ESENCIALIZACIÓN DE LA PRÁCTICA CONTROL DE CAMBIOS DE SOFTWARE DE RUP UTILIZANDO EL MODELO PARA LA DEFINICIÓN DE PRÁCTICAS EN INGENIERÍA DE SOFTWARE

Jairo Arévalo Acosta, Nicolás Barrios Carvajal, Alexander Barón Salazar

Universidad de Nariño
Pasto, Colombia

Resumen

La aplicación de buenas prácticas para el control de cambios de software permite economizar costos, esfuerzo de trabajo y tiempo. También, permite conservar la integridad del producto. En el ciclo de vida del software y sin importar la etapa, los cambios se presentan de forma frecuente. Para controlar estos cambios, en ingeniería de software, se proponen diversas prácticas. Una de las prácticas más conocidas es la práctica de Control de Cambios de Software de RUP (CCS-RUP). La comunidad de la ingeniería de software define esta práctica de diferentes maneras. En estas definiciones no se presenta una estructura clara para la práctica CCS-RUP, es decir, es complejo identificar y definir los elementos que constituyen la práctica. Este hecho genera dificultad para entender, aplicar y evaluar la práctica en contextos reales. En este trabajo de grado se aplica el Modelo para la Definición de Prácticas en Ingeniería de Software a la práctica CCS-RUP. A partir de la aplicación del modelo, se obtiene una práctica bien formada y nombrada, fácil de entender, aplicar y evaluar. Este proceso se denomina esencialización. Una práctica esencializada facilita a los practicantes entender, aplicar y evaluar la práctica. La práctica esencializada tiene una estructura definida con elementos que se integran de manera sistémica. La práctica CCS-RUP esencializada se valida mediante un estudio de caso que permite simular un contexto real.

Palabras clave: RUP; Ingeniería de Software; control de cambios; esencialización de prácticas; Essence

Abstract

The application of good practices for the control of software changes allows saving costs, work effort and time. Also, it allows to preserve the integrity of the product. In the software life cycle and regardless of the stage, changes occur frequently. To control these changes, in software engineering, various practices are proposed. One of the best-known practices is RUP's Control Changes to Software (RUP-CCS) practice. The software engineering community defines this practice in different ways. These definitions do not present a clear structure for the RUP-CCS practice, ergo, it is complex to identify and define the elements that constitute the practice. This fact creates difficulty in understanding, applying and evaluating the practice in real contexts. In this degree paper, the Model for the Definition of Practices in Software Engineering is applied to the RUP-CCS practice. From the application of the model, a well-formed and named practice is obtained, easy to understand, apply and evaluate. This process is called essentialization. An essentialized practice makes it easier for practitioners to understand, apply, and evaluate the practice. Essentialized practice has a defined structure with elements that are integrated in a systemic way. The essentialized RUP-CCS practice is validated through a case study that allows a real context to be simulated.

Keywords*: RUP; software engineering; control of software changes; practice essentialization; Essence*

1. Introducción

La aplicación de buenas prácticas para el control de cambios de software permite economizar costos, esfuerzo de trabajo y tiempo. También, permite conservar la integridad del producto. (Durango & Zapata, 2015). Mediante las metodologías de desarrollo y las formas como en estas, los practicantes generan soluciones, se evidencia que, los cambios en los requisitos son aceptados y tratados incluso en etapas avanzadas del ciclo de vida. Dicho de otra manera, para llevar a cabo con éxito el cambio y lograr la satisfacción del cliente, se debe crear una comunicación clara y así evitar el sobre esfuerzo en un mismo producto de trabajo (Beck et al., 2001). Así, un equipo de trabajo puede alcanzar un producto software funcional que evidencia el progreso del proyecto, pues los cambios son inevitables (Berzal, 2004). Atendiendo la importancia del control de cambios, es importante contar con una práctica fácil de entender, aplicar y evaluar a fin de mejorar la eficacia de la empresa y la satisfacción del cliente.

Para controlar los cambios, en ingeniería de software, se proponen diversas prácticas. Una de las prácticas más conocidas es la práctica de Control de Cambios de Software de la metodología Rational Unified Process (RUP) (Jacobson et al., 2001). La práctica CCS-RUP permite administrar los cambios en un proceso de desarrollo iterativo exitoso.

La comunidad de la ingeniería de software define esta práctica de diferentes maneras. En general, en estas definiciones no se presenta la estructura clara de la práctica CCS-RUP, es decir, es complejo identificar y definir los elementos que constituyen la práctica. Este hecho genera dificultad para entender, aplicar y evaluar la práctica en contextos reales (Kruchten, 2004).

En este trabajo de grado se aplica el Modelo para la Definición de Prácticas en Ingeniería de Software a la práctica CCS-RUP. A partir de la aplicación del modelo, se obtiene una práctica bien formada y nombrada, fácil de entender, aplicar y evaluar. Este proceso se denomina esencialización. El modelo permite identificar de manera fácil, los elementos que componen la práctica y la manera como se integran. Además, facilita al practicante entender, aplicar y evaluar la práctica. De esta manera, se consolida una definición de la práctica CCS-RUP bien formada y nombrada (Barón, 2019).

La práctica esencializada tiene una estructura definida con elementos que se integran de manera sistémica (Jacobson et al., 2012). El practicante entiende la práctica porque los elementos que la constituyen se definen e identifican claramente siguiendo el Modelo para la Definición de Prácticas en Ingeniería de Software. El practicante aplica fácilmente la práctica porque el criterio de entrada, el criterio de finalización, las actividades y los productos de trabajo están claramente definidos. La definición clara de productos de trabajo, criterios de entrada y criterios de finalización, permite evidenciar la finalización exitosa de la práctica. Así, se facilita la evaluación de la práctica (Barón, 2019).

2. Planteamiento del problema

En el ciclo de vida del software y sin importar la etapa, los cambios se presentan de forma frecuente. Estos cambios se dan cuando se desea modificar los requisitos, también, para disminuir o ampliar el alcance e impacto del proyecto, o igualmente, es necesario realizar modificaciones al enfoque técnico. Todos estos cambios son una acción de respuesta a los reportes de problemas sin importar la fuente (IEEE Computer Society, 2014). Estos cambios pueden ser solicitados por los clientes o por el mismo equipo de desarrollo.

Por tanto, en ingeniería de software se evidencia la necesidad de controlar los cambios del software para mantener la calidad del producto. La carencia de una buena práctica para abordar los cambios afecta las características del software, vida útil, calidad, diseño, reutilización, productividad y la competitividad del producto, factores que influyen directamente aumentando los costos (IEEE Computer Society, 2014).

Una de las prácticas más conocidas para el control de cambios de software es la que se propone en RUP (Jacobson et al., 2001). La comunidad de la ingeniería de software define esta práctica de diferentes maneras. Por ejemplo, (Jiménez, 2016) la define de manera gráfica. A pesar de usar diagramas y figuras, es difícil identificar los criterios de entrada y criterios de finalización de la práctica, pues no se encuentran claramente definidos. Esto dificulta entender, aplicar y evaluar la práctica.

La definición que propone Kruchten (2004) es textual y parcialmente gráfica. Aunque se identifican algunos elementos de la práctica tales como las actividades, no se puede identificar y definir elementos de la práctica, tales como criterios de entrada, criterios de salida y productos de trabajo, esto hace difícil entender, aplicar y evaluar la práctica. En la definición de (Rational Software

Company, 2001), se halla una descripción textual, esta representación resulta insuficiente, debido que, no se identifican ni se definen claramente los elementos que componen la práctica, no se evidencian criterios de entrada, criterios de salida y productos de trabajo, en consecuencia, se dificulta entender aplicar y evaluar la práctica.

En general, en estas formas de definición de la práctica CCS-RUP es complejo identificar y definir los elementos que constituyen la práctica. Este hecho genera dificultad para entender, aplicar y evaluar la práctica en contextos reales (Kruchten, 2004). En este trabajo de grado se identifican como elementos constitutivos de la práctica, los que propone Barón (2019) en el Modelo para la Definición de Prácticas en Ingeniería de Software.

