Revista Cubana de Ciencias Informáticas
Vol. 8, No. Especial UCIENCIA 2014, Mayo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 41-51
http://rcci.uci.cu

Tipo de artículo: Artículo original
Temática: Ingeniería y Gestión de Software
Recibido: 7/05/2014 | Aceptado: 21/05/2014

Modelando el proceso de desarrollo de software de la UCI con cbSPEM

Modeling UCI's software development process with cbSPEM

Yanet Vega Miniet [1*], Michel Ramos Navarro[1], Daimara Mustelier Sanchidrian [1], Yadenis Piñero Pérez[1]

[1]Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, km 2 ½, Torrens, Boyeros, La Habana, Cuba. CP.: 19370. Correo-e:{ram, dmustelier}@uci.cu

* Autor para correspondencia: yvegam@uci.cu

Resumen

El objetivo de este trabajo es definir un meta-modelo para representar el proceso de desarrollo de software de la Universidad de las Ciencias Informáticas. El proceso de desarrollo de software es un proceso de conocimiento intensivo y debe modelarse utilizando el enfoque de la gestión de casos, por lo que se definió un meta-modelo que extiende los meta-modelos *Case Management Model and Notation* y *Software and Systems Process Engineering Meta-Model Specification*. Este meta-modelo se utilizó para representar el proceso de desarrollo de software de la universidad utilizando un *plugin* creado para este proceso, lo cual demuestra que es factible modelar procesos de desarrollo de software utilizando el meta-modelo propuesto: C*ase Based SPEM*. El aporte fundamental de este trabajo radica en la elaboración y aplicación de un meta-modelo que permite crear modelos de proceso de desarrollo de software más flexibles.

Palabras clave: meta-modelo, modelo de proceso de desarrollo de software

Revista Cubana de Ciencias Informáticas
Vol. 8, No. Especial UCIENCIA 2014, Mayo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 41-51
http://rcci.uci.cu

Abstract

The aim of this paper is to define a meta-model to represent University of Information Science's development process. The software development is a knowledge-intensive process and must be modeled using case management approach. Because of that, a meta-model that extends Case Management Model and Notation and Software and Systems Process Engineering Meta-Model Specification meta-models was defined. This meta-model was used to represent university's development process using a plugin created for this process. This shows that is possible to model software development process using Case Based SPEM. The main result of the work is the elaboration and aplicacition of a meta-model that allows to create more flexible software process models.

Keywords: *Case Based SPEM, Case Management Model and Notation, meta-model, Software and Systems Process Engineering Meta-Model Specification, software development process.*

Introducción

La gestión del proceso de desarrollo de software se basa en el principio de que la calidad de un sistema informático está altamente influenciada por la calidad del proceso utilizado para adquirirlo, desarrollarlo y mantenerlo. Varios estudios han mostrado que la mejora de los procesos de desarrollo de software tiene un impacto positivo en los desarrolladores, está relacionada con mejoras en el desempeño de los proyectos, la calidad de los productos obtenidos y el éxito de las organizaciones (Lavallée and Robillard, 2012; Harter et al., 2012; Clarke and O'Connor, 2012; Subramanian et al., 2007); esto ha motivado un creciente interés por gestionar y mejorar continuamente los procesos de desarrollo de software. La gestión del proceso de desarrollo de software consiste en cuatro actividades secuenciales desarrolladas en un ciclo iterativo: establecimiento, planificación, implementación y cambios, y evaluación (IEEE, 2004). En las últimas tres actividades es útil contar con un modelo del proceso, por lo que los modelos de proceso son un elemento importante en la mejora continua de los procesos de desarrollo de software.

En el 2008 se inició en la Universidad de las Ciencias Informáticas (UCI) un proyecto de mejora del proceso de desarrollo de software basado en *Capability Maturity Model Integrated for Development (CMMI for Development)* (ESI Center, 2006). El proyecto de mejora de la universidad se trazó como meta alcanzar el nivel 2 dentro de este modelo. Los principales resultados de este proyecto fueron que el Centro de Informatización de la Gestión de Entidades (CEIGE), el Centro de Informática Industrial (CEDIN) y el Centro de Informática Médica (CESIM) se certificaron en el nivel 2 de *CMMI for Development* en el año 2011 (SEI, 2011) y la definición del modelo de proceso

de desarrollo de software de la universidad -en lo adelante se hará referencia a este como Modelo de Proceso de Desarrollo de Software de la UCI.

En el 2011 se inició la implantación del Modelo de Proceso de Desarrollo de Software de la UCI en el resto de los centros de desarrollo de software de la universidad, pero un diagnóstico realizado a la actividad productiva en el año 2012 (CALISOFT, 2013) permitió identificar problemas relacionados con la comprensión del proceso y consecuentemente su correcta aplicación. El proceso se especificó en libros de procesos por cada área del nivel 2 de CMMI y algunos documentos generales, lo cual dificulta la identificación de relaciones entre elementos de diferentes áreas de proceso. Por otra parte, las representaciones gráficas utilizadas para modelar el proceso no se basan en un lenguaje de modelado o notación con una semántica bien definida. El objetivo de este trabajo es definir un meta-modelo para representar el proceso de desarrollo de software de la UCI de manera que se facilite su comprensión.

Materiales y métodos

Múltiples definiciones de proceso de desarrollo de software existen en la bibliografía, algunas de las más conocidas son las dadas por (IEEE, 1990; Sommerville, 2007; Pressman, 2010). Sin embargo, en todas ellas se evidencian dos cuestiones fundamentales: el desarrollo de software es un proceso de negocio, el objetivo de este proceso es obtener un producto que satisfaga las necesidades de los interesados. Por otra parte, se observa que el desarrollo de software es un proceso de conocimiento intensivo según la definición dada por (Weber and Reichert, 2012) al respecto. Un modelo de proceso es una representación abstracta del proceso que se modela. Los modelos de proceso se basan en meta-modelos que definen los componentes y la semántica de los modelos.

Generalmente los meta-modelos de proceso asumen como único mecanismo para dirigir las instancias de procesos el enrutamiento de las tareas basándose en un modelo de proceso que describe completamente el proceso antes de su ejecución (Dumas et al., 2005; Weber and Reichert, 2012). Esto permite el modelado y ejecución de los procesos de negocio estructurados y repetitivos (OMG, 2009; Weber and Reichert, 2012), sin embargo, resulta muy restrictivo y genera problemas para el tratamiento de los cambios (Van der Aalst et al., 2003; Ferreira, 2008; Weber and Reichert, 2012) en los procesos de conocimiento intensivo, como el proceso de desarrollo de software. Una solución relativamente nueva para estos problemas es la gestión de casos (Van der Aalst et al., 2003; Van der Aalst et al., 2005).

Revista Cubana de Ciencias Informáticas
Vol. 8, No. Especial UCIENCIA 2014, Mayo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 41-51
http://rcci.uci.cu

El *Object Management Group* (OMG) publicó el Modelo y Notación de Gestión de Casos (CMMN1) (OMG, 2013) con el objetivo de modelar los procesos utilizando el enfoque de la gestión de casos.

Atendiendo a que el desarrollo de software es un proceso de conocimiento intensivo este debe modelarse utilizando el enfoque de gestión de casos, como se muestra en (Oldenhave, 2010). Sin embargo, la propia OMG definió el meta-modelo *Software and Systems Process Engineering Meta-Model Specification* (SPEM) (OMG, 2008) específicamente para modelar los procesos de desarrollo de software. Este meta-modelo no incluye una semántica específica para modelar el comportamiento de los procesos, sino que provee un conjunto de interfaces para su integración con otros meta-modelos como *Unified Modeling Language* (UML) (OMG, 2011); pero estas interfaces no brindan soporte a algunos elementos de CMMN como las restricciones y la planeación en tiempo de ejecución.

Considerando lo anterior se decidió extender el meta-modelo SPEM con los elementos necesarios de CMMN para soportar el modelado del proceso de desarrollo de software siguiendo el enfoque de la gestión de casos. Este meta-modelo extendido se nombrará *Case Based SPEM* (cbSPEM).

