Modelado del proceso de desarrollo de software con el Modelo y Notación de Gestión de Casos (CMMN)

Yanet Vega Miniet
Universidad de las Ciencias Informáticas, La Habana, Cuba, yvegam@uci.cu

Michel Ramos Navarro
Universidad de las Ciencias Informáticas, La Habana, Cuba, ram@uci.cu

ABSTRACT

When software processes are modeled the traditional approach of workflows is used. The software process is a knowledge-intensive process and this approach is too restrictive and generates problems to manage change in this processes. A relatively new solution to these problems is case management. The Case Management Model and Notation (CMMN) is a notation to model process using this approach. The aim of this article is to assess the feasibility of modeling the software development process using CMMN. Basic concepts related to case management are described, the software development process is defined as a knowledge-intensive process, and a relationship between the requirements to model it, case management and CMMN is established.
This work concluded that is possible to model the software development process using the case management approach and CMMN in particular. But due to relative novelty and change of paradigm the process models can result little understandable. Is out of the scope of case management the process variability support, later studies are required to add an appropriate mechanism for variability support in the software development process.

Keywords: software development process, case management, Case Management Model and Notation

RESUMEN

Cuando se modela el proceso de desarrollo de software se utiliza el enfoque tradicional de *workflow*. El desarrollo de software es un proceso de conocimiento intensivo y este enfoque es muy restrictivo y genera problemas para gestionar los cambios en estos procesos. Una solución relativamente nueva para estos problemas es la gestión de casos. El Modelo y Notación de Gestión de Casos (CMMN) es una notación para modelar procesos usando este enfoque. El objetivo de este artículo es valorar la factibilidad de modelar el proceso de desarrollo de software utilizando CMMN. Se describen los conceptos básicos de la gestión de casos, los elementos fundamentales de CMMN, se define el desarrollo de software como un proceso de conocimiento intensivo y se establece una relación entre los requisitos para modelarlo, la gestión de casos y CMMN.
Este trabajo concluyó que es posible modelar el proceso de desarrollo de software utilizando la gestión de casos y CMMN en particular. Pero debido a su relativa novedad y cambio de paradigma los modelos de proceso pueden resultar poco comprensibles. Está fuera del alcance de CMMN tratar la variabilidad de los procesos, se requieren estudios posteriores para incorporarle un mecanismo apropiado para tratar la variabilidad en el proceso de desarrollo de software.

Palabras claves: modelado del proceso de desarrollo de software, gestión de casos, Modelo y Notación de Gestión de Casos

1. INTRODUCCIÓN

La gestión del proceso de desarrollo de software se basa en el principio de que la calidad de un sistema está altamente influenciada por la calidad del proceso utilizado para adquirirlo, desarrollarlo y mantenerlo. Este

principio ha motivado un creciente interés por gestionar y mejorar continuamente los procesos de desarrollo de software. Estudios empíricos han mostrado que la mejora de los procesos de desarrollo de software tiene un impacto positivo en los desarrolladores (Lavallée and Robillard, 2012) y está relacionada con mejoras en el desempeño de los proyectos, la calidad de los productos obtenidos y el éxito de las organizaciones (Clarke and O'Connor, 2012; Subramanian et al., 2007).

Durante las últimas dos décadas se han publicado numerosos trabajos relacionados con la gestión y mejora de los procesos de desarrollo de software que pueden agruparse en dos grandes áreas. Una de ellas relacionada con los aspectos metodológicos de la gestión del proceso de desarrollo de software, donde CMM, CMMI, ISO/IEC 15504: 2004 y SPICE son algunas de las iniciativas más conocidas y utilizadas según Pino en (Pino et al., 2008).

La otra área de investigación se relaciona con los aspectos técnicos de la gestión del proceso de desarrollo de software. En este sentido se han desarrollado desde los '90 los ambientes de ingeniería de software orientados a procesos *(PSEE*, por las siglas de *process-centered software engineering environments*) con el propósito de apoyar el modelado, análisis y ejecución de procesos de desarrollo de software (Gruhn, 2002). Los PSEE, como el resto de los sistemas de información orientados a procesos, generalmente utilizan como único mecanismo para dirigir una instancia de proceso el enrutamiento de las actividades 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 (Object Management Group, 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), por ejemplo, en los procesos de conocimiento intensivo.

Los procesos de conocimiento intensivo se caracterizan por ser no repetitivos (los modelos de dos instancias de procesos no son completamente iguales), no predecibles (el curso exacto de acciones depende de parámetros específicos de la situación) y emergentes (el conocimiento obtenido durante la ejecución del proceso determina el futuro curso de las acciones) (Weber and Reichert, 2012).

Una solución relativamente nueva para estos problemas de flexibilidad es la gestión de casos. Esta área de investigación surgió como respuesta a algunos de los problemas generados por el uso del enrutamiento de las actividades como único mecanismo para el control de los procesos de negocio (Van der Aalst et al., 2003; Van der Aalst et al., 2005). Por otra parte, el *Object Management Group* (OMG) publicó el Modelo y Notación de Gestión de Casos (CMMN, por las siglas de *Case Management Model and Notation*) (Object Management Group, 2013) con el objetivo de modelar este tipo de procesos.

El desarrollo de software es un proceso no repetitivo, las instancias de este proceso difieren debido a que los requisitos, recursos disponibles, compromisos y riesgos no son los mismos en todos los proyectos e inevitablemente el equipo de desarrollo deberá modificar el proceso mientras este se ejecuta. Por otra parte, no es posible predecir durante el diseño del proceso todas las incidencias que puedan ocurrir, y el equipo de desarrollo deberá decidir qué acciones ejecutar para resolverlas y evitar desviaciones en los planes de proyecto. Además, el conocimiento obtenido durante la ejecución del proceso determina el curso futuro de las acciones. Por ejemplo, los requisitos identificados se utilizan como base para identificar qué componentes se pueden reutilizar y qué pruebas son requeridas. Por lo tanto, el desarrollo de software es un proceso de conocimiento intensivo. Esto coincide con lo concluido por Oldenhave en (Oldenhave, 2010), quien utiliza la gestión de casos para modelar y ejecutar un proceso de desarrollo de software basado en Scrum.

El objetivo de este trabajo es valorar la factibilidad de modelar el proceso de desarrollo de software utilizando CMMN.

El resto del trabajo está estructurado como se describe a continuación. La sección 2 describe las bases y conceptos en los que se sustenta el modelado del proceso de desarrollo de software utilizando CMMN. Esta sección está dividida en dos partes. La sección 2.1 describe los conceptos básicos relacionados con la gestión de casos. La sección 2.2 describe los elementos fundamentales de CMMN. La sección 3 define el desarrollo de software como un proceso de conocimiento intensivo y establece una relación entre los requisitos para modelarlo con lo tratado en las dos secciones anteriores. Finalmente se concluye el estudio y se indican elementos que deben tratarse en futuros trabajos de investigación.

2. BASES PARA EL MODELADO DEL PROCESO DE DESARROLLO DE SOFTWARE UTILIZANDO CMMN

2.1 GESTIÓN DE CASOS

La gestión de casos fue definida formalmente por Van der Aalst en (Van der Aalst et al., 2005) con el objetivo de resolver algunos problemas de flexibilidad generados por el uso del enrutamiento de las tareas como único mecanismo para dirigir una instancia de proceso (Van der Aalst et al., 2003; Van der Aalst et al., 2005). En (Van der Aalst et al., 2005) estos autores proponen la gestión de casos como "un nuevo paradigma para apoyar los procesos de conocimiento intensivo". En los *workflow* el estado de las instancias de proceso se determina por el estado del flujo de control, pero en la gestión de casos se determina por la presencia de los objetos de datos. "Esto es verdaderamente un cambio de paradigma: la gestión de casos es también dirigida por el flujo de datos y no exclusivamente por el flujo de control." (Van der Aalst et al., 2005)

La Figura 1 muestra un esquema del meta-modelo de la gestión de casos según (Van der Aalst et al., 2005) y a continuación se explican los conceptos fundamentales según este mismo trabajo.

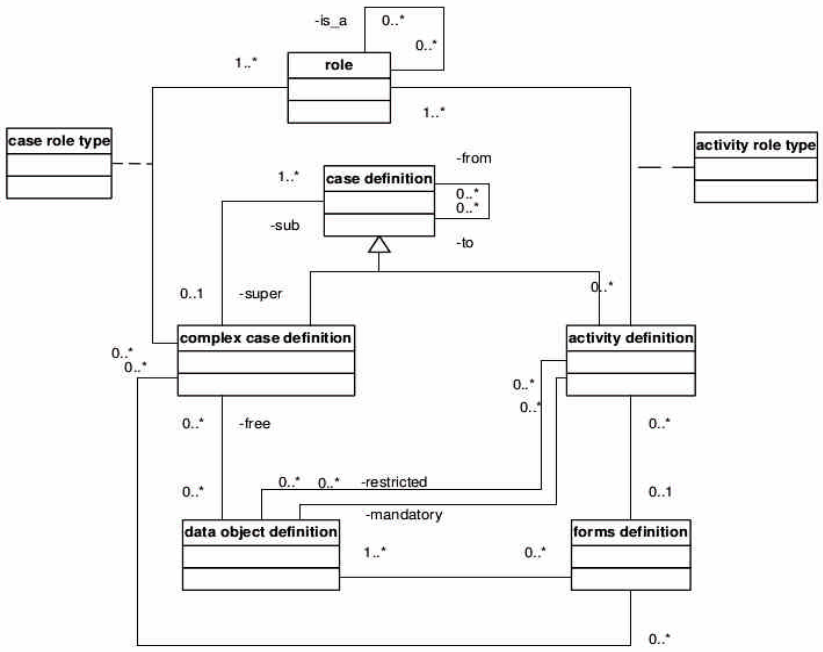

Figura 1: Esquema del meta-modelo de gestión de casos. Tomado de (Van der Aalst et al., 2005).

El concepto central de la gestión de casos es el caso (case *definition*), siendo este el "producto" que será obtenido como resultado de la ejecución del proceso. Este producto tiene estructura y estado. Las definiciones de casos pueden ser complejas (*complex case definition*) o atómicas (*activity definition*). Las definiciones complejas de casos consisten en un conjunto de definiciones de casos, resultando en una estructuración jerárquica de los casos.

El estado y la estructura de un caso se basan en una colección de objetos de datos. Un objeto de datos (*data object definition*) es una pieza de información que está presente o no, y cuando está presente tiene un valor. Los objetos de datos pueden ser libres (*free*), estos se pueden modificar en cualquier momento mientras se ejecuta el caso; o pueden estar asociados a actividades como obligatorios (*mandatory*) y/o restringidos (*restricted*). Cuando un objeto de datos es obligatorio para una actividad para completar esta actividad hay que introducir ese dato. Si un objeto de datos es restringido para una actividad ese dato solo puede ser introducido en esa actividad o en cualquier otra para la cual también sea restringido.

Para gestionar un caso se requiere ejecutar actividades. Una actividad (*activity definition*) es una unidad lógica de trabajo. Las actividades se relacionan y los casos siguen patrones típicos, es decir, procesos. Un proceso (*process*) es el patrón típico de actividades para gestionar un tipo de caso.

Cuando los trabajadores del caso ejecutan una actividad deben poder acceder a toda la información del caso que necesiten, según los permisos que tengan. Los formularios (*forms definition*) se utilizan para presentar diferentes vistas de los objetos de datos asociados a un caso. Las actividades pueden estar relacionadas con formularios para mostrar los datos más relevantes.