3. Marco teórico

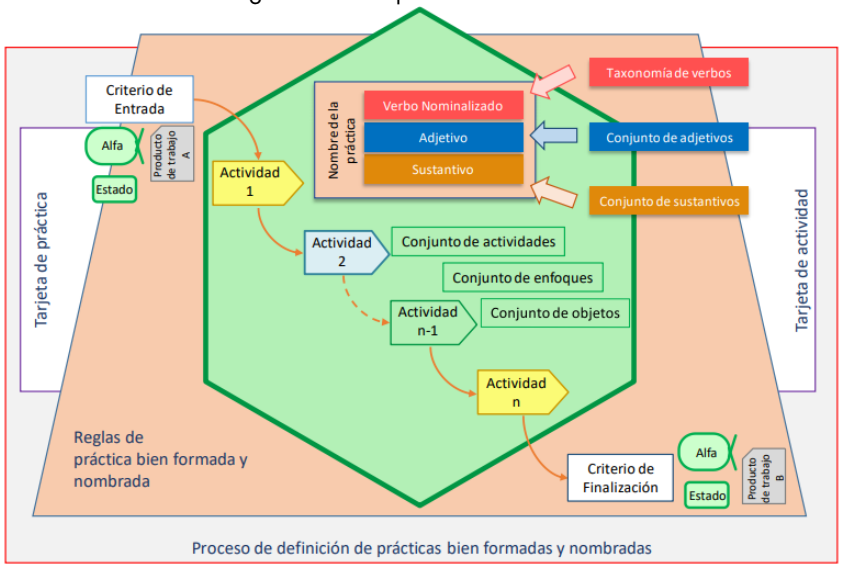
RUP: RUP es un proceso de desarrollo de sistemas que se basa en principios sólidos de ingeniería de software, como adoptar un enfoque iterativo, basado en requisitos y centrado en la arquitectura para el desarrollo de software y que proporciona varios mecanismos como iteraciones y puntos de decisión para proporcionar visibilidad de gestión en el proceso de desarrollo. Además, proporciona un enfoque disciplinado para asignar y administrar tareas y responsabilidades en una organización de desarrollo de software. Al aplicar este proceso, los equipos de desarrollo de software pueden producir software de alta calidad. Que satisface las necesidades de sus usuarios finales y lo hace dentro de un cronograma y presupuesto predecible (Somerville, 2011).

RUP presenta un grupo de 6 prácticas de desarrollo llamadas “Mejores Prácticas”; no por la cuantificación de su valor, sino porque se observa que son utilizadas por organizaciones de forma exitosa (Rational Software Company, 2001), las prácticas son las siguientes: Desarrollo Iterativo de Software, Arquitectura Basada en Componentes, Modelado Visual de Software, Verificación de la Calidad del Software, Gestión de Requisitos de Software, Control de Cambios de Software.

Essence: La comunidad Software Engineering Method and Theory (SEMAT por sus siglas en inglés), consolida un marco de pensamiento denominado Essence que involucra restructurar la Ingeniería de Software y mejorar sus métodos. El estándar define a los métodos como un conjunto de prácticas que pueden ser usadas y que además representan las actividades que se llevan a cabo durante el desarrollo. La práctica en Essence, se define como la forma de abordar una actividad y que puede ser repetitiva cumpliendo un propósito en específico. Además, una práctica puede ser reutilizable por cualquier método.

Modelo para la Definición de Prácticas en Ingeniería de Software (Barón, 2019): Este es un modelo para la definición unificada y carente de ambigüedad de la práctica como constructo teórico en Ingeniería de Software. Los constructos teóricos y las proposiciones que sirven para caracterizar la práctica se identifican a partir del análisis de prácticas de diferentes enfoques y se integran en los componentes del modelo (Barón, 2019). Los componentes se integran de manera sistémica en el modelo. Cada componente cumple una función específica que se orientan a definir prácticas bien formadas y nombradas.

Figura 3-1 Componentes del modelo



Fuente: (Barón, 2019)

4. RSL sobre la definición de la práctica CCS-RUP

Una revisión sistemática de la literatura (a menudo denominada revisión sistemática) es un medio de identificar, evaluar e interpretar la investigación disponible y relevante para una pregunta en particular de investigación, área temática o fenómeno de interés. Los estudios individuales que contribuyen a una revisión sistemática se denominan estudios primarios (Kitchenham & Charters, 2007).

La RSL que se desarrolla en esta investigación se realiza siguiendo un proceso que es una adaptación de la estrategia metodológica para elaborar síntesis conceptuales en ingeniería de software que se propone en (Barón, 2019).

Esta RSL permite seleccionar los estudios primarios para la investigación. Inicialmente, se realiza la planeación de la RSL para definir los elementos que la guían. El primero de estos elementos que guían esta RSL es la necesidad de la RSL. Dado que, la comunidad de ingeniería de software define la práctica CCS-RUP de diferentes maneras, se necesita resumir y sintetizar las definiciones que la comunidad de ingeniería de software ofrece. Por lo tanto, la necesidad de esta RSL es: Identificar la manera como la comunidad de ingeniería de software define la práctica CCS-RUP.

El segundo y último elemento que guía esta RSL es la pregunta de la investigación. Esta pregunta guía las actividades de búsqueda de estudios primarios para realizar la investigación. En consecuencia, se define la siguiente pregunta: ¿De qué manera la comunidad de ingeniería de software define la práctica CCS-RUP?

Posteriormente, se realiza la RSL, es decir, se identifican los estudios primarios de las fuentes digitales y se aplican unos criterios de inclusión y exclusión para obtener los estudios realmente relevantes. Para lo cual, primero se definen cadenas de búsqueda las cuales son aplicadas en

diferentes fuentes de estudios y así se obtiene un universo de estudios posiblemente relevantes para la investigación. Con el universo de estudios como insumo, se definen los criterios de inclusión y exclusión que se aplican para obtener así, los estudios realmente relevantes.

Tabla 4-1 Criterios de inclusión y exclusión

Criterios de inclusión
▪ Práctica de Control de Cambios de Software de RUP ▪ Definición de la práctica Control de Cambios de Software de RUP ▪ Aplicaciones de la práctica Control de Cambios de Software de RUP en proyectos reales
Criterios de exclusión
▪ Estudios que hablen de RUP como una metodología ágil ▪ Estudios que presenten la metodología RUP con descripciones cortas o insuficientes

Fuente: Elaboración propia

Una vez obtenidos los estudios relevantes, se extrae y se sintetiza la información.

Finalmente, se realiza el reporte de la RSL, donde se presentan los estudios realmente relevantes para la investigación.

Tabla 4-2 Estudios relevantes

Estudios seleccionados	**Cita**
Representación en el Núcleo de SEMAT de Prácticas de Métodos de Desarrollo Basados en Planes	(Jiménez, 2016)
Rational Unified Process Best Practices for Software Development Teams	*(Rational Software Company,* 2001)
The Rational Unified Process: An Introduction	(Kruchten, 2004)
Aplicación de la metodología RUP y el patrón de diseño MVC en la construcción de un sistema de gestión académica para la Unidad Educativa Ángel De La Guarda	(Jaramillo, 2016)
Metodología para el desarrollo de software escalable para el departamento de pensiones del IESS	(Pozo, 2015)
Implementación de un portal web mediante la metodología RUP para optimizar los procesos de prestación de servicios de la empresa Programadores Web Perú S.A.C.	(Cerron, 2017)

Fuente: Elaboración propia

5. Análisis sobre la definición de la práctica CCS-RUP

El proceso análisis sobre la definición de la práctica CCS-RUP se compone de dos actividades. La primera es identificar en cada estudio, los componentes que constituyen una práctica bien formada y nombrada según Barón (2019). Segundo, se analiza si la forma en que cada estudio define la práctica CCS-RUP cumple las reglas que propone (Barón, 2019) para una práctica bien formada y nombrada.

En la primera actividad, se busca identificar en los estudios que son objeto de análisis, los elementos que constituyen una práctica según (Barón, 2019) y que definen una práctica bien formada y bien

nombrada. Los componentes de una práctica bien formada y bien nombrada según (Barón, 2019) son:

- Criterio de entrada que establece las condiciones para llevar a cabo la inicialización de la práctica, en este criterio de entrada se debe identificar un estado que permita definir el inicio de la práctica.
- Criterio de finalización que establece las condiciones necesarias para finalizar la práctica, en este criterio de finalización se debe identificar un estado que permita definir como finaliza la práctica.
- Conjunto de actividades que permite definir el proceso de aplicación de la práctica. Entre actividades existen relaciones de secuencias y de transferencia de recursos.
- Flujo de actividades que permite definir el orden de ejecución de las actividades y la transferencia de recursos entre ellas.

En la tabla 5-1 se presentan los componentes, que se utilizan en cada estudio para definir la práctica CCS-RUP. El resultado del análisis se describe de la siguiente manera: si en el estudio se encuentra el componente se marca la casilla con un "SI", en caso contrario se marca "NO".

Tabla 5-1 Análisis de componentes

Estudio	**Componente de práctica según Modelo para la definición de prácticas en ingeniería de software Barón (2019)**				
	Criterio de entrada	Criterio de finalización	Conjunto de actividades	Actividad	Flujo de actividades
(Jiménez, 2016)	NO	NO	SI	NO	NO
(Rational Software Company, 2001)	NO	NO	NO	NO	NO
(Kruchten, 2004)	NO	NO	SI	NO	NO
(Jaramillo, 2016)	NO	NO	NO	NO	NO
(Pozo, 2015)	NO	NO	SI	NO	NO
(Cerron, 2017)	NO	NO	SI	NO	NO

Fuente: Elaboración propia

En la segunda y última actividad del análisis, se analizan los estudios relevantes para determinar si la forma como describen la práctica, constituye una práctica bien formada y nombrada.

Según (Barón, 2019), una práctica bien formada es aquella cuyo conjunto de actividades cumple con las reglas de: (i) coherencia, o sea que el criterio de finalización de cada actividad aporta en el progreso del sustantivo hacia el criterio de finalización de la práctica; (ii) consistencia, dice que existe al menos una actividad cuyo criterio de entrada es igual al criterio de entrada de la práctica, también que existe al menos una actividad cuyo criterio de finalización es igual al criterio de finalización de la práctica, también, para cada actividad, su criterio de entrada es igual al criterio de finalización de al menos otra actividad o el de entrada de la práctica, por último, para cada actividad su criterio de finalización es igual al criterio de entrada de al menos otra actividad o el criterio de finalización de la práctica; (iii) suficiencia, que el cumplimiento del criterio de finalización de cada actividad permite el progreso del sustantivo hasta lograr el cumplimiento del criterio de finalización de la práctica.

Igualmente, (Barón, 2019) define una práctica bien nombrada como aquella práctica que tiene un nombre estructurado y adecuado de práctica que permite identificar sus elementos esenciales, compuesto por un verbo nominalizado que integra el conjunto de actividades que permite conducir el progreso del sustantivo hasta alcanzar el estado que se indica en el criterio de finalización de la práctica. Un adjetivo que integra los enfoques que se utilizan para realizar cada una de las actividades de la práctica y un sustantivo que se indica en el criterio de finalización de la práctica, dado el caso de que no exista criterio de finalización definido. El sustantivo que se utiliza para nombrar la práctica, es el sustantivo que tiene mayor progreso como resultado de la práctica.
En la tabla 5-2 se identifican los componentes anteriormente mencionados, al igual que en el anterior análisis, se marca con "SI", si el componente se identifica y con "NO" en caso contrario.

Tabla 5-2 Análisis de práctica bien formada y nombrada

Estudio	**Práctica bien formada**			**Práctica bien nombrada**		
	RCH	RCS	RSF	VN	A	S
(Jiménez, 2016)	NO	NO	NO	SI	NO	SI
(Rational Software Company, 2001)	NO	NO	NO	SI	NO	SI
(Kruchten, 2004)	NO	NO	NO	SI	NO	SI
(Jaramillo, 2016)	NO	NO	NO	SI	NO	SI
(Pozo, 2015)	NO	NO	NO	SI	NO	SI
(Cerron, 2017)	NO	NO	NO	SI	NO	SI

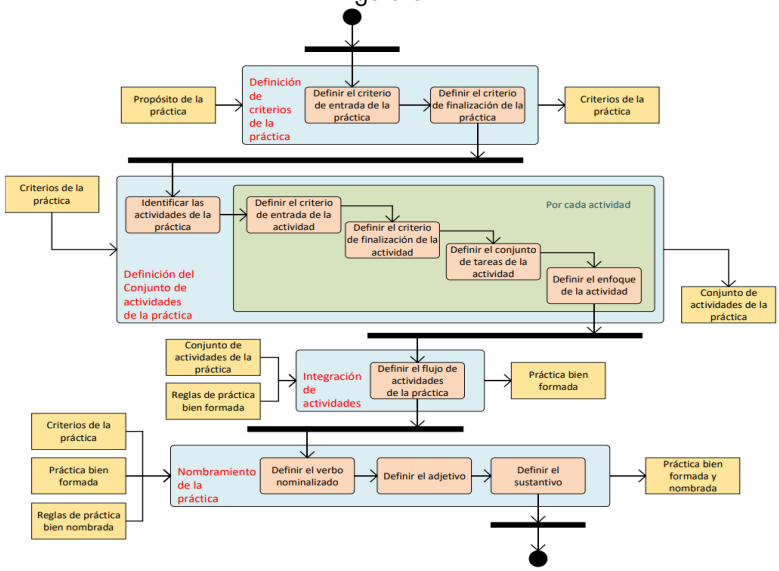
Fuente: Elaboración propia

Como se puede evidenciar, no existe, en los estudios objeto de análisis de esta investigación, una definición que cumpla en su totalidad, con las reglas de práctica bien formada y nombrada.

6. Conversión de la práctica CCS-RUP en una práctica bien formada y nombrada

Para la conversión de la práctica CCS-RUP, se identifica un estudio que defina la práctica CCS-RUP como referente. En este caso se usa la definición obtenida de Jiménez (2016), dado que ofrece la definición con mayor número de elementos. Posteriormente se realiza la conversión de la práctica CCS-RUP siguiendo el Modelo para la Definición de Prácticas en Ingeniería de Software que propone Barón (2019) para la definición de prácticas bien formadas y nombradas. En la figura 6-1 se describe este proceso de manera detallada.

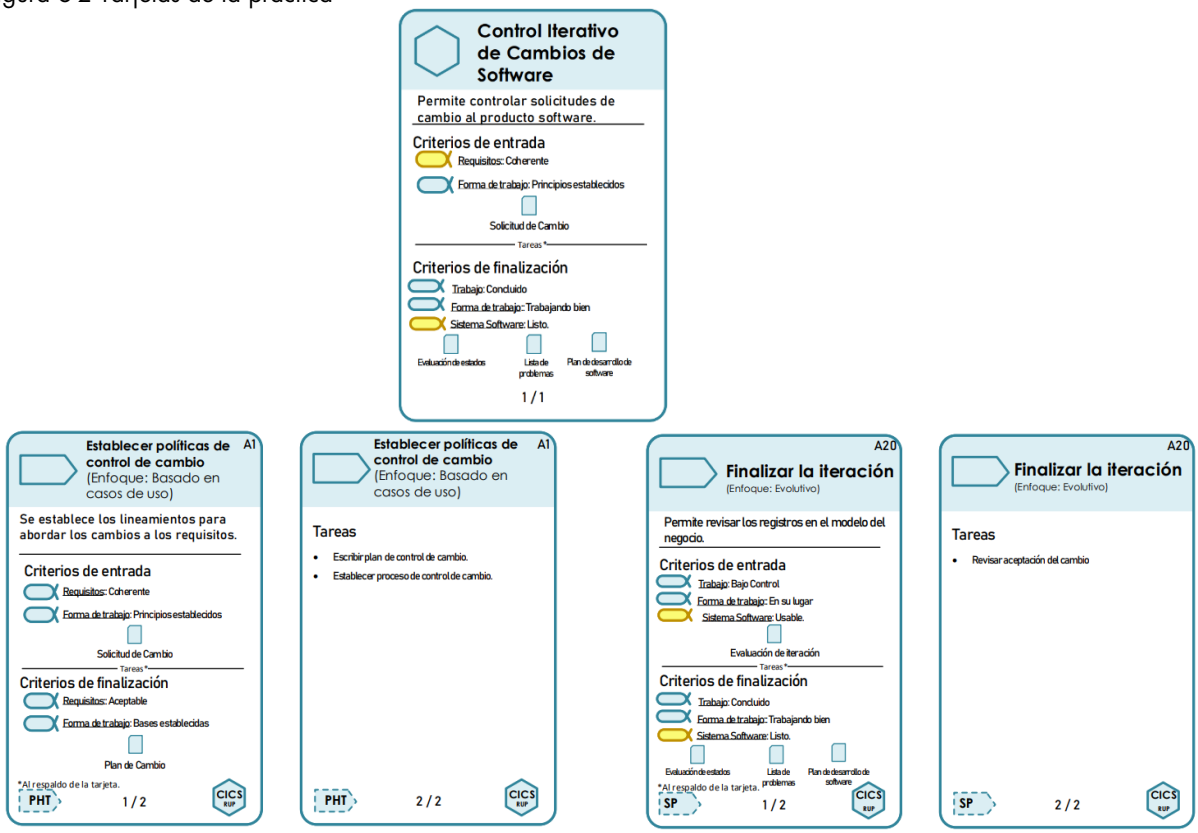
Figura 6-1



Fuente: (Barón, 2019)

En la figura 6-2 se presentan las tarjetas obtenidas luego de aplicar Modelo para la Definición de Prácticas en Ingeniería de Software que propone Barón (2019) bajo el estándar de Essence (Tarjeta de práctica y dos actividades).

Figura 6-2 Tarjetas de la práctica



Fuente: Elaboración propia

7. Conclusiones

- Con la finalización de este trabajo para la definición de la práctica CCS-RUP, se aporta una definición a la comunidad de ingeniería de software fácil de entender aplicar y evaluar en contextos reales para sus practicantes.
- La definición de la práctica CCS-RUP aporta a la comunidad de ingeniera de software una herramienta para economizar costos esfuerzos de trabajo y tiempo, puesto que, durante el ciclo de vida del desarrollo, se presentan cambios que deben ser tratados y aceptados a fin de agregar calidad y competencia del producto.
- La práctica se define siguiendo el Modelo para la Definición Unificada de la Práctica como Constructo Teórico en Ingeniería de Software (Barón, 2019). Esto permite una definición unificada y aceptada para la comunidad de ingeniería de software.

8. Referencias

- Barón, A. (2019). Modelo para la Definición Unificada de la Práctica como Constructo Teórico en Ingeniería de Software (Tesis de doctorado). Universidad Nacional, Medellín.
- Beck et al. (2001). Manifesto for Agile Software Development. Consultado el 20 de Enero de 2020 en http://agilemanifesto.org/
- Berzal. (2004). El ciclo de vida de un sistema de información. Consultado el 20 de Enero de 2020 en http://flanagan.ugr.es/docencia/2005-2006/2/apuntes/ciclovida.pdf
- Cerron, D. (2017). Implementación de un portal web mediante le metodología RUP para optimizar procesos de prestación de servicios de la empresa Programadores Web Perú S.A.C. (Tesis de pregrado). Escuela profesional de ingeniería de sistemas e informática, Lima.
- Durango, & Zapata. (2015). Una representación basada en Semat y RUP para el Método de Desarrollo SIG del Instituto Geográfico Agustín Codazzi. Revista Ingenierías USBMed, Vol. 6, No. 1, pp. 24-37.
- IEEE Computer Society. (2014). Guide to the Software Engineering Body of Knowledge. IEEE Computer Society Press.
- Jacobson, I., Booch, G., & Rumbaugh, J. (2001). El Proceso Unificado de Desarrollo de Software. Addison-Wesley.
- Jacobson, I., Ng, P., McMahon, P., Spence, I., & Lidman, S. (2012). The Essence of Software Engineering: The SEMAT Kernel. Communications of the ACM, Vol. 55, No. 12, pp. 42-49.
- Jaramillo, W. (2016). Aplicación de la metodología RUP y el patrón de diseño MVC den la construcción de un sistema de gestión académica para la Unidad Educativa Angel De La Guarda (Tesis de pregrado). Pontificia Universidad Católica del Ecuador, Quito.
- Jiménez, L. (2016). Representación en el Núcleo de SEMAT de Prácticas de Métodos de Desarrollo Basados en Planes (Tesis de Maestría). Universidad Nacional, Medellin.
- Kitchenham, B., & Charters, S. (2007). Guidelines for performing systematic literature. EBSE, Technical report.
- Kruchten, P. (2004). The Rational Unified Process: An Introduction. Boston, USA: Addison-Wesley.
- Pozo, W. (2015). Metodología para el desarrollo de software escalable para el departamento de pensiones del IESS (Tesis de maestría). Universidad Central del Ecuador, Quito.
- Rational Software Company. (2001). Rational Unified Process: Best practices for software development teams. Consultado el 20 de Enero de 2020 en https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

- Somerville, I. (2011). Software Engineering. Boston, Massachusetts, USA: Addison-Wesley.

Sobre los autores

- **Jairo Arévalo Acosta**: Egresado de Ingeniería de Sistemas de la Universidad de Nariño. jairoarevalo4@gmail.com
- **Nicolás Barrios Carvajal**: Egresado de Ingeniería de Sistemas de la Universidad de Nariño. nico9607@outlook.com
- **Alexander Barón Salazar**: Ingeniero de sistemas, Magister en Ingeniería Informática, Doctor en Ingeniería – Sistemas e Informática de la Universidad Nacional. Decano de la facultad de ingeniería de la Universidad de Nariño. abaron_98@udenar.edu.co

---