Resultados y discusión

Case Based SPEM (cbSPEM)

CbSPEM está organizado en los siguientes paquetes:

- *cbSPEM_Core*. Contiene los elementos abstractos sobre los que se construye el resto del meta-modelo. Extiende los paquetes *CoreInfraestructure* de CMMN y *Core* de SPEM.
- *cbSPEM_ManagedContent.* Define los conceptos requeridos para gestionar las descripciones textuales de los elementos base y procesos. Extiende el paquete *ManagedContent* de SPEM y *cbSPEM_Core*.
- *cbSPEM_MethodContent.* Define los elementos base de cualquier proceso, por ejemplo: tareas, roles y productos de trabajo. Este paquete extiende al paquete *MethodContent* de SPEM y c*bSPEM_ManagedContent*.

[1] Por las siglas de *Case Management Model and Notation.*

Revista Cubana de Ciencias Informáticas
Vol. 8, No. Especial UCIENCIA 2014, Mayo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 41-51
http://rcci.uci.cu

- *cbSPEM_Process*. Contiene los elementos estructurales para representar el proceso de desarrollo de software y los elementos requeridos para definir su comportamiento. Este paquete extiende a los paquetes *ProcessStructure*, *ProcessBehavior* y *ProcessWithMethods* de SPEM, al paquete *PlanModelElements* de CMMN y a *cbSPEM_ManagedContent* añadiendo los elementos requeridos para modelar el proceso de desarrollo de software según el enfoque propuesto en (Van der Aalst et al., 2005) sobre la gestión de casos.
- *cbSPEM_Package*. Define los conceptos requeridos para agrupar los elementos base y elementos de proceso. Extiende los paquetes *cbSPEM_Process* y *cbSPEM_MethodContent*.
- *cbSPEM_MethodPlugin*. Define los conceptos requeridos para gestionar librerías de elementos base y procesos. Extiende los paquetes *cbSPEM_Process* y *cbSPEM_MethodContent, y* se basa en el paquete *MethodPlugin* de SPEM.

Plugin base de cbSPEM para modelar el proceso de desarrollo de software de la UCI

El paquete cbSPEM_MethodPlugin define las capacidades requeridas para gestionar librerías de elementos base de proceso *MethodContentElement* (cbSPEM_MethodContent)[2] y de procesos Activity (cbSPEM_Process). Este paquete brinda la capacidad de que todos los elementos de cbSPEM estén organizados en paquetes. Sus clases más importantes son: MethodLibrary, MethodPlugin y MethodConfiguration, que es una colección de elementos seleccionados que permite definir cómo se publicará el modelo de proceso. Utilizando estas capacidades de cbSPEM se creó el **Plugin Base para la UCI**[3] en el que se separan los elementos base y los procesos. Como la UCI está basando su programa de mejora en CMMI y cada iteración incluye la definición de un conjunto de áreas de proceso, el paquete **MethodContent** se organizó en paquetes de contenido por áreas de proceso y un paquete con los elementos comunes. Estos paquetes instancian la clase del meta-modelo MethodContentPackage (cbSPEM_MethodPlugin).

[2] Para evitar confusiones en la descripción del plugin base, los nombres de las clases del meta-modelo se subrayarán y cuando sea necesario especificar el paquete al que pertenecen se utilizará el formato establecido por UML para esto: <nombre de la clase> (<nombre del paquete al que pertenece>). Los nombres de las clases abstractas se escribirán en cursiva. Por ejemplo: *MethodContentElement* (cbSPEM_MethodContent).

[3] Para evitar confusiones en la descripción del plugin base, los nombres de los elementos del plugin base para la UCI se mostrarán en negrita.

Revista Cubana de Ciencias Informáticas
Vol. 8, No. Especial UCIENCIA 2014, Mayo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 41-51
http://rcci.uci.cu

Dentro de cada paquete correspondiente a un área de proceso se definen los elementos base sobre las que se construye el proceso de desarrollo de software de la UCI. Estos elementos base instancian las siguientes clases del paquete cbSPEM_MethodContent: RoleDefinition, TaskDefinition, ToolDefinition, WorkProductDefinition y DataObjectDefinition. Estas clases especializan a la clase *MethodContentElement*, que a su vez es un *DescribableElement* (cbSPEM_ManagedContent) por lo que se les pueden asociar descripciones textuales. De cada uno de los elementos bases se describirán los siguientes elementos:

- DataObjectDefinition: nombre, descripción, tipo, y DataObjectDefinitions que contiene.
- RoleDefinition: nombre, descripción, competencias, criterios para la asignación, tareas de las que es responsable, tareas en las que participa, productos de trabajo que crea y productos de trabajo que modifica.
- TaskDefinition: nombre, descripción, objetivo.
- ToolDefinition: nombre, descripción.
- WorkProductDefinition: nombre, descripción, objetivo, DataObjectDefinitions y WorkProductDefinitions que contiene.

Además, los paquetes de las áreas de procesos pueden contener guías, que pueden asociarse a los elementos base de proceso. En el proceso de desarrollo de software de la UCI se utilizarán los siguientes tipos de guías:

- *CheckList*. Es un conjunto de elementos que necesitan ser completados o verificados. Se utilizan en las pruebas, revisiones y auditorías. Las listas de chequeo se describen mediante un nombre, descripción, y elementos de chequeo que contienen.
- *Example*. Es un ejemplo de un producto de trabajo o fragmento de este. Tienen un nombre y descripción. Además, se les debe anexar el fichero correspondiente.
- *ProcessRequeriment*. Los requisitos del proceso son prácticas que deben cumplirse por el proceso para satisfacer un modelo de referencia o estándar de calidad. También pueden ser prácticas requeridas para satisfacer necesidades o expectativas de los interesados. Los requisitos se describirán mediante un título y una breve descripción.

- *TermDefinition*. Define un término clave utilizado en el modelo de proceso. De cada término se describe su nombre, significado y sinónimos.
- *ToolMentor*. Describe cómo utilizar una herramienta para realizar determinadas tareas. Tienen un nombre y descripción. Además, se les anexará el fichero correspondiente.
- *Whitepaper*. Es un documento que ha sido revisado o publicado de manera independiente del modelo de proceso. Tienen un nombre y descripción. Además, se les anexará el fichero correspondiente a la publicación.

Los elementos base de procesos definidos dentro del paquete **MethodContent** se utilizan en el paquete **Process** del **Plugin Base para la UCI** para definir los procesos. Un elemento base de proceso es una definición que se puede utilizar muchas veces dentro de un proceso, instanciando una especialización de la clase *MethodContentUse* (cbSPEM_Process). Por ejemplo, la definición de tarea: Notificar Revisión, se puede utilizar dentro del proceso Revisión Técnica Formal y dentro del proceso Revisión de Adherencia a Proceso.

Cada proceso en cbSPEM se define como una actividad Activity (cbSPEM_Process) o como un caso Case (cbSPEM_Process). Activity representa una unidad de trabajo compuesta que contiene un conjunto de instancias de *BreakdownElement* (cbSPEM_Process); mientras que *BreakdownElement* es una generalización abstracta para los elementos que pueden incluirse en un proceso. Los elementos que pueden incluirse en un proceso son los usos de los elementos base *MethodContentUse* (cbSPEM_Process), los capturadores de eventos EventListener (cbSPEM_Process) y los hitos Milestone (cbSPEM_Process). Activity puede representar un modelo de proceso, mediante su especialización Case o simplemente representar un agrupamiento de instancias de otros *BreakdownElement* como Milestone, Activity, ProcessPerformer, etc. Por ejemplo, una instancia válida de Activity puede contener solo instancias de RoleUse (cbSPEM_Process), expresando que en la ejecución de esa instancia de Activity participan esos roles. Esto provee la flexibilidad para modelar procesos que no están completamente definidos pero se conocen los participantes.

Por otra parte, un caso: Case, representa el concepto central de la gestión de casos. Los casos siguen patrones típicos, es decir, procesos, para su ejecución. Esta clase del meta-modelo representa el concepto de proceso asociado a un caso, por lo tanto, incluye actividades: Activity, tareas: TaskUse (cbSPEM_Process), restricciones de ejecución entre los elementos que representan trabajo Constraint (cbSPEM_Core), usos de los productos de trabajo WorkProductUse

(cbSPEM_Process) y usos de los roles RoleUse. Además, dentro de un caso se pueden asociar criterios para la planificación en tiempo de ejecución PlanningTable (cbSPEM_Process) a los elementos de trabajo.