Los actores (*actor*) son los trabajadores que ejecutan las actividades y se agrupan en roles (*role definition*). Un actor puede tener varios roles y cada rol puede tener varios actores. Los roles son específicos para cada proceso. Para una actividad hay tres tipos de roles: para ejecutarla (*execute*), para rehacerla (*redo*) y para ignorarla (*skip*). Los tipos de roles para una actividad se representan por la relación entre actividad (*activity*) y rol (*role*), representada en el diagrama por la clase *activity role type*. Los tipos de roles asociados con las definiciones complejas de casos típicamente se refieren a los roles de la organización. Los tipos de roles para una definición compleja de caso se representan por la relación entre rol (*role*) y definición compleja de caso (*complex case definition*), representada en el diagrama por la clase (*case role type*).

2.2 CASE MANAGEMENT MODEL AND NOTATION

En 2009 el OMG publicó una solicitud de propuestas para extender *Business Process Modeling Notation (BPMN)* con el objetivo de soportar el modelado de casos (Object Management Group, 2009). En enero de 2013 se publicó la versión beta del *Case Management Model and Notation* (CMMN) (Object Management Group, 2013) como respuesta a la publicación realizada en 2009. A continuación se explican los elementos fundamentales de esta notación, una descripción detallada puede encontrarse en (Object Management Group, 2013).

La Figura 2 muestra los conceptos fundamentales de CMMN y a continuación se explica brevemente esta figura.

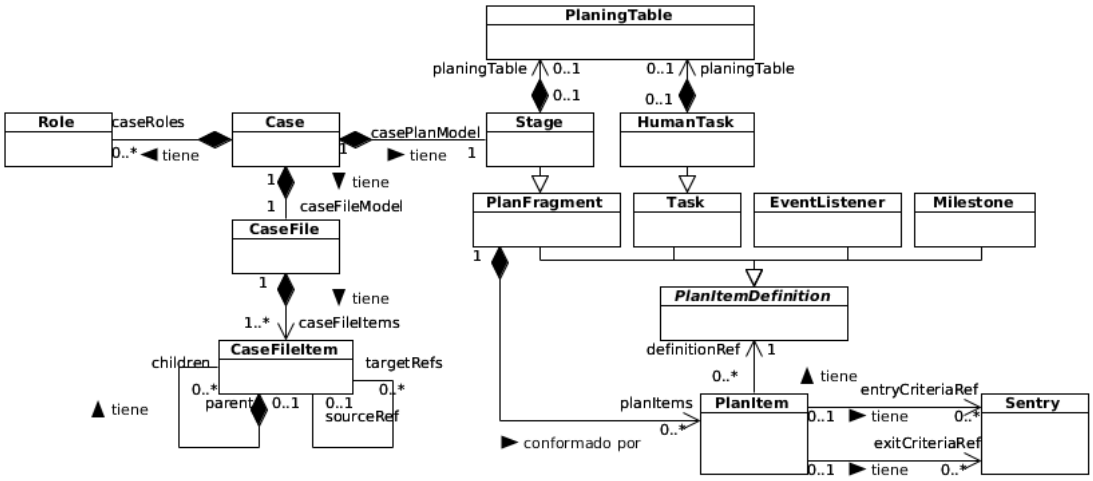

Figura 2: Conceptos fundamentales de CMMN. Elaboración propia basada en (Object Management Group, 2013)).

En CMMN el concepto de más alto nivel es el caso, representado por la clase *Case*. Un caso individual puede ser resuelto completamente *ad-hoc*, pero cuando crece la experiencia resolviendo casos similares, emergen patrones de comportamiento para resolver casos similares. Cada caso (*Case*) tiene asociado un plan que contiene estos patrones de comportamiento y las restricciones para la planeación en tiempo de ejecución de un tipo de caso. El plan de un caso se representa por la clase (*Stage*). Un (*Stage*) contiene otros elementos que se utilizan para resolver un caso: *Task*, *EventListener*, *Milestone* y *PlanFragment*. Todos estos elementos pueden tener condiciones asociadas (*Sentry*) que establecen los criterios de entrada y salida de cada uno de ellos.

La gestión de casos requiere planeación en tiempo de ejecución, por lo que el modelado debe expresar que la selección de las tareas a ejecutar, el orden de estas y los trabajadores que van a participar pueda hacerse durante la ejecución del proceso. Es por esto que CMMN incorpora un grupo de clases que permiten la planeación durante la ejecución del caso, por ejemplo *PlaningTable*. Los *Stage* y *HumanTask*, que es una especialización de *Task*, pueden tener *PlaningTable* asociadas.

