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

ARÉVALO ACOSTA JAIRO
BARRIOS CARVAJAL NICOLAS

PROGRAMA DE INGENIERÍA DE SISTEMAS
DEPARTAMENTO DE SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE NARIÑO
AGOSTO, 2022

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

Autores

ARÉVALO ACOSTA JAIRO, jairoarevalo4@gmail.com

BARRIOS CARVAJAL NICOLAS, nicko9607@gmail.com

Informe final de trabajo de grado presentado como requisito para optar al título de Ingeniero de Sistemas, en modalidad investigación.

Director

PhD, BARÓN SALAZAR ALEXANDER

PROGRAMA DE INGENIERÍA DE SISTEMAS

DEPARTAMENTO DE SISTEMAS

FACULTAD DE INGENIERÍA

UNIVERSIDAD DE NARIÑO

AGOSTO, 2022

NOTA EXCLUSIÓN DE RESPONSABILIDAD INTELECTUAL

"Las ideas y conclusiones aportadas en este Trabajo de Grado son responsabilidad de los autores".

Artículo 1° del Acuerdo No. 324 de octubre 11 de 1966, emanado del Honorable Consejo Directivo de la Universidad de Nariño.

NOTA DE ACEPTACIÓN

_____
_____
_____
_____
_____
_____

_____
Firma del director

_____
Firma del jurado evaluador

_____
Firma del jurado evaluador

_____
San Juan de Pasto, octubre de 2022

AGRADECIMIENTOS

Este trabajo de grado se realizó con el apoyo de la vicerrectoría de investigaciones postgrados y relaciones internacionales. Así mismo, el grupo de investigación Galeras.Net. Les agradecemos enormemente el confiar en este proyecto y brindarnos todo su apoyo.

También expresamos nuestra infinita gratitud y profundo agradecimiento al Phd. Alexander Barón Salazar, por su perseverancia e incansable labor como docente y director de este trabajo de grado. Por enseñarnos que no existen limitaciones en el camino del conocimiento y motivarnos a indagar en el maravilloso campo de la investigación.

A la Universidad de Nariño, nuestra alma mater por brindarnos la oportunidad de estudiar, por permitirnos alcanzar nuestros sueños y más importante, brindarnos el apoyo para enfrentar una nueva vida como seres humanos formados en la universalidad del conocimiento.

A nuestros compañeros y docentes de la facultad de Ingeniería de la Universidad de Nariño por su aprecio, confianza, apoyo y acompañamiento en esta etapa de nuestras vidas.

“Rodéate de personas que crean en tus sueños, animen tus ideas, apoyen tus ambiciones y saquen lo mejor de ti”

-Roy T. Bennet

Dedico este trabajo a mis padres, Olga Carvajal y Carlos Barrios. Las personas que más amo, valoro y admiro en este vasto mundo. Por sus incansables esfuerzos, por su interminable apoyo y comprensión estaré siempre agradecido.

A mi abuela, mi segunda madre, Olga Hernández, por bríndame su sabiduría y su amor incondicional. A mi hermano que siempre ha estado a mi lado, por sus enseñanzas y motivación.
A mis amigos, quienes se han convertido en una parte de mí, con quienes comparto tantísimos recuerdos y experiencias. Por su afecto y acompañamiento en los momentos más difíciles.
A ellos dedico este y todos mis logros, porque sin ellos no sería la persona que soy.
Nicolás Barrios Carvajal.

Dedico este trabajo a quienes han creído en mi desde siempre, a mis padres, quienes me han brindado su apoyo y su cariño sin limitaciones.

Durante los años he compartido con personas los momentos que me han traído a este punto de mi vida, estoy agradecido con todos y cada una de esas personas porque me han dejado una huella, algunas en la mente y otras en el corazón.

Lo dedico a mis amigos que se han convertido en mi familia y me han apoyado durante los años, han confiado en mí y me han brindado su más profundo cariño.

Para todas estas personas les dedico con mucho cariño el esfuerzo y la dedicación que ha representado llegar hasta este punto, muchas gracias.

Jairo Arévalo Acosta.

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.

TABLA DE CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

Pág.

GLOSARIO

ANGULAR: Es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página.

API: La interfaz de programación de aplicaciones, conocida también por la sigla API, en inglés, application programming interface, es un conjunto de sub rutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizada por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas de programación. Una API representa la capacidad de comunicación entre componentes de software. Se trata del conjunto de llamadas a ciertas bibliotecas que ofrecen acceso a ciertos servicios desde los procesos y representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software.

ESSENCE: Se trata del núcleo conceptual de la Ingeniería del Software que establece los conceptos fundamentales que requiere conocer todo ingeniero de software para trabajar con métodos de la Ingeniería del Software.

IEEE: El Instituto de Ingenieros Eléctricos y Electrónicos, conocido por sus siglas IEEE, en inglés Institute of Electrical and Electronics Engineers, es una asociación mundial de ingenieros dedicada a la normalización y el desarrollo en áreas técnicas, es la mayor asociación internacional sin ánimo de lucro formada por profesionales de las nuevas tecnologías, como ingenieros electricistas, ingenieros electrónicos, ingenieros de sistemas, ingenieros en computación, matemáticos aplicados, entre otras ramas de la ingenieria.

NODE: Es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

POSTGRES: Es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto, publicado bajo la licencia PostgreSQL, similar a la BSD o la MIT.

RUP: El Proceso Racional Unificado o RUP, por sus siglas en inglés de Rational Unified Process, es un proceso de desarrollo de software desarrollado por la empresa Rational Software. Junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos. Es un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

SOFTWARE: Es un término informático que hace referencia a un programa o conjunto de programas de cómputo, así como datos, procedimientos y pautas que permiten realizar distintas tareas en un sistema informático. Comúnmente se utiliza este término para referirse de una forma muy genérica a los programas de un dispositivo informático, sin embargo, el software abarca todo aquello que es intangible en un sistema computacional.

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.

En el ciclo de vida del software y sin importar la etapa, los cambios se presentan de forma frecuente. Estos cambios pueden ser solicitados por el cliente o por el mismo equipo de desarrollo. Así, se evidencia la necesidad de realizar cambios al software, a fin de adaptarlo a las condiciones variables del contexto y para optimizar los procesos (Pressman, 2010).

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 de Control de Cambios de Software de RUP (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. Existen propuestas en las que se presenta la práctica CCS-RUP de manera textual. Por ejemplo, la propuesta *Rational Unified Process Best Practices for Software Development Teams* (*Rational Software Company*, 2001). Otras propuestas, definen y representan esta práctica de manera gráfica. Por ejemplo, en la Representación en el Núcleo de SEMAT de Prácticas de Métodos de Desarrollo Basados en Planes (Jiménez, 2016). 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).

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 (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).

La práctica CCS-RUP esencializada se valida mediante un estudio de caso que permite simular un contexto real. El estudio de caso es una estrategia adecuada para validar el resultado de la investigación ya que permite simular un escenario de desarrollo de software similar al de una organización de software (Barón, 2019). La práctica esencializada se valida mediante su aplicación en un estudio de caso que simula un contexto real.

I. CONTEXTUALIZACIÓN

En este capítulo se presentan los elementos de identificación de la investigación. Inicialmente se relaciona el título de la investigación. Posteriormente, se define la línea de investigación en la que se inscribe este trabajo. Asimismo, se describen el alcance y delimitaciones de este trabajo. Luego se identifica la modalidad sobre la que aplica esta investigación. Seguido de una descripción del problema. Posteriormente se definen los objetivos a alcanzar para este trabajo. Por último, justificación de por qué se realiza esta investigación.

A. LÍNEA DE INVESTIGACIÓN

La línea de investigación en la que se enmarca este trabajo de grado es: Ingeniería de Software y Manejo de Información. La investigación está adscrita al Grupo de Investigación Galeras .NET del Departamento de Sistemas de la Universidad de Nariño.

B. PLANTEAMIENTO DEL PROBLEMA

Inicialmente se realiza el planteamiento del problema que trata esta investigación. Seguido se formula el problema en forma de pregunta, la cual sirve de base para definir el objetivo de la investigación. Finalmente se sistematiza este problema en preguntas específicas, estas son la base para definir los objetivos específicos.

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 Soriano (2005) 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.

1) Formulación del Problema

¿Cómo se debe definir la práctica CCS-RUP de tal manera que se facilite entender, aplicar y evaluar?

2) Sistematización del Problema

- ¿De qué manera la comunidad de ingeniería de software define la práctica CCS-RUP?
- ¿Cuáles son los elementos que constituyen la práctica CCS-RUP?
- ¿Cómo se debe definir la práctica CCS-RUP de tal manera que sus elementos estén claramente definidos?
- ¿De qué manera se debe validar que la práctica esencializada de CCS-RUP es fácil de entender, aplicar y evaluar?

C. JUSTIFICACIÓN

La naturaleza dinámica del contexto del software implica cambios continuos en el software. Los cambios se presentan en cualquier etapa del ciclo de vida como respuesta a problemas de diversa índole. El control de cambios es la actividad que permite preservar la calidad, las características originales, la vida útil, el diseño, la reutilización, la productividad y la competitividad del producto software (IEEE *Computer Society*, 2014). En ingeniería de software, diversos autores proponen prácticas para abordar el control de cambios. Por ejemplo, en (*Scrum Study*, 2013) se presenta la práctica Panel de Control de Cambios. Igualmente, (*Software Engineering Institute*, 2010) propone la práctica Cambios a Requisitos y Control de Cambios. Asimismo, en (*International Organization for Standardization*, 2015) en el estándar ISO 9001 se propone la práctica Planificación de los Cambios. También, en (*Rational Software Company*, 2001) se presenta la práctica CCS-RUP.

La práctica CCS-RUP es una de las prácticas más conocidas para el control de cambios (Jacobson *et al.*, 2001). En ingeniería de software, esta práctica se define de manera diversa: textual, gráfica y combinando estas dos formas de representación (Jiménez, 2016; Soriano, 2005; *Rational Software Company*, 2001).

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 dificulta, entender, aplicar y evaluar la práctica. Como un aporte a la solución de esta problemática, en este trabajo de grado, se identifica los elementos constitutivos de la práctica, se define la práctica CCS-RUP según el modelo que propone Barón (2019). De esta manera, se propone a la comunidad de la ingeniería de software una definición de la práctica CCS-RUP que facilita su aplicación en contextos reales. Así, los practicantes pueden preservar la calidad, las características originales, la vida útil, el diseño, la reutilización, la productividad y la competitividad del producto software.

D. OBJETIVOS

A continuación, se establecen el objetivo general y objetivos específicos, como respuesta a las preguntas establecidas para esta investigación.

a) Objetivo General

Definir la práctica CCS-RUP aplicando el Modelo para la Definición de Prácticas en Ingeniería de Software de tal manera que se facilite entender, aplicar y evaluar la práctica.

1) Objetivos Específicos

- Identificar los mecanismos que la comunidad de ingeniería de software utiliza para definir la práctica CCS-RUP.
- Identificar elementos que constituyen la práctica CCS-RUP de acuerdo con la manera como la comunidad de ingeniería de software la define.
- Definir la práctica CCS-RUP de acuerdo con el Modelo para la Definición de Practicas en Ingeniería de Software.
- Validar la práctica esencializada CCS-RUP mediante su aplicación en un estudio de caso que simula un contexto real.

II. . MARCO TEÓRICO

En este capítulo se presentan los referentes conceptuales que son el fundamento teórico de la investigación y que son necesarios para el desarrollo de esta investigación. Inicialmente se define la metodología RUP y sus mejores prácticas. Seguido de la presentación del núcleo de SEMAT que se utiliza para la esencialización de prácticas de software. Posteriormente se describe el Modelo para la Definición de Prácticas en Ingeniería de Software que proporciona los mecanismos para obtener una práctica bien formada y nombrada.

A. RUP Y EL CONTROL DE CAMBIOS DE SOFTWARE

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 (Somerville, 2011).

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).

1) Ciclo de vida

El ciclo de vida de RUP está estructurado en dos dimensiones, dinámica y estática.

a) Estructura dinámica (Horizontal)

La estructura dinámica representa la dimensión del tiempo del proceso. Muestra como el proceso el proceso expresado en ciclos, fases, iteraciones e hitos, se desarrolla durante el ciclo de vida de un proyecto (Kroll & Kruchten, 2003).

En cuanto a la organización dinámica del proceso, ciclo de vida se divide en ciclos, haciendo que cada uno de los ciclos trabaje en una nueva generación del producto. Cada ciclo se divide en 4 fases, inicio, elaboración, construcción y transición (Somerville, 2011). Además, las fases también pueden desglosarse en iteraciones. Una iteración es un ciclo de desarrollo completo que da como resultado una versión (interna o externa) de un producto que crece de forma incremental de iteración en iteración para convertirse en el sistema final; el usar iteraciones trae consigo varios beneficios para el desarrollo del producto, como, por ejemplo, mitigación de riesgos o una mayor facilidad para adaptarse a cambios (*Rational Software Company*, 2001).

Cada una de las fases concluye con un hito, decir, un punto en el tiempo en el cual se evalúa el desarrollo del proyecto y en el cual se deben tomar críticas. Los hitos de RUP de acuerdo a su fase son: objetivos del ciclo de vida (fase de inicio), arquitectura del ciclo de vida (fase de elaboración), capacidad operativa inicial (fase de construcción) y lanzamiento del producto (fase de transición) (*Rational Software Company*, 2001).

b) Estructura estática (Vertical)

La estructura estática describe como los elementos del proceso (actividades, disciplinas, artefactos y roles) se agrupan lógicamente las disciplinas o flujos de trabajo (Kroll & Kruchten, 2003).

Se describen los elementos así:

- Rol o trabajador: Define el comportamiento y las responsabilidades de un individuo o un grupo de individuos que trabajan como equipo. (Kroll & Kruchten, 2003).
- Actividad: Es una unidad de trabajo que se puede solicitar a un individuo con ese rol que la realice; una actividad tiene un objetivo claro y por lo general involucran a un trabajador y un artefacto. (Kroll & Kruchten, 2003).
- Artefacto: Es una pieza de información producida, modificada o utilizada mientras se trabaja hacia el producto final. Los artefactos son utilizados como entrada por los trabajadores para realizar una actividad y son el resultado o salida de tales actividades (Kroll & Kruchten, 2003).
- Disciplina o flujo de trabajo: Es una secuencia de actividades que produce un resultado de valor observable y que en términos de UML se puede expresar como un diagrama de secuencia o actividad. (Kroll & Kruchten, 2003).

Existen 9 flujos de trabajo principales que a su vez se dividen en dos grupos que son:

- Flujos de trabajo principales de ingeniería: Modelado de negocio, requisitos, análisis y diseño, implementación, pruebas y despliegue. (*Rational Software Company*, 2001).
- Flujos de trabajo principales de apoyo: Configuración y gestión de cambios, gestión de proyectos y entorno (*Rational Software Company*, 2001).

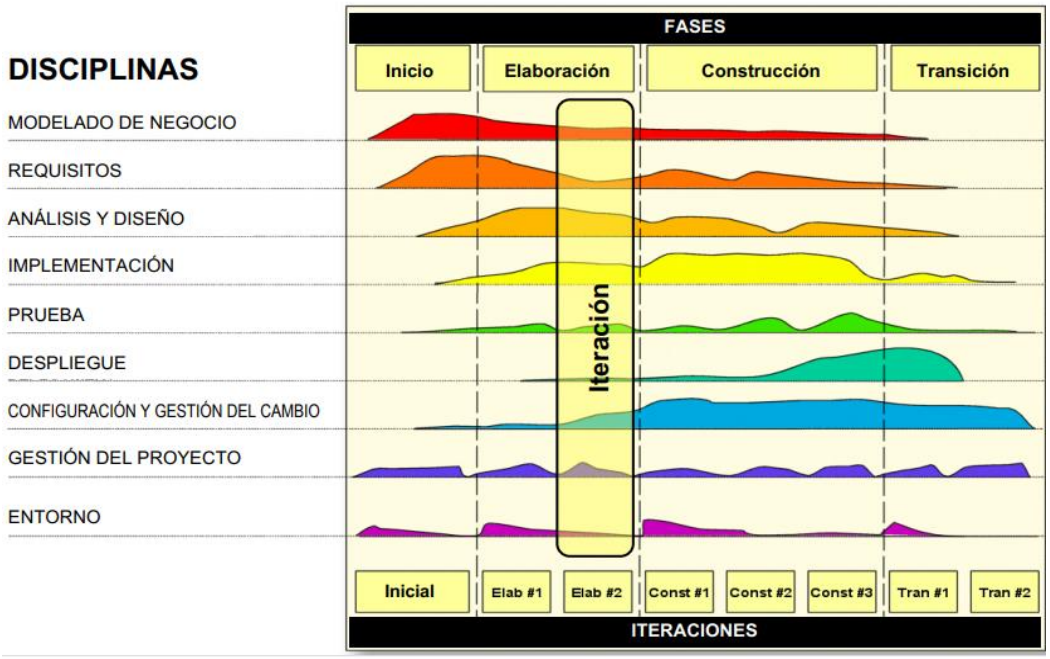


Fig. 1. Ciclo de vida de RUP.

Fuente: (García & García, 2018).

2) Mejores prácticas de RUP

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:

a) Desarrollo Iterativo de Software

Dados los sofisticados sistemas de software actuales, no es posible primero definir secuencialmente todo el problema, diseñar la solución completa, construir el software y luego probar el producto al final. Por lo tanto, se requiere adoptar un enfoque iterativo que permita una mayor comprensión del problema a solucionar. RUP con este enfoque iterativo de desarrollo aborda los elementos de mayor riesgo en cada etapa del ciclo de vida del proyecto, reduciendo significativamente los riegos, atacando cada riesgo a través de avances demostrables y lanzamientos frecuentes y ejecutables que permiten la participación del usuario final haciendo que se facilite el acomodo de cambios tácticos en los requisitos, características o cronogramas (*Rational Software Company*, 2001).

b) Arquitectura Basada en Componentes

RUP se enfoca en el desarrollo temprano y la referencia de una arquitectura ejecutable robusta, antes de comprometer recursos para el desarrollo a gran escala. Describe cómo diseñar una arquitectura resistente que sea flexible, que se acomode el cambio, sea intuitivamente comprensible y promueva una reutilización de software más efectiva. RUP admite el desarrollo de software basado en componentes. Los componentes son módulos no triviales, subsistemas que cumplen una función clara; además, proporciona un enfoque sistemático para definir una arquitectura utilizando componentes nuevos y existentes. Estos se ensamblan en una arquitectura bien definida, ya sea Ad-Hoc, o en una infraestructura de componentes como Internet, CORBA y COM (*Rational Software Company*, 2001).

c) Modelado Visual de Software

RUP muestra cómo modelar visualmente el software para capturar la estructura y el comportamiento de arquitecturas y componentes. Esto le permite ocultar los detalles y escribir código utilizando "bloques de construcción gráficos". Las abstracciones visuales lo ayudan a comunicar diferentes aspectos de su software; ver cómo encajan los elementos del sistema; asegúrese de que los bloques de construcción sean consistentes con su código; mantener la coherencia entre un diseño y su implementación; y promueve una comunicación inequívoca. El lenguaje de modelado unificado (UML) estándar de la industria, creado por *Rational Software*, es la base para un modelado visual exitoso (*Rational Software Company*, 2001).

d) Verificación de la Calidad del Software

El poder asegurar la calidad del software es el punto de fallo más común en los proyectos de desarrollo de software, debido a que anteriormente era un punto en el cual no se había pensado. RUP ayuda a planificar, diseñar, implementar, ejecutar y evaluar estos tipos de pruebas. La evaluación de la calidad no es una tarea que esté dirigida especialmente a la calidad, más bien está integrada en el proceso, en todas las actividades, involucrando a todos los participantes, utilizando medidas objetivas y criterios (*Rational Software Company*, 2001).

e) Gestión de Requisitos de Software

RUP describe cómo obtener, organizar y documentar la funcionalidad y restricciones requeridas; rastrear y documentar las compensaciones y decisiones; y capturar y comunicar fácilmente los requisitos comerciales. Las nociones de casos de uso y escenarios proscritos en el proceso han demostrado ser una excelente manera de capturar los requisitos funcionales y garantizar que estos impulsen el diseño, la implementación y las pruebas de software, lo que hace más probable que el sistema final satisfaga las necesidades del usuario final (*Rational Software Company*, 2001).

f) Control de Cambios de Software

En todos los proyectos de software los cambios son inevitables por lo tanto la capacidad de gestionar los cambios son un punto clave en el desarrollo de software; para esto RUP define métodos para controlar, rastrear y monitorear los cambios para permitir un desarrollo iterativo exitoso. RUP define también una guía sobre cómo establecer espacios de trabajo seguros para cada desarrollador al proporcionar aislamiento de los cambios realizados en otros espacios de trabajo y al controlar los cambios de todos los artefactos de software (por ejemplo, modelos, códigos, documentos, etc.) logrando así que un sistema no se vea afectado por cambios realizados en otro sistema (*Rational Software Company*, 2001).

B. ESSENCE – EL NÚCLEO Y EL LENGUAJE PARA LOS MÉTODOS DE INGENIERÍA DE SOFTWARE

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. Para la conceptualización del estándar es necesario reconocer un conjunto de conceptos que se presentan a continuación.

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. En la figura 2, se muestran los elementos que conforman *Essence*.

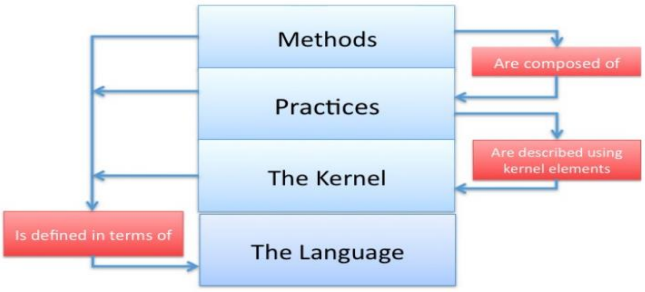


Fig. 2. Elementos de *Essence*.

Fuente: (*Object Management Group*, 2018)

El propósito del núcleo es proporcionar un conjunto de elementos que permitan establecer y definir de mejor manera el desarrollo de software y sus prácticas, incluyendo elementos esenciales que prevalecen en cada esfuerzo de ingeniería de software. El núcleo permite componer prácticas para crear métodos específicos, adaptados a las necesidades particulares de una comunidad, un proyecto, un equipo o una organización (*Object Management Group*, 2018).

El núcleo de *Essence* incluye: (i) las áreas de interés, que permiten organizar los elementos del núcleo; (ii) los alfas, las cosas con las que siempre se trabaja; (iii) los espacios de actividad, las cosas que siempre se hacen; (iv) las competencias, las habilidades necesarias. Estos elementos se definen a continuación:

1) Las áreas de interés

El núcleo se organiza en tres áreas de interés: cliente, solución y esfuerzo. Cada área de interés se enfoca en un aspecto específico de la Ingeniería de Software. En *Essence* cada área de interés y sus elementos se identifican con colores específicos: el área de interés cliente de color verde, el área de interés solución de color amarillo y el área de interés esfuerzo con color azul. El área de interés cliente contiene todo lo relacionado con el uso y la explotación del sistema de software. El área de interés solución contiene todo lo necesario para especificar y desarrollar el sistema de software. El área de interés esfuerzo contiene todo lo relacionado con el equipo y la forma en la que realiza su trabajo. Cada área de interés contiene alfas, espacios de actividad y competencias (*Object Management Group*, 2018). En la Figura 3, se muestran las áreas de interés del núcleo.

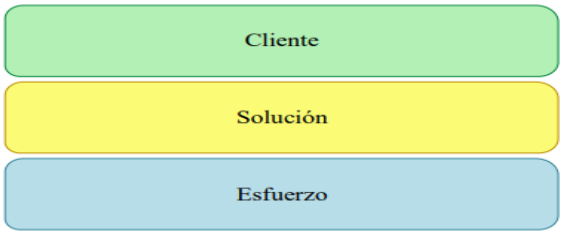


Fig. 3. Áreas de interés del núcleo

Fuente: (*Object Management Group*, 2018)

2) Los alfas

Las alfas se pueden definir como la representación de las cosas esenciales para trabajar. Dichos alfas dimensionan el esfuerzo de Ingeniería de Software que un equipo realiza, a través de un conjunto de estados, donde cada uno de estos contiene una lista de chequeo que permite realizar seguimiento a la salud y al progreso que va teniendo.

En el área de interés cliente se encuentran los siguientes alfas:

a) Alfa Oportunidad

Es el conjunto de circunstancias adecuado para desarrollar o modificar un sistema de software. La oportunidad articula la razón para la creación de lo nuevo o modificado del sistema software (*Object Management Group*, 2018).

b) Alfa Interesados

Las personas, grupos u organizaciones que afecta o se afectan con un sistema de software. Los interesados proporcionan la oportunidad y son fuente de los requisitos y la financiación para el sistema de software (*Object Management Group*, 2018).

El área de interés solución tiene los siguientes alfas:

a) Alfa Requisitos

Lo que el sistema de software debe hacer para tratar la oportunidad y satisfacer los interesados (*Object Management Group*, 2018).

b) Alfa Sistema de Software

Un sistema se compone de software, hardware y los datos que proporciona el valor primario de la ejecución del software (*Object Management Group*, 2018).

En el área de interés esfuerzo se encuentran los siguientes alfas:

a) Alfa Trabajo

Actividad que implica un esfuerzo mental o físico realizado con el fin de lograr un resultado. En el contexto de la Ingeniería de software, el trabajo es todo lo que hace el equipo para cumplir con las metas de producción de un sistema de software que coincida con los requisitos (*Object Management Group*, 2018).

b) Alfa Equipo

Un grupo de personas que participa activamente en el desarrollo, mantenimiento o entrega de un sistema de software (*Object Management Group*, 2018).

c) Alfa Forma de Trabajo

El conjunto adaptado de prácticas y herramientas que utiliza un equipo para orientar y apoyar su trabajo (*Object Management Group*, 2018).

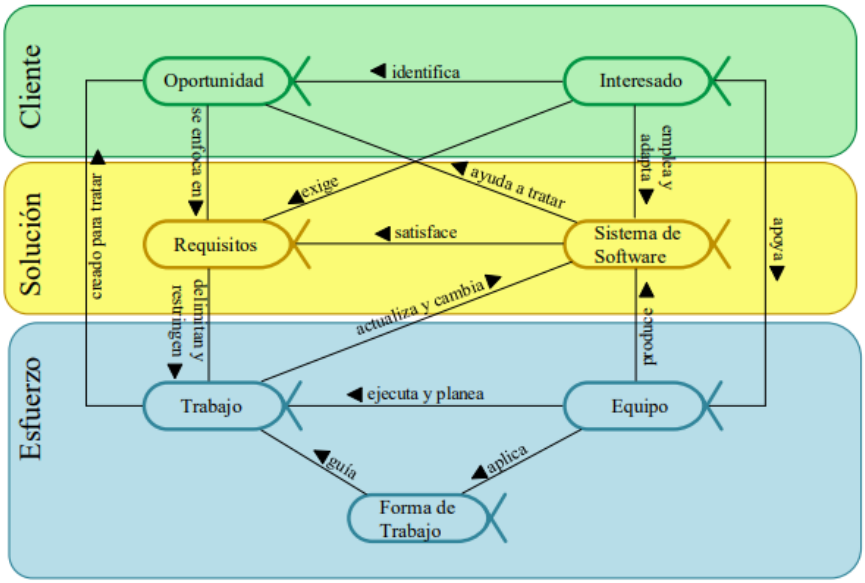


Fig. 4. Alfas del núcleo

Fuente: (*Object Management Group*, 2018)

3) Los espacios de actividad

Los espacios de actividad se definen como las cosas que siempre se hacen y contemplan cada uno de los alfas, estos espacios proveen una visión basada en la actividad de Ingeniería de Software. Se agrupan de acuerdo a las áreas de interés de cliente, solución y esfuerzo. El trabajo de un espacio de actividad se considera completo cuando se cumplen sus criterios de finalización (*Object Management Group*, 2018).

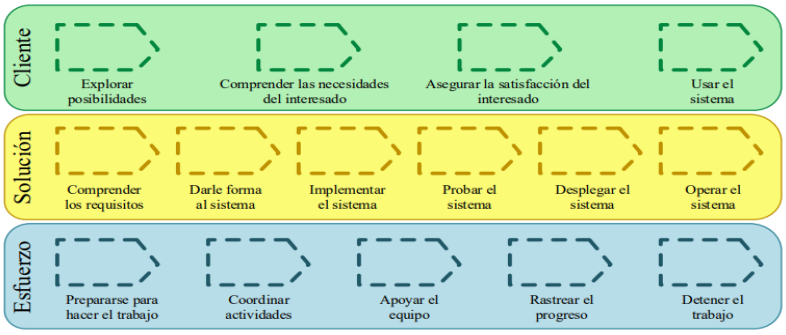


Fig. 5. Espacios de actividad del núcleo

Fuente: (Object Management Group, 2018)

4) Las competencias

Las competencias representan las habilidades clave que se requieren para el desarrollo de software. Por el área de cliente se encuentra la representación del interesado, en el área de solución está el

análisis, el desarrollo y las pruebas y en el área de esfuerzo está el liderazgo y la gestión (*Object Management Group*, 2018).

Fig. 6. Competencias del núcleo

Fuente: (*Object Management Group*, 2018)

C. 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).

1) Los componentes del modelo

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.

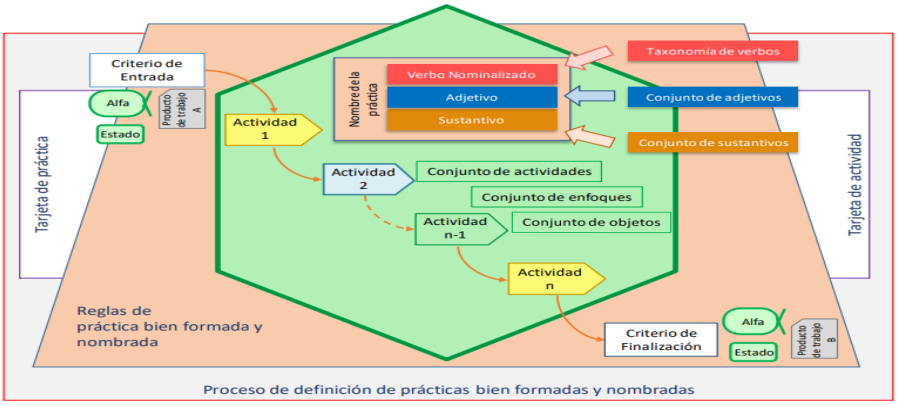


Fig. 7. Componentes del modelo

Fuente: (Barón, 2019)

2) La práctica bien formada en Ingeniería de Software

Una práctica se define como bien formada si el conjunto de actividades cumple con las reglas de coherencia, consistencia y suficiencia.

Los componentes que constituyen una práctica bien formada son los propuestos por Barón (2019) y se describen a continuación:

a) Criterio de entrada de la práctica

Permite determinar las condiciones que se deben cumplir para iniciar la práctica. El criterio de entrada se expresa en términos de sustantivos que pueden ser alfas y sub-alfas y un estado parcial o completo. El producto de trabajo es la evidencia de que un alfa se encuentra total o parcialmente en un estado.

b) Criterio de finalización de la práctica

Permite determinar las condiciones que se deben cumplir para terminar exitosamente la práctica. El criterio de finalización se expresa en términos de un sustantivo que puede ser alfa o sub-alfa y un estado parcial o completo. El producto de trabajo es la evidencia de que un alfa se encuentra total o parcialmente en un estado.

c) Conjunto de actividades

Permite definir el proceso de aplicación de la práctica. Entre actividades existen relaciones de secuencias y de transferencia de recursos. Cada actividad del conjunto de actividades de la práctica se conforma con acciones menores que se denominan tarea. La actividad tiene criterio de entrada y criterio de finalización que determinan las condiciones de inicio y finalización exitosa de la actividad. Los criterios de entrada y finalización de la actividad se expresan en términos de alfa, sub-alfas y estados o productos de trabajo y niveles de detalle. Además, a cada actividad se le define un enfoque que permite conocer la forma en la que se aborda está en términos de un adjetivo.

d) Flujo de actividades

Permite definir el orden de ejecución de las actividades y la transferencia de recursos entre ellas.

e) Reglas de la práctica bien formada

Son reglas de coherencia, consistencia y suficiencia que permiten describir las relaciones que se deben presentar entre la práctica y las actividades, a fin de definir prácticas bien formadas.

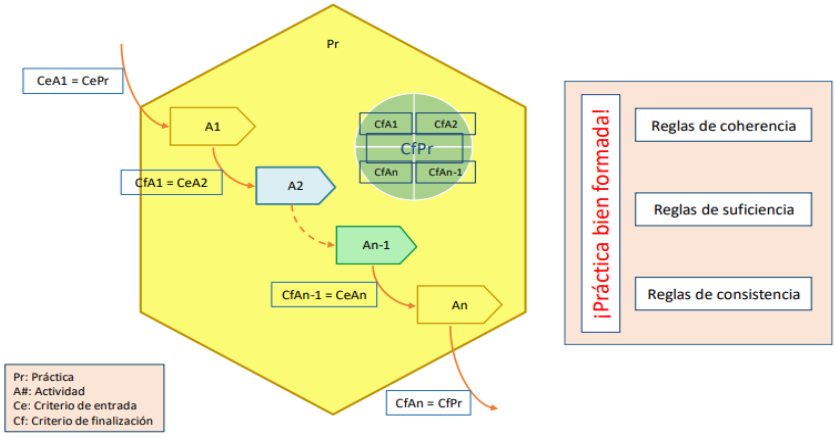


Fig. 8. Reglas de práctica bien formada

Fuente: (Barón, 2019)

f) Regla de coherencia

Un conjunto de actividades de una práctica es coherente si el cumplimiento del criterio de finalización de cada actividad aporta en el progreso del sustantivo hacia el criterio de finalización de la práctica (Barón, 2019).

g) Reglas de consistencia

Un conjunto de actividades de una práctica es consistente si:

R1. Existe al menos una actividad cuyo criterio de entrada es igual al criterio de entrada de la práctica.

R2. Existe al menos una actividad cuyo criterio de finalización es igual al criterio de finalización de la práctica.

R3. Para cada actividad del conjunto de actividades de la práctica, su criterio de entrada es igual al criterio de finalización de al menos otra actividad o al criterio de entrada de la práctica.

R4. Para cada actividad del conjunto de actividades de la práctica, su criterio de finalización es igual al criterio de entrada de al menos otra actividad o al criterio de finalización de la práctica.

Las reglas R3 y R4 son válidas excluyendo las actividades que cumplen las reglas R1 y R2 (Barón, 2019).

h) Regla de suficiencia

Un conjunto de actividades de una práctica es suficiente si 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 (Barón, 2019).

3) La práctica bien nombrada en Ingeniería de Software

Un nombre estructurado y adecuado de práctica permite determinar sus elementos esenciales. A partir del nombre se puede identificar inequívocamente prácticas (Barón, 2019).

a) Modelo general para el nombramiento de prácticas

Una práctica es bien nombrada si su nombre incluye: un verbo nominalizado, que indica lo que se hace con la práctica, un adjetivo que indica cómo se hace y un sustantivo que indica el objeto sobre el cual se aplica la práctica. Para definir los elementos que conforman el nombre de la práctica, en el modelo se propone una taxonomía de verbos nominalizados, un conjunto de adjetivos y un conjunto de sustantivos.

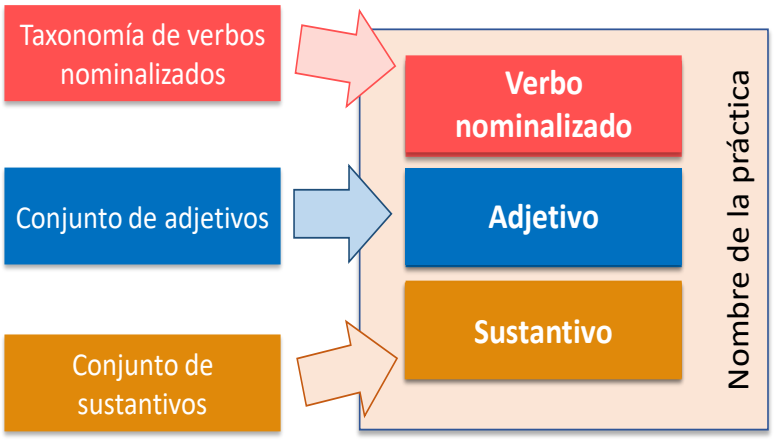


Fig. 9. Modelo general para el nombramiento de prácticas

Fuente: (Barón, 2019)

b) Los componentes que constituyen una práctica bien nombrada

Los componentes que constituyen una práctica bien nombrada son los propuestos por Barón (2019) y se describen a continuación:

Nombre: El nombre de una práctica bien nombrada debe estar formado por un verbo nominalizado, un sustantivo y uno o varios adjetivos.

Verbo nominalizado: Es el verbo 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.

Adjetivo: Es un adjetivo que integra los enfoques que se utilizan para realizar cada una de las actividades de la práctica.

Sustantivo: Es 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.

c) Taxonomía de verbos nominalizados

TABLA I

VERBOS NOMINALIZADOS

Taxonomia de verbos nominalizados								
Analisis	Recoleccion	Identification	Negociacion	Especificacion	Valoracion	Elicitacion	Educcion	Priorizacion
Modelado	Diseno	Definicion						
Implementacion	Construccion	Integracion	Mejoramiento	Desarrollo				
Prueba	Construccion	Realizacion	Identification	Resolucion	Mejoramiento	Evaluacion		
Despliegue	Liberacion	Habilitacion						
Mantenimiento	Operacion	Monitoreo	Apoyo					
Verificación	Aprobacion	Aceptacion						
Validacion	Aprobacion	Aceptacion	Entrenamiento					
Planificacion	Estimacion	Organizacion	Definicion	Financiacion				
Conformación	Seleccion	Contratacion	Entrenamiento					
Cooperacion	Certificacion	Mejoramiento	Comunicacion	Guia	Entrenamiento			
Coordinacion	Control	Aseguramiento	Inspeccion	Monitoreo	Seguimiento	Revision		
Cierre	Suspension	Retiro	Entrega					

Fuente: (Barón, 2019)

d) Conjunto de adjetivos

TABLA II

ADJETIVOS

Conjunto consolidado de adjetivos			
Ágil	Colaborativo	Evolutivo	Probado
Automatizado	Compartido	Formal	Progresivo
Auto-organizado	Concurrente	Funcional	Prospectivo
Basado en actividades	Conjunto	Global	Repetible
Basado en casos de uso	Continuo	Incremental	Reutilizable
Basado en componentes	Cualitativo	Integrado	Sistemático
Basado en escenarios	Cuantitativo	Interdisciplinario	Sistémico
Basado en riesgos	Dirigido por pruebas	Iterativo	Temprano
Basado en valor	Disciplinario	Metódico	Valor compartido
Bien organizado	Empírico	Orientado al producto	Visual
Casual	Escalonado	Orientado por objetos	
Arquitectura	Estructurado	Personalizado	

Fuente: (Barón, 2019)

e) Conjunto de sustantivos

TABLA III

SUSTANTIVOS

Conjunto de sustantivos	
Alfas	**Sub- Alfas**
Interesados	Representante de los interesados
Oportunidad	Necesidad
Requisitos	Ítem de requisito
Sistema de software	Elemento de sistema de software
Equipo	Miembro del equipo

Fuente: (Barón, 2019)

f) Reglas de práctica bien nombrada

Las reglas que rigen la manera de definir los elementos que constituyen el nombre de una práctica son las siguientes:

R1. El verbo nominalizado que se utiliza para nombrar la práctica es el 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. Responde a la pregunta: ¿Cuál es el verbo nominalizado que resume el conjunto de actividades mediante el cual el sustantivo alcanza el estado que se indica en el criterio de finalización de la práctica? (Barón, 2019).

R2. El adjetivo que se utiliza para nombrar la práctica es el adjetivo que integra los enfoques que se utilizan para realizar cada una de las actividades de la práctica. Responde a la pregunta: ¿Cuál es el adjetivo que resume la manera cómo se realiza el conjunto de actividades para llevar el sustantivo al estado que se indica en el criterio de finalización de la práctica? (Barón, 2019).

R3. El sustantivo que se utiliza para nombrar la práctica es el sustantivo que se indica en el criterio de finalización de la práctica. En el caso de no tener el criterio de finalización definido, el sustantivo que se utiliza para nombrar la práctica es el sustantivo que tiene mayor progreso como resultado de realizar la práctica. Responde a la pregunta: ¿Cuál es el sustantivo que, asociado con un estado, permite definir la realización exitosa de la práctica? (Barón, 2019).

4) Tarjetas de práctica y actividad

La tarjeta de práctica permite visualizar los elementos esenciales que describen la práctica de Software. De la misma manera, la tarjeta de actividad permite visualizar los elementos esenciales que describen la actividad. Se utilizan para facilitar el seguimiento a la creación o conversión de prácticas bien formadas y nombradas (Barón, 2019).

	**Práctica de ingeniería de software** **Nombre:** [Verbo nominalizado] [Adjetivo][Sustantivo]
Descripción: [Breve descripción de la práctica]	
Criterio de entrada	
[Parcialmente en:] [(Sustantivo: Estado)]	
Productos de trabajo asociados con el criterio de entrada **1.** [Listar los productos de trabajo que son evidencias del cumplimiento del criterio de entrada] **2.** …	
Criterio de finalización	
[Contribuye con:] [(Sustantivo: Estado)]	
Productos de trabajo asociados con el criterio de finalización **1.** [Listar los productos de trabajo que son evidencias del cumplimiento del criterio de finalización] **2.** …	

Fig. 10. Tarjeta de práctica

Fuente: (Barón, 2019)

	Actividad [#]: [Nombre de la actividad]
	Enfoque: [enfoque de la actividad]
	Espacio de actividad: [Espacio de actividad que agrupa la actividad]
Descripción: [Breve descripción de la actividad]	
Criterio de entrada	
[(Sustantivo: Estado) o (Producto de trabajo: Nivel de detalle)]	
Productos de trabajo asociados con el criterio de entrada **1.** [Listar los productos de trabajo que son evidencias del cumplimiento del criterio de entrada] **2.** …	
Tareas	
1. [Listar tareas de la actividad] **2.** …	
Criterio de finalización	
[(Sustantivo: Estado) o (Producto de trabajo: Nivel de detalle)]	
Productos de trabajo asociados con el criterio de finalización **1.** [Listar los productos de trabajo que son evidencias del cumplimiento del criterio de finalización] **2.** …	

Fig. 11. Tarjeta de actividad

Fuente: (Barón, 2019)

5) Proceso de definición de prácticas bien formadas y nombradas

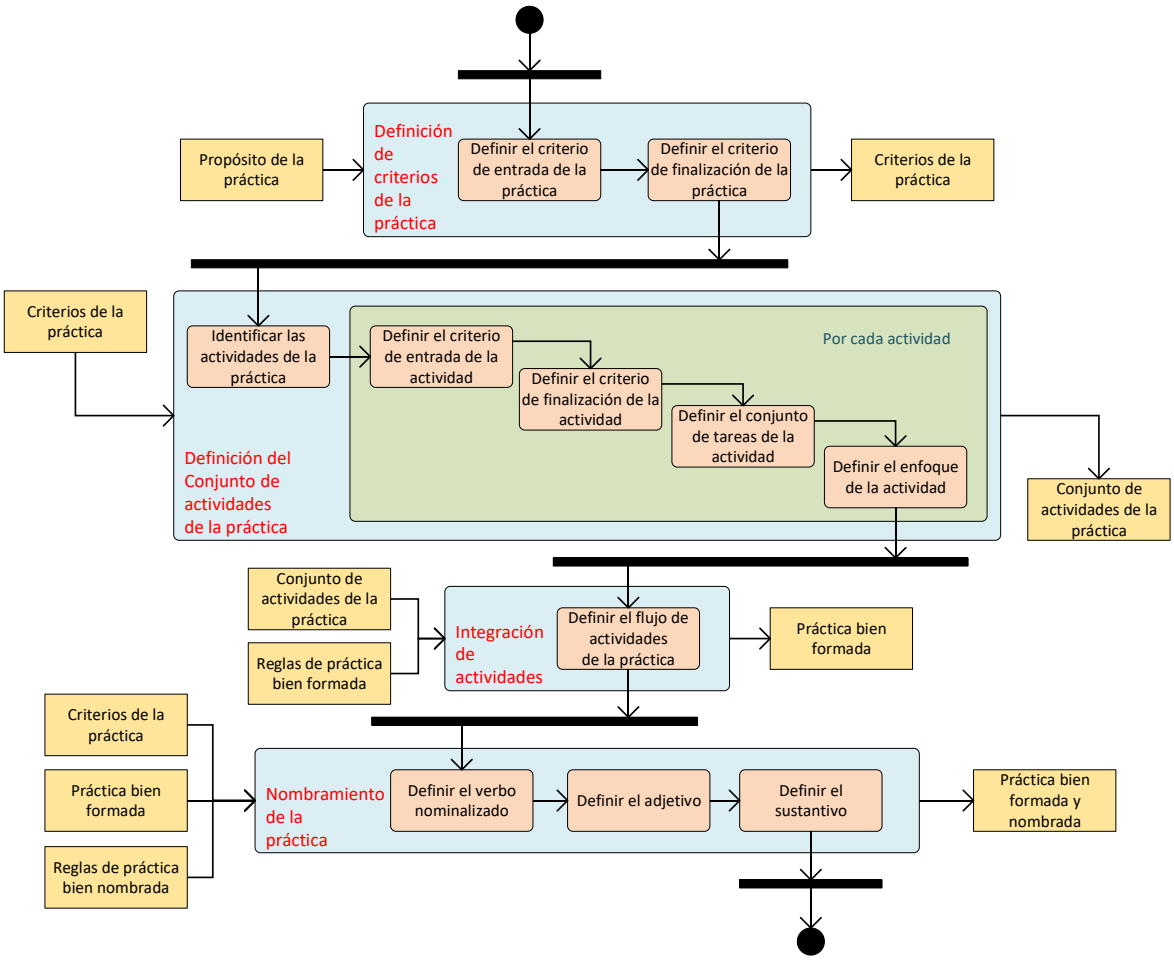


Fig. 12. Proceso de definición de prácticas bien formadas y nombradas

Fuente: (Barón, 2019)

III. METODOLOGIA

A. TIPO DE INVESTIGACIÓN

En este trabajo de grado se aplica el Modelo para la Definición de Prácticas en Ingeniería de Software que se propone en (Barón, 2019). El modelo se aplica a la práctica CCS-RUP. Con el desarrollo de este trabajo de grado se busca identificar y definir los componentes de la práctica según el modelo. Este proceso se denomina esencialización. La esencialización de una práctica, facilita entender, aplicar y evaluar la práctica. La práctica CCS-RUP esencializada se valida mediante su aplicación en un estudio de caso que permite simular un contexto real.

B. DESARROLLO DE LA PROPUESTA

1) 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). En la figura 13 se muestra la estrategia metodológica para realizar síntesis conceptuales de (Barón, 2019). En la figura 14 se muestra el proceso de RSL, que se emplea en este trabajo, como una adaptación de la propuesta de (Barón, 2019).

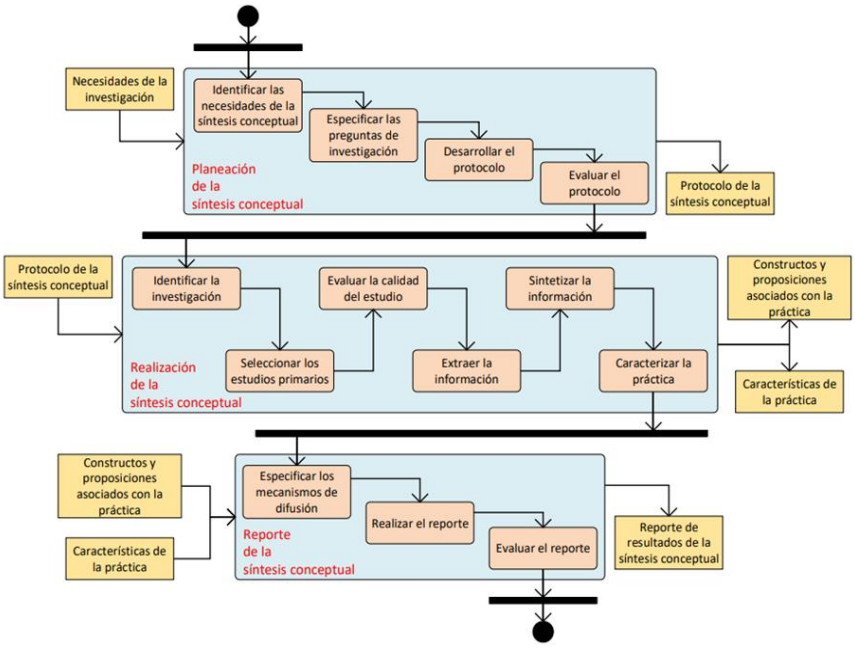


Fig. 13. Estrategia metodológica para elaborar síntesis conceptuales en ingeniería de software

Fuente: (Barón, 2019)

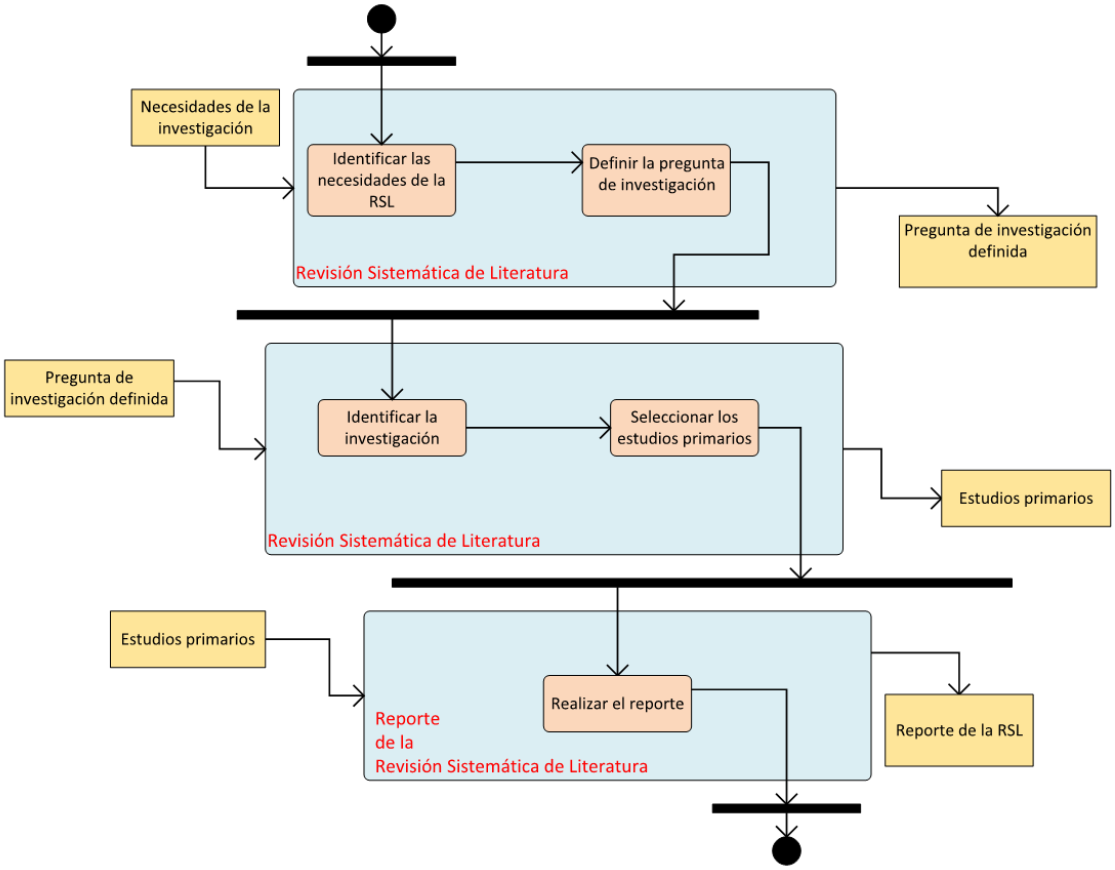


Fig. 14. Proceso de RSL

Fuente: Elaboración propia adaptada de (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. 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, 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.

a) Fase 1: Planeación de la RSL

En la fase de planeación se realizan 2 actividades que permiten identificar todos los elementos necesarios para llevar a cabo la realización de la RSL. Las entradas para esta fase son las necesidades de la investigación. La salida es la pregunta de investigación definida.

Actividad 1.1: Identificar la necesidad de la RSL

La comunidad de ingeniería de software define la práctica CCS-RUP de diferentes maneras, por tanto, se necesita resumir y sintetizar las definiciones que la comunidad de ingeniería de software ofrece. Por lo tanto, para esta investigación se identifica la siguiente necesidad: Identificar la manera como la comunidad de ingeniería de software define la práctica CCS-RUP.

Actividad 1.2: Definir la pregunta de la investigación

Para determinar cuáles son los estudios relevantes, se formula una pregunta de 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?

b) Fase 2: Realización de la RSL

Para llevar a cabo la realización de la RSL se realizan las siguientes actividades:

Actividad 2.1: Identificar la investigación

Para identificar la investigación, con base en la pregunta de investigación, se definen las cadenas de búsqueda. Estas cadenas se utilizan como criterios de búsqueda en las fuentes de estudios para obtener los estudios primarios. En la tabla 4 se encuentra la definición de las cadenas de búsqueda.

TABLA IV

CADENAS DE BÚSQUEDA

Cadena 1	"RUP"
Cadena 2	"Práctica control de cambios de RUP" o "*RUP control of changes practice"*

Fuente: Elaboración propia

Actividad 2.2: Seleccionar los estudios primarios

Para llevar a cabo la selección de estudios primarios, se realizan las siguientes tareas:

Tarea 2.2.1: Seleccionar bases de datos digitales

Inicialmente se seleccionan las bases de datos digitales donde se usan las cadenas de búsqueda definidas anteriormente. Las bases de datos digitales seleccionadas para esta investigación son las siguientes: EBSCO, ACM digital library, Google Scholar, Microsoft Academic.

Tarea 2.2.2: Aplicar cadenas de búsqueda

Al aplicar las cadenas de búsqueda en las bases de datos digitales se obtiene un universo de posibles estudios relevantes para la investigación. En la tabla 5 se muestra la tabla con el número de estudios primarios encontrados en cada fuente y para cada cadena, además de estudios aportados por expertos.

TABLA V

ESTUDIOS PRIMARIOS

Fuente	Resultados encontrados		
	Cadena 1	Cadena 2	Total
EBSCO	27	55	82
ACM digital library	35	61	96
Google Scholar	32	47	79
Microsoft Academic	28	34	62
Expertos		3	
Total			**322**

Fuente: Elaboración propia

Tarea 2.2.3: Definir criterios de inclusión y exclusión

Luego, para identificar los estudios realmente relevantes para la investigación, se definen unos criterios de inclusión y exclusión. En la tabla 6 a continuación se definen los criterios de inclusión y exclusión para esta RSL.

TABLA VI

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 traten a RUP como una metodología ágil
▪ Estudios que presenten la metodología RUP con descripciones cortas o insuficientes

Fuente: Elaboración propia

En este punto del proceso de la RSL, se aplican estos criterios de inclusión y exclusión al total de los estudios primarios encontrados, para así obtener los estudios relevantes para esta investigación. La aplicación de criterios de inclusión y exclusión se realiza en 3 vueltas aplicando el método de marcación de colores denominado semáforo de la siguiente manera: Se marcan de color verde los estudios que cumplen con los criterios de inclusión, en amarillo aquellos que cumplen parcialmente con los criterios de inclusión, estos pasarán a la siguiente vuelta para una evaluación más exhaustiva. Se marcan de color rojo los estudios que se descartan de la investigación. En la primera vuelta, el análisis se enfoca en el título y el resumen de los estudios. En la segunda vuelta, el análisis es más riguroso y se concentra en la introducción. En la tercer y última vuelta, el análisis se hace sobre el cuerpo del estudio.

Paso 1: Primera vuelta

En la tabla 7 se muestran los estudios seleccionados a partir del análisis de los 322, trabajos que son el universo de estudios para la investigación. Este análisis se concentra en el título y resumen. Por extensión se omite incluir la tabla de los 322 estudios completa, se presenta únicamente los estudios que se marcan con color amarillo.

TABLA VII

PRIMERA VUELTA

Resultado de la primera vuelta	Cita
Representación en el Núcleo de SEMAT de Prácticas de Métodos de Desarrollo Basados en Planes	(Jiménez, 2016)
RUP: Disciplina de Manejo de Cambios y Configuraciones	(Soriano A, 2005)
Rational Unified Process Best Practices for Software Development Teams	*(Rational Software Company*, 2001)
Ciclo Vital para Proyectos Pequeños de RUP	(IBM *Corporation*, 2006)
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)
Aplicación del Proceso Unificado de Desarrollo a proyectos de software	(Hernández, 2004)
A Reduced Set of RUP Roles to Small Software Development Teams	(Monteiro *et al.*, 2012)

Resultado de la primera vuelta	Cita
Software Engineering: Methods, modeling and teaching	(Zapata & Castro, 2014)
Estructura básica del proceso unificado de desarrollo de software	(Castro, 2004)
Definición de buenas prácticas de desarrollo de sistemas de información geográfica utilizando el núcleo de Semat	(Durango, 2019)
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)
Métodos de desarrollo de software: El desafío pendiente de la estandarización	(Gacitúa, 2003)
Ingeniería de software orientada a agentes: Roles y metodologías	(Moreno *et al.*, 2006)
Personalización de RUP para proyectos académicos de desarrollo de software	(Tabares, 2011)
Improvement in RUP Project Management via Service Monitoring: Best Practice of SOA	(Saqib *et al.*, 2010)
Adaptación de RUP para PSP: experiencia en el PIS	(Peréz & Vallespir, 2009)
The Rational Unified Process Made Easy - A Practitioner's Guide to the RUP	(Kroll & Kruchten, 2003)

Fuente: Elaboración propia

Paso 2: Segunda vuelta

En la tabla 8 se muestran los estudios resultantes de un análisis enfocado en la introducción.

TABLA VIII

SEGUNDA VUELTA

Resultado de la segunda vuelta	Cita
Representación en el Núcleo de SEMAT de Prácticas de Métodos de Desarrollo Basados en Planes	(Jiménez, 2016)
RUP: Disciplina de Manejo de Cambios y Configuraciones	(Soriano A, 2005)
Rational Unified Process Best Practices for Software Development Teams	*(Rational Software Company*, 2001)
Ciclo Vital para Proyectos Pequeños de RUP	(IBM *Corporation*, 2006)
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)
Aplicación del Proceso Unificado de Desarrollo a proyectos de software	(Hernández, 2004)
A Reduced Set of RUP Roles to Small Software Development Teams	(Monteiro *et al.*, 2012)
Software Engineering: Methods, modeling and teaching	(Zapata & Castro, 2014)
Estructura básica del proceso unificado de desarrollo de software	(Castro, 2004)
Definición de buenas prácticas de desarrollo de sistemas de información geográfica utilizando el núcleo de Semat.	(Durango, 2019)
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)
Métodos de desarrollo de software: El desafío pendiente de la estandarización	(Gacitúa, 2003)

Resultado de la segunda vuelta	Cita
Ingeniería de software orientada a agentes: Roles y metodologías	(Moreno *et al.*, 2006)
Personalización de RUP para proyectos académicos de desarrollo de software	(Tabares, 2011)
Improvement in RUP Project Management via Service Monitoring: Best Practice of SOA	(Saqib *et al.*, 2010)
Adaptación de RUP para PSP: experiencia en el PIS	(Peréz & Vallespir, 2009)
The Rational Unified Process Made Easy - A Practitioner's Guide to the RUP	*(*Kroll & Kruchten, 2003)

Fuente: Elaboración propia

Paso 3: Tercera vuelta

En la tabla 9 se muestran los estudios resultantes de un análisis enfocado en el cuerpo de los estudios.

TABLA IX

TERCERA VUELTA

Resultado de la tercera vuelta	**Cita**
Representación en el Núcleo de SEMAT de Prácticas de Métodos de Desarrollo Basados en Planes	(Jiménez, 2016)
RUP: Disciplina de Manejo de Cambios y Configuraciones	(Soriano A, 2005)
Rational Unified Process Best Practices for Software Development Teams	*(Rational Software Company*, 2001)
Aplicación del Proceso Unificado de Desarrollo a proyectos de software	(Hernández, 2004)
Definición de buenas prácticas de desarrollo de sistemas de información geográfica utilizando el núcleo de Semat.	(Durango, 2019)

Resultado de la tercera Vuelta	*Cita*
A Reduced Set of RUP Roles to Small Software Development Teams	(Monteiro *et al.*, 2012)
Adaptación de RUP para PSP: experiencia en el PIS	(Pérez-Queiruga & Vallespir, 2009)
The Rational Unified Process Made Easy - A Practitioner's Guide to the RUP	*(*Kroll & Kruchten, 2003)

Fuente: Elaboración propia

c) Fase 3: Reporte de la RSL

Actividad 3.1: Realizar el reporte

En la tabla 10 se muestran los estudios realmente relevantes para la investigación, estos estudios son usados para las siguientes actividades de la RSL, además, del análisis de como los diferentes autores definen los elementos constitutivos de la práctica CCS-RUP.

TABLA X

ESTUDIOS REALMENTE 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

2) Análisis sobre la definición de la práctica CCS-RUP

En este capítulo, se realiza un análisis de la manera cómo varios autores definen la práctica CCS-RUP, a partir de los estudios realmente relevantes que se obtuvieron por medio de la RSL. Este análisis permite identificar en cada estudio los elementos de la práctica que se utilizan para describirla utilizando como criterio de análisis el Modelo para la Definición de Prácticas en Ingeniería de Software (Barón,2019).

El proceso de 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.

a) Análisis de los elementos constitutivos de la práctica

En esta 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. Cada actividad del conjunto de actividades de la práctica se conforma con acciones menores que se denominan tarea. La actividad tiene criterio de entrada y criterio de finalización que determinan las condiciones de inicio y finalización exitosa de la actividad. Además, a cada actividad se le define un enfoque que permite conocer la forma en la que se aborda está en términos de un adjetivo.

Flujo de actividades que permite definir el orden de ejecución de las actividades y la transferencia de recursos entre ellas.

En la tabla 11 se presentan los componentes, que se utilizan en cada estudio para definir la práctica CCS-RUP. Los elementos que son criterio de análisis corresponden a los componentes de una práctica bien formada y nombrada según Barón (2019). 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”. En seguida, se describe de manera detallada la manera como se llega a este resultado en cada estudio.

TABLA XI

ANÁLISIS A LOS 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

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
(Jaramillo, 2016)	NO	NO	NO	NO	NO
(Pozo, 2015)	NO	NO	SI	NO	NO
(Cerrón, 2017)	NO	NO	SI	NO	NO

Fuente: elaboración propia

En (Jiménez, 2016), no se puede identificar un criterio de entrada para establecer las condiciones en que debe iniciar la práctica, si bien, define en lenguaje *Essence* los alfas necesarios para iniciar la práctica o actividades, no se especifica el estado en que se encuentra este alfa. Tampoco se logra identificar los criterios de finalización para la práctica, dado que, al igual que en el caso de los criterios de entrada, se indican los alfas, pero no sus estados, siendo estos necesarios para saber cómo inicia y termina la práctica. Presenta una tabla donde se identifica el conjunto de actividades en cada etapa, sin embargo, estas actividades no constan de un criterio de entrada o finalización que permita entender cuando una actividad inicia o termina exitosamente. A pesar de definir una tabla de actividades, no es suficiente para identificar los recursos que se transmiten entre ellas, dificultando evaluar el flujo de las actividades.

(Rational Software Company, 2001) es una definición propuesta de manera textual, no permite identificar los criterios de entrada o finalización, tampoco estados que establecen las condiciones para iniciar o finalizar la práctica CCS-RUP. No se puede identificar cuáles son las actividades

propias de la práctica, o el flujo de actividades que constituyen la práctica, esta definición carece de los elementos que componen una práctica.

En (Kruchten, 2004), no se identifican los criterios de entrada y tampoco se identifican los criterios de finalización dado que, en ningún lugar hace referencia al estado que debe tener la práctica para iniciar o finalizar. Esta definición identifica algunas actividades de la práctica de manera textual según el desarrollo de la metodología RUP, lo que permite identificar el conjunto de actividades de la práctica CCS-RUP, sin embargo, las actividades identificadas no definen un criterio de entrada ni de finalización, este conjunto de carencias hace imposible identificar el flujo de actividades.

En (Jaramillo, 2016), no se identifica con claridad los criterios de entrada o salida de la práctica, puesto que, no se definen los estados con el que inicia y termina la práctica, por tanto, no se puede establecer las condiciones en que debe iniciar y finalizar la práctica. Esta definición tampoco identifica un flujo de actividades, si bien se puede identificar algunas actividades de la práctica CCS-RUP, estas no generan productos de trabajo que actúen como insumo para la inicialización de una actividad siguiente o permitan definir la finalización exitosa de la práctica. Lo que demuestra las carencias significativas de los componentes de una práctica.

En (Pozo, 2015), se encuentra una definición dada por una tabla secuenciada, sin embargo, no permite identificar los criterios de entrada y finalización o los estados que estable las condiciones para iniciar y finalizar la práctica. La definición mediante tabla permite identificar un conjunto de actividades, a pesar de que cada actividad cuenta de un numeral, no se definen los criterios de entrada ni finalización de las actividades, en consecuencia, no permite identificar si un producto de trabajo es un insumo para iniciar una actividad posterior o es un criterio de finalización de la práctica. Estas carencias impiden obtener un flujo de actividades y presenta carencia de los componentes de una práctica.

En (Cerron, 2017), se encuentra un ejemplo de la aplicación de la práctica CCS-RUP en un proyecto real. A pesar de definir una tabla donde se encuentra un conjunto de actividades enumeradas, esta definición no nos permite identificar con claridad cuál es un criterio de entrada para la inicialización de la práctica y tampoco permite identificar cual es un criterio de finalización para la práctica. Aunque define estados para las actividades, no es posible identificar si estos estados corresponden a un criterio de entrada o finalización. El conjunto de actividades está definido, sin embargo, en estas actividades no es posible identificar la transferencia de recursos que indique el progreso de la práctica. Estas carencias evidencian la carencia de los componentes de una práctica bien formada.

b) Análisis de la práctica bien formada y bien 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 este punto de la investigación se analizan los estudios relevantes para determinar si la forma como describen la práctica, constituye una práctica bien formada y nombrada. El análisis consiste en determinar: (i) si el conjunto de actividades cumple con las reglas de coherencia (RCH), consistencia (RCS) y suficiencia (RSF); (ii) si el nombre de la práctica se constituye de los componentes que definen una práctica bien nombrada, que son, el verbo nominalizado (VN), el adjetivo (A) y el sustantivo (S).

En la tabla 12 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 XII

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

Estudio	Práctica bien formada			Práctica bien nombrada		
	RCH	RCS	RSF	VN	A	S
(Pozo, 2015)	NO	NO	NO	SI	NO	SI
(Cerron, 2017)	NO	NO	NO	SI	NO	SI

Fuente: Elaboración propia

En (Jiménez, 2016), se presenta una tabla donde se identifica el conjunto de actividades en cada etapa, sin embargo, estas actividades no constan de un criterio de entrada o finalización que permita entender cuando una actividad inicia o termina exitosamente. Por lo tanto, no cumple con ninguna de las reglas de práctica bien formada, puesto que, la carencia de estos criterios de entrada y finalización dificulta seguir el progreso del sustantivo a través de la aplicación de la práctica CCS-RUP. Se identifica un verbo nominalizado "Control" y el sustantivo "Cambios", sin embargo, no se identifica un adjetivo que integre los enfoques que se utilizan para realizar cada una de las actividades de la práctica.

(Rational Software Company, 2001) no permite identificar los criterios de entrada o finalización, tampoco estados que establecen las condiciones para iniciar o finalizar las actividades de la práctica CCS-RUP. Esto hace que esta definición no cumpla con las reglas de práctica bien formada y nombrada, dado que es imposible evaluar el progreso del sustantivo a través de la aplicación de la práctica CCS-RUP. Se encuentra el verbo nominalizado "Control" y el sustantivo "Cambios" pero no se encuentra el adjetivo.

Dado que en (Kruchten, 2004) las actividades no definen los criterios de entrada o de finalización, se puede decir que no cumple con ninguna de las reglas de práctica bien formada. Tampoco presenta las reglas de práctica bien nombrada en su totalidad, puesto que, a pesar de definir un verbo nominalizado "Control" y un sustantivo "Cambios", no define un adjetivo que indique la forma como se realiza la práctica.

En (Jaramillo, 2016) no se puede identificar el flujo de actividades, y las actividades que define no generan productos de trabajo que actúen como insumo para la inicialización de una actividad siguiente. Esto hace que no cumpla las reglas de práctica bien formada, puesto que, la falta de estos elementos hace imposible evaluar el progreso del sustantivo al aplicar la práctica. Define el verbo nominalizado "Control" y el sustantivo "Cambios", pero no se define el adjetivo.

En (Pozo, 2015) no se logra identificar criterios de entrada o finalización en las actividades que define, por lo tanto, no cumple con ninguna de las reglas de práctica bien formada. Tampoco cumple, en su totalidad, con las reglas de práctica bien nombrada. Si bien define un verbo nominalizado y un sustantivo, carece de un adjetivo.

Puesto que en (Cerron, 2017) no se puede identificar los criterios de entrada o finalización de las actividades, se puede afirmar que no cumple con ninguna de las reglas de práctica bien formada. Tampoco cumple con las reglas de práctica bien nombrada, puesto que carece de un adjetivo para integrar los enfoques que se pueden usar en las actividades de la práctica.

3) Conclusiones del análisis

Los resultados del análisis sobre la definición de la práctica CCS-RUP, permiten concluir lo siguiente:

a) Conclusión sobre la identificación de los elementos constitutivos de la práctica

Como se puede observar, los estudios primarios objeto de este análisis no presentan una definición de la práctica CCS-RUP donde se pueda identificar, en su totalidad, los elementos que propone (Barón, 2019) como constitutivos de una práctica. Por ejemplo, en (Jiménez, 2016) si bien se definen objetos tales como las actividades, estas no definen su debido criterio de entrada y finalización. Se evidencia este tipo de carencias en los estudios objeto de análisis. Estas carencias dificultan entender, aplicar y evaluar la práctica CCS-RUP en contextos reales.

b) Conclusión sobre la práctica bien formada

Como se puede observar, no existe, en los estudios que hacen parte de la investigación, una definición de la práctica CCS-RUP que se pueda calificar como práctica bien formada. La carencia de los elementos constitutivos, tales como los criterios de entrada y finalización, tanto de la práctica como de las actividades, dificulta evaluar el progreso del sustantivo conforme se ejecuta la práctica. Esto dificulta entender, aplicar y evaluar la práctica CCS-RUP en contextos reales.

c) Conclusión sobre la práctica bien nombrada

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 nombrada. Todas estas definiciones heredan el nombre definido por RUP, este nombre consta de un verbo nominalizado “Control” y de un sustantivo “Software”, pero como se menciona anteriormente, no define un adjetivo que integre los enfoques que se utilizan para realizar cada una de las actividades de la práctica.

d) Conversión de la práctica CCS-RUP en una práctica bien formada y nombrada

Como se puede observar, la comunidad de la ingeniería de software define la práctica CCS-RUP de diferentes maneras. En general, en estas formas de definición de la práctica 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. Por lo tanto, es necesario convertir la práctica CCS-RUP en un a práctica bien formada y nombrada.

En este capítulo se realiza la conversión de la práctica CCS-RUP en una práctica bien formada y nombrada. Para lo cual, 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, a pesar de carecer de elementos que permitan identificar un estado en sus actividades, 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. Así, se obtiene una práctica bien formada y nombrada que sea fácil de entender, aplicar y evaluar en contextos reales.

e) Presentación de la práctica CCS-RUP

La práctica CCS-RUP permite conservar la integridad del producto software, facilitar la comunicación clara para las solicitudes de cambio, economizando así, costos, esfuerzos de trabajo y tiempo. Por lo tanto, la práctica permite (i) definir el flujo de trabajo de los cambios; (ii) la comunicación clara de los cambios; (iii) proporcionar un entorno estable para desarrollar el producto; (iv) evaluar el estado del proyecto; (v) asegurar la satisfacción del cliente.

Como se trata de la conversión de prácticas que existen de forma tácita o expresadas desde la particularidad de un enfoque de ingeniería de software, el metodólogo define los elementos que caracterizan la práctica a partir de la información que la fuente aporta (Barón, 2019).

Como fuente para la conversión de la práctica CCS-RUP, se usa la definición obtenida de Jiménez (2016) en su trabajo Representación en el Núcleo de SEMAT de Prácticas de Métodos de Desarrollo Basados en Planes, donde propone una definicion de RUP bajo el nucleo de SEMAT. Se define una tabla donde se encuentran las tareas realizadas en cada etapa, además se usa una representación gráfica que permite identificar los productos de trabajo de dichas actividades acompañado del rol del encargado de esta actividad. Como esta definición está bajo el marco de *Essence*, se facilita el desarrollo de esta conversión. En las siguientes figuras, se muestra la forma como Jiménez (2016) define la práctica CCS-RUP.

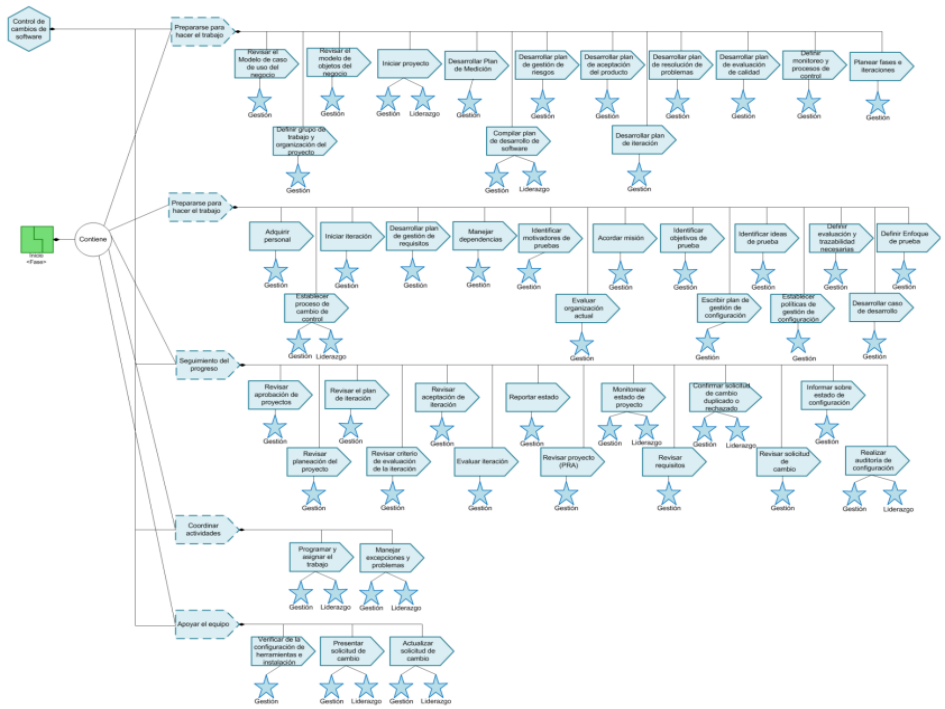


Fig. 15. Control de Cambios de Software de RUP (1)

Fuente: (Jiménez, 2016)

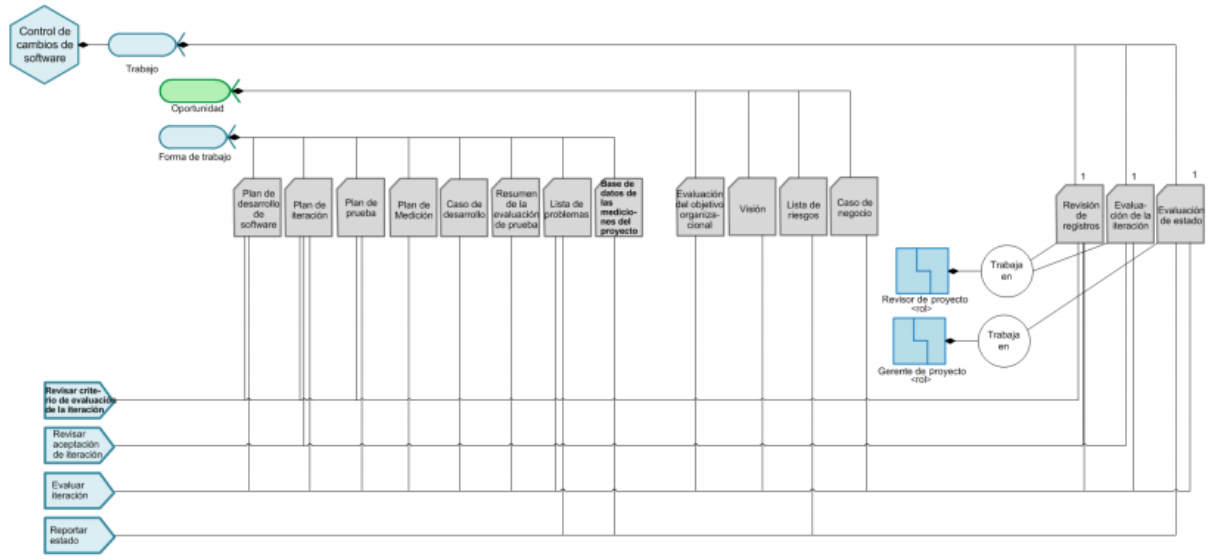


Fig. 16. Control de Cambios de Software de RUP (2)

Fuente: (Jiménez, 2016)

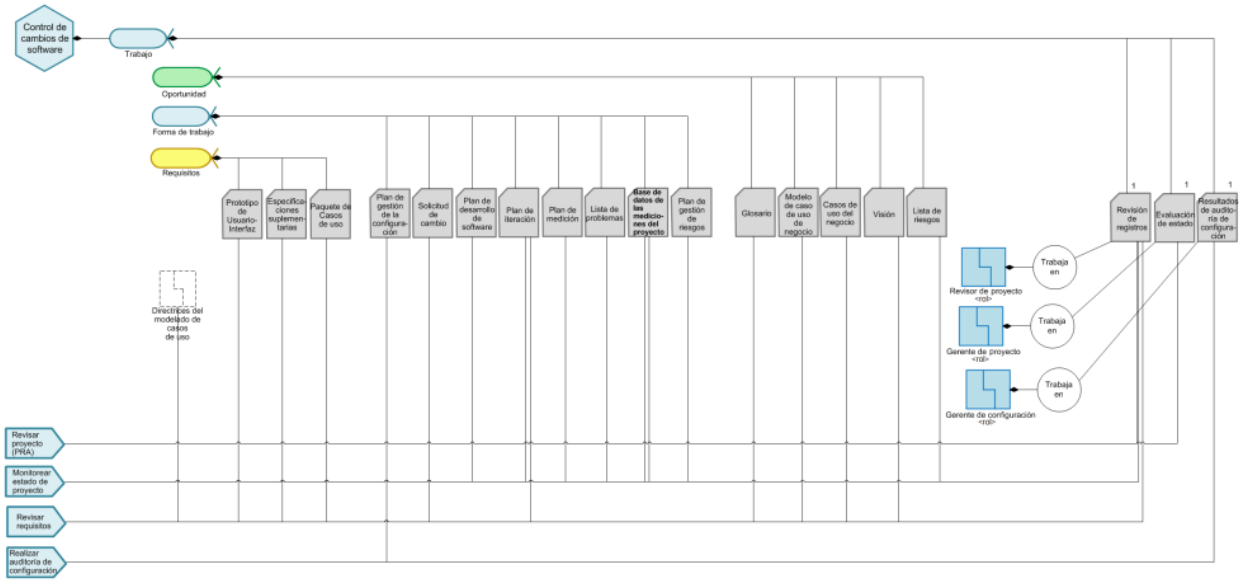


Figura. 17. Control de Cambios de Software de RUP (3)

Fuente: (Jiménez, 2016)

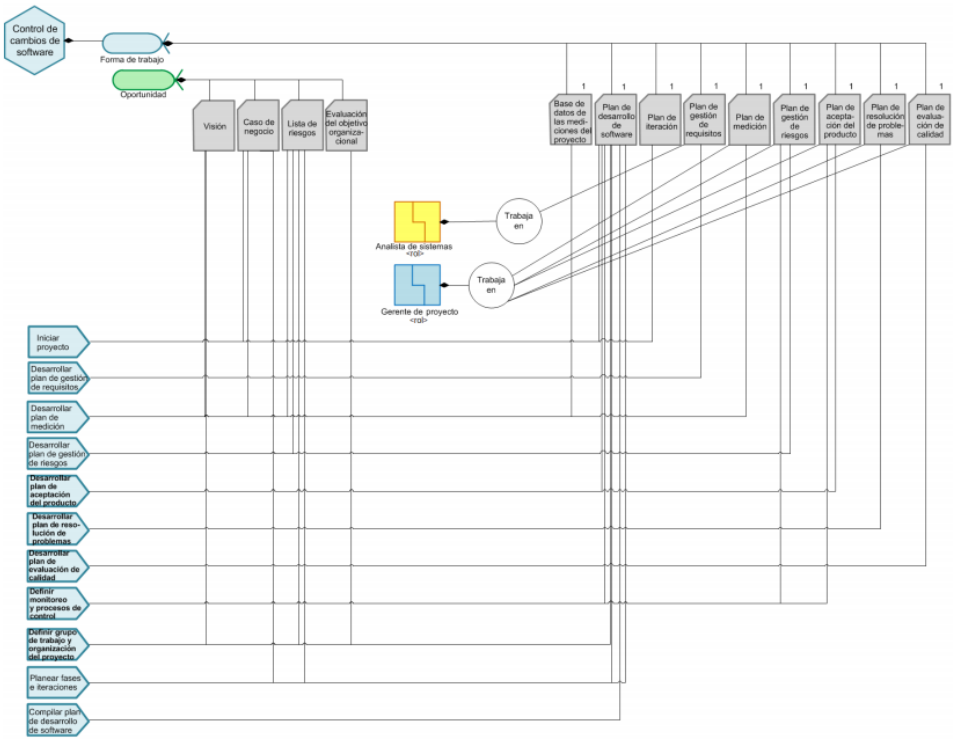


Fig. 18. Control de Cambios de Software de RUP (4)

Fuente: (Jiménez, 2016)

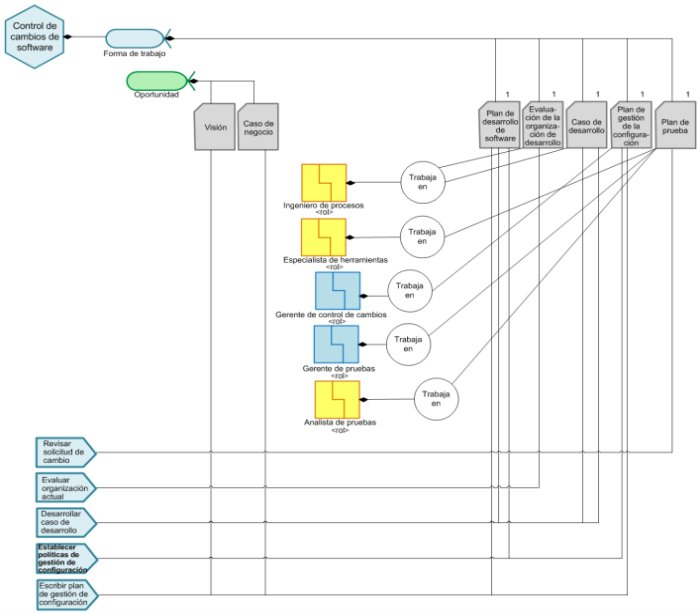


Fig. 19. Control de Cambios de Software de RUP (5)

Fuente: (Jiménez, 2016)

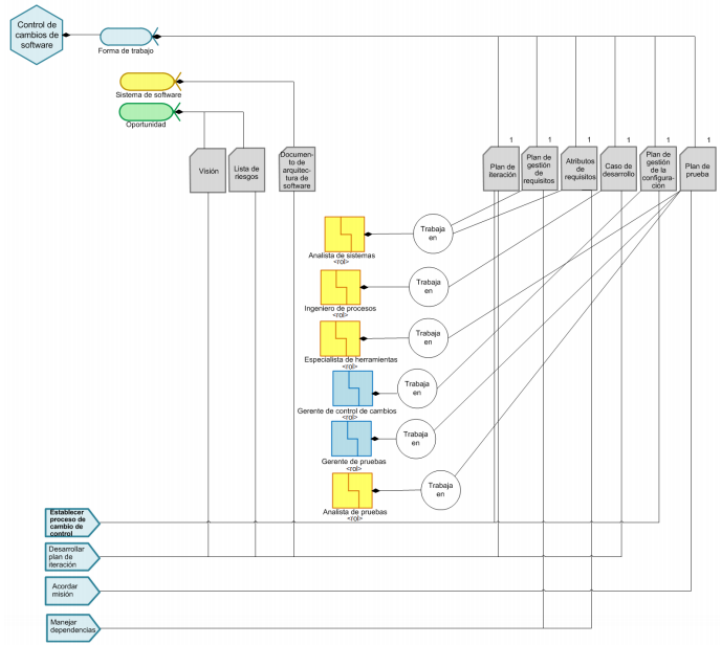


Fig. 20. Control de Cambios de Software de RUP (6)

Fuente: (Jiménez, 2016)

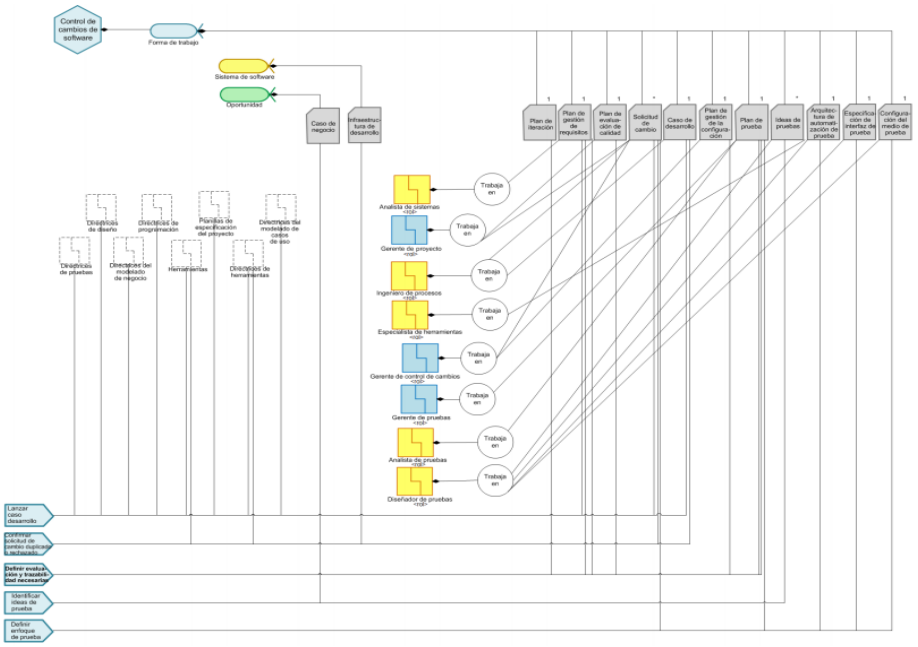


Fig. 21. Control de Cambios de Software de RUP (7)

Fuente: (Jiménez, 2016)

4) Conversión de la práctica CCS-RUP

Para la conversión de la práctica CCS-RUP, se sigue el proceso de definición de prácticas bien formadas y nombradas propuesto por Barón (2019). En la figura 5-8 se describe este proceso de manera detallada.

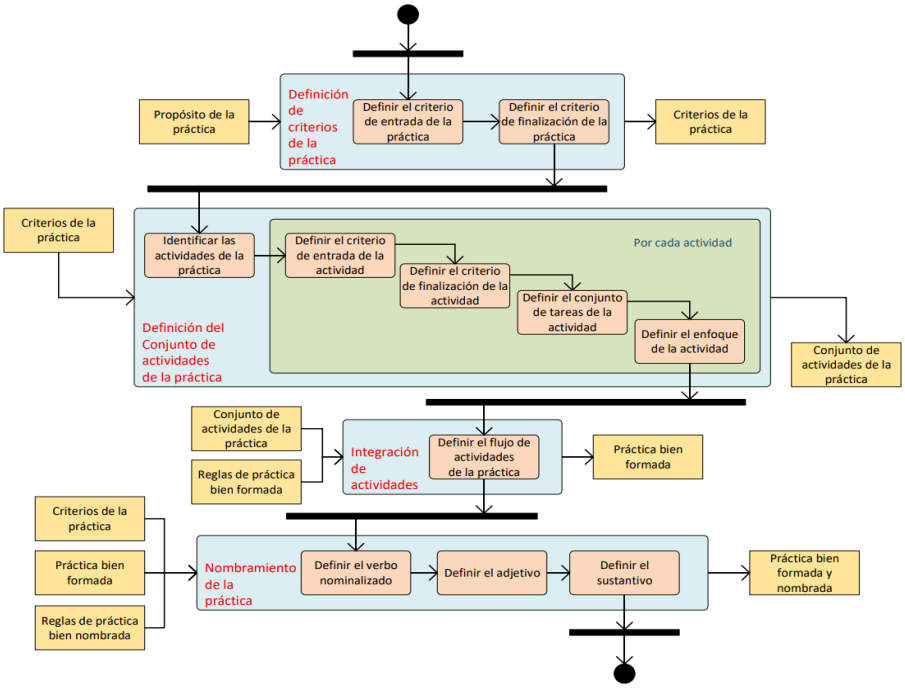


Fig. 22. Proceso de definición de prácticas bien formadas y nombradas

Fuente: (Barón, 2019)

a) Fase 1: Definición de los criterios de la práctica

En esta fase, teniendo en cuenta propósito de la práctica se definen cuáles son los criterios de entrada y finalización de la práctica. Obteniendo como resultado los criterios de la práctica. Para lo cual se ejecutan las siguientes actividades:

Actividad 1.1: Definir el criterio de entrada de la práctica

Teniendo en cuenta el propósito de la práctica, se definen los criterios de entrada como “Requisitos: Coherentes”, dado que existe una visión general clara para realizar el cambio, se han tratado y comprendido los conflictos. Además, se tiene claro cuáles son las prioridades. También, “Forma de trabajo: Con principios establecidos” puesto que se entiende el contexto en que el equipo debe trabajar.

Actividad 1.2: Definir el criterio de finalización de la práctica

Teniendo en cuenta el propósito de la práctica, se define los criterios de finalización como “Forma de trabajo: Trabajando bien”, La forma de trabajo está trabajando bien para el equipo. También, “Sistema software: Listo”, dado que existe la documentación de usuario con la información del cambio, además, los representantes de los interesados aceptaron el sistema y quieren que se haga operacional. Y “Trabajo: Concluido” dado se termina el trabajo cuando el cliente ha aceptado el sistema software resultante.

b) Fase 2: Definición del conjunto de actividades de la práctica

En esta fase, con los criterios de la práctica definidos, se procede a definir el conjunto de actividades de la práctica. Para ello, se desarrollan las siguientes actividades.

Actividad 2.1: Identificar las actividades de la práctica

Para ejecutar la práctica se identifican las actividades: (i) Establecer políticas de control de cambio; (ii) Iniciar iteración; (iii) Desarrollar plan de iteración; (iv) Desarrollar plan de medición; (v) Desarrollar plan de gestión de problemas y riesgos; (vi) Desarrollar plan de aceptación del producto; (vii) Definir monitoreo y procesos de control; (viii) Definir grupo de trabajo y organización del proyecto; (ix) Definir evaluación y trazabilidad necesarias; (x) Desarrollar caso de desarrollo; (xi) Desarrollar plan de despliegue; (xii) Compilar plan de desarrollo de software; (xiii) Programar y asignar trabajo; (xiv) Manejar excepciones y problemas; (xv) Verificar la configuración de herramientas e instalación; (xvi) Presentar o actualizar solicitud de cambio; (xvii) Gestionar las pruebas de aceptabilidad; (xviii) Revisar aceptación del cambio; (ixx) Evaluar iteración; (xx) Finalizar iteración.

Actividad 2.2: Definir el criterio de entrada de la actividad

En esta actividad, se definen las condiciones necesarias para iniciar cada actividad en términos de uno o varios alfas, con su respectivo estado y producto de trabajo. El criterio de entrada de la práctica se ve reflejado en el criterio de entrada de la primera actividad de la práctica.

Actividad 2.3: Definir el criterio de finalización de la actividad

En esta actividad, se definen las condiciones necesarias para finalizar cada actividad en términos de uno o varios alfas, con su respectivo estado y producto de trabajo. El criterio de finalización de la práctica se ve reflejado en el criterio de finalización de la última actividad de la práctica.

Actividad 2.4: Definir el conjunto de tareas de la actividad

En esta actividad se definen las tareas que se deben realizar en cada actividad de la práctica.

Actividad 2.5: Definir el enfoque de la actividad

Se define la forma en que se debe abordar cada actividad en términos de un adjetivo. Puesto que, la fuente base de esta definición de práctica no define enfoques o adjetivos para sus actividades, se utiliza el conjunto consolidado de adjetivos obtenidos del modelo.

El conjunto de actividades con sus respectivos criterios de entrada y finalización se presenta en las tablas 13 y 14 En las tablas 15 y 16 se indica el conjunto de actividades con sus respectivas tareas y enfoques.

TABLA XIII

CONJUNTO DE ACTIVIDADES CON CRITERIOS DE ENTRADA Y FINALIZACIÓN (1)

Espacio de actividad	Actividad	Criterio de entrada		Criterio de finalización	
		Alfa	Estado	Alfa	Estado
Prepararse para hacer el trabajo	Establecer políticas de control cambio	Requisitos	Coherente	Requisitos	Aceptable
		Forma de trabajo	Con principios establecidos	Forma de trabajo	Con bases establecidas
	Iniciar iteración	Equipo	Sembrado	Equipo	Formado
	Desarrollar plan de iteración	Trabajo	Preparado	Trabajo	Comenzado
	Desarrollar plan de medición	Forma de trabajo	Con principios establecidos	Forma de trabajo	Con bases establecidas
		Interesados	Involucrado	Interesados	De acuerdo
	Desarrollar plan de gestión de problemas y riesgos	Forma de trabajo	Con principios establecidos	Forma de trabajo	Con bases establecidas
		Interesados	Involucrado	Interesados	De acuerdo
	Desarrollar plan de aceptación del producto	Forma de trabajo	Con principios establecidos	Forma de trabajo	Con bases establecidas
		Interesados	Involucrado	Interesados	De acuerdo
	Definir monitoreo y procesos de control	Forma de trabajo	Con principios establecidos	Forma de trabajo	Con bases establecidas
	Definir grupo de trabajo y organización del proyecto	Forma de trabajo	Con principios establecidos	Forma de trabajo	Con bases establecidas

Espacio de actividad	Actividad	Criterio de entrada		Criterio de finalización	
		Alfa	Estado	Alfa	Estado
	Definir evaluación y trazabilidad necesarias	Forma de trabajo	Con principios establecidos	Forma de trabajo	Con bases establecidas
		Interesados	Involucrado	Interesados	De acuerdo
	Desarrollar caso de desarrollo	Forma de trabajo	Con principios establecidos	Forma de trabajo	Con bases establecidas
Espacio de actividad	Actividad	Criterio de entrada		Criterio de finalización	
		Alfa	Estado	Alfa	Estado
	Desarrollar plan de despliegue	Forma de trabajo	Con principios establecidos	Forma de trabajo	Con bases establecidas
		Interesados	Involucrado	Interesados	De acuerdo
	Compilar plan de desarrollo de software	Forma de trabajo	Con bases establecidas	Forma de trabajo	En uso
		Requisitos	Aceptable	Requisitos	Tratado

Fuente: Elaboración propia

TABLA XIV

CONJUNTO DE ACTIVIDADES CON CRITERIOS DE ENTRADA Y FINALIZACIÓN (2)

Espacio de actividad	Actividad	Criterio de entrada		Criterio de finalización	
		Alfa	Estado	Alfa	Estado
Coordinar actividades	Programar y asignar trabajo	Equipo	Sembrado	Equipo	Formado
	Manejar excepciones y problemas	Equipo	Formado	Equipo	Colaborando
		Sistema software	Demostrable	Sistema software	usable
Apoyar al equipo	Verificar la configuración de herramientas e instalación	Forma de trabajo	Con bases establecidas	Forma de trabajo	En uso
	Presentar o actualizar solicitud de cambio	Forma de trabajo	Con bases establecidas	Forma de trabajo	En uso
Seguimiento del proceso	Gestionar las pruebas de aceptabilidad	Forma de trabajo	Con bases establecidas	Forma de trabajo	En uso

Espacio de actividad	Actividad	Criterio de entrada		Criterio de finalización	
		Alfa	Estado	Alfa	Estado
	Revisar aceptación del cambio	Forma de trabajo	En uso	Forma de trabajo	En su lugar
		Interesados	De acuerdo	Interesados	Satisfecho para despliegue
	Evaluar iteración	Trabajo	Comenzado	Trabajo	Bajo control
		Trabajo	Bajo control	Trabajo	Concluido
	Finalizar la iteración	Forma de trabajo	En su lugar	Forma de trabajo	Trabajando bien
		Sistema software	Usable	Sistema software	Listo

Fuente: Elaboración propia

TABLA XV

CONJUNTO DE ACTIVIDADES Y TAREAS (1)

Actividad	Tareas	Enfoque
Establecer políticas de control cambio	1. Escribir plan de gestión de configuración. 2. Establecer proceso de cambio de control.	Basado en casos de uso
Iniciar iteración	1. Identificar solicitud de cambio. 2. Generar orden de trabajo.	Iterativo
Desarrollar plan de iteración	1. Identificar orden de trabajo. 2. Desarrollar plan de iteración.	Iterativo
Desarrollar plan de medición	1. Identificar objetivos del cambio. 2. Establecer métricas.	Cuantitativo

Actividad	Tareas	Enfoque
Desarrollar plan de gestión de problemas y riesgos	1. Evaluar posibles problemas. 2. Desarrollar plan de resolución de problemas. 3. Identificar posibles riesgos. 4. Desarrollar plan de gestión de riesgos.	Basado en escenarios
Desarrollar plan de aceptación del producto	1. Identificar objetivos del cambio. 2. Definir plan de evaluación.	Basado en valor
Definir monitoreo y procesos de control	1. Identificar prioridades de la solicitud de cambio. 2. Establecer plan de monitoreo.	Iterativo
Definir grupo de trabajo y organización del proyecto	1. Definir grupo de trabajo. 2. Organizar grupo de trabajo.	Bien organizado
Definir evaluación y trazabilidad necesarias	1. Identificar ideas de prueba. 2. Definir enfoques de prueba. 3. Identificar objetivos de prueba.	Basado en casos de uso
Desarrollar caso de desarrollo	1. Diseñar base de datos. 2. Planear integración. 3. Planear integración de subsistema.	Basado en casos de uso

Fuente: Elaboración propia

TABLA XVI

CONJUNTO DE ACTIVIDADES Y TAREAS (2)

Actividad	Tareas	Enfoque
Desarrollar plan de despliegue	1. Identificar el ambiente de despliegue. 2. Desarrollar plan de despliegue.	Basado en escenarios
Compilar plan de desarrollo de software	1. Comprobar prioridades del cambio. 2. Confirmar grupo de trabajo.	Estructurado
Programar y asignar trabajo	1. Programar trabajo. 2. Asignar trabajo.	Integrado

Actividad	Tareas	Enfoque
Manejar excepciones y problemas	1. Revisar plan de resolución de problemas.	Iterativo
Verificar la configuración de herramientas e instalación	1. Verificar instalación de herramientas. 2. Verificar configuración de herramientas.	Integrado
Presentar o actualizar solicitud de cambio	1. Enviar solicitud de cambio.	Iterativo
Gestionar las pruebas de aceptabilidad	1. Revisar plan de pruebas. 2. Obtener resultado de pruebas. 3. Reportar resultado de pruebas.	Basado en escenarios
Revisar aceptación del cambio	1. Revisar solicitud de cambio. 2. Revisar estado del software. 3. Revisar la arquitectura. 4. Revisar código. 5. Obtener comentarios de pruebas. 6. Evaluar calidad.	Basado en valor
Evaluar iteración	1. Revisar plan de iteración. 2. Revisar criterio de evaluación de iteración. 3. Revisar aceptación de iteración.	Iterativo
Finalizar la iteración	1. Revisar aceptación del cambio.	Iterativo

Fuente: Elaboración propia

c) Fase 3: Integración de actividades

En esta fase, con el conjunto de actividades definido y las reglas de práctica bien formada, se define el flujo de actividades el cual permite obtener una práctica bien formada.

Actividad 3.1: Definir el flujo de actividades de la práctica

En esta actividad, se define el flujo de actividades de la práctica, este flujo permite identificar el progreso del alfa hasta lograr el cumplimiento del criterio de finalización de la práctica. Esto permite cumplir las reglas de práctica bien formada.

TABLA XVII

FLUJO DE ACTIVIDADES (1)

Numeral de actividad	Actividad	Predecesor
1	Establecer políticas de control cambio	
2	Iniciar iteración	1
3	Desarrollar plan de iteración	1
4	Desarrollar plan de medición	3
5	Desarrollar plan de gestión de problemas y riesgos	3
6	Desarrollar plan de aceptación del producto	3
7	Definir monitoreo y procesos de control	6
8	Definir grupo de trabajo y organización del proyecto	2; 4; 5; 7
9	Definir evaluación y trazabilidad necesarias	4; 7
10	Desarrollar caso de desarrollo	8; 9
11	Desarrollar plan de despliegue	10
12	Compilar plan de desarrollo de software	11

Fuente: Elaboración propia

TABLA XVIII

FLUJO DE ACTIVIDADES (2)

Numeral de actividad	Actividad	Predecesor
13	Programar y asignar trabajo	12
14	Manejar excepciones y problemas	5; 12
15	Verificar la configuración de herramientas e instalación	10; 12
16	Presentar o actualizar solicitud de cambio	13; 14; 15
17	Gestionar las pruebas de aceptabilidad	10
18	Revisar aceptación del cambio	17
19	Evaluar iteración	18
20	Finalizar la iteración	19

Fuente: Elaboración propia

d) Fase 4: Nombramiento de la práctica

En esta fase se definen los elementos necesarios para obtener una práctica bien nombrada. Una práctica es bien nombrada si su nombre incluye un verbo nominalizado, que indica lo que se hace con la práctica, un adjetivo que indica cómo se hace y un sustantivo que indica el objeto sobre el cual se aplica la práctica.

Actividad 4.1: Definir el verbo nominalizado

Para esta práctica, el verbo nominalizado que indica lo que se hace con la práctica es “Control”. Este verbo nominalizado se puede identificar en la tabla 19 de taxonomía de verbos nominalizados obtenida de Barón (2019).

TABLA XIX

VERBOS NOMINALIZADOS

Taxonomia de verbos nominalizados								
Analisis	Recoleccion	Identification	Negociacion	Especificacion	Valoracion	Elicitacion	Educcion	Priorizacion
Modelado	Diseno	Definicion						
Implementacion	Construccion	Integracion	Mejoramiento	Desarrollo				
Prueba	Construccion	Realizacion	Identification	Resolucion	Mejoramiento	Evaluacion		
Despliegue	Liberacion	Habilitacion						
Mantenimiento	Operacion	Monitoreo	Apoyo					
Verificación	Aprobacion	Aceptacion						
Validacion	Aprobacion	Aceptacion	Entrenamiento					
Planificacion	Estimacion	Organizacion	Definicion	Financiacion				
Conformación	Seleccion	Contratacion	Entrenamiento					
Cooperacion	Certificacion	Mejoramiento	Comunicacion	Guia	Entrenamiento			
Coordinacion	Control	Aseguramiento	Inspeccion	Monitoreo	Seguimiento	Revision		
Cierre	Suspension	Retiro	Entrega					

Fuente: (Barón, 2019)

Actividad 4.2: Definir el adjetivo

Para esa práctica, el adjetivo que indica cómo se hace la práctica es “Iterativo”. Este adjetivo se puede identificar en la tabla 20 de adjetivos obtenida de Barón (2019).

TABLA XX

ADJETIVOS

Conjunto consolidado de adjetivos			
Ágil	Colaborativo	Evolutivo	Probado
Automatizado	Compartido	Formal	Progresivo
Auto-organizado	Concurrente	Funcional	Prospectivo
Basado en actividades	Conjunto	Global	Repetible
Basado en casos de uso	Continuo	Incremental	Reutilizable
Basado en componentes	Cualitativo	Integrado	Sistemático
Basado en escenarios	Cuantitativo	Interdisciplinario	Sistémico
Basado en riesgos	Dirigido por pruebas	Iterativo	Temprano
Basado en valor	Disciplinario	Metódico	Valor compartido
Bien organizado	Empírico	Orientado al producto	Visual
Casual	Escalonado	Orientado por objetos	
Arquitectura	Estructurado	Personalizado	

Fuente: (Barón, 2019)

Actividad 4.3: Definir el sustantivo

Para esta práctica, el sustantivo que indica el objeto sobre el cual se aplica la práctica es “Sistema Software”. Este sustantivo se puede identificar en la tabla XXI de sustantivos obtenida de Barón (2019).

TABLA XXI

SUSTANTIVOS

Conjunto de sustantivos	
Alfas	**Sub- Alfas**
Interesados	Representante de los interesados
Oportunidad	Necesidad
Requisitos	Ítem de requisito
Sistema de software	Elemento de sistema de software
Equipo	Miembro del equipo

Fuente: (Barón, 2019)

Por lo tanto, se obtiene el nombre de práctica Control Iterativo de Cambios del Sistema Software. Desde la figura 5-9 en adelante se encuentran la tarjeta de práctica, así como las tarjetas de actividades de la práctica Control Iterativo de Cambios del Sistema Software.

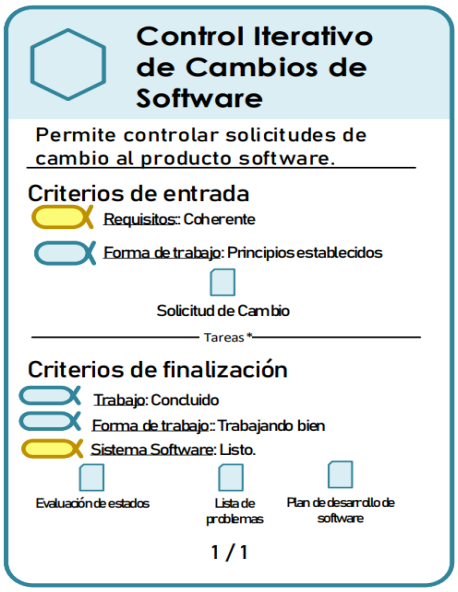


Fig. 23. Tarjeta de la práctica Control Iterativo de Cambios del Sistema Software

Fuente: Elaboración propia

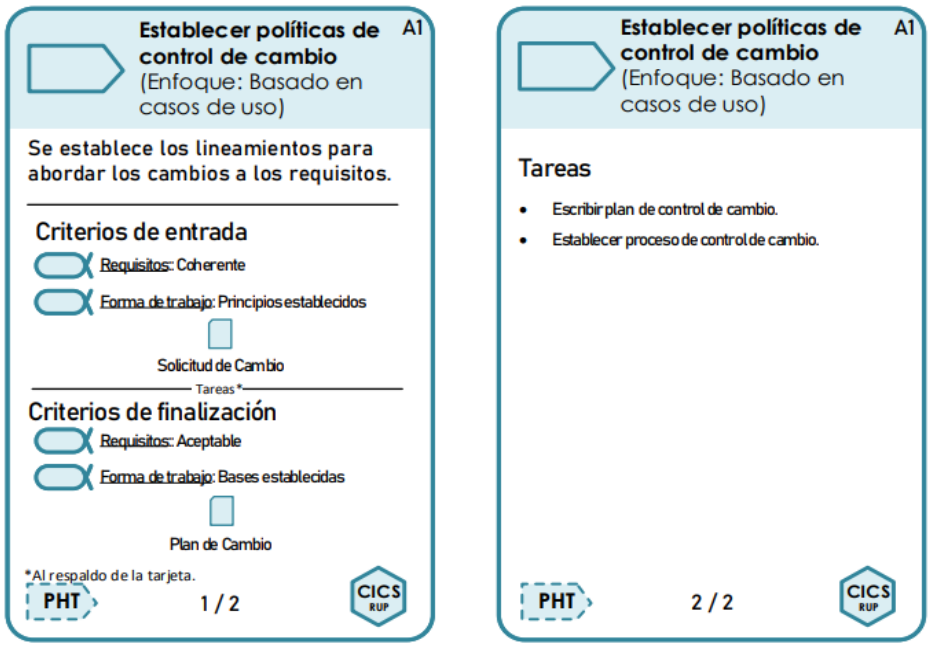


Fig. 24. Tarjeta de actividad 1: Establecer políticas de control de cambio

Fuente: Elaboración propia

Fig. 25. Tarjeta de actividad 2: Iniciar iteración

Fuente: Elaboración propia

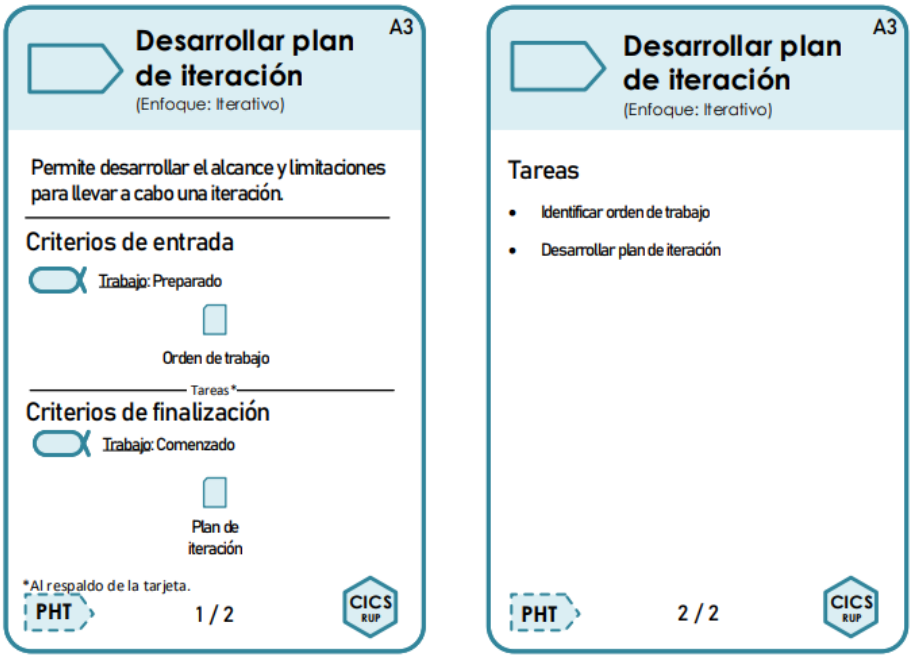


Fig. 26. Tarjeta de actividad 3: Desarrollar plan de iteración

Fuente: Elaboración propia

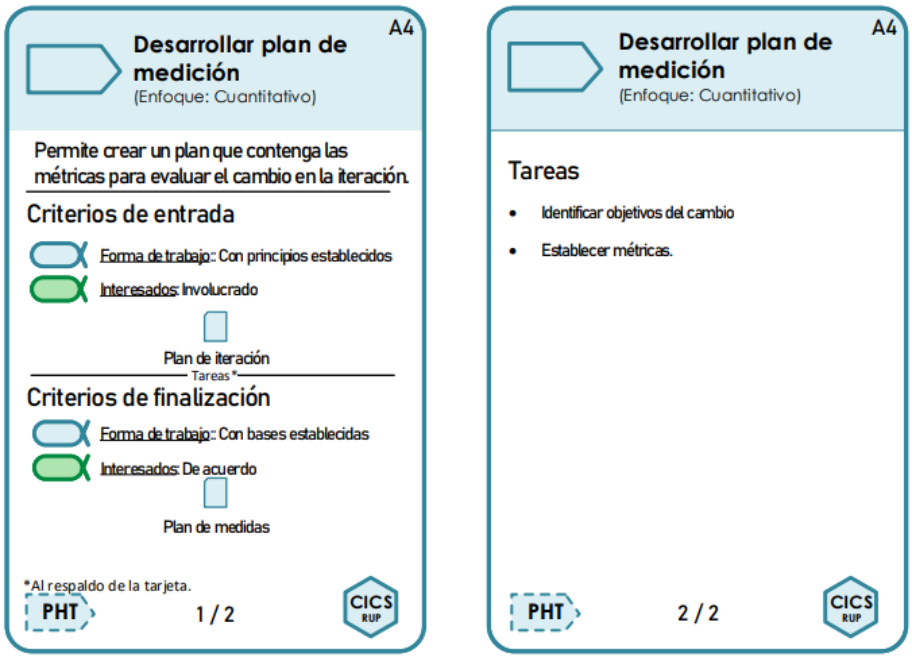


Fig. 27. Tarjeta de actividad 4: Desarrollar plan de medición

Fuente: Elaboración propia

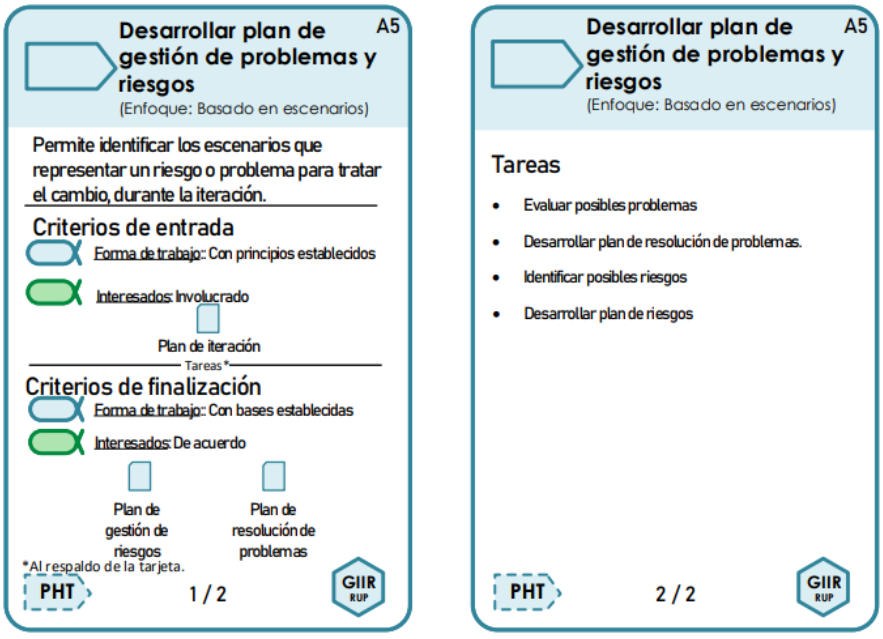


Fig. 28. Tarjeta de actividad 5: Desarrollar plan de gestión de problemas y riesgos

Fuente: Elaboración propia

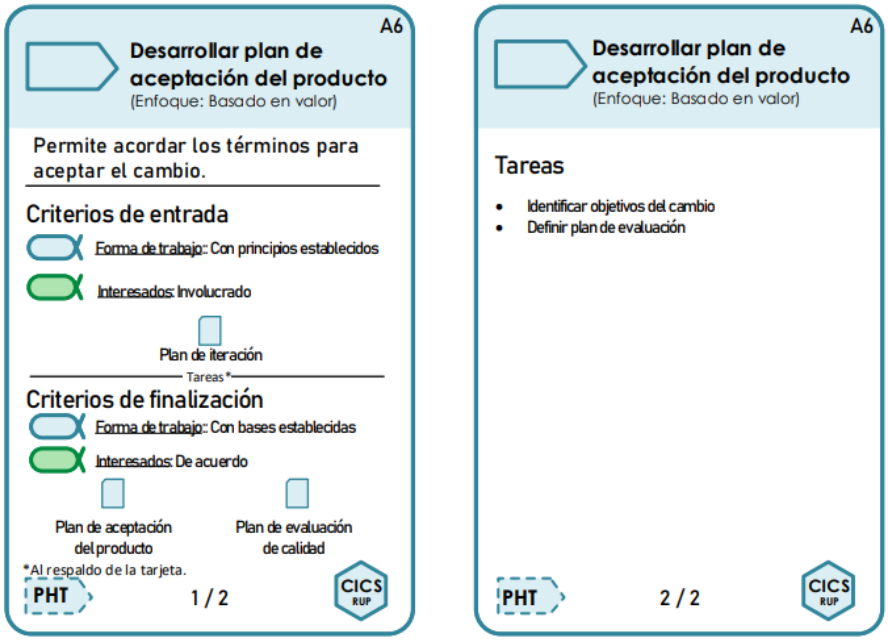


Fig. 29. Tarjeta de actividad 6: Aceptación del producto

Fuente: Elaboración propia

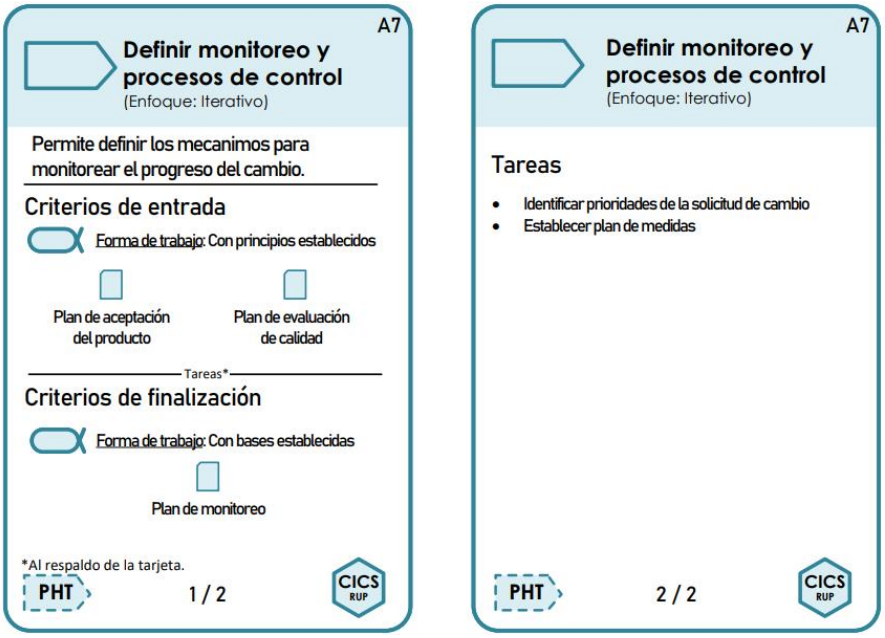


Fig. 30. Tarjeta de actividad 7: Definir monitoreo y procesos de control

Fuente: Elaboración propia

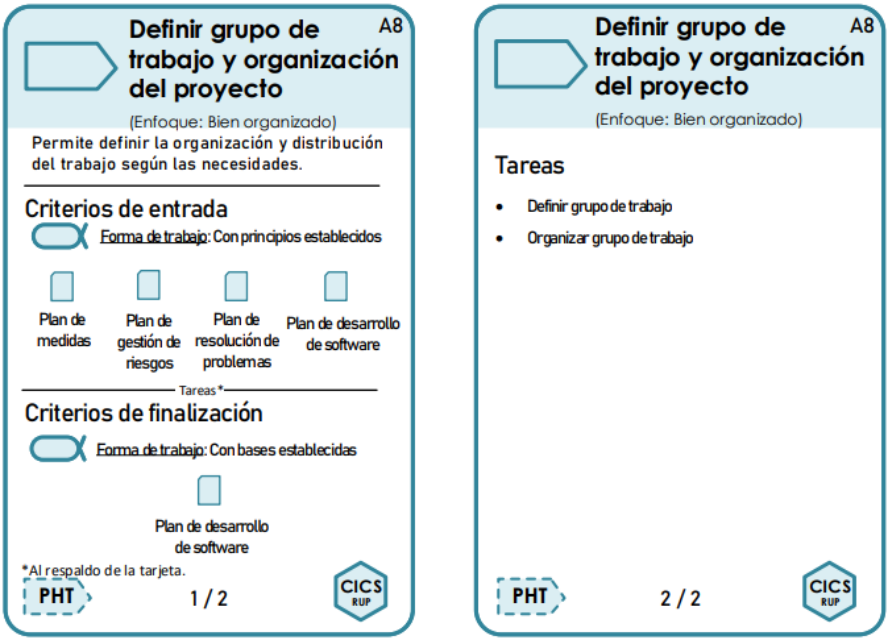


Fig. 31. Tarjeta de actividad 8: Definir grupo de trabajo y organización del proyecto

Fuente: Elaboración propia

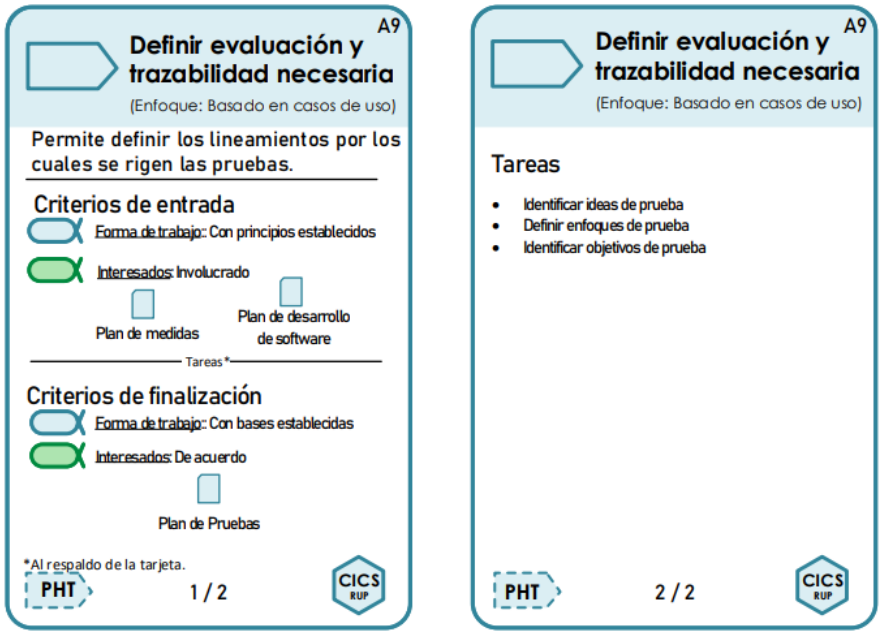


Fig. 32. Tarjeta de actividad 9: Definir evaluación y trazabilidad necesaria

Fuente: Elaboración propia

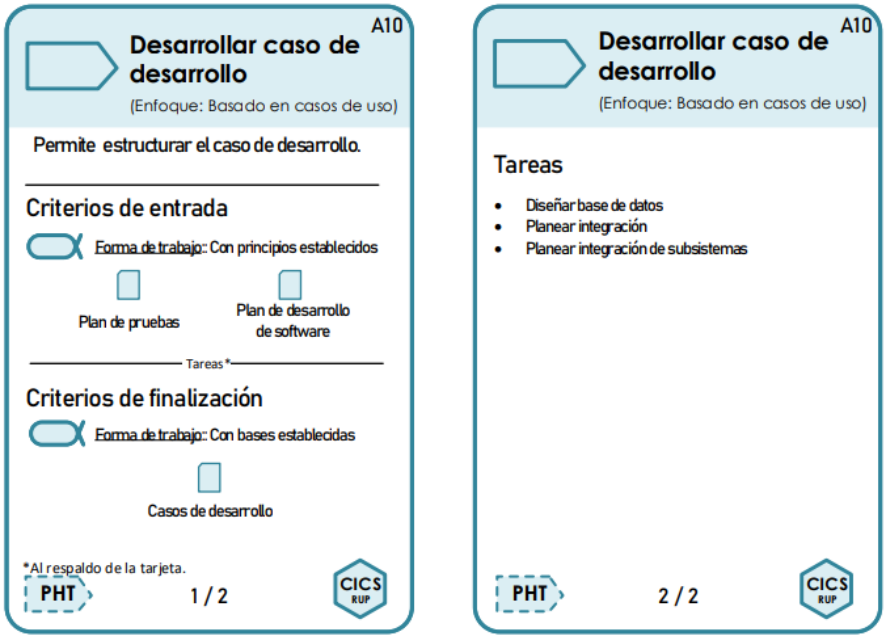


Fig. 33. Tarjeta de actividad 10: Desarrollar caso de desarrollo

Fuente: Elaboración propia

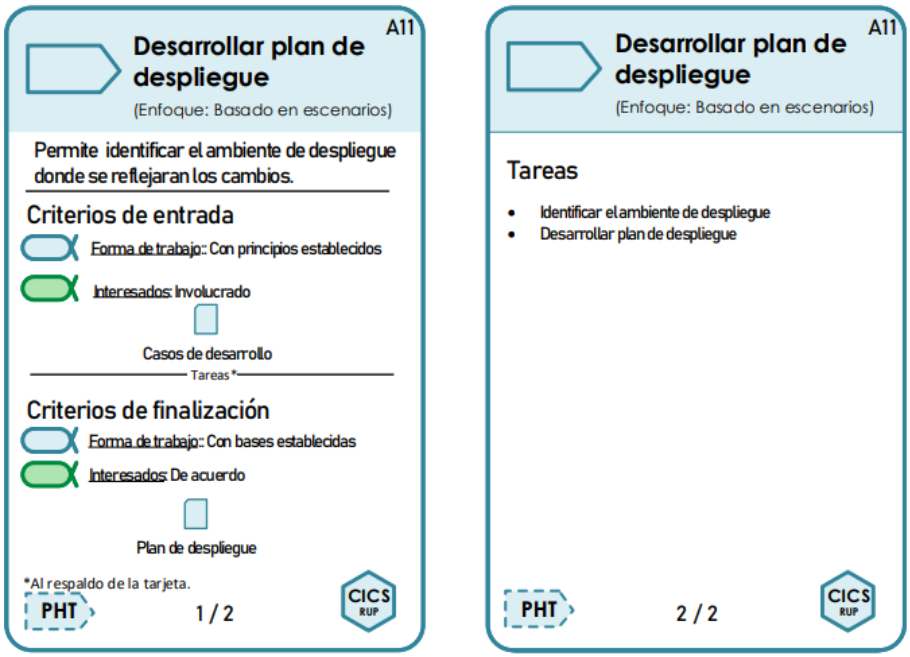


Fig. 34. Tarjeta de actividad 11: Desarrollar plan de despliegue

Fuente: Elaboración propia

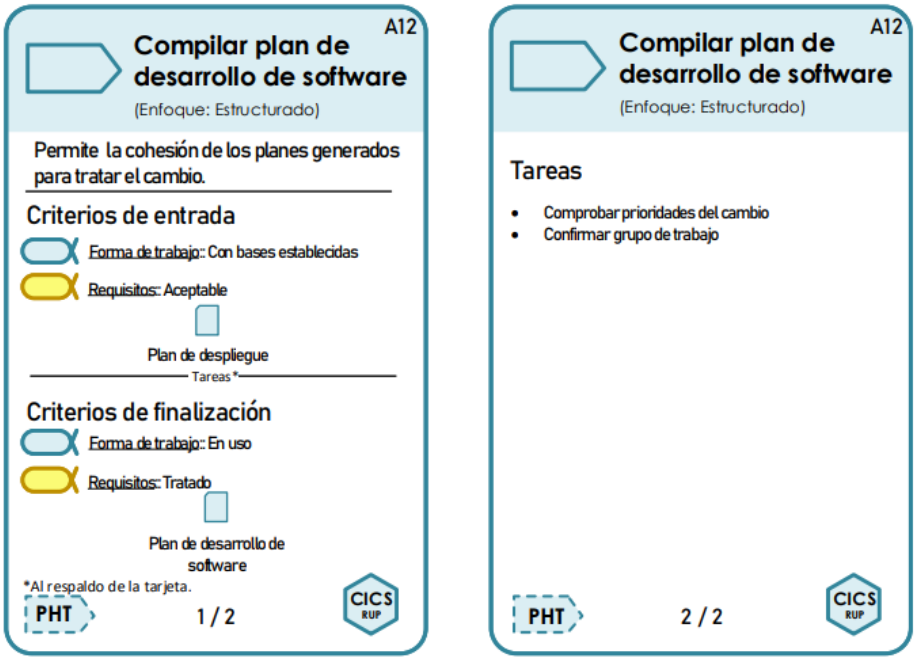


Fig. 35. Tarjeta de actividad 12: Compilar plan de desarrollo de software

Fuente: Elaboración propia

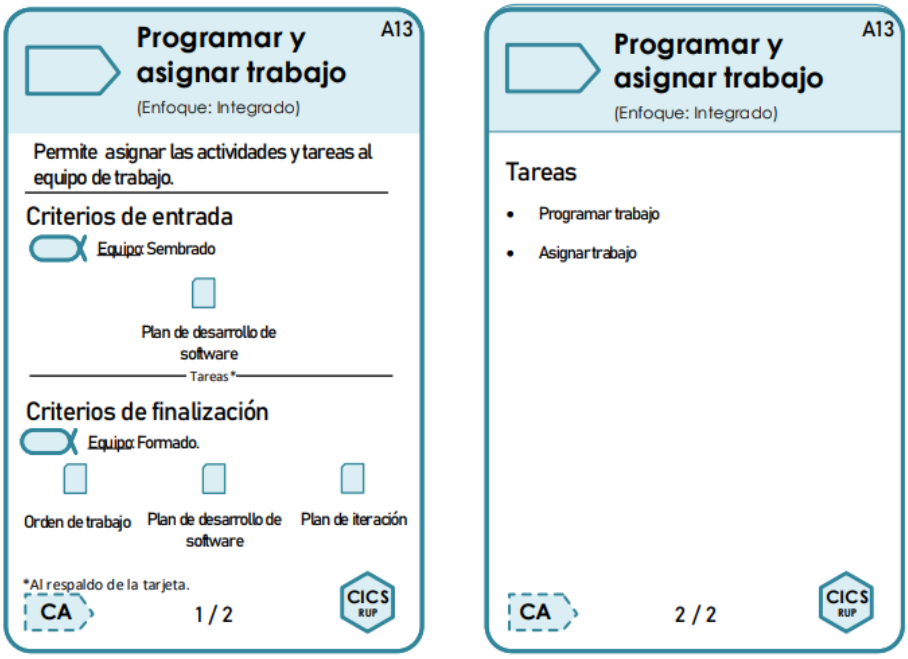


Fig. 36. Tarjeta de actividad 13: Programar y asignar trabajo

Fuente: Elaboración propia

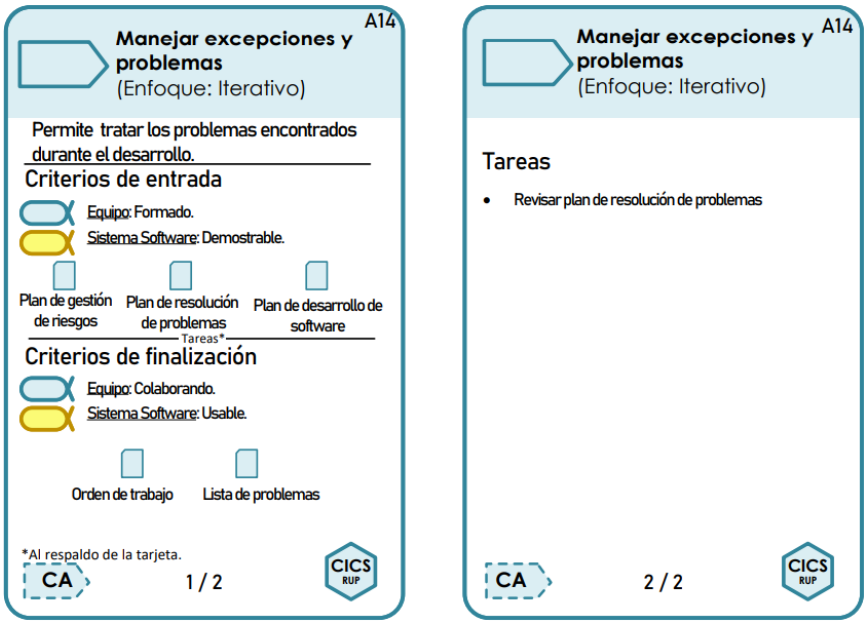


Fig. 37. Tarjeta de actividad 14: Manejar excepciones y problemas

Fuente: Elaboración propia

Fig. 38. Tarjeta de actividad 15: Verificar la configuración de herramientas e instalación

Fuente: Elaboración propia

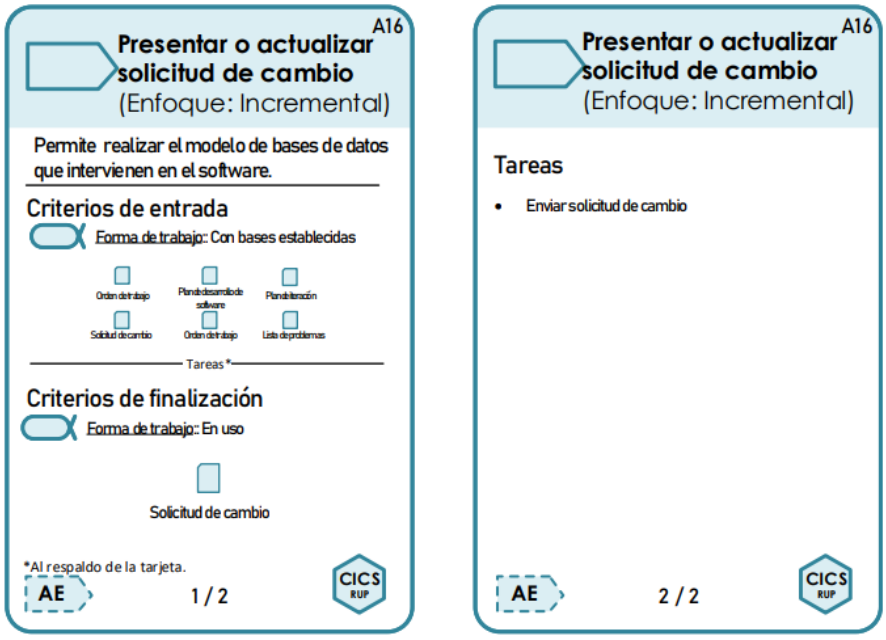


Fig. 39. Tarjeta de actividad 16: Presentar o actualizar solicitud de cambio

Fuente: Elaboración propia

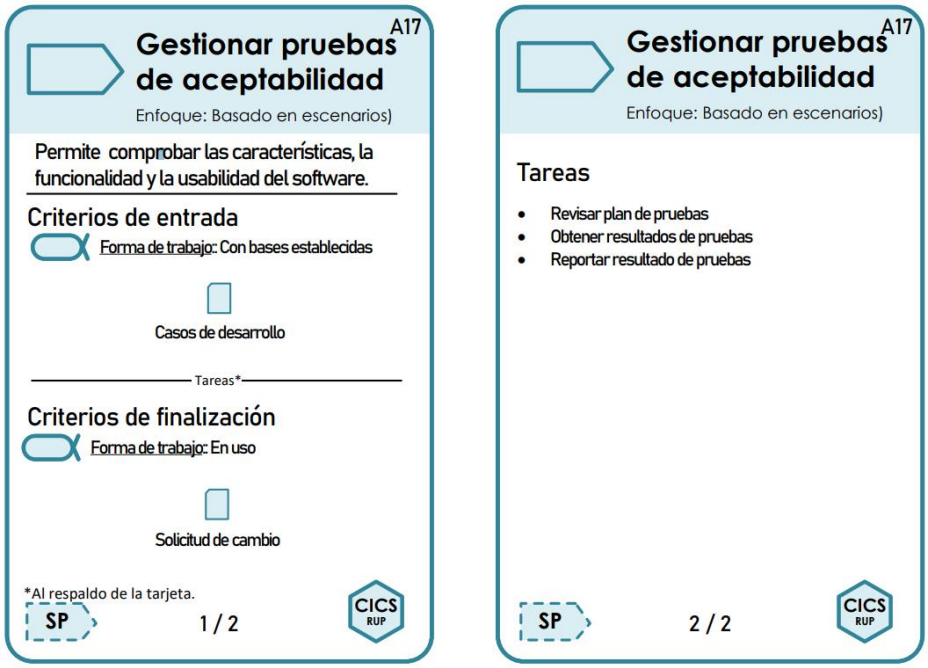


Fig. 40. Tarjeta de actividad 17: Gestionar pruebas de aceptabilidad

Fuente: Elaboración propia

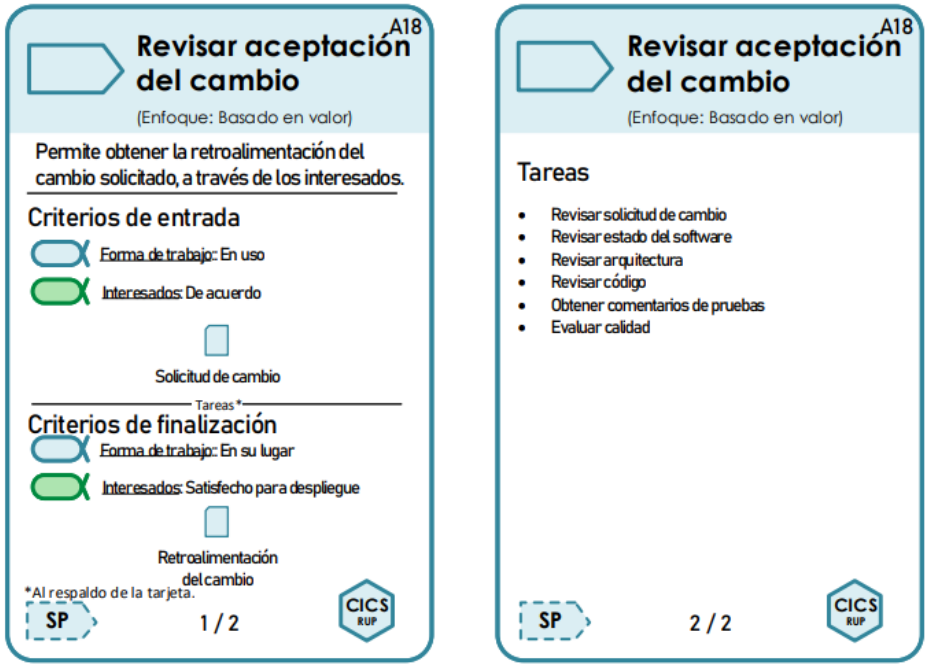


Fig. 41. Tarjeta de actividad 18: Revisar aceptación del cambio

Fuente: Elaboración propia

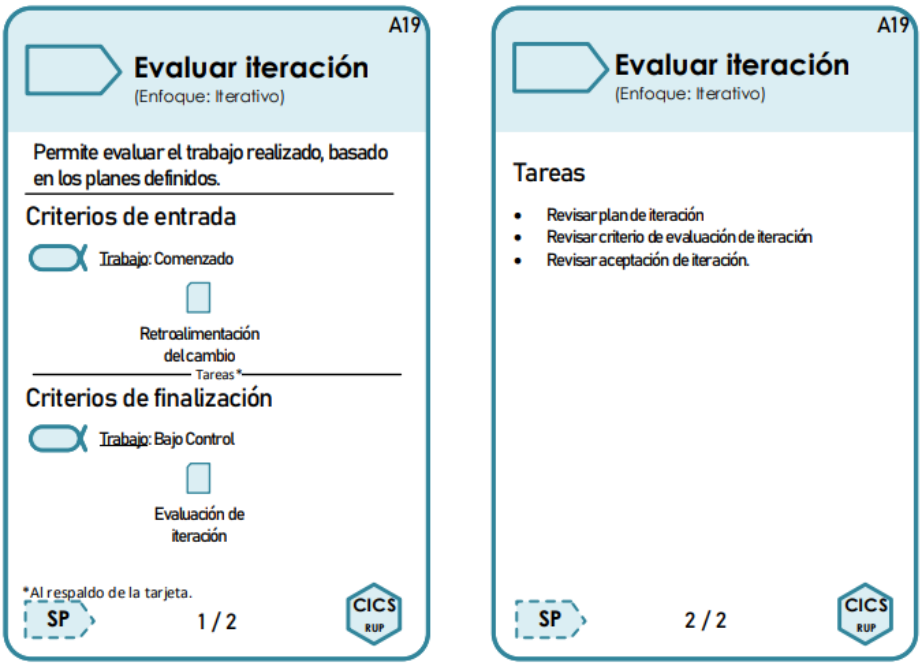


Fig. 42. Tarjeta de actividad 19: Evaluar iteración

Fuente: Elaboración propia

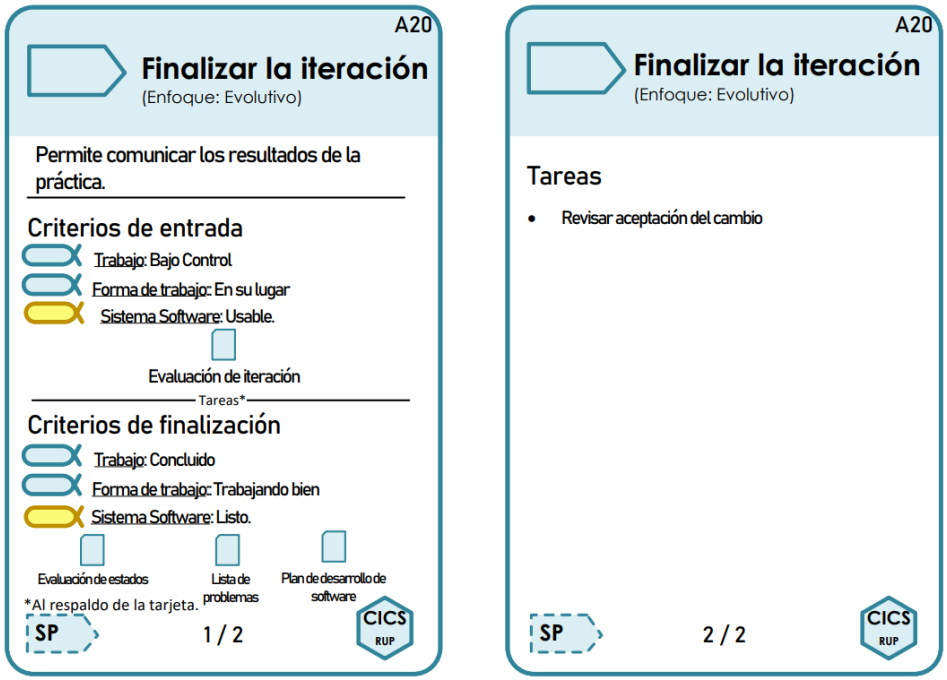


Fig. 43. Tarjeta de actividad 20: Finalizar la iteración

Fuente: Elaboración propia

C. VALIDACIÓN DE LA INVESTIGACIÓN

En este capítulo se realiza la validación de la práctica CICS-RUP mediante un estudio de caso que permite simular un contexto real siguiendo la práctica y haciendo uso de las tarjetas de practica y actividades establecidas en el capítulo anterior. Como contexto de validación se parte de una aplicación base la cual es modificada para crear la aplicación "Práctica CICS" que permite controlar los cambios en un sistema software haciendo uso de la práctica CICS-RUP.

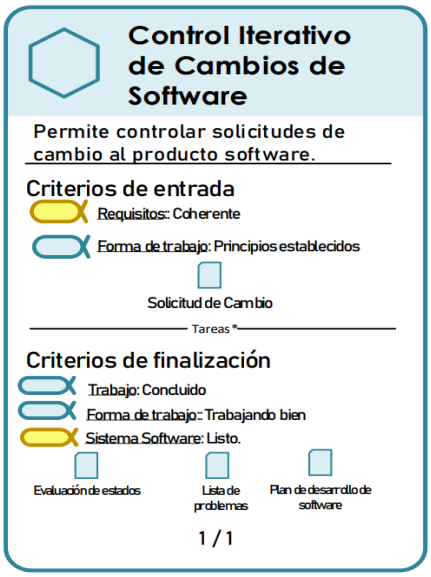


Fig. 44. Tarjeta de la práctica Control Iterativo de Cambios del Sistema Software

Fuente: Elaboración propia

1) Modificación de software hospitalario

Actividad 1:

En la tabla 44 se observa el producto de trabajo, además del resultado de la actividad 1 reflejada en la figura 6-2.

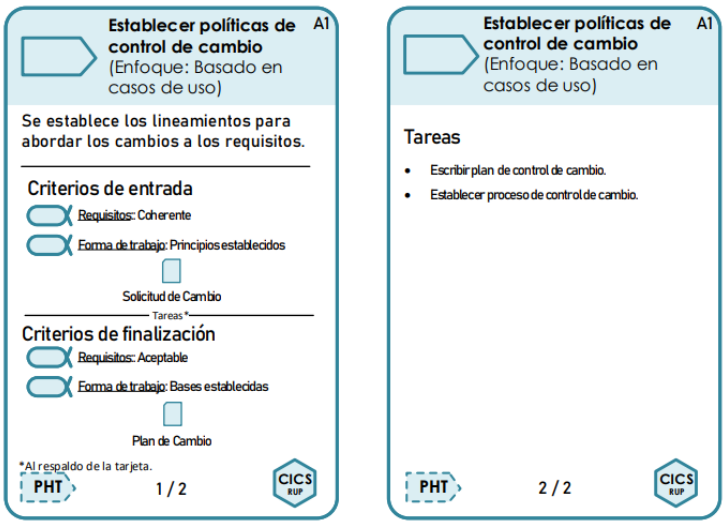


Fig. 45. Tarjeta de actividad Establecer políticas de control de cambio

Fuente: Elaboración propia

TABLA XXII

RESULTADOS DE LA ACTIVIDAD ESTABLECER POLÍTICAS DE CONTROL DE CAMBIO

Producto de entrada Actividad 1:
Solicitud de cambio.
CICS-001: Teniendo en cuenta el software existente que permite la creación de usuarios, hospitales y asignación de roles. Se requiere la adaptación de la interfaz y configuración del backend con el fin de controlar los cambios y visualizar su progreso en un proyecto de software. Haciendo uso de la práctica CICS-RUP.
Producto de trabajo Actividad 1:
Tarea 1: Escribir plan de control de cambio. Tarea 2: Establecer proceso de control de cambio.
Plan de cambio
Se debe modificar la configuración a los puertos establecidos para el nuevo proyecto, en donde se va a alojar la nueva aplicación. Identificar las peticiones existentes y modificar las necesarias para adaptarse a la nueva necesidad. Establecer nuevas vistas para el nuevo proyecto. Se debe implementar un mecanismo que permita identificar el progreso de la practica (CICS).

Fuente: Elaboración propia

Actividad 2:

Con el plan de cambio como insumo se puede iniciar la iteración como se indica en la figura 44, el resultado de la actividad se encuentra en la tabla 23.

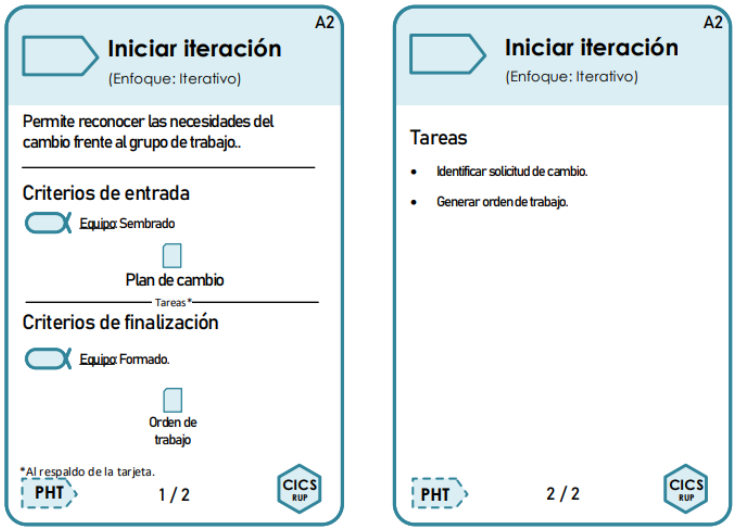


Fig. 46. Tarjeta de actividad Iniciar iteración

Fuente: Elaboración propia

TABLA XXIII

RESULTADO DE LA ACTIVIDAD INICIAR ITERACIÓN

Producto de trabajo de entrada Actividad 2:
Plan de cambio.
Se debe modificar la configuración a los puertos establecidos para el nuevo proyecto, en donde se va a alojar la nueva aplicación. Identificar las peticiones existentes y modificar las necesarias para adaptarse a la nueva necesidad. Establecer nuevas vistas para el nuevo proyecto. Se debe implementar un mecanismo que permita identificar el progreso de la practica (CICS).
Producto de trabajo de salida Actividad 2:
Tarea 1: Identificar solicitud de cambio. Tarea 2: Generar orden de trabajo.
Orden de trabajo
Modificar las interfaces utilizando las librerías existentes que son de carácter público (sin licenciamiento para su uso). Cambiar interfaz software hospitalario a uno que permita seguir la practica CICS-RUP. Adaptar controladores para realizar las nuevas peticiones necesarias para controlar los cambios.

Fuente: Elaboración propia

Actividad 3:

Para desarrollar el plan de iteración se obtiene la orden de trabajo, producto de trabajo de la actividad anterior, como se indica en la figura 46 y su resultado en la tabla XXIV.

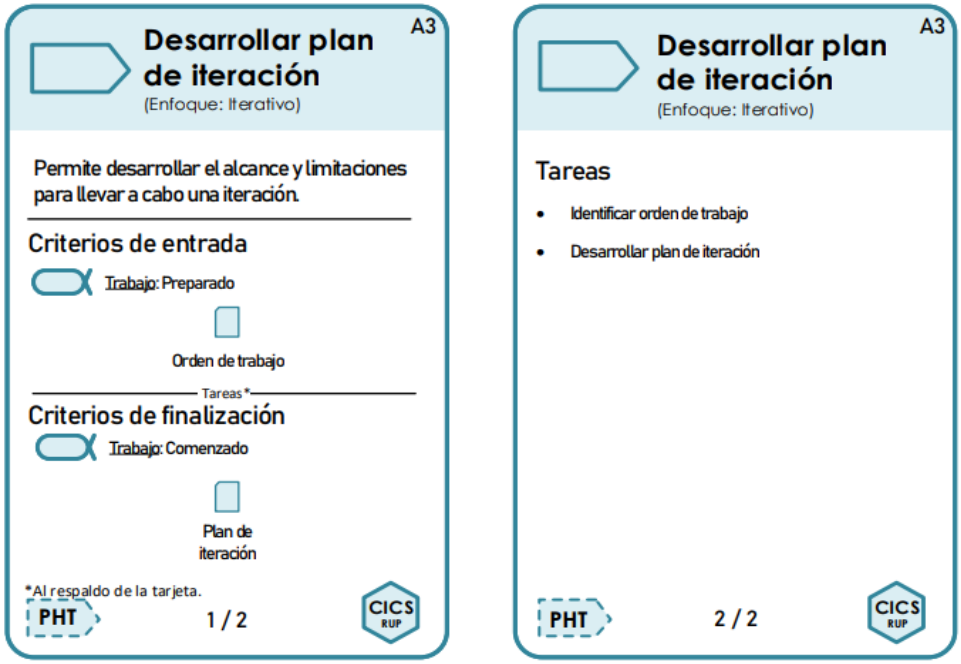


Fig. 47. Tarjeta de actividad Desarrollar plan de iteración

Fuente: Elaboración propia

TABLA XXIV

RESULTADO DE LA ACTIVIDAD DESARROLLAR PLAN DE ITERACIÓN

Producto de trabajo de entrada Actividad 3:
Orden de trabajo
Modificar las interfaces utilizando las librerías existentes que son de carácter público (sin licenciamiento para su uso). Cambiar interfaz software hospitalario a uno que permita seguir la practica CICS-RUP. Adaptar controladores para realizar las nuevas peticiones necesarias para controlar los cambios.

Producto de trabajo de salida Actividad 3:
Tarea 1: Identificar orden de trabajo. Tarea 2: Desarrollar plan de iteración.
Plan de iteración
El cambio de interfaz y controladores se realiza en una iteración con duración total de un mes (160 horas), se requiere de dos desarrolladores quienes deben contar con las herramientas necesarias para llevar a cabo el cambio. Se realizarán 3 reuniones (25 minutos) semanales durante la duración de la iteración en las cuales se debe comunicar el avance realizado con el fin de identificar posibles problemas y oportunidades. Para ello, se identificaron 3 vistas que se deben modificar y 6 nuevas. Así mismo, se identifica 3 peticiones que se pueden reutilizar, se deben crear 8 nuevos.

Fuente: Elaboración propia

Actividad 4:

La tabla XXV muestra el resultado de la actividad Desarrollar plan de medición, como indica la figura 47.

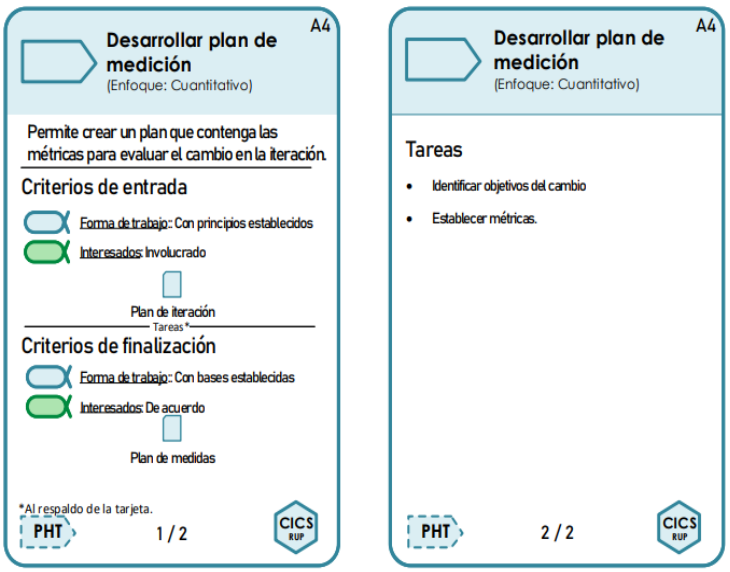


Fig. 47. Tarjeta de actividad Desarrollar plan de medición

Fuente: Elaboración propia

TABLA XXV

RESULTADO DE LA ACTIVIDAD DESARROLLAR PLAN DE MEDICIÓN

Producto de trabajo de entrada Actividad 4:
Plan de iteración
El cambio de interfaz y controladores se realiza en una iteración con duración total de un mes (160 horas), se requiere de dos desarrolladores quienes deben contar con las herramientas necesarias para llevar a cabo el cambio. Se realizarán 3 reuniones (25 minutos) semanales durante la duración de la iteración en las cuales se debe comunicar el avance realizado con el fin de identificar posibles problemas y oportunidades. Para ello, se identificaron 3 vistas que se deben modificar y 6 nuevas. Así mismo, se identifica 3 peticiones que se pueden reutilizar, se deben crear 8 nuevas.
Producto de trabajo de salida Actividad 4:
Tarea 1: Identificar objetivos del cambio. Tarea 2: Establecer métricas.
Plan de medidas
En total se deben trabajar en 20 artefactos diferentes, se evaluará el progreso por medio de los artefactos terminados. Teniendo en cuenta la duración de la iteración se establece en promedio un tiempo de 8 horas para la finalización de cada artefacto

Fuente: Elaboración propia

Actividad 5:

En las tablas XXVI y XXVII se indican los resultados de la actividad Desarrollar plan de gestión de problemas y riesgos, como se indica en la figura 48.

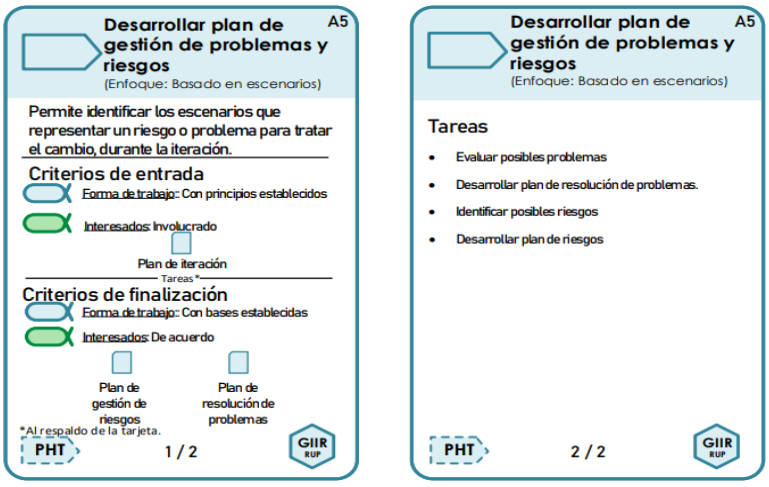


Fig. 48. Tarjeta de actividad Desarrollar plan de gestión de problemas y riesgos

Fuente: Elaboración propia

TABLA XXVI

RESULTADO DE LA ACTIVIDAD DESARROLLAR PLAN DE GESTIÓN DE PROBLEMAS Y RIESGOS PARTE (1)

Producto de trabajo de entrada Actividad 5:
Plan de iteración
El cambio de interfaz y controladores se realiza en una iteración con duración total de un mes (160 horas), se requiere de dos desarrolladores quienes deben contar con las herramientas necesarias para llevar a cabo el cambio. Se realizarán 3 reuniones (25 minutos) semanales durante la duración de la iteración en las cuales se debe comunicar el avance realizado con el fin de identificar posibles problemas y oportunidades. Para ello, se identificaron 3 vistas que se deben modificar y 6 nuevas. Así mismo, se identifica 3 peticiones que se pueden reutilizar, se deben crear 8 nuevos.
Producto de trabajo de salida Actividad 5:
Tarea 1: Evaluar posibles problemas. Tarea 2: Desarrollar plan de resolución de problemas. Tarea 3: Identificar posibles riesgos. Tarea 4: Desarrollar plan de riesgos.

Fuente: Elaboración propia

TABLA XXVII

RESULTADO DE LA ACTIVIDAD DESARROLLAR PLAN DE GESTIÓN DE PROBLEMAS Y RIESGOS PARTE (2)

Plan de gestión de riesgos
Al momento de realizar la planificación de gestión de riesgos, no se encuentra un riesgo que afecte la implementación del cambio solicitado. Por lo tanto, se sugiere que se cree un espacio dentro de las 3 reuniones semanales donde se comunicarán los riesgos inminentes encontrados durante el desarrollo, es deber del equipo determinar un plan de acción acorde para mitigarlo.
Plan de resolución de problemas
Al modificar un software existente se identifican posibles problemas de versión en las dependencias del proyecto. Antes de realizar cualquier modificación al software existente se debe actualizar todas sus dependencias a las más actuales.

Fuente: Elaboración propia

Actividad 6:

Como indica la figura 49, con el plan de iteración como insumo para la actividad Desarrollar plan de aceptación del producto, se obtiene el resultado de la tabla XXVIII.

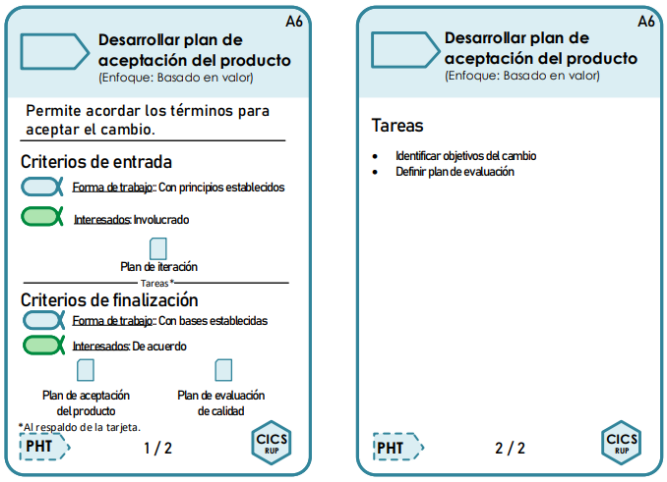


Fig. 49. Tarjeta de actividad Desarrollar plan de aceptación del producto

Fuente: Elaboración propia

TABLA XXVIII

PRODUCTO DE TRABAJO DE ENTRADA

Producto de trabajo de entrada Actividad 6:
Plan de iteración
El cambio de interfaz y controladores se realiza en una iteración con duración total de un mes (160 horas), se requiere de dos desarrolladores quienes deben contar con las herramientas necesarias para llevar a cabo el cambio. Se realizarán 3 reuniones (25 minutos) semanales durante la duración de la iteración en las cuales se debe comunicar el avance realizado con el fin de identificar posibles problemas y oportunidades. Para ello, se identificaron 3 vistas que se deben modificar y 6 nuevas. Así mismo, se identifica 3 peticiones que se pueden reutilizar, se deben crear 8 nuevos.

Producto de trabajo de salida Actividad 6:
Tarea 1: Identificar objetivos del cambio. Tarea 2: Definir plan de evaluación.
Plan de aceptación del producto
El producto será aceptado cuando los 20 artefactos sean finalizados e implementados en su totalidad.
Plan de evaluación de calidad
El software cumple con: ● Cumple con los requerimientos funcionales. ● El software es mantenible y escalable.

Fuente: Elaboración propia

Actividad 7:

En la tabla XXIX se encuentra el resultado de la actividad Definir monitoreo y procesos de control, como lo describe la figura 50.

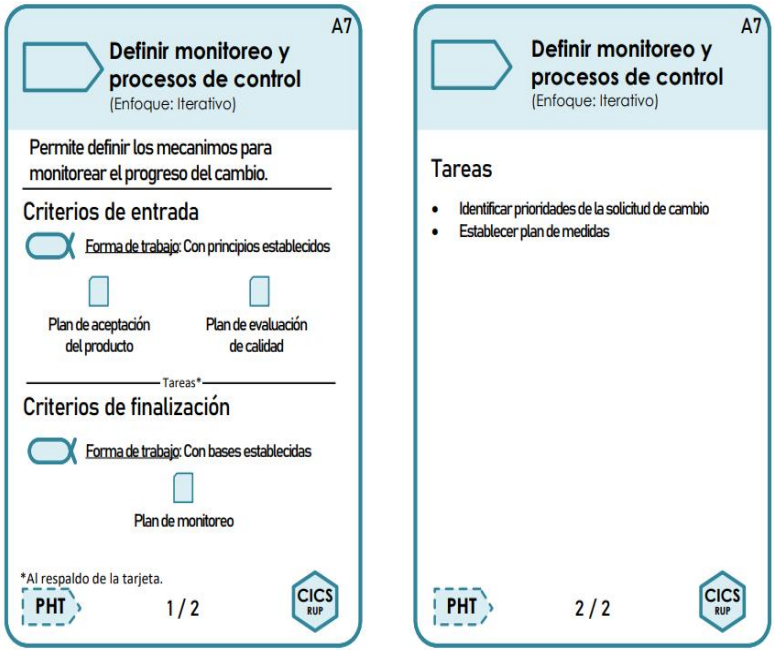


Fig. 50. Tarjeta de actividad Definir monitoreo y procesos de control

Fuente: Elaboración propia

TABLA XXIX

RESULTADO DE LA ACTIVIDAD DEFINIR MONITOREO Y PROCESOS DE CONTROL

Producto de trabajo de entrada Actividad 7:
Plan de aceptación del producto
El producto será aceptado cuando los 20 artefactos sean finalizados e implementados en su totalidad.
Plan de evaluación de calidad
El software cumple con: ● Cumple con los requerimientos funcionales. ● El software es mantenible y escalable.
Producto de trabajo de salida Actividad 7:
Tarea 1: Identificar prioridades de solicitud de cambio. Tarea 2: Evaluar plan de medidas.
Plan de monitoreo
Teniendo en cuenta que se parte de un software existente, se prioriza los artefactos nuevos que son necesarios para la implementación de la práctica CICS-RUP. Las revisiones se harán cada vez se termine un artefacto, además de la continua evaluación en las reuniones semanales.

Fuente: Elaboración propia

Actividad 8:

En la tabla XXX se encuentra el resultado de la actividad Definir grupo de trabajo y organización del proyecto, como lo describe la figura 51.

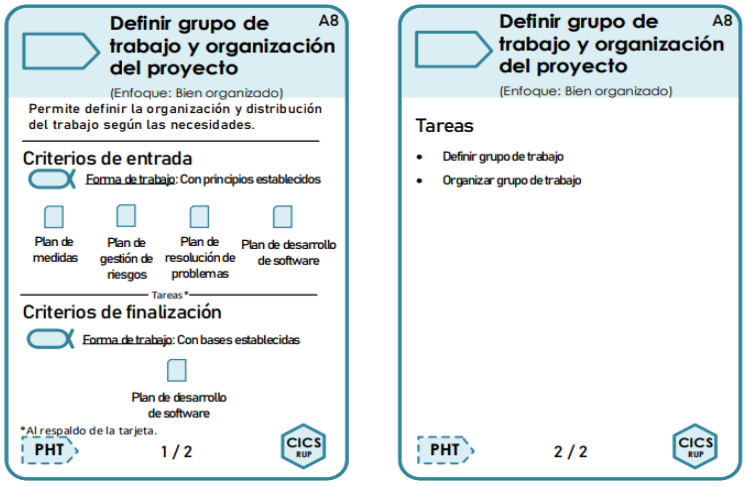


Fig. 51. Tarjeta de actividad Definir grupo de trabajo y organización del proyecto

Fuente: Elaboración propia

TABLA XXX

RESULTADO DE LA ACTIVIDAD DEFINIR GRUPO DE TRABAJO Y ORGANIZACIÓN DEL PROYECTO

Producto de trabajo de entrada Actividad 8:
Plan de medidas
En total se deben trabajar en 20 artefactos diferentes, se evaluará el progreso por medio de los artefactos terminados. Teniendo en cuenta la duración de la iteración se establece en promedio un tiempo de 8 horas para la finalización de cada artefacto
Plan de gestión de riesgos
Al momento de realizar la planificación de gestión de riesgos, no se encuentra un riesgo que afecte la implementación del cambio solicitado. Por lo tanto, se sugiere que se cree un espacio dentro de las 3 reuniones semanales donde se identificaran los riesgos inminentes, es deber del equipo determinar un plan de acción acorde para mitigarlo.
Plan de resolución de problemas
Al modificar un software existente se identifican posibles problemas de versión en las dependencias del proyecto. Antes de realizar cualquier modificación al software existente se debe actualizar todas sus dependencias a las más actuales.
Plan de monitoreo
Teniendo en cuenta que se parte de un software existente, se prioriza los artefactos nuevos que son necesarios para la implementación de la práctica CICS-RUP. Las revisiones se harán cada vez se termine un artefacto, además de la continua evaluación en las reuniones semanales.
Producto de trabajo de salida Actividad 8:
Tarea 1: Definir grupo de trabajo. Tarea 2: Organizar grupos de trabajo.
Plan de desarrollo de software
Se requiere de dos desarrolladores, uno especializado en frontend y el otro en backend. Además de un líder de equipo.

Fuente: Elaboración propia

Actividad 9:

En la tabla XXXI se encuentra el resultado de la actividad Definir evaluación y trazabilidad necesaria, como lo describe la figura 52.

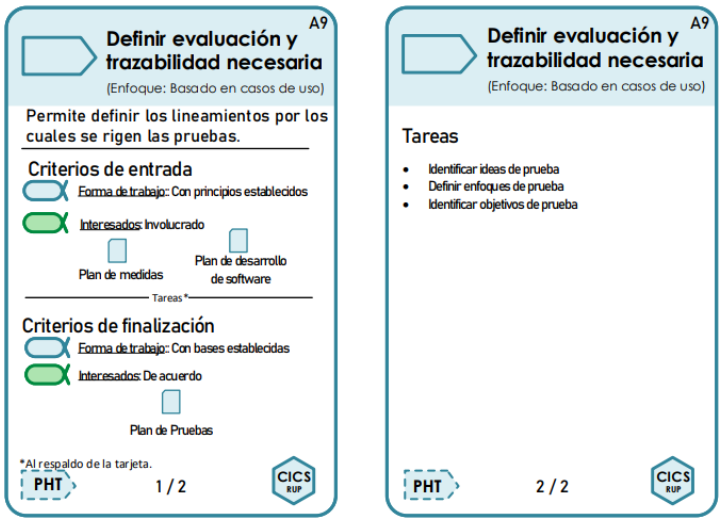


Fig. 52. Tarjeta de actividad Definir evaluación y trazabilidad necesaria

Fuente: Elaboración propia

TABLA XXXI

RESULTADO DE LA ACTIVIDAD DEFINIR EVALUACIÓN Y TRAZABILIDAD NECESARIA

Producto de trabajo de entrada Actividad 9:
Plan de medidas
En total se deben trabajar en 20 artefactos diferentes, se evaluará el progreso por medio de los artefactos terminados. Teniendo en cuenta la duración de la iteración se establece en promedio un tiempo de 8 horas para la finalización de cada artefacto
Plan de desarrollo de software
Se requiere de dos desarrolladores, uno especializado en frontend y el otro en backend. Además de un líder de equipo.
Producto de trabajo de salida Actividad 9:
Tarea 1: Identificar ideas de prueba. Tarea 2: Definir enfoques de prueba. Tarea 3: Identificar objetivos de prueba.
Plan de pruebas
Se debe probar funcionalmente todos los escenarios posibles que se hayan diseñado para cumplir con las funcionalidades requeridas.

Fuente: Elaboración propia

Actividad 10:

En las tablas XXXII y XXXIII se encuentra el resultado de la actividad Desarrollar caso de desarrollo, como lo describe la figura 53.

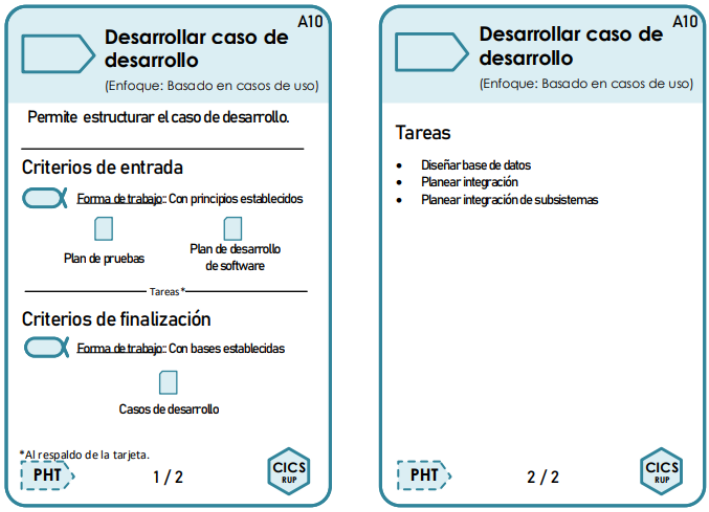


Fig. 53. Tarjeta de actividad Desarrollar caso de desarrollo

Fuente: Elaboración propia

TABLA XXXII

RESULTADO DE LA ACTIVIDAD DESARROLLAR CASO DE DESARROLLO (1)

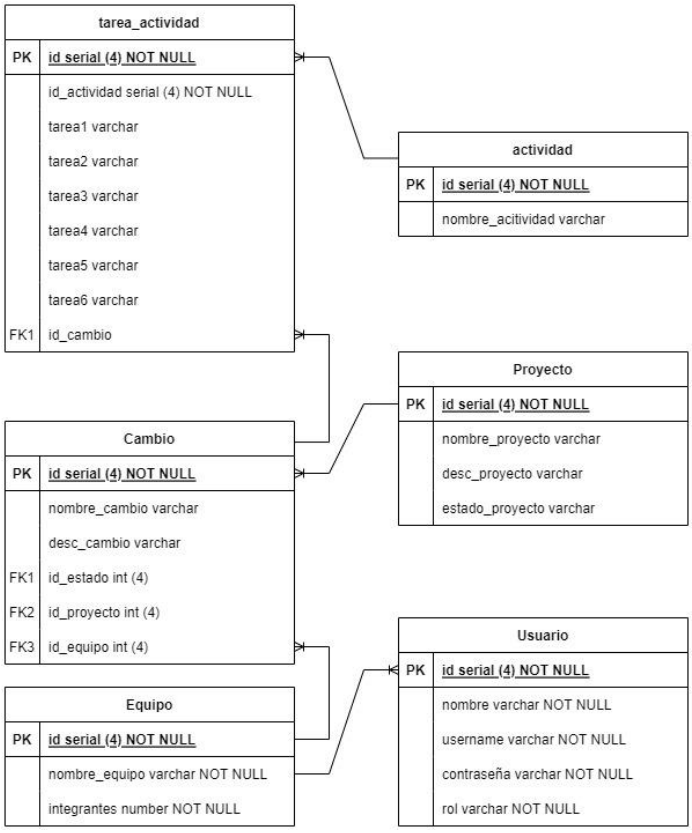
Producto de trabajo de entrada Actividad 10:
Plan de pruebas
Se debe probar funcionalmente todos los escenarios posibles que se hayan diseñado para cumplir con las funcionalidades requeridas.
Plan de desarrollo de software
Se requiere de dos desarrolladores, uno especializado en frontón y el otro en backend. Además de un líder de equipo.
Producto de trabajo de salida Actividad 10:
Tarea 1: Diseñar base de datos. Tarea 2: Planear integración. Tarea 3: Planear integración de subsistemas.

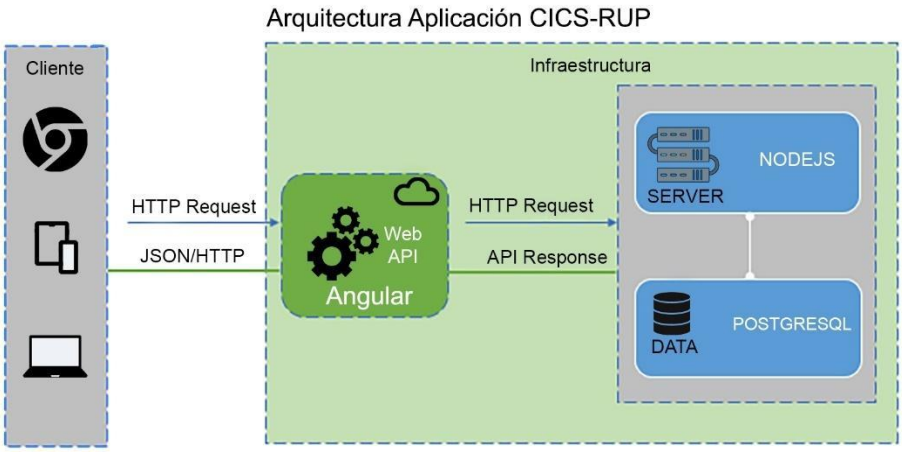
Fuente: Elaboración propia

TABLA XXXIII

RESULTADO DE LA ACTIVIDAD DESARROLLAR CASO DE DESARROLLO (2)

Casos de desarrollo
Base de datos:





Fuente: Elaboración propia

Actividad 11:

En la tabla XXXIV se encuentra el resultado de la actividad Desarrollar plan de despliegue, como lo describe la figura 54.

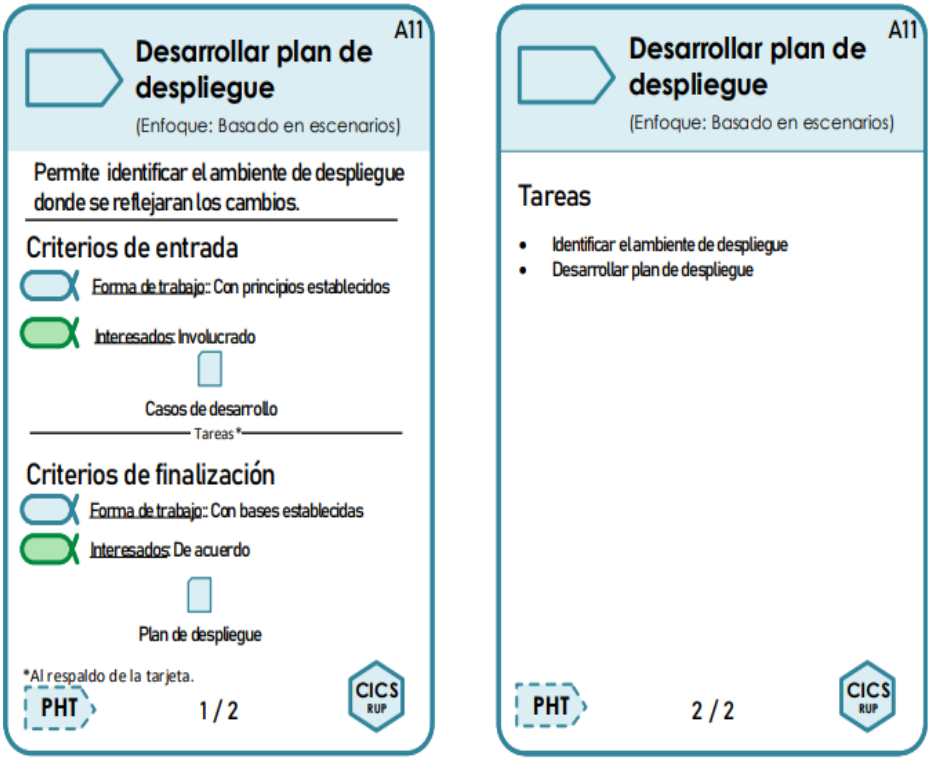


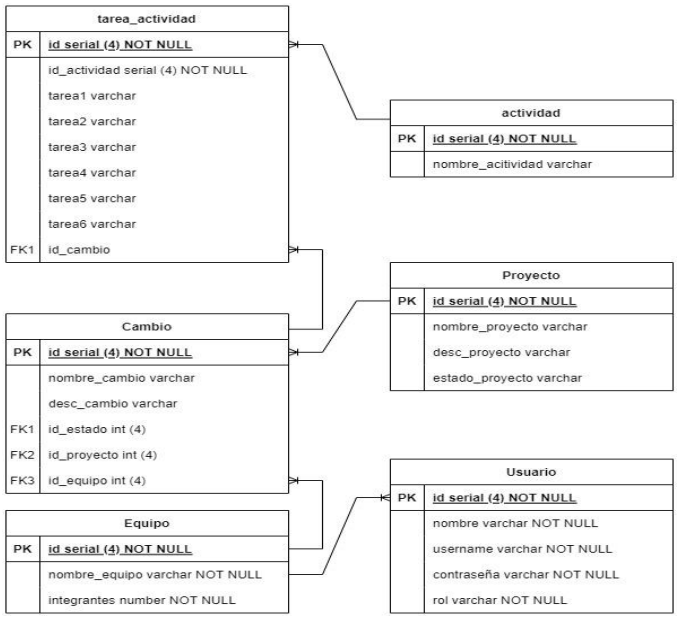
Fig. 54. Tarjeta de actividad Desarrollar plan de despliegue

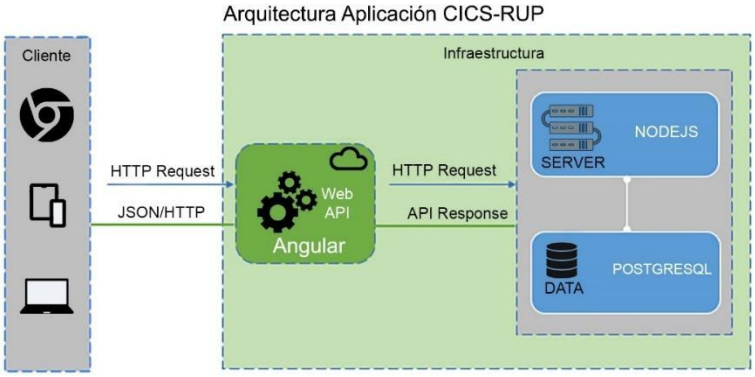
Fuente: Elaboración propia

TABLA XXXIV

RESULTADO DE LA ACTIVIDAD DESARROLLAR PLAN DE DESPLIEGUE

Producto de trabajo de entrada Actividad 11:
Casos de desarrollo
Base de datos:





Producto de trabajo de salida Actividad 11:
Tarea 1: Identificar el ambiente de despliegue.
Tarea 2: Desarrollar plan de despliegue.
Plan de despliegue
Con base a la finalidad del software final, se establece que el software será desplegado en las maquinas locales solamente.

Fuente: Elaboración propia

Actividad 12:

En la tabla XXXV se encuentra el resultado de la actividad Compilar plan de desarrollo, como lo describe la figura 55.

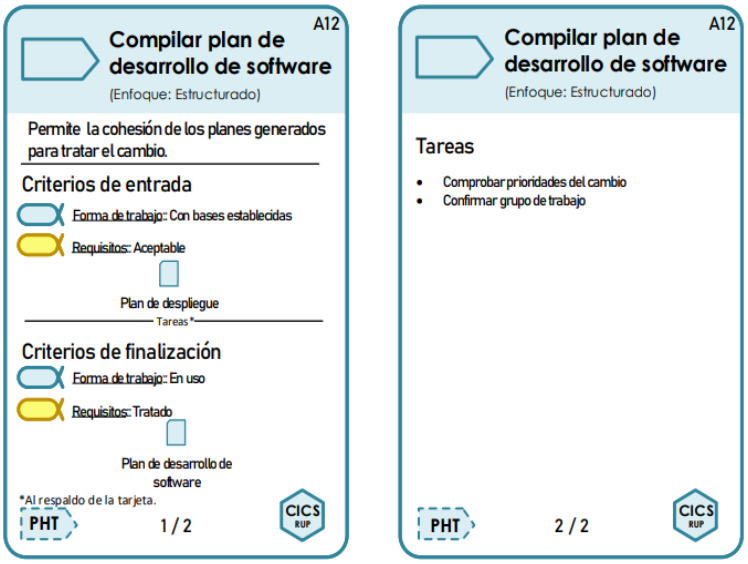


Fig. 55. Tarjeta de actividad Compilar plan de desarrollo

Fuente: Elaboración propia

TABLA XXXV

PRODUCTO DE TRABAJO DE ENTRADA

Producto de trabajo de entrada Actividad 12:
Plan de despliegue
Con base a la finalidad del software final, se establece que el software será desplegado en las maquinas locales solamente.
Producto de trabajo de salida Actividad 12:
Tarea 1: Comprobar prioridades del cambio. Tarea 2: Confirmar grupo de trabajo.
Plan de desarrollo de software
• Se requiere de dos desarrolladores, uno especializado en frontend y el otro en backend y un líder de equipo. • Se prioriza los artefactos necesarios para implementar la práctica CICS-RUP.

Fuente: Elaboración propia

Actividad 13:

En las tablas XXXVI y XXXVII se encuentra el resultado de la actividad Programar y asignar trabajo, como lo describe la figura 56.

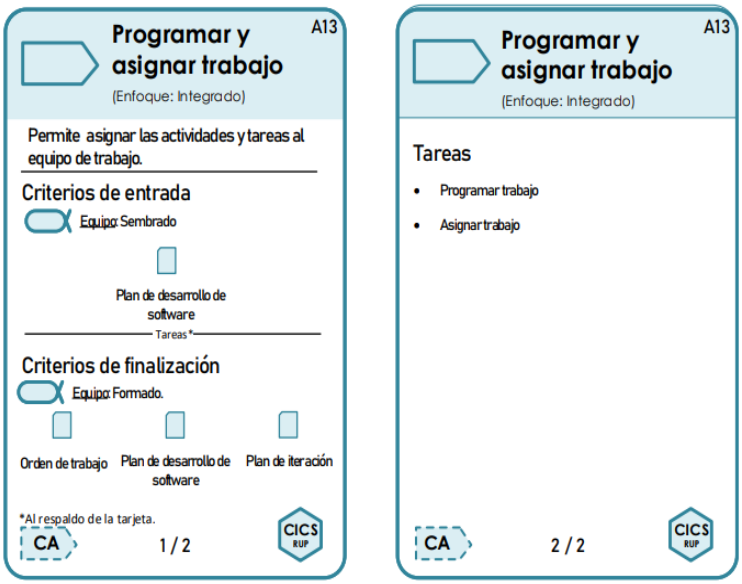


Fig. 56. Tarjeta de actividad Programar y asignar trabajo

Fuente: Elaboración propia

TABLA XXXVI

RESULTADO DE LA ACTIVIDAD PROGRAMAR Y ASIGNAR TRABAJO

Producto de trabajo de entrada Actividad 13:
Plan de desarrollo de software
• Se requiere de dos desarrolladores, uno especializado en frontend y el otro en backend y un líder de equipo. • Se prioriza los artefactos necesarios para implementar la práctica CICS-RUP.
Producto de trabajo de salida Actividad 13:
Tarea 1: Programar trabajo. Tarea 2: Asignar trabajo.

Fuente: Elaboración propia

TABLA XXXVII

RESULTADO DE LA ACTIVIDAD PROGRAMAR Y ASIGNAR TRABAJO (2)

Orden de trabajo

Numero orden	Cambio asociado
1	CICS-001
Encargado	
Especialidad	Jairo Arévalo
Desarrollador angular	
Entregables	
Artefacto	Tiempo estimado
Vista login	8 hrs
Vista de registro	8 hrs
Vista de dashboard	8 hrs
Vista de equipos	8 hrs
Vista de cambios	8 hrs
Vista del progreso de cambios	8 hrs
Vista del progreso de la practica	8 hrs
Vista de agregar cambio	8 hrs
Vista de tarjetas de cambio	8 hrs
Detalles	
Verificar la versión de dependencias	
Asistir a las reuniones de control	

Numero orden	Cambio asociado
2	CICS-001
Encargado	
Especialidad	Nicolas Barrios
Desarrollador Javascript	
Entregables	
Artefacto	Tiempo estimado
Controlador de autenticación	8 hrs
Controlador de registro	8 hrs
Controlador de página de cambios	8 hrs
Controlador de adicionar cambios	8 hrs
Controlador de página de practica	8 hrs
Controlador de manejo de equipos	8 hrs
Controlador de cambio	8 hrs
Detalles	
Verificar la versión de dependencias	
Asistir a las reuniones de control	

Numero orden	Cambio asociado
3	CICS-001
Encargado	
Especialidad	Jairo Arévalo
Líder de equipo	Nicolas Barrios
Entregables	
Artefacto	Tiempo estimado
N/A	N/A
Detalles	
Organizar equipo	
Liderar reuniones en búsqueda de problemas	

Plan de desarrollo de software

- Se requiere de dos desarrolladores, uno especializado en frontend y el otro en backend y un líder de equipo.
- Se prioriza los artefactos necesarios para implementar la práctica CICS-RUP.
- Al desarrollador frontend le corresponden los 9 artefactos de interfaz y el desarrollador backend de los 11 correspondientes a las peticiones. El líder mediará las reuniones semanales y estará encargado de guiar a los desarrolladores durante todo el proceso.

Plan de iteración

El cambio de interfaz y controladores se realiza en una iteración con duración total de un mes (160 horas), se requiere de dos desarrolladores quienes deben contar con las herramientas necesarias para llevar a cabo el cambio. Se realizarán 3 reuniones (25 minutos) semanales durante la duración de la iteración en las cuales se debe comunicar el avance realizado con el fin de identificar posibles problemas y oportunidades. Para ello, se identificaron 3 vistas que se deben modificar y 6 nuevas. Así mismo, se identifica 3 peticiones que se pueden reutilizar, se deben crear 8 nuevos.

Fuente: Elaboración propia

Actividad 14:

En las tablas XXXVIII y XXXIX se encuentra el resultado de la actividad Manejar excepciones y problemas, como lo describe la figura 57.

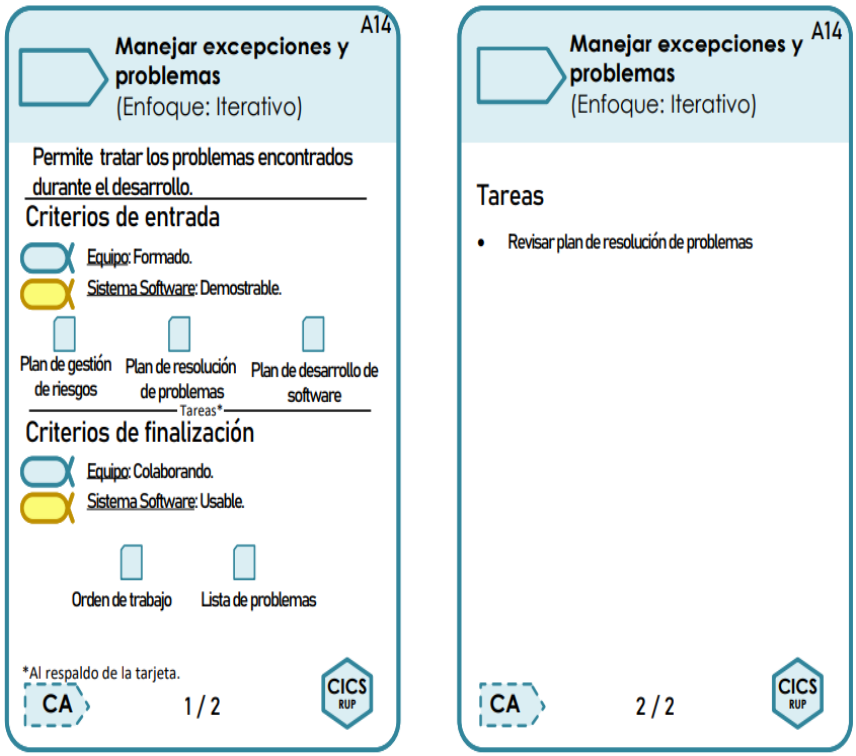


Fig. 57. Tarjeta de actividad Manejar excepciones y problemas

Fuente: Elaboración propia

TABLA XXXVIII

RESULTADO DE LA ACTIVIDAD MANEJAR EXCEPCIONES Y PROBLEMAS (1)

Producto de trabajo de entrada Actividad 14:
Plan de gestión de riesgos
Al momento de realizar la planificación de gestión de riesgos, no se encuentra un riesgo que afecte la implementación del cambio solicitado. Por lo tanto, se sugiere que se cree un espacio dentro de las 3 reuniones semanales donde se identificaran los riesgos inminente, es deber del equipo determinar un plan de acción acorde para mitigarlo.

Producto de trabajo de entrada Actividad 14:
Plan de resolución de problemas
Al modificar un software existente se identifican posibles problemas de versión en las dependencias del proyecto. Antes de realizar cualquier modificación al software existente se debe actualizar todas sus dependencias a las más actuales.
Plan de desarrollo de software
● Se requiere de dos desarrolladores, uno especializado en frontend y el otro en backend y un líder de equipo. ● Se prioriza los artefactos necesarios para implementar la práctica CICS-RUP. ● Al desarrollador frontend le corresponden los 9 artefactos de interfaz y el desarrollador backend de los 11 correspondientes a las peticiones. El líder mediará las reuniones semanales y estará encargado de guiar a los desarrolladores durante todo el proceso.

Fuente: Elaboración propia

TABLA XXXIX

RESULTADO DE LA ACTIVIDAD MANEJAR EXCEPCIONES Y PROBLEMAS (2)

Producto de trabajo de salida Actividad 14:
Tarea 1: Revisar plan de resolución de problemas.
Orden de trabajo

Numero orden	Cambio asociado
4	CICS-001
Encargado	
Especialidad	Jairo Arévalo
Desarrollador Angular	
Entregables	
Artefacto	Tiempo estimado
Notificación de verificación	4 hrs
Detalles	
Alta prioridad	

Numero orden	Cambio asociado
5	CICS-001
Encargado	
Especialidad	Nicolas Barrios
Desarrollador Javascript	
Entregables	
Artefacto	Tiempo estimado
Notificación de verificación	4 hrs
Detalles	
Alta prioridad	

Numero orden	Cambio asociado
6	CICS-001
Encargado	
Especialidad	Jairo Arévalo
Líder de equipo	Nicolas Barrios
Entregables	
Artefacto	Tiempo estimado
Estimado de esfuerzo de desarrollo	8 hrs

Producto de trabajo de entrada Actividad 14:
Lista de problemas
• La versión de las dependencias actualizadas no es retro compatible, se debe actualizar a la última que funcione para el proyecto. • Se identifica la necesidad de controlar los cambios de múltiples proyectos y visualizarlos de manera independiente.

Fuente: Elaboración propia

Actividad 15:

En las tablas XL y XLI se encuentra el resultado de la actividad Verificar la configuración de herramientas e instalación, como lo describe la figura 58.

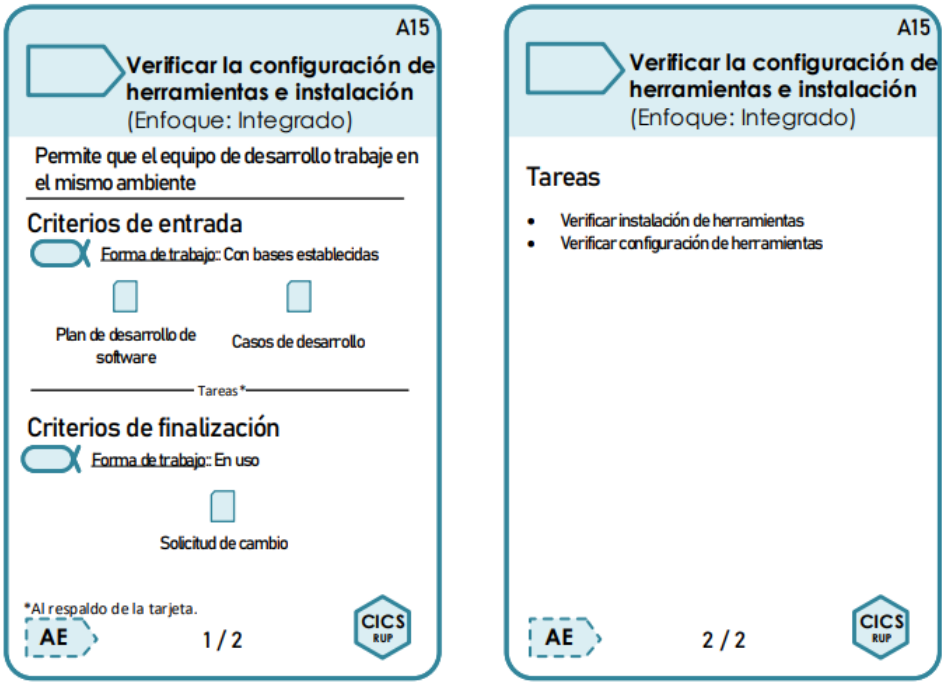


Fig. 58. Tarjeta de actividad Verificar la configuración de herramientas e instalación

Fuente: Elaboración propia

TABLA XL

RESULTADO DE LA ACTIVIDAD VERIFICAR LA CONFIGURACIÓN DE HERRAMIENTAS E INSTALACIÓN (1)

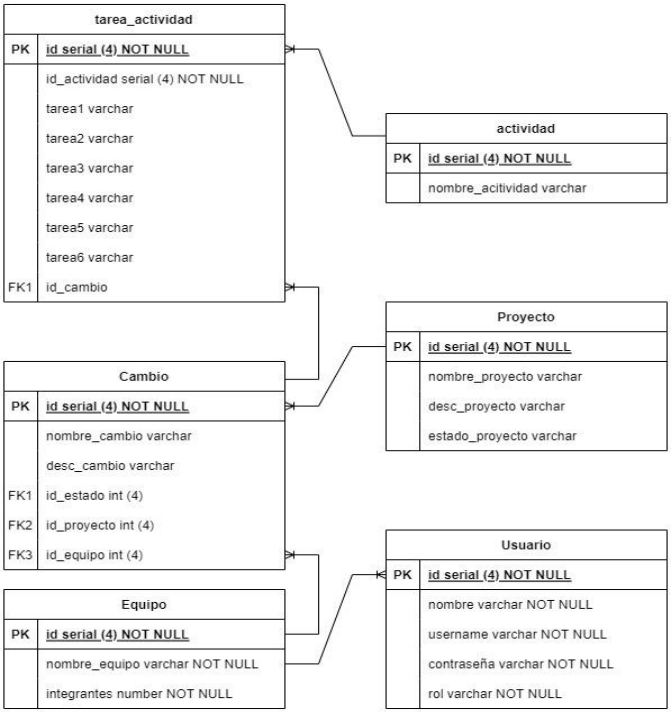
Producto de trabajo de entrada Actividad 15:
Plan de desarrollo de software

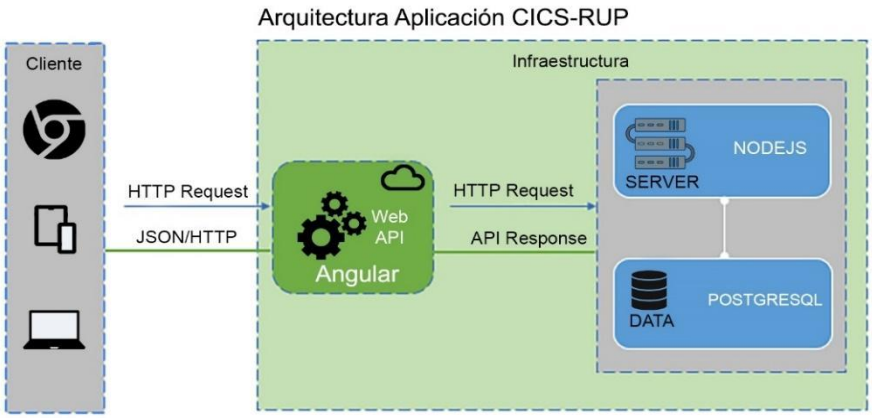
- Se requiere de dos desarrolladores, uno especializado en frontend y el otro en backend y un líder de equipo.
- Se prioriza los artefactos necesarios para implementar la práctica CICS-RUP.
- Al desarrollador frontend le corresponden los 9 artefactos de interfaz y el desarrollador backend de los 11 correspondientes a las peticiones. El líder mediará las reuniones semanales y estará encargado de guiar a los desarrolladores durante todo el proceso.

Fuente: Elaboración propia

TABLA XLI
RESULTADO DE LA ACTIVIDAD VERIFICAR LA CONFIGURACIÓN DE HERRAMIENTAS E INSTALACIÓN (2)

Casos de desarrollo
Base de datos:





Producto de trabajo de salida Actividad 15:
Tarea 1: Verificar instalación de herramientas.
Tarea 2: Verificar configuración de herramientas.
Solicitud de cambio
Los desarrolladores se encuentran trabajando en un ambiente estable. No hay cambio

Fuente: Elaboración propia

Actividad 16:

En las tablas XLII y XLIII se encuentra el resultado de la actividad Presentar o actualizar solicitud de cambio, como lo describe la figura 59.

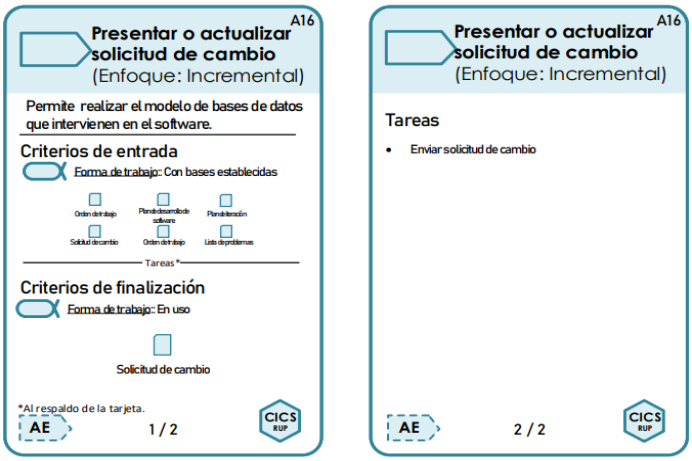


Fig. 59. Tarjeta de actividad Presentar o actualizar solicitud de cambio

Fuente: Elaboración propia

TABLA XLII

RESULTADO DE LA ACTIVIDAD PRESENTAR O ACTUALIZAR SOLICITUD DE CAMBIO (1)

Producto de trabajo de entrada Actividad 16:
Orden de trabajo

Numero orden	Cambio asociado
4	CICS-001
Encargado	
Especialidad	Jairo Arévalo
Desarrollador Angular	
Entregables	
Artefacto	Tiempo estimado
Notificación de verificación	4 hrs
Detalles	
Alta prioridad	

Numero orden	Cambio asociado
5	CICS-001
Encargado	
Especialidad	Nicolas Barrios
Desarrollador Javascript	
Entregables	
Artefacto	Tiempo estimado
Notificación de verificación	4 hrs
Detalles	
Alta prioridad	

Numero orden	Cambio asociado
6	CICS-001
Encargado	
Especialidad	Jairo Arévalo
Líder de equipo	Nicolas Barrios
Entregables	
Artefacto	Tiempo estimado
Estimado de esfuerzo de desarrollo	8 hrs

Producto de trabajo de entrada Actividad 16:
Plan de desarrollo de software
● Se requiere de dos desarrolladores, uno especializado en frontend y el otro en backend y un líder de equipo. ● Se prioriza los artefactos necesarios para implementar la práctica CICS-RUP. ● Al desarrollador frontend le corresponden los 9 artefactos de interfaz y el desarrollador backend de los 11 correspondientes a las peticiones. El líder mediará las reuniones semanales y estará encargado de guiar a los desarrolladores durante todo el proceso.
Plan de iteración
El cambio de interfaz y controladores se realiza en una iteración con duración total de un mes (160 horas), se requiere de dos desarrolladores quienes deben contar con las herramientas necesarias para llevar a cabo el cambio. Se realizarán 3 reuniones (25 minutos) semanales durante la duración de la iteración en las cuales se debe comunicar el avance realizado con el fin de identificar posibles problemas y oportunidades. Para ello, se identificaron 3 vistas que se deben modificar y 6 nuevas. Así mismo, se identifica 3 peticiones que se pueden reutilizar, se deben crear 8 nuevos.
Solicitud de cambio
CICS-001: Teniendo en cuenta el software existente que permite la creación de usuarios, hospitales y asignación de roles. Se requiere la adaptación de la interfaz y configuración del backend con el fin de controlar los cambios y visualizar su progreso en un proyecto de software. Haciendo uso de la práctica CICS-RUP
Lista de problemas
● La versión de las dependencias actualizadas no es retro compatible, se debe actualizar a la última que funcione para el proyecto. ● Se identifica la necesidad de controlar los cambios de múltiples proyectos y visualizarlos de manera independiente.

Fuente: Elaboración propia

TABLA XLIII

RESULTADO DE LA ACTIVIDAD PRESENTAR O ACTUALIZAR SOLICITUD DE CAMBIO (2)

Producto de trabajo de salida Actividad 16:
Tarea 1: Enviar solicitud de cambio.
Solicitud de cambio
CICS-002: Se identifica la necesidad de controlar los cambios de diferentes proyectos, es necesario acceder a múltiples proyectos, asignar cambios y usuarios a estos.

Fuente: Elaboración propia

Actividad 17:

En la tabla XLIV se encuentra el resultado de la actividad Gestionar pruebas de aceptabilidad, como lo describe la figura 60.

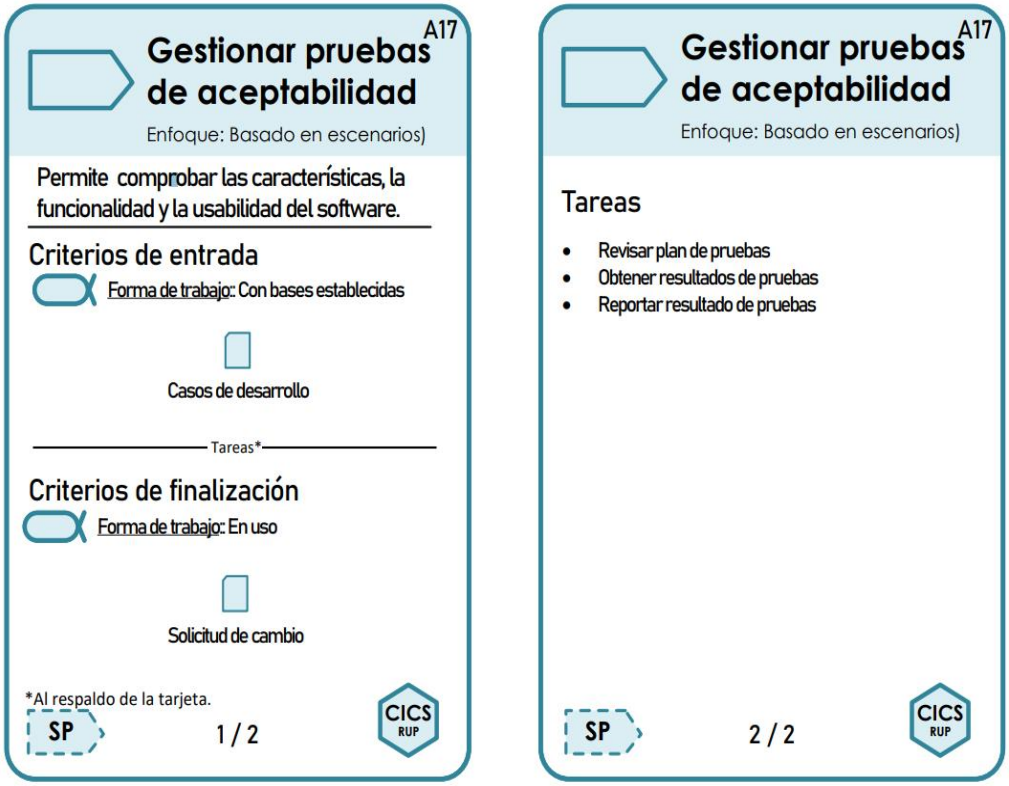


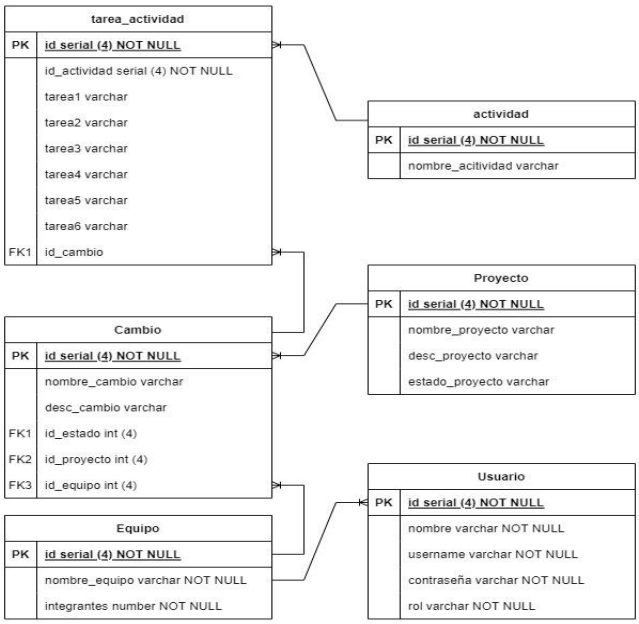
Fig. 60. Tarjeta de actividad Gestionar pruebas de aceptabilidad

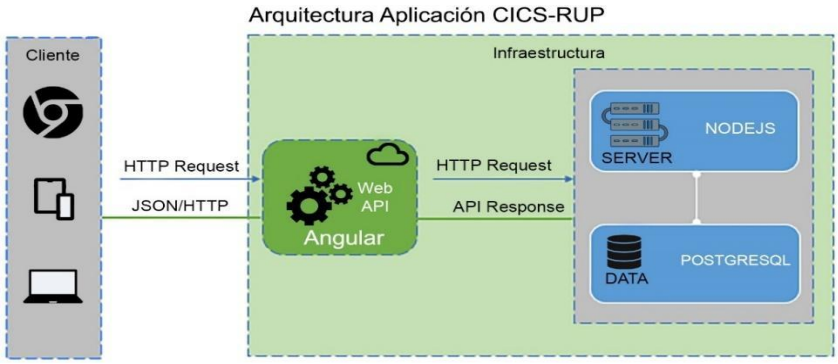
Fuente: Elaboración propia

TABLA XLIV

RESULTADO DE LA ACTIVIDAD GESTIONAR PRUEBAS DE ACEPTABILIDAD

Producto de trabajo de entrada Actividad 17:
Casos de desarrollo
Base de datos:





Producto de trabajo de salida Actividad 17:
Tarea 1: Revisar plan de pruebas.
Tarea 2: Obtener resultados de pruebas.
Tarea 3: Reportar resultado de pruebas.
Solicitud de cambio
No se identifican cambios

Fuente: Elaboración propia

Actividad 18:

En la tabla XLV se encuentra el resultado de la actividad Revisar aceptación del cambio, como lo describe la figura 61.

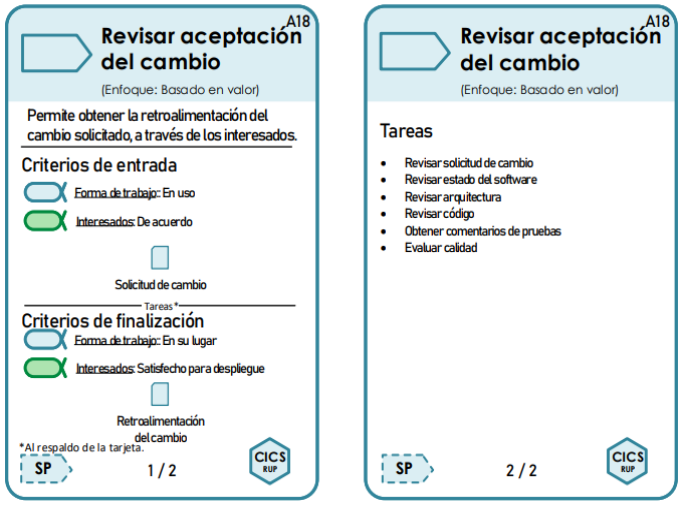


Fig. 61. Tarjeta de actividad Revisar aceptación del cambio

Fuente: Elaboración propia

TABLA XLV

RESULTADO DE LA ACTIVIDAD REVISAR ACEPTACIÓN DEL CAMBIO

Producto de trabajo de entrada Actividad 18:
Solicitud de cambio
CICS-001: Teniendo en cuenta el software existente que permite la creación de usuarios, hospitales y asignación de roles. Se requiere la adaptación de la interfaz y configuración del backend con el fin de controlar los cambios y visualizar su progreso en un proyecto de software. Haciendo uso de la práctica CICS-RUP
Producto de trabajo de salida Actividad 18:
Tarea 1: Revisar solicitud de cambio.
Tarea 2: Revisar estado del software.
Tarea 3: Revisar arquitectura.
Tarea 4: Revisar código.
Tarea 5: Obtener comentarios de pruebas.
Tarea 6: Evaluar calidad.

Producto de trabajo de salida Actividad 18:
Retroalimentación del cambio
• Software: Funcional. • Pruebas: Aprobadas satisfactoriamente. • Calidad: Aceptable. • Los problemas encontrados durante la ejecución del cambio se solucionaron de manera exitosa. • El cambio se ha realizado exitosamente, los artefactos fueron finalizados en su totalidad y entregados a tiempo.

Fuente: Elaboración propia

Actividad 19:

En la tabla XLVI se encuentra el resultado de la actividad Evaluar iteración, como lo describe la figura 62.

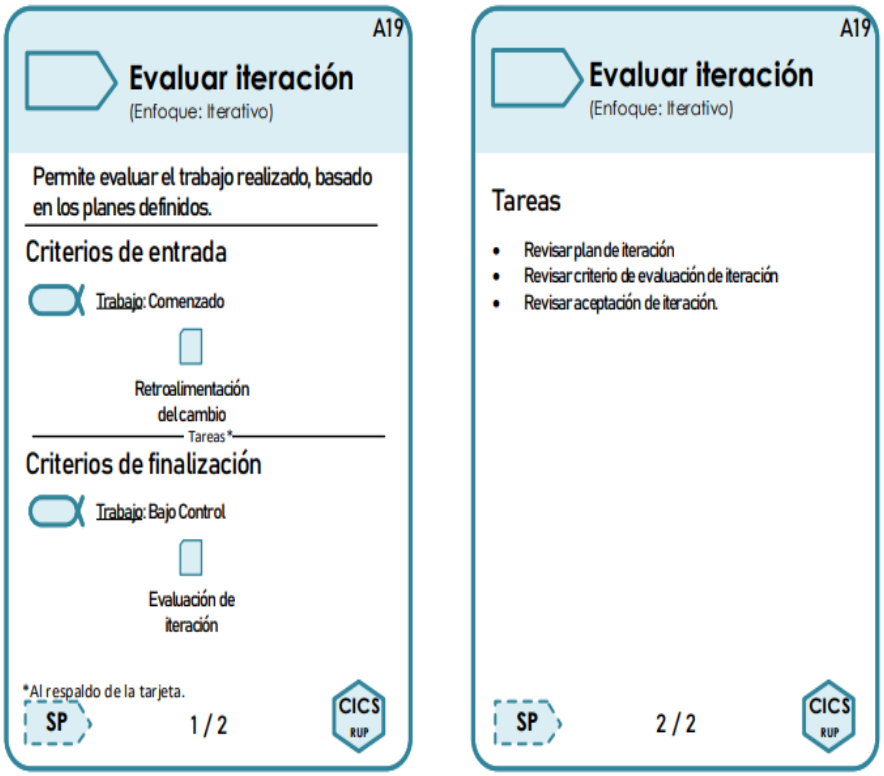


Fig. 62. Tarjeta de actividad Evaluar iteración

Fuente: Elaboración propia

TABLA XLVI

RESULTADO DE LA ACTIVIDAD EVALUAR ITERACIÓN

Producto de trabajo de entrada Actividad 19:
Retroalimentación del cambio
● Software: Funcional. ● Pruebas: Aprobadas satisfactoriamente. ● Calidad: Aceptable. ● Los problemas encontrados durante la ejecución del cambio se solucionaron de manera exitosa. ● El cambio se ha realizado exitosamente, los artefactos fueron finalizados en su totalidad y entregados a tiempo.
Producto de trabajo de salida Actividad 19:
Tarea 1: Revisar plan de iteración. Tarea 2: Revisar criterio de evaluación de iteración. Tarea 3: Revisar aceptación de iteración.
Evaluación de iteración
● Artefactos identificados/Artefactos entregados: 20/20. ● Tiempo estimado/Tiempo contabilizado: 160h/160h. ● Problemas encontrados/Problemas solucionados en la iteración: 2/2. ● Nuevos cambios encontrados/Cambios tratados: 1/0. Comentarios: Debido a la complejidad y estado del desarrollo, se decide continuar con el cambio CICS-002 en una siguiente iteración.

Fuente: Elaboración propia

Actividad 20:

En la tabla XLVII se encuentra el resultado de la actividad Finalizar iteración, como lo describe la figura 63.

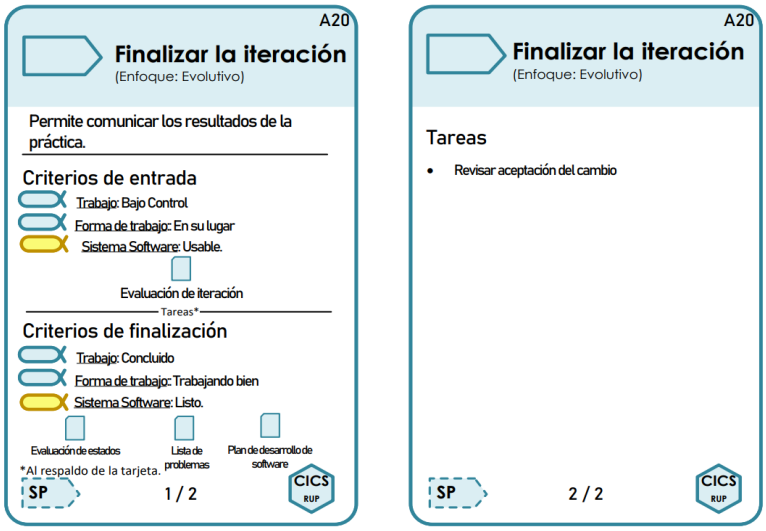


Fig. 63. Tarjeta de actividad Finalizar iteración

Fuente: Elaboración propia

TABLA XLVII

RESULTADO DE LA ACTIVIDAD FINALIZAR ITERACIÓN

Producto de trabajo de entrada Actividad 20:
Evaluación de iteración
• Artefactos identificados/Artefactos entregados: 20/20. • Tiempo estimado/Tiempo contabilizado: 160h/160h. • Problemas encontrados/Problemas solucionados en la iteración: 2/2. • Nuevos cambios encontrados/Cambios tratados: 1/0. Comentarios: Debido a la complejidad y estado del desarrollo, se decide continuar con el cambio CICS-002 en una siguiente iteración.
Producto de trabajo de salida Actividad 20:
Tarea 1: Programar trabajo. Tarea 2: Asignar trabajo.
Evaluación de estados
• Artefactos entregados 20 (100%) • Tiempo utilizado: 160hs (100%) • Estado final: Iteración exitosa.
Lista de problemas
• Nueva necesidad no estimada en el cambio inicial, se requiere de una nueva iteración para determinar su ejecución.

Fuente: Elaboración propia

Creación componentes de visualización de la práctica

Actividad 1:

En la tabla XLVIII se observa el producto de trabajo, además del resultado de la actividad 1 reflejada en la figura 64.

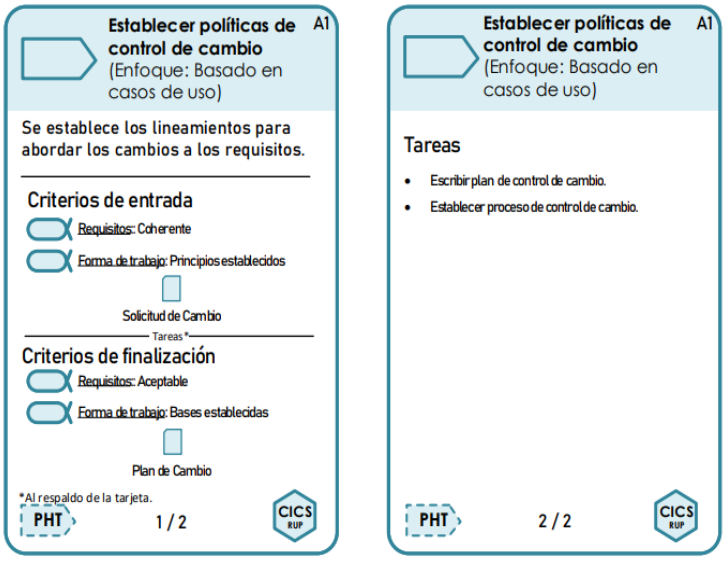


Fig. 64. Tarjeta de actividad Establecer políticas de control de cambio

Fuente: Elaboración propia

TABLA XLVIII

RESULTADOS DE LA ACTIVIDAD ESTABLECER POLÍTICAS DE CONTROL DE CAMBIO

Producto de entrada Actividad 1:
Solicitud de cambio.
CICS-002: Se identifica la necesidad de controlar los cambios de diferentes proyectos, es necesario acceder a múltiples proyectos, asignar cambios y usuarios a estos.
Producto de trabajo Actividad 1:
Tarea 1: Escribir plan de control de cambio. Tarea 2: Establecer proceso de control de cambio.
Plan de cambio
Se debe agregar los controladores y vistas necesarias para: ● Poder crear múltiples proyectos. ● Asignar varios cambios a múltiples. ● Asignar usuarios a más de un proyecto. ● Visualizar el progreso de los cambios en cada proyecto.

Fuente: Elaboración propia

Actividad 2:

Con el plan de cambio como insumo se puede iniciar la iteración como se indica en la figura 65, el resultado de la actividad se encuentra en la tabla XLIX.

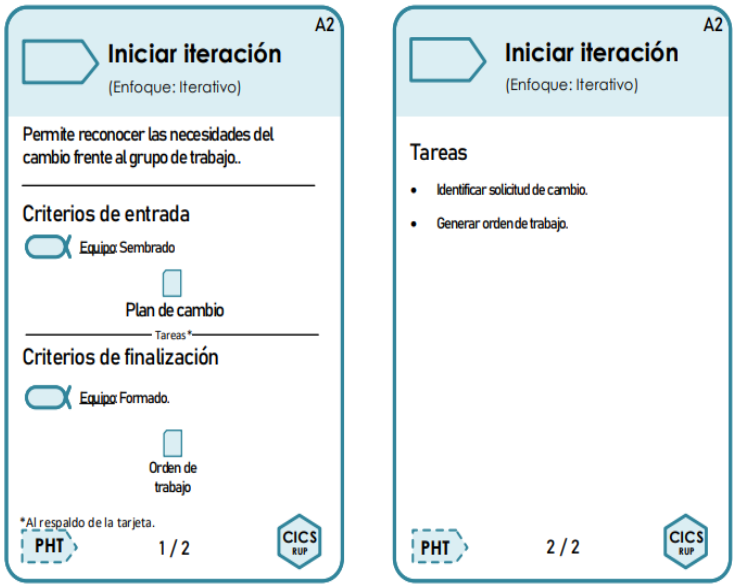


Fig. 65. Tarjeta de actividad Iniciar iteración

Fuente: Elaboración propia

TABLA XLIX

RESULTADO DE LA ACTIVIDAD INICIAR ITERACIÓN

Producto de trabajo de entrada Actividad 2:
Plan de cambio.
Se debe modificar la configuración a los puertos establecidos para el nuevo proyecto, en donde se va a alojar la nueva aplicación. Identificar las peticiones existentes y modificar las necesarias para adaptarse a la nueva necesidad. Establecer nuevas vistas para el nuevo proyecto. Se debe implementar un mecanismo que permita identificar el progreso de la practica (CICS).
Producto de trabajo de salida Actividad 2:
Tarea 1: Identificar solicitud de cambio. Tarea 2: Generar orden de trabajo.
Orden de trabajo
Crear vistas y controladores necesarios para que la aplicación pueda controlar los cambios de múltiples proyectos, independientemente.

Fuente: Elaboración propia

Actividad 3:

Para desarrollar el plan de iteración se obtiene la orden de trabajo, producto de trabajo de la actividad anterior, como se indica en la figura 66 y su resultado en la tabla L.

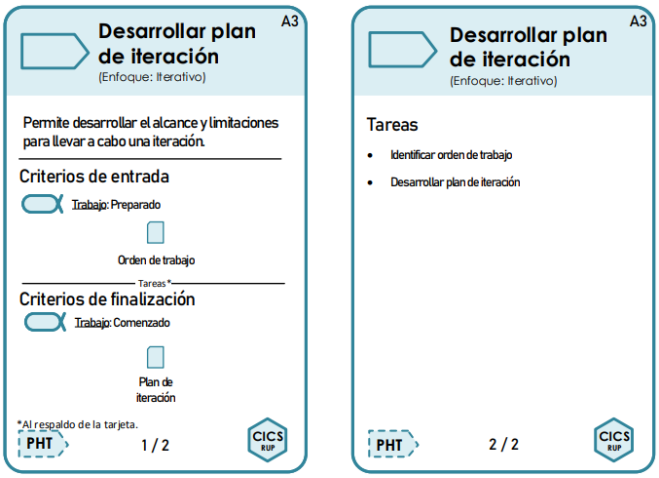


Fig. 66. Tarjeta de actividad Desarrollar plan de iteración

Fuente: Elaboración propia

TABLA L

RESULTADO DE LA ACTIVIDAD DESARROLLAR PLAN DE ITERACIÓN

Producto de trabajo de entrada Actividad 3:
Orden de trabajo
Modificar las interfaces utilizando las librerías existentes que son de carácter público (sin licenciamiento para su uso). Cambiar interfaz software hospitalario a uno que permita seguir la practica CICS-RUP. Adaptar controladores para realizar las nuevas peticiones necesarias para controlar los cambios.
Producto de trabajo de salida Actividad 3:
Tarea 1: Identificar orden de trabajo. Tarea 2: Desarrollar plan de iteración.
Plan de iteración
La creación de interfaz y controladores se realiza en una iteración con duración total de un mes (160 horas), se requiere de dos desarrolladores quienes deben contar con las herramientas necesarias para llevar a cabo el cambio. Se realizarán 3 reuniones (25 minutos) semanales durante la duración de la iteración en las cuales se debe comunicar el avance realizado con el fin de identificar posibles problemas y oportunidades. Para ello, se identificaron 4 vistas nuevas. Así mismo, 4 peticiones para suplir la nueva necesidad.

Fuente: Elaboración propia

Actividad 4:

La tabla LI muestra el resultado de la actividad Desarrollar plan de medición, como indica la figura 67.

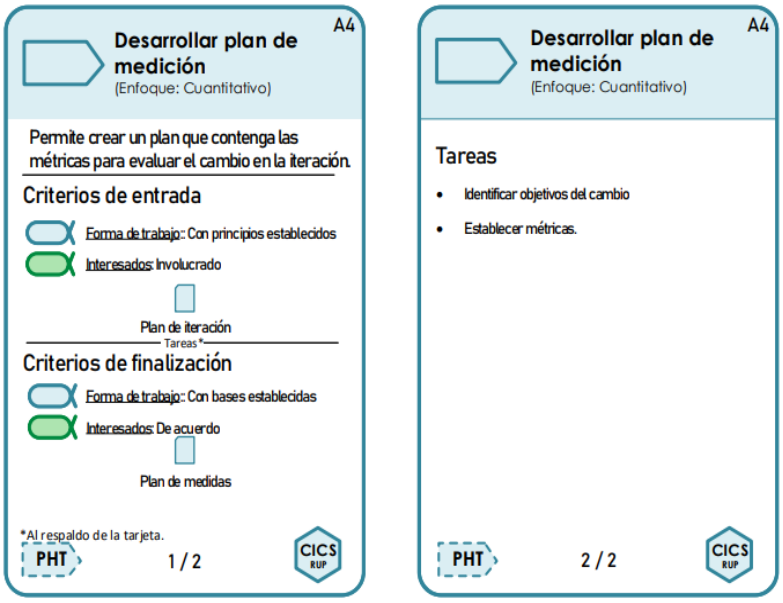


Fig. 67. Tarjeta de actividad Desarrollar plan de medición

Fuente: Elaboración propia

TABLA LI

RESULTADO DE LA ACTIVIDAD DESARROLLAR PLAN DE MEDICIÓN

Producto de trabajo de entrada Actividad 4:
Plan de iteración
El cambio de interfaz y controladores se realiza en una iteración con duración total de un mes (160 horas), se requiere de dos desarrolladores quienes deben contar con las herramientas necesarias para llevar a cabo el cambio. Se realizarán 3 reuniones (25 minutos) semanales durante la duración de la iteración en las cuales se debe comunicar el avance realizado con el fin de identificar posibles problemas y oportunidades. Para ello, se identificaron 3 vistas que se deben modificar y 6 nuevas. Así mismo, se identifica 3 peticiones que se pueden reutilizar, se deben crear 8 nuevas.

Producto de trabajo de salida Actividad 4:
Tarea 1: Identificar objetivos del cambio. Tarea 2: Establecer métricas.
Plan de medidas
En total se deben trabajar en 8 artefactos diferentes, se evaluará el progreso por medio de los artefactos terminados. Teniendo en cuenta la duración de la iteración se establece en promedio un tiempo de 20 horas para la finalización de cada artefacto. Teniendo en cuenta posibles problemas de compatibilidad con la lógica existente.

Fuente: Elaboración propia

Actividad 5:

En las tablas LII y LIII se indican los resultados de la actividad Desarrollar plan de gestión de problemas y riesgos, como se indica en la figura 68.

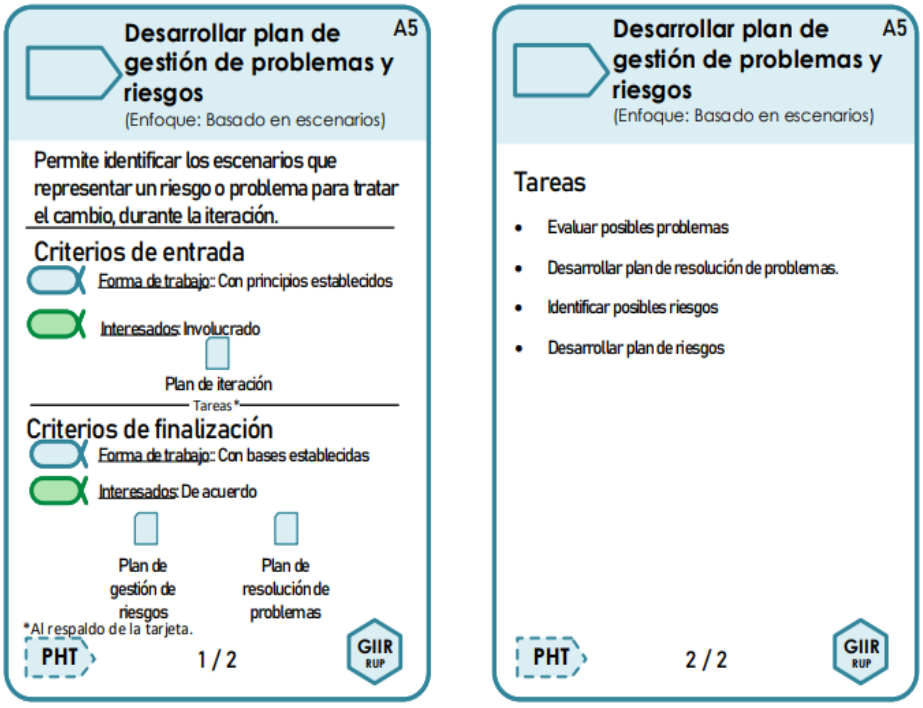


Fig. 68. Tarjeta de actividad Desarrollar plan de gestión de problemas y riesgos

Fuente: Elaboración propia

TABLA LII

RESULTADO DE LA ACTIVIDAD DESARROLLAR PLAN DE GESTIÓN DE PROBLEMAS Y RIESGOS (1)

Producto de trabajo de entrada Actividad 5:
Plan de iteración
La creación de interfaz y controladores se realiza en una iteración con duración total de un mes (160 horas), se requiere de dos desarrolladores quienes deben contar con las herramientas necesarias para llevar a cabo el cambio. Se realizarán 3 reuniones (25 minutos) semanales durante la duración de la iteración en las cuales se debe comunicar el avance realizado con el fin de identificar posibles problemas y oportunidades. Para ello, se identificaron 4 vistas nuevas. Así mismo, 4 peticiones para suplir la nueva necesidad.
Producto de trabajo de salida Actividad 5:
Tarea 1: Evaluar posibles problemas.
Tarea 2: Desarrollar plan de resolución de problemas.
Tarea 3: Identificar posibles riesgos.
Tarea 4: Desarrollar plan de riesgos.

Fuente: Elaboración propia

TABLA LIII

RESULTADO DE LA ACTIVIDAD DESARROLLAR PLAN DE GESTIÓN DE PROBLEMAS Y RIESGOS (2)

Plan de gestión de riesgos
Al momento de realizar la planificación de gestión de riesgos, no se encuentra un riesgo que afecte la implementación del cambio solicitado. Por lo tanto, se sugiere que se cree un espacio dentro de las 3 reuniones semanales donde se comunicarán los riesgos inminentes encontrados durante el desarrollo, es deber del equipo determinar un plan de acción acorde para mitigarlo.
Plan de resolución de problemas
Se debe asegurar la integración de la nueva lógica con la existente, a pesar de contar con un software escalable y mantenible, se debe verificar la funcionalidad de todo el software con cada cambio que se realice. Manteniendo así, la integridad del producto.

Fuente: Elaboración propia

Actividad 6:

Como indica la figura 69, con el plan de iteración como insumo para la actividad Desarrollar plan de aceptación del producto, se obtiene el resultado de la tabla LIV.

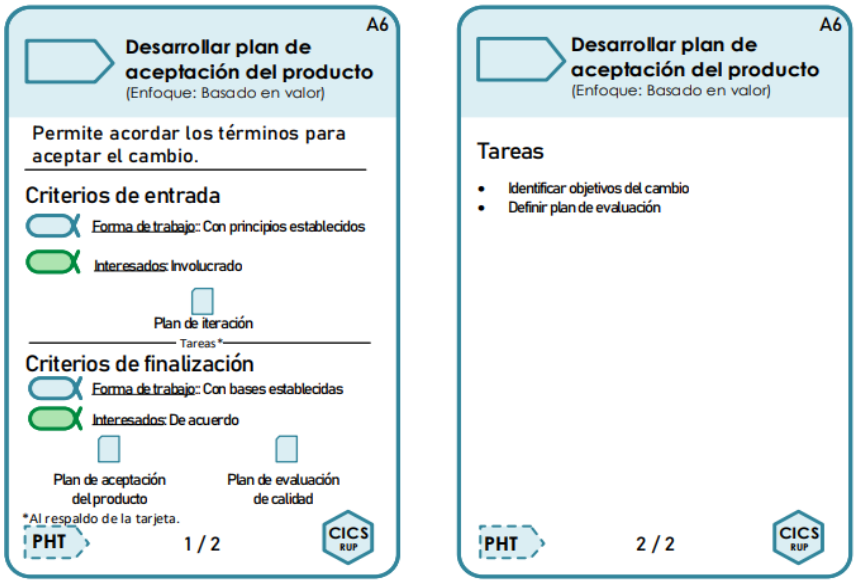


Fig. 69. Tarjeta de actividad Desarrollar plan de aceptación del producto
Fuente: Elaboración propia

TABLA LIV

RESULTADOS DE LA ACTIVIDAD DESARROLLAR PLAN DE ACEPTACIÓN DEL PRODUCTO

Producto de trabajo de entrada Actividad 6:
Plan de iteración
La creación de interfaz y controladores se realiza en una iteración con duración total de un mes (160 horas), se requiere de dos desarrolladores quienes deben contar con las herramientas necesarias para llevar a cabo el cambio. Se realizarán 3 reuniones (25 minutos) semanales durante la duración de la iteración en las cuales se debe comunicar el avance realizado con el fin de identificar posibles problemas y oportunidades. Para ello, se identificaron 4 vistas nuevas. Así mismo, 4 peticiones para suplir la nueva necesidad.
Producto de trabajo de salida Actividad 6:
Tarea 1: Identificar objetivos del cambio. Tarea 2: Definir plan de evaluación.
Plan de aceptación del producto
El producto será aceptado cuando los 8 artefactos sean finalizados e implementados en su totalidad.
Plan de evaluación de calidad
El software cumple con: • Cumple con los requerimientos funcionales. • El software es mantenible y escalable.

Fuente: Elaboración propia

Actividad 7:

En la tabla LV se encuentra el resultado de la actividad Definir monitoreo y procesos de control, como lo describe la figura 70.

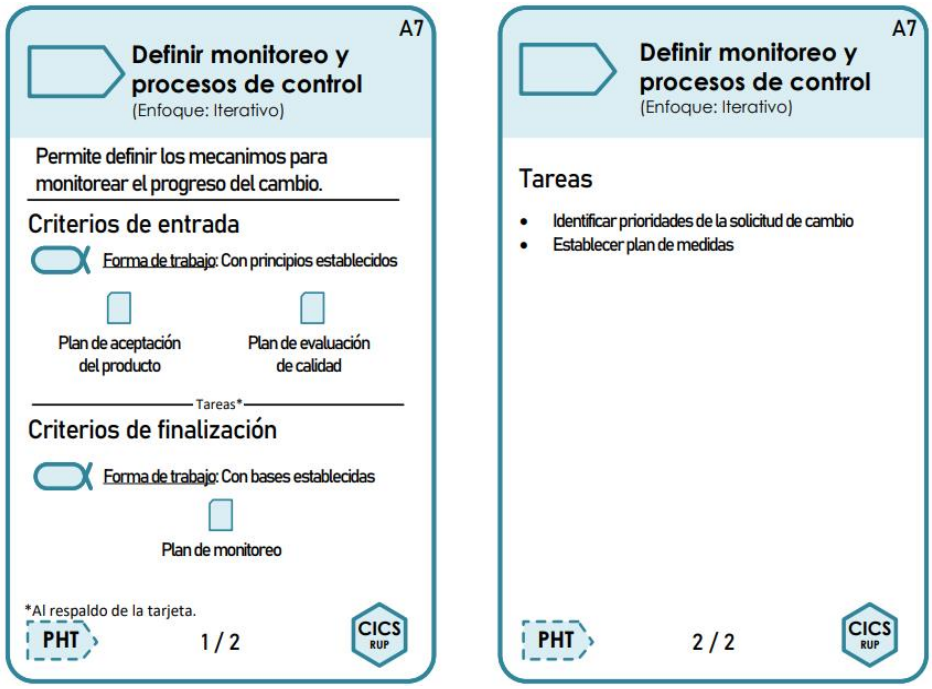


Fig. 70. Tarjeta de actividad Definir monitoreo y procesos de control

Fuente: Elaboración propia

TABLA LV

RESULTADO DE LA ACTIVIDAD DEFINIR MONITOREO Y PROCESOS DE CONTROL

Producto de trabajo de entrada Actividad 7:
Plan de aceptación del producto
El producto será aceptado cuando los 8 artefactos sean finalizados e implementados en su totalidad.
Plan de evaluación de calidad
El software cumple con: ● Cumple con los requerimientos funcionales. ● El software es mantenible y escalable.

Producto de trabajo de salida Actividad 7:
Tarea 1: Identificar prioridades de solicitud de cambio. Tarea 2: Evaluar plan de medidas.
Plan de monitoreo
Teniendo en cuenta que se debe acoplar la funcionalidad existente con la nueva, se hará un monitoreo a cada paso dado, cada cambio debe ser evaluado por todo el equipo, para verificar el funcionamiento de las funcionalidades nuevas, así como las existentes. Se debe informar cualquier problema en las reuniones pautadas.

Fuente: Elaboración propia

Actividad 8:

En la tabla LVI se encuentra el resultado de la actividad Definir grupo de trabajo y organización del proyecto, como lo describe la figura 71.

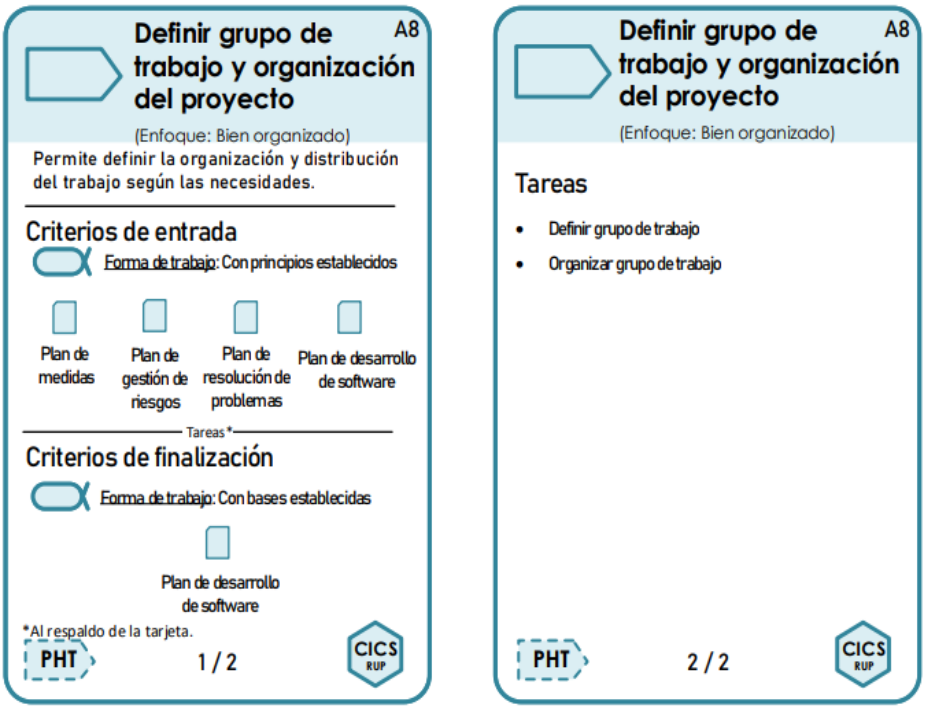


Fig. 71. Tarjeta de actividad Definir grupo de trabajo y organización del proyecto

Fuente: Elaboración propia

TABLA LVI

RESULTADO DE LA ACTIVIDAD DEFINIR GRUPO DE TRABAJO Y ORGANIZACIÓN DEL PROYECTO

Producto de trabajo de entrada Actividad 8:
Plan de medidas
En total se deben trabajar en 8 artefactos diferentes, se evaluará el progreso por medio de los artefactos terminados. Teniendo en cuenta la duración de la iteración se establece en promedio un tiempo de 20 horas para la finalización de cada artefacto. Teniendo en cuenta posibles problemas de compatibilidad con la lógica existente.
Plan de gestión de riesgos
Al momento de realizar la planificación de gestión de riesgos, no se encuentra un riesgo que afecte la implementación del cambio solicitado. Por lo tanto, se sugiere que se cree un espacio dentro de las 3 reuniones semanales donde se comunicarán los riesgos inminentes encontrados durante el desarrollo, es deber del equipo determinar un plan de acción acorde para mitigarlo.
Plan de resolución de problemas
Se debe asegurar la integración de la nueva lógica con la existente, a pesar de contar con un software escalable y mantenible, se debe verificar la funcionalidad de todo el software con cada cambio que se realice. Manteniendo así, la integridad del producto.
Plan de monitoreo
Teniendo en cuenta que se debe acoplar la funcionalidad existente con la nueva, se hará un monitoreo a cada paso dado, cada cambio debe ser evaluado por todo el equipo, para verificar el funcionamiento de las funcionalidades nuevas, así como las existentes. Se debe informar cualquier problema en las reuniones pautadas.
Producto de trabajo de salida Actividad 8:
Tarea 1: Definir grupo de trabajo. Tarea 2: Organizar grupos de trabajo.
Plan de desarrollo de software
Se requiere de dos desarrolladores, uno especializado en frontend y el otro en backend. Además de un líder de equipo.

Fuente: Elaboración propia

Actividad 9:

En la tabla LVII se encuentra el resultado de la actividad Definir evaluación y trazabilidad necesaria, como lo describe la figura 72.

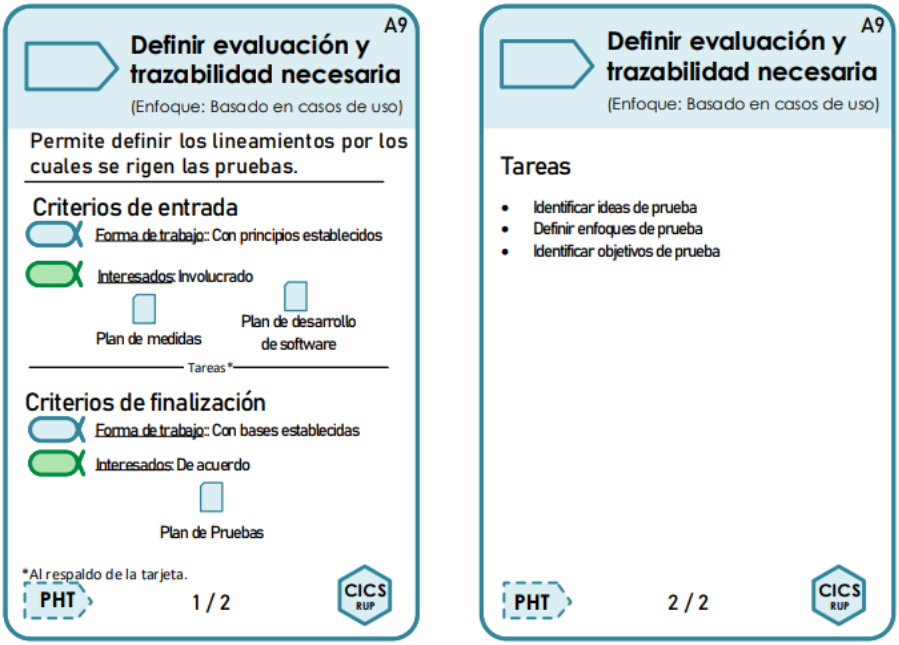


Fig. 72. Tarjeta de actividad Definir evaluación y trazabilidad necesaria

Fuente: Elaboración propia

TABLA LVII

RESULTADO DE LA ACTIVIDAD DEFINIR EVALUACIÓN Y TRAZABILIDAD NECESARIA

Producto de trabajo de entrada Actividad 9:
Plan de medidas
En total se deben trabajar en 8 artefactos diferentes, se evaluará el progreso por medio de los artefactos terminados. Teniendo en cuenta la duración de la iteración se establece en promedio un tiempo de 20 horas para la finalización de cada artefacto. Teniendo en cuenta posibles problemas de compatibilidad con la lógica existente.
Plan de desarrollo de software
Se requiere de dos desarrolladores, uno especializado en frontend y el otro en backend. Además de un líder de equipo.

Producto de trabajo de salida Actividad 9:
Tarea 1: Identificar ideas de prueba. Tarea 2: Definir enfoques de prueba. Tarea 3: Identificar objetivos de prueba.
Plan de pruebas
Se debe probar funcionalmente todos los escenarios posibles que se hayan diseñado para cumplir con las funcionalidades requeridas. Además de las funcionalidades existentes.

Fuente: Elaboración propia

Actividad 10:

En las tablas LVIII y LIX se encuentra el resultado de la actividad Desarrollar caso de desarrollo, como lo describe la figura 73.

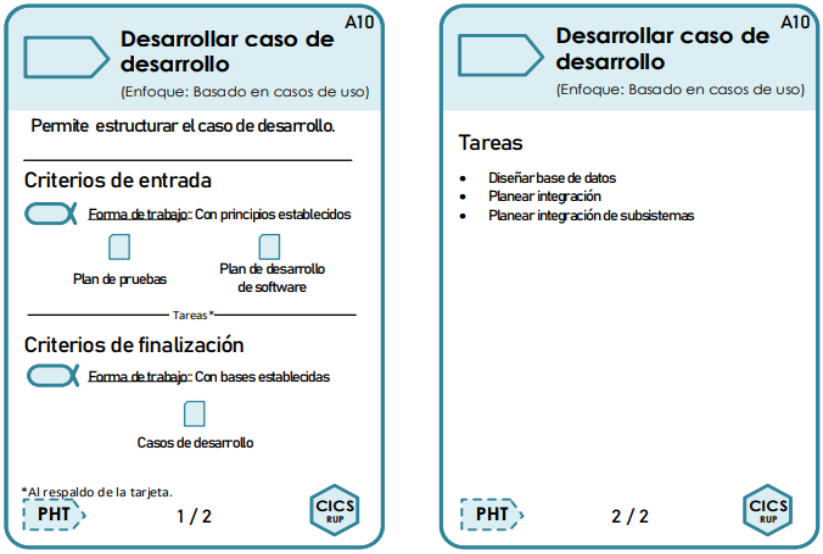


Fig. 73. Tarjeta de actividad Desarrollar caso de desarrollo

Fuente: Elaboración propia

TABLA LVIII

RESULTADO DE LA ACTIVIDAD DESARROLLAR CASO DE DESARROLLO (1)

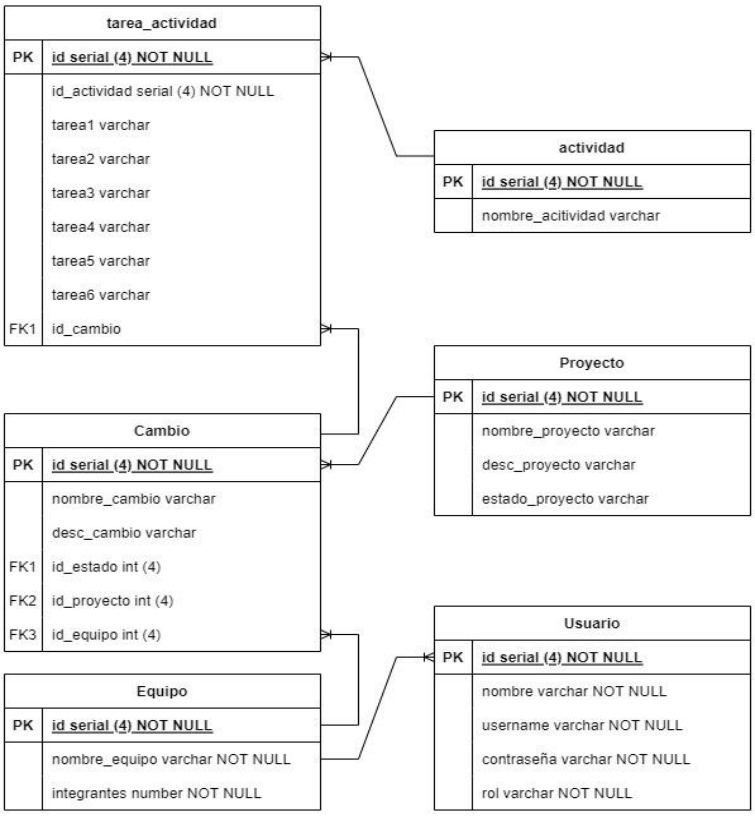
Producto de trabajo de entrada Actividad 10:
Plan de pruebas
Se debe probar funcionalmente todos los escenarios posibles que se hayan diseñado para cumplir con las funcionalidades requeridas. Además de las funcionalidades existentes.
Plan de desarrollo de software
Se requiere de dos desarrolladores, uno especializado en frontón y el otro en backend. Además de un líder de equipo.
Producto de trabajo de salida Actividad 10:
Tarea 1: Diseñar base de datos. Tarea 2: Planear integración. Tarea 3: Planear integración de subsistemas.

Fuente: Elaboración propia

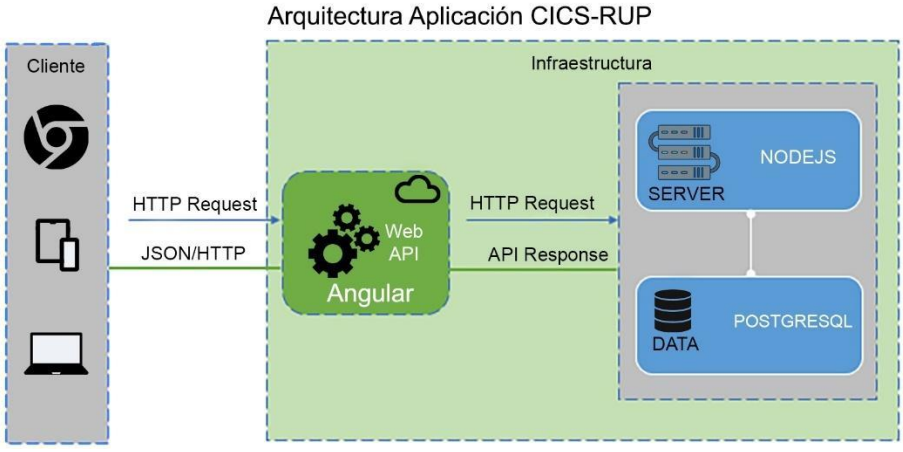
TABLA LIX

RESULTADO DE LA ACTIVIDAD DESARROLLAR CASO DE DESARROLLO (2)

Casos de desarrollo
Base de datos:



Casos de desarrollo



Fuente: Elaboración propia

Actividad 11:

En las tablas LI y VI-LXI se encuentra el resultado de la actividad Desarrollar plan de despliegue, como lo describe la figura 74.

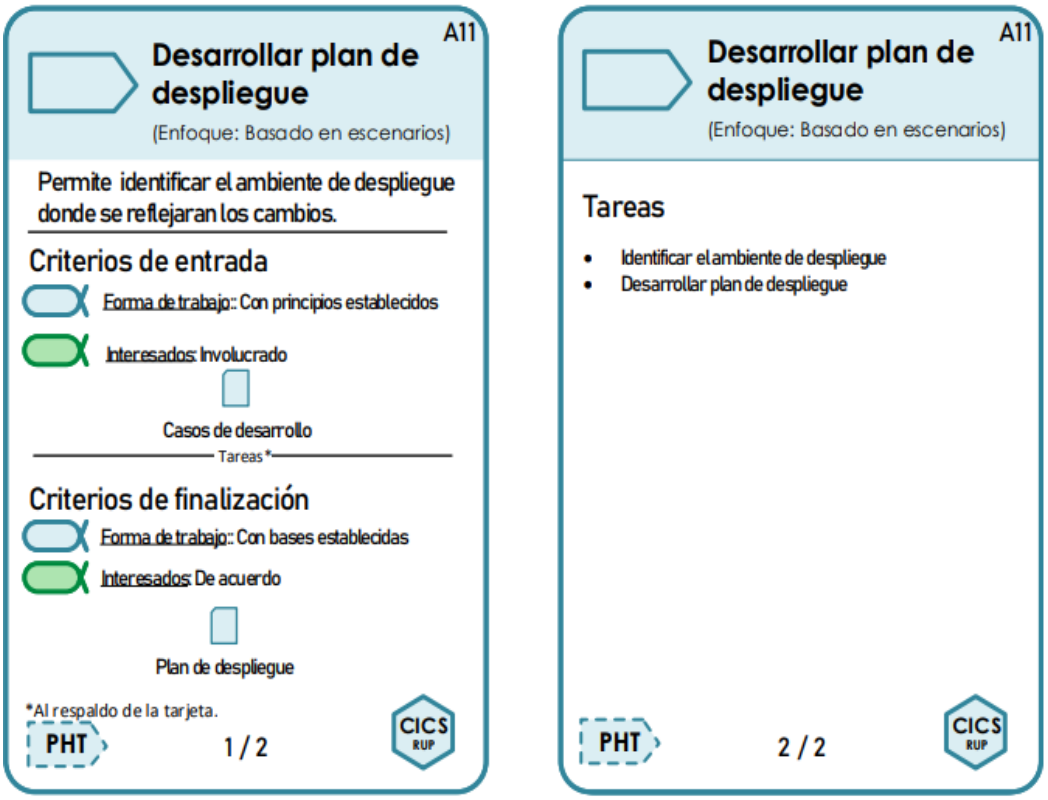


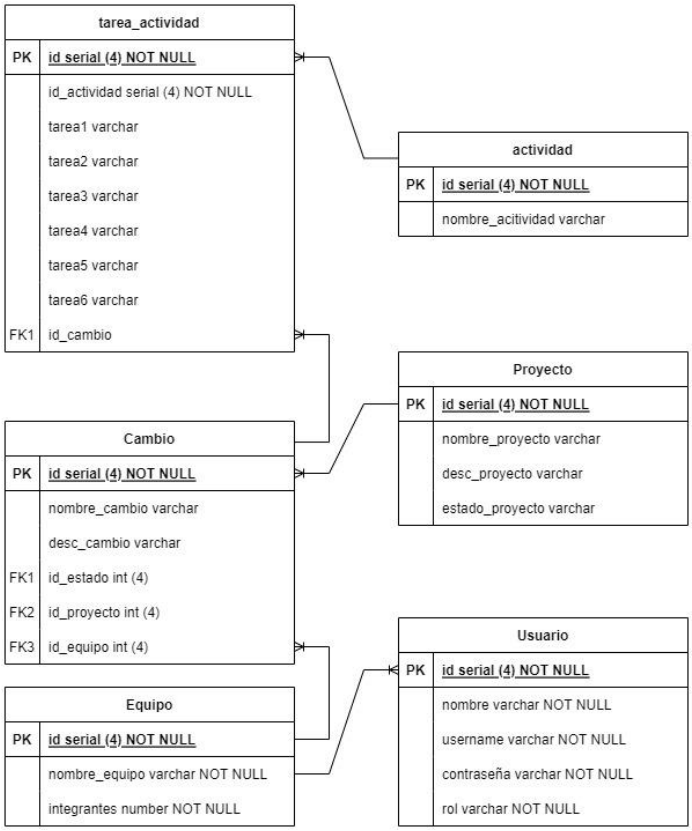
Fig. 74. Tarjeta de actividad Desarrollar plan de despliegue

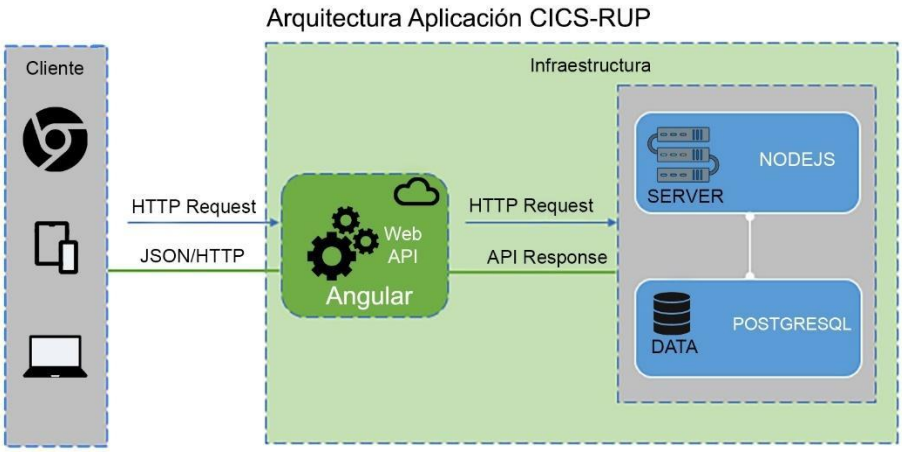
Fuente: Elaboración propia

TABLA LX

RESULTADO DE LA ACTIVIDAD DESARROLLAR PLAN DE DESPLIEGUE (1)

Producto de trabajo de entrada Actividad 11:
Casos de desarrollo
Base de datos:





Fuente: Elaboración propia

TABLA LXI

RESULTADO DE LA ACTIVIDAD DESARROLLAR PLAN DE DESPLIEGUE (2)

Producto de trabajo de salida Actividad 11:
Tarea 1: Identificar el ambiente de despliegue.
Tarea 2: Desarrollar plan de despliegue.
Plan de despliegue
Con base a la finalidad del software final, se establece que el software será desplegado en las maquinas locales solamente.

Fuente: Elaboración propia

Actividad 12:

En la tabla LXI se encuentra el resultado de la actividad Compilar plan de desarrollo, como lo describe la figura 75.

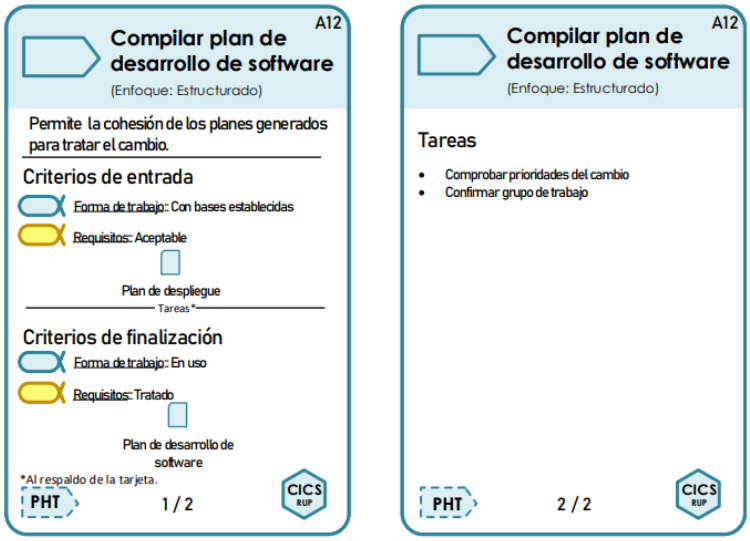


Fig. 75. Tarjeta de actividad Compilar plan de desarrollo

Fuente: Elaboración propia

TABLA LXII

PRODUCTO DE TRABAJO DE ENTRADA

Producto de trabajo de entrada Actividad 12:
Plan de despliegue
Con base a la finalidad del software final, se establece que el software será desplegado en las maquinas locales solamente.

Producto de trabajo de salida Actividad 12:
Tarea 1: Comprobar prioridades del cambio. Tarea 2: Confirmar grupo de trabajo.
Plan de desarrollo de software
• Se requiere de dos desarrolladores, uno especializado en frontend y el otro en backend y un líder de equipo. • Se prioriza la integración de la nueva lógica y estabilidad del sistema software, para aplicar la práctica CICS-RUP en múltiples proyectos.

Fuente: Elaboración propia

Actividad 13:

En las tablas LXIII y LXIV se encuentra el resultado de la actividad Programar y asignar trabajo, como lo describe la figura 76.

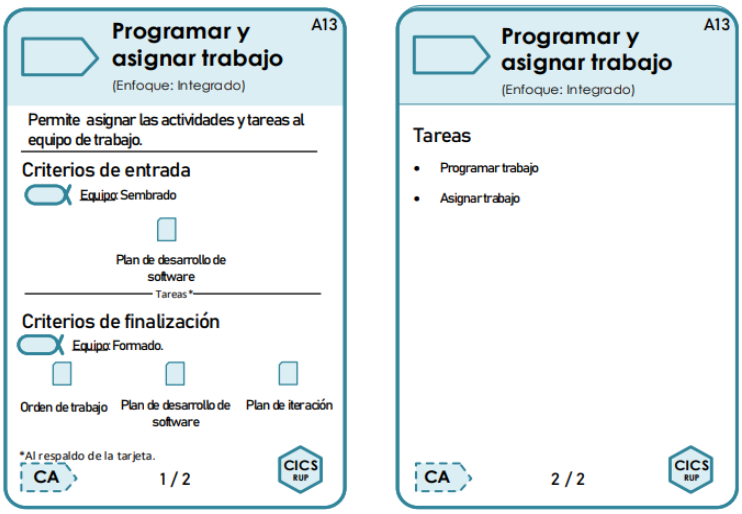


Fig. 76. Tarjeta de actividad Programar y asignar trabajo

Fuente: Elaboración propia

TABLA LXIII

RESULTADO DE LA ACTIVIDAD PROGRAMAR Y ASIGNAR TRABAJO (1)

Producto de trabajo de entrada Actividad 13:
Plan de desarrollo de software
• Se requiere de dos desarrolladores, uno especializado en frontend y el otro en backend y un líder de equipo. • Se prioriza los artefactos necesarios para implementar la práctica CICS-RUP.

Producto de trabajo de salida Actividad 13:

Tarea 1: Programar trabajo.

Tarea 2: Asignar trabajo.

Orden de trabajo

Numero orden	Cambio asociado
7	CICS-001
Encargado	
Especialidad	Jairo Arévalo
Desarrollador angular	
Entregables	
Artefacto	Tiempo estimado
Vista dashboard, nuevo elemento	8 hrs
Vista de página de proyectos	8 hrs
Vista de progreso del cambio del proyecto	8 hrs
Vista de actividad por proyecto	8 hrs
Detalles	
Verificar la versión de dependencias	
Asistir a las reuniones de control	
Verificar funcionalidad de cada cambio	

Numero orden	Cambio asociado
8	CICS-001
Encargado	
Especialidad	Nicolas Barrios
Desarrollador de Javascript	
Entregables	
Artefacto	Tiempo estimado
Controlador de dashboard	8 hrs
Controlador de proyectos	8 hrs
Controlador de progreso del cambio del proyecto	8 hrs
Controlador asignar usuarios a varios proyectos	8 hrs
Detalles	
Verificar la versión de dependencias	
Asistir a las reuniones de control	
Verificar funcionalidad de cada cambio	

Numero orden	Cambio asociado
9	CICS-001
Encargado	
Especialidad	Jairo Arévalo
Líder de equipo	Nicolas Barrios
Entregables	
Artefacto	Tiempo estimado
N/A	N/A
Detalles	
Organizar equipo	
Liderar reuniones en búsqueda de problemas	

Plan de desarrollo de software

- Se requiere de dos desarrolladores, uno especializado en frontend y el otro en backend y un líder de equipo.
- Se prioriza la integración de la nueva lógica y estabilidad del sistema software.
- Al desarrollador frontend le corresponden los 4 artefactos de interfaz y el desarrollador backend de los 4 correspondientes a las peticiones. El líder mediará las reuniones semanales y estará encargado de guiar a los desarrolladores durante todo el proceso.

Fuente: Elaboración propia

TABLA LXIV

RESULTADO DE LA ACTIVIDAD PROGRAMAR Y ASIGNAR TRABAJO (2)

Plan de iteración
La creación de interfaz y controladores se realiza en una iteración con duración total de un mes (160 horas), se requiere de dos desarrolladores quienes deben contar con las herramientas necesarias para llevar a cabo el cambio. Se realizarán 3 reuniones (25 minutos) semanales durante la duración de la iteración en las cuales se debe comunicar el avance realizado con el fin de identificar posibles problemas y oportunidades. Para ello, se identificaron 4 vistas nuevas. Así mismo, 4 peticiones para suplir la nueva necesidad.

Fuente: Elaboración propia

Actividad 14:

En la tabla LXV se encuentra el resultado de la actividad Manejar excepciones y problemas, como lo describe la figura 77.

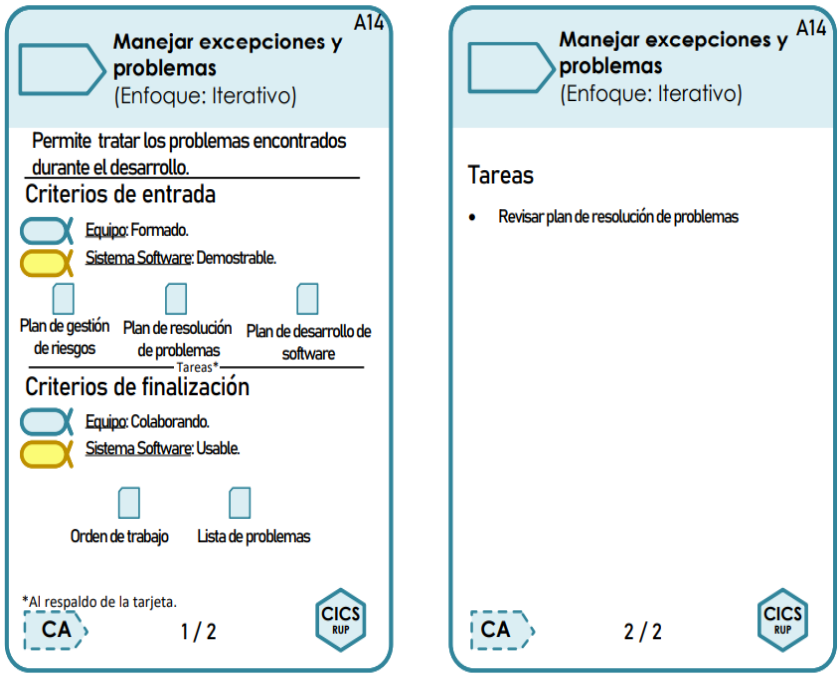


Fig. 77. Tarjeta de actividad Manejar excepciones y problemas

Fuente: Elaboración propia

TABLA LXV

RESULTADO DE LA ACTIVIDAD MANEJAR EXCEPCIONES Y PROBLEMAS

Producto de trabajo de entrada Actividad 14:
Plan de gestión de riesgos
Al momento de realizar la planificación de gestión de riesgos, no se encuentra un riesgo que afecte la implementación del cambio solicitado. Por lo tanto, se sugiere que se cree un espacio dentro de las 3 reuniones semanales donde se comunicarán los riesgos inminentes encontrados durante el desarrollo, es deber del equipo determinar un plan de acción acorde para mitigarlo.
Plan de resolución de problemas
Se debe asegurar la integración de la nueva lógica con la existente, a pesar de contar con un software escalable y mantenible, se debe verificar la funcionalidad de todo el software con cada cambio que se realice. Manteniendo así, la integridad del producto.
Plan de desarrollo de software
· Se requiere de dos desarrolladores, uno especializado en frontend y el otro en backend y un líder de equipo.
Producto de trabajo de salida Actividad 14:
Tarea 1: Revisar plan de resolución de problemas.
Orden de trabajo

Numero orden	Cambio asociado
7	CICS-001
Encargado	
Especialidad	Jairo Arévalo
Desarrollador angular	
Entregables	
Artefacto	Tiempo estimado
Vista dashboard, nuevo elemento	8 hrs
Vista de página de proyectos	8 hrs
Vista de progreso del cambio del proyecto	8 hrs
Vista de actividad por proyecto	8 hrs
Detalles	
Verificar la versión de dependencias	
Asistir a las reuniones de control	
Verificar funcionalidad de cada cambio	

Numero orden	Cambio asociado
8	CICS-001
Encargado	
Especialidad	Nicolas Barrios
Desarrollador de Javascript	
Entregables	
Artefacto	Tiempo estimado
Controlador de dashboard	8 hrs
Controlador de proyectos	8 hrs
Controlador de progreso del cambio del proyecto	8 hrs
Controlador asignar usuarios a varios proyectos	8 hrs
Detalles	
Verificar la versión de dependencias	
Asistir a las reuniones de control	
Verificar funcionalidad de cada cambio	

Numero orden	Cambio asociado
9	CICS-001
Encargado	
Especialidad	Jairo Arévalo
Líder de equipo	Nicolas Barrios
Entregables	
Artefacto	Tiempo estimado
N/A	N/A
Detalles	
Organizar equipo	
Liderar reuniones en búsqueda de problemas	

Lista de problemas
No se encontraron problