana de Inteligencia Artificial* 10(3), 55-68.

Delgado, M.D.D. (2006). Un modelo para la gestión de Revisiones en proyectos de software utilizando Razonamiento Basado en Casos. Tesis presentada en opción al grado científico de Doctor en Ciencias Técnicas., 2006. Facultad de Informática. CUJAE.

Elbaum, S., Rothermel, G., Kanduri, S., Malishevsky, A. (2004). Selecting a Cost-Effective Test Case Prioritization Technique. *Software Quality Journal* 12(3), 185-210

Febles, A. (2004). Un modelo de referencia para la gestión de configuración en la pequeña y mediana empresa de software. Tesis presentada en opción del grado de Doctor en Ciencias Técnicas. Instituto Superior José Antonio Echeverría.

Fraser, G., Wotawa, F. (2007). Test-case prioritization with model-checkers, 25th conference on IASTED International, 2007.

Ganesan, K. . Allen, E. (2000). Case-based software quality prediction´, International Journal of Software Engineering and Knowledge Engineering, 10(2):139-152.

Garcıa, F.J., Corchado, J.M.,& Laguna, M.A. (2001). CBR Applied to Development with Reuse Based on mecanos. In *Proceedings of the 13th International Conference on Software Engineering and Knowledge Engineering*, june 13-15, 2001. Buenos Aires, 2001

Jacobson, I. (2000). El Proceso Unificado de Desarrollo de Software, Addison Wesley Longman Inc, 2000.

Last, M., Friedman, M., Kandel, A. (2003). The Data Mining Approach to Automated Software Testing", Proceedings of the Ninth ACM SIGKDD International Conference, 2003.

Last, M. (2006). The Uncertainty Principle of Cross-Validation, 2006 IEEE International Conference on Granular Computing, 2006.

Manjares, A. (2001). Razonamiento basado en casos. Universidad Nacional de Educación a Distancia, Departamento de Inteligencia Artificial, Madrid, España.

Markosian, L. (2003). Hosted Servicesfor Advanced V&V Technologies: An Approach to Achieving Adoption with out the Woes of Usage, 3rd International Workshop on Adoption-Centric Software Engineering, ICSE 2003 IEEE/ACM International Conference on Software Engineering, Portland, Oregon, May, 2003.

McEwen, S. (2004). Requirements: An introduction, IBM Rational, IBM Corporation 2004, http://www-136.ibm.com/developerworks/rational/library/4166.html, (2004).

Object Management Group. (2003). Unified Modeling Language. Version 1.5, Object Management Group Inc., March 2003.

O'Neill, D. (2000). National Software Quality Experiment: Results 1992-1999. Software Technology Conference, Salt Lake City, 2000.

OOReuse. (2003). Research Projects. CBR-WEB & MLNET. Http://Www.Ai-Cbr.Org.

Pressman, R. (2005). Ingeniería del Software: un enfoque práctico. Parte 1, Editorial Félix Varela, 5ta edición, 2005.

Rational Corporation. (2001). Rational Unified Process. Version 2001A.04.00, Copyright 1987-2001.

Reitzig, R. (2003). Using Rational Software Solutions to Achieve CMMI Level 2, Rational Software 2003, http://www.therationaledge.com/content/jan_03/f_CMMI_rr.jsp.

ROSA. (2003). Research Projects. CBR-WEB & MLNET. http://www.ai-cbr.org.

Rumbaugh, J. (1999). The Unified Modeling Language. Reference Manual, Addison Wesley Longman Inc, 1999.

Schulmeyer, G. (1997). Handbook of Software Quality Assurance, Prentice may, 1997.

Usaola, M.P. (2006). Curso de doctorado sobre Proceso software y gestión del conocimiento. Pruebas del Software. Universidad de Castilla-La Mancha. Departamento de Tecnologías y Sistemas de Información, 2006.

Vanmali, M., Last, M., Kandel, A. (2002). Using a Neural Network in the Software Testing Process, International Journal of Intelligent Systems, 2002.

Vicinanza, S. (1991). Software effort estimation: an exploratory study of expert performance´, Information Systems Research 2(4), 243-262.

Walcott, K. R., Soffa, M.L., Kapfhammer, G.M.,& Roos, R.S. (2006). TimeAware Test Suite Prioritization, Proceedings of the 2006 international symposium on Software.