Es conveniente tener una vista del proceso de desarrollo de software por cada área de proceso y una vista global del proceso. **Process** contiene un paquete para las vistas por área de proceso y otro para la vista global. El paquete para las vistas por área de proceso contiene una instancia de Activity para cada área de proceso. El paquete para la vista global contiene una instancia de la clase Case nombrada **Proceso_de_Desarrollo**.

Utilizando el Plugin Base UCI para modelar el proceso de desarrollo de software de la UCI

Para mostrar cómo se utilizó el **Plugin Base UCI** para modelar el proceso de desarrollo se utilizará como ejemplo el área de proceso Aseguramiento de la Calidad del Proceso y del Producto.

Dentro del paquete **MethodContent** se creó un paquete nombrado **aseguramiento_calidad**. Este paquete contiene las definiciones de roles: Administrador de la Calidad, Alta Gerencia, Coordinador de la Calidad en la Alta Gerencia, Coordinador de la Calidad en la Entidad Desarrolladora, Jefe de Proyecto, Miembro del Equipo de Proyecto y Revisor; las definiciones de tareas: Planificar Revisiones de Adherencia, Elaborar Plan de Revisiones, Realizar Evaluación, Monitorear No Conformidades, entre otras; las definiciones de productos de trabajo: Informe de Evaluación, Informe de Tendencias, Minuta de Reunión, Registro de No Conformidades, entre otros; y las guías: Guía de Evaluación y Lista de Chequeo de PPQA.

En el paquete **Process** se creó una actividad para la vista del proceso desde la perspectiva del aseguramiento a la calidad nombrada **act_aseguramiento_calidad**, que contiene el fragmento del modelo que corresponde a esta área de proceso. Finalmente se creó una configuración que permite consultar el Modelo de Proceso de Desarrollo de Software de la UCI desde tres perspectivas: una de control de flujo, llamada **Modelo de Proceso**; una de datos, llamada **Productos de Trabajo** y una de áreas de proceso, llamada **Áreas de Proceso**.

Se utilizó *Eclipse Process Framework Composer* (EPF) y *Visual Paradigm* (VP) para elaborar el Modelo de Proceso de Desarrollo de Software de la UCI según lo descrito anteriormente. Todos los elementos pudieron representarse sin

dificultad, excepto aquellos propios de la gestión de casos, porque ninguna de los dos herramientas brinda soporte para su modelado. Para resolver este problema se elaboraron los diagramas utilizando *Business Process Modeling Notation* (BPMN) con la semántica de la gestión de casos, aunque algunos elementos como las restricciones y las tablas de planeación no pudieron ser representados. Utilizando el EPF se publicó el Modelo de Proceso de Desarrollo de Software de la UCI como un sitio web estático que se encuentra publicado actualmente para toda la comunidad universitaria. Esto mejoró el acceso a la documentación del proceso de desarrollo de software, no obstante, queda pendiente evaluar si aumentó la facilidad de comprensión del Modelo de Proceso de Desarrollo de Software de la UCI.

Conclusiones

El desarrollo de software es un proceso de conocimiento intensivo y debe ser modelado utilizando el enfoque de la gestión de casos. Esto motivó la extensión del meta-modelo SPEM para integrarlo con el meta-modelo CMMN, obteniéndose como resultado cbSPEM. El principal aporte de cbSPEM radica en que este incorpora elementos clave de la gestión de casos como las restricciones y la planificación en tiempo de ejecución, a la vez que mantiene la separación clara entre los elementos base de proceso y su utilización en los procesos. Este meta-modelo se utilizó para representar el proceso de desarrollo de software de la UCI utilizando un *plugin* creado particularmente para este proceso, lo cual demuestra que es factible modelar procesos de desarrollo de software utilizando cbSPEM. La publicación del Modelo de Proceso de Desarrollo de Software de la UCI mejoró la facilidad de acceso a la descripción de este modelo, no obstante, queda pendiente evaluar si aumentó la facilidad de comprensión y si esto repercutió en una mejor aplicación del modelo.

Las limitaciones identificadas en las herramientas EPF y VP para representar elementos propios de la gestión de casos deben ser resueltas mediante la extensión de estas herramientas para brindar un soporte total al meta-modelo propuesto.

Referencias

- CALISOFT. Libro del Diagnóstico UCI 2012. La Habana, CALISOFT, 2013. 92 p.
- CLARKE, P. and O'CONNOR, R. V. The influence of SPI on business success in software SMEs: An empirical study. Automated Software Evolution, 2012, 85 (10): p. 2356–2367.

- DUMAS, M.; VAN DER AALST, W. and TER HOFSTEDE, A. Process-aware information systems. Bridging people and software through process technology. New Jersey, John Wiley & Sons, Inc., 2005. ISBN 978-0-471-66306-5.
- ESI CENTER. Propuesta de Servicios de Consultoría y Formación basados en CMMI para la Universidad de las Ciencias Informáticas. Monterrey, ESI Center, 2006. 54 p.
- FERREIRA, H. Automatic Plan Generation and Adaptation by Observation: Supporting complex human planning. PhD in Philosophy, University of Porto, Porto. 2008.
- HARTER, D. E., KEMERER, C. F. and SLAUGHTER, S. A. Does software process improvement reduce the severity of defects? A longitudinal field study. IEEE Transactions on Software Engineering, 2012, 38(4): p. 810–827.
- IEEE, C. S. P. P. C. Guide to the Software Engineering Body of Knowledge. California, IEEE Computer Society, 2004. 204 p. ISBN 0-7695-2330-7.
- IEEE, C. S. IEEE Standard Glossary of Software Engineering Terminology. New York, Institute of Electrical and Electronics Engineers, Inc., 1990. 94 p. ISBN 0-7381-0391-8.
- LAVALLÉE, M. and ROBILLARD, P.N. The impacts of software process improvement on developers: A systematic review. In: 34th International Conference on Software Engineering (ICSE). Zurich: IEEE Computer Society, 2012, p. 113–122.
- OMG. Software & Systems Process Engineering Meta-Model Specification. MA, Object Management Group, 2008. 236 p.
- OMG. Case Management Process Modeling (CMPM). Request For Proposal. MA, Object Management Group, 2009. 43 p.
- OMG. OMG Unified Modeling LanguageTM (OMG UML), Superstructure. MA, Object Management Group, 2011. 538 p.
- OMG. Case Management Model and Notation (CMMN). MA, Object Management Group, 2013. 89 p.
- OLDENHAVE, D. Formalism for a standard Case Management and its application on Scrum. Master thesis in Information Science, Radboud University of Nijmegen, 2010.
- PRESSMAN, R. S. Software Engineering. A Practitioner's Approach. 7th ed. New York, McGraw-Hill, 2010. 895 p ISBN 978-0-07-337597-7.
- SEI. Entidades certificadas. [online]. 2011. [Accessed: 25 April 2013]. Available from: https://sas.cmmiinstitute.com/pars/pars.aspx.

– SOMMERVILLE, I. Software Engineering. 7th ed. People's Republic of China, Addison-Wesley and Pearson Education, 2007. 865 p. ISBN 978-0-321-31379-9.
– SUBRAMANIAN, G H., JIANG, J. J. and KLEIN, G. Software quality and IS project performance improvements from software development process maturity and IS implementation strategies. Journal of Systems and Software, 2007, 80(4): p. 616–627. ISSN 0164-1212.
– VAN DER AALST, W. M. P, STOFFELE, M. and WAMELINK, J. W. F. Case handling in construction. Automation in Construction, 2003, 12(3): p. 303–320.
– VAN DER AALST, W. M. P, WESKE, M. and GRÜNBAUER, D. Case handling: a new paradigm for business process support. Data Know Eng, 2005, 53(2): p. 129–162.
– WEBER, B and REICHERT, M. Chapter 3. Flexibility Issues in Process-Aware Information Systems. In: WEBER, B and REICHERT, M. Enabling Flexibility in Process-Aware Information Systems. Challenges, Methods, Technologies. Berlin, Springer-Verlag, 2012: p. 43-55.