Cada caso maneja información, la clase *Case* tiene asociado un archivo del caso (*CaseFile*), que a su vez contiene las informaciones en las instancias de la clase *CaseFileItem*. Un *CaseFileItem* representa una pieza de información de cualquier naturaleza, puede ser estructurada o no, simple o compleja. Puede ser, por ejemplo, un documento, una carpeta o una jerarquía de directorios. Un *CaseFileItem* puede contener otros *CaseFileItem*.

Por otra parte, cada caso tiene asociado un conjunto de roles (*Role*) que autorizan a las trabajadores a realizar las tareas humanas definidas en el plan del caso.

En CMMN las relaciones de dependencia entre las tareas se expresan por las condiciones en los *Sentry* aunque también se pueden representar gráficamente, esta representación gráfica no tiene ninguna semántica de ejecución asociada. Lo mismo ocurre con las relaciones entre las tareas y los *CaseFileItem*.

En general, CMMN mantiene y amplía los conceptos introducidos por (Van der Aalst Aalst et al., 2005) dándole más flexibilidad a los modelos.

3. MODELADO DEL PROCESO DE DESARROLLO DE SOFTWARE UTILIZANDO CMMN

Múltiples definiciones de proceso de desarrollo de software existen en la bibliografía sobre Ingeniería de Software, 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.

El desarrollo de software es un proceso de conocimiento intensivo según la definición dada por (Weber and Reichert, 2012) al respecto. Esta característica del proceso de desarrollo de software hace necesario considerar nuevos enfoques para modelar este proceso.

La Tabla 1 muestra la relación entre la definición dada por (Weber and Reichert, 2012) y las características del proceso de desarrollo de software.

Habiendo mostrado que el proceso de desarrollo de software es un proceso de conocimiento intensivo, es conveniente valorar, atendiendo a sus componentes y características, si es posible modelarlo utilizando el enfoque de la gestión de casos.

En (Vega and Ramos, 2013) se definen las siguientes características como deseables en los lenguajes para modelar el proceso de desarrollo de software:

- El lenguaje de modelado permite representar los componentes fundamentales del proceso de desarrollo de software. La Figura 3 muestra un modelo conceptual de estos componentes.

  Según varios de los principales estudiosos de la Ingeniería de Software (Cockburn, 2001; Sommerville, 2007; Presuman, 2010) el proceso de desarrollo de software está conformado por un conjunto de tareas, que son las unidades de trabajo que conforman los procesos, pueden ser atómicas o compuestas. La clase Tarea representa este concepto. Las tareas utilizan, generan o modifican productos de trabajo, representados por la clase Producto de trabajo, se realizan por un rol (Rol), y pueden realizarse según un Método. Además, los roles son responsables de los productos de trabajo y de la utilización de los métodos. Un rol define un conjunto de competencias y responsabilidades relacionadas que se requieren para ejecutar las tareas. Los Métodos constituyen guías para realizar las tareas.

Tabla 1: Relación entre las características de los procesos de conocimiento intensivo y las características del proceso de desarrollo de software. Elaboración propia.

Características de los procesos de conocimiento intensivo (Weber and Reichert, 2012)	Características del proceso de desarrollo de software
Se caracterizan por ser no repetitivos (los modelos de dos instancias de procesos no son completamente iguales)	Los requisitos, recursos disponibles, compromisos y riesgos no son los mismos en todos los proyectos y cada equipo de desarrollo deberá modificar el proceso mientras este se ejecuta para adaptarlo a su situación. Por esto las instancias de este proceso no son completamente iguales. Como consecuencia el proceso de desarrollo de software es no repetitivo.
No predecibles (el curso exacto de acciones depende de parámetros específicos de la situación)	No es posible predecir durante el diseño del proceso de desarrollo de software todas los incidentes que puedan ocurrir, y el equipo de desarrollo deberá decidir qué acciones ejecutar para resolverlas y evitar desviaciones en los planes de proyecto.
Emergentes (el conocimiento obtenido durante la ejecución del proceso determina el futuro curso de las acciones)	El conocimiento obtenido durante la ejecución del proceso determina cómo se debe proceder en el resto del proceso. Por ejemplo, los requisitos identificados se utilizan como base para identificar qué componentes se pueden reutilizar y qué pruebas son requeridas.

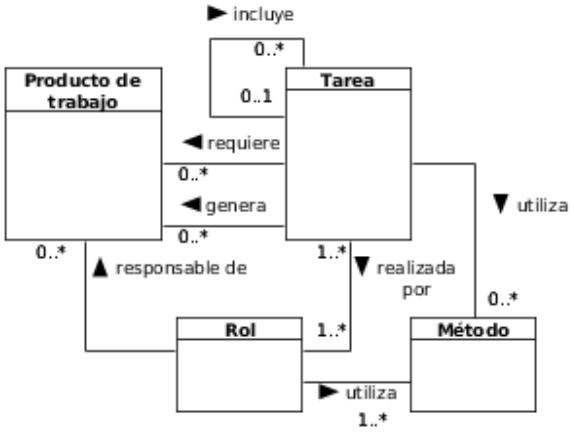

Figura 3: Componentes fundamentales del proceso de desarrollo de software. Elaboración propia.

- Los modelos de proceso obtenidos utilizando el lenguaje son comprensibles por humanos. El lenguaje de modelado de procesos tiene una representación gráfica. Esta representación gráfica resulta familiar para los interesados y cuenta con un conjunto de elementos básicos que pueden extenderse para añadir información adicional.
- El lenguaje de modelado de procesos permite representar subprocesos (Ellner et al., 2010).

- Los modelos de procesos obtenidos utilizando el lenguaje son ejecutables. Es posible ejecutar el modelo de proceso utilizando un motor de procesos o transformar los modelos en un lenguaje ejecutable.
- El lenguaje permite representar procesos así como variantes de estos y mantener la traza entre ellos.
- Los modelos de proceso pueden modificarse durante su ejecución. Es posible mientras se ejecuta el proceso crear e ignorar tareas e instancias de tareas, así como modificar las relaciones existentes entre estas. También se pueden crear y eliminar productos de trabajo y sus relaciones con las tareas.

La Tabla 2 muestra un resumen de la relación entre las características deseables en los lenguajes para modelar el proceso de desarrollo de software (Vega and Ramos, 2013), CMMN (Object Management Group, 2013) y los conceptos básicos de la gestión de casos introducidos por (Van der Aalst et al., 2005). Aquí se evidencia que la gestión de casos y CMMN satisfacen la mayoría de las características deseables para modelar el proceso de desarrollo de software. El único elemento que no está dentro del alcance de la gestión de casos y CMMN es el tratamiento de la variabilidad del proceso. Esto en sí mismo no constituye una limitación, ya que algunos trabajos han demostrado que es conveniente que los mecanismos para representar la variabilidad de los procesos sean independientes de los lenguajes de modelado de procesos.

Tabla 2: Relación entre las características deseables en los lenguajes para modelar el proceso de desarrollo de software, CMMN y los conceptos básicos de la gestión de casos (elaboración propia).

Características deseables en los lenguajes para modelar el proceso de desarrollo de software (Vega and Ramos, 2013)	**Conceptos básicos de la gestión de casos introducidos por (Van der Aalst et al., 2005)**	**CMMN (Object Management Group, 2013)**
El lenguaje de modelado permite representar los componentes fundamentales del proceso de desarrollo de software:	La clase case *definition* representa el proceso, en tanto contiene las acciones requeridas para cumplir un objetivo. Las unidades de trabajo se representan con las clases *complex* case *definition* y *activity definition.* Las unidades de trabajo se representan con las clases *complex* case *definition* y *activity definition.* Solo se pueden representar objetos de datos individuales con data *object definition*. No se pueden representar elementos de compuestos, los productos de trabajo, en general, son elementos compuestos.	La clase Case representa el proceso, en tanto contiene las acciones requeridas para cumplir un objetivo. Las unidades de trabajo se representan como *Stage* y *Task.* Amplió el concepto de objeto de datos introducido en (Van der Aalst et al., 2005), permitiendo crear objetos de datos compuestos, que incluyen desde datos individuales hasta documentos, carpetas y estructuras de directorios. Esto se logra utilizando las clases *CaseFile* y *CaseFileItem*. De esta manera es posible modelar no solo los productos de trabajo, sino su organización. Aunque CMMN define la semántica para esto, no provee una representación gráfica para definir la estructura. Esto puede hacerse utilizando otros lenguajes, como *Unified Model Language* (UML).
Rol	Los roles se representan con la clase role *definiton.*	Los roles se representan con la clase Role.

Características deseables en los lenguajes para modelar el proceso de desarrollo de software (Vega and Ramos, 2013)	**Conceptos básicos de la gestión de casos introducidos por (Van der Aalst et al., 2005)**	**CMMN (Object Management Group, 2013)**
Método	No define ninguna notación ni semántica particular para esto. No obstante, siendo un método una guía para realizar una tarea, esa tarea puede representarse como un *complex case definition* que contiene las actividades (*activity definition*) indicadas por el método.	No define ninguna notación ni semántica particular para esto. No obstante, siendo un método una guía para realizar una tarea, esa tarea puede representarse como un *Stage* que contiene las tareas indicadas por el método.
Los modelos de procesos obtenidos utilizando el lenguaje son comprensibles por humanos.	No define una notación gráfica para representar los procesos.	Define una notación gráfica para representar los procesos con un grupo de 11 elementos básicos y 7 decoradores para estos elementos, que añaden información adicional. No obstante, al ser una notación relativamente nueva es poco conocida y esto puede afectar su comprensión. Además, implica un cambio de paradigma en relación a las notaciones de modelado de procesos que generalmente se utilizan.
El lenguaje de modelado de procesos permite representar subprocesos.	Es posible representar subprocesos mediante la clase definición compleja de caso (*complex case definition*).	Es posible representar subprocesos mediante la clase *Stage*.
Los modelos de procesos obtenidos utilizando el lenguaje son ejecutables.	Define una semántica para la ejecución de los procesos.	Define una semántica para la ejecución de los procesos.
El lenguaje permite representar procesos así como variantes de estos y mantener la traza entre ellos.	No está dentro del alcance de la gestión de casos resolver la variabilidad de los procesos.	No está dentro del alcance de CMMN resolver la variabilidad de los procesos.
Los modelos de procesos pueden modificarse durante su ejecución.	No está definido.	Permite la planeación del proceso durante su ejecución mediante *PlaningTable* y otros conceptos asociados.

Considerando que el desarrollo de software es un proceso de conocimiento intensivo, y que satisface la mayoría de los requisitos definidos en (Vega and Ramos, 2013) para modelar este proceso, se concluye que el enfoque de gestión de casos, y particularmente CMMN, se puede utilizar para modelar el proceso de desarrollo de software. Esto coincide con lo demostrado por (Oldenhave, 2010).

4. CONCLUSIONES

En este trabajo se argumentó que el desarrollo de software es un proceso de conocimiento intensivo y se analizó la factibilidad de modelarlo utilizando CMMN. Este análisis concluyó que es posible modelar el proceso de desarrollo de software utilizando el enfoque de la gestión de casos y en particular CMMN.

Además, este estudió permitió identificar como una debilidad del uso de CMMN para modelar el proceso de desarrollo de software su relativa novedad y el cambio de paradigma en relación a las notaciones de modelado de procesos que generalmente se utilizan. Esto puede conducir a una comprensión insuficiente de los modelos elaborados con CMMN. En este sentido se debe analizar en futuros trabajos la comprensión de los modelos de proceso utilizando CMMN.

Por otra parte no está dentro del alcance de CMMN el tratamiento de la variabilidad de los procesos. Esto en sí mismo no constituye una limitación, ya que es conveniente que los mecanismos para tratar la variabilidad de los procesos sean independientes de los lenguajes de modelado de procesos. Pero se requieren estudios posteriores para seleccionar e incorporar un mecanismo apropiado para tratar la variabilidad en el proceso de desarrollo de software.

REFERENCIAS

Clarke, P., and O'connor, R.V. (2012). "The influence of SPI on business success in software SMEs: An empirical study". *Automated Software Evolution* [online], Vol. 85, No. 10, pp. 2356–2367.

Cockburn, A. (2001). *Agile Software Development*. 1st edition, Addison Wesley.

Dumas, M, Van der Aalst, W. and Ter Hofstede, A. (2005). *PROCESS-AWARE INFORMATION SYSTEMS. Bridging People and Software Through Process Technology*. New Jersey. ISBN 978-0-471-66306-5.

Ellner, R., Al-Hilank, S., Drexler, J., Jung, M., Kips, D. and Philippsen, M. (2010). "eSPEM – A SPEM Extension for Enactable Behavior Modeling". *Lecture Notes in Computer Science* [online]. Springer Berlin/Heidelberg. pp. 116–131. ISBN 978-3-642-13594-1.

Ferreira, H. (2008). *Automatic Plan Generation and Adaptation by Observation: Supporting complex human planning*. Tesis de doctorado. Portugal: Universidad de Oporto.

Gruhn, V. (2002). "Process-Centered Software Engineering Environments, A Brief History and Future Challenges". *Annals of Software Engineering* [online], Vol. 14, No. 1, pp. 363–382.

IEEE. (1990) *IEEE Standard Glossary of Software Engineering Terminology*. Institute of Electrical and Electronics Engineers, Inc.

Lavallée, M. and Robillard, P.N. (2012). "The impacts of software process improvement on developers: A systematic review". In: *Software Engineering (ICSE), 34th International Conference on*. pp. 113–122.

OMG. (2009). *Case Management Process Modeling (CMPM). Request for Proposal.*

OMG. (2013) *Case Management Model and Notation (CMMN)*.

Oldenhave, D. (2010). *Formalism for a standard Case Management and its application on Scrum*. Maestría. Holanda: Radboud University of Nijmegen.

Pino, F., García, F. and Piattini, M. (2008). "Software process improvement in small and medium software enterprises: a systematic review". *Software Quality Journal* [online], Vol. 16, No. 2, pp. 237–261.

Pressman, R. S. (2010) *Software Engineering. A Practitioner's Approach*. 7rd edition. New York: McGraw-Hill. ISBN 978-0-07-337597-7.

Sommerville, I. (2007). *Software Engineering*. 8rd edition. República de China: Pearson Education Limited. ISBN 978-0-321-31379-9.

Subramanian, G. H., Jiang, J. J. and Klein, G. (2007) “Software quality and IS project performance improvements from software development process maturity and IS implementation strategies”. *Software Performance 5th International Workshop on Software and Performance* [online], Vol. 80, No. 4, pp. 616–627.

Van der Aalst, W., Stoffele, M. and Wamelink, J. (2003). “Case handling in construction”. *Automation in Construction*, Vol. 12, No. 3, pp. 303–320.

Van der Aalst, W., Weske, M. and Grünbauer, D. (2005). “Case handling: a new paradigm for business process support”. *Data Know Eng*, Vol. 53, no. 2, pp. 129–162.

Vega, Y. and Ramos, M. (2013). “Características deseables en los lenguajes de modelado de procesos para modelar el proceso de desarrollo de software”. In: *Memorias de Informática 2013*. La Habana. 2013. ISBN 978-959-7213-02-4.

Weber, B. and Reichert, M. (2012). *Enabling Flexibility in Process-Aware Information Systems.Challenges, Methods, Technologies.*

Authorization and Disclaimer

Authors authorize LACCEI to publish the paper in the conference proceedings. Neither LACCEI nor the editors are responsible either for the content or for the implications of what is expressed in the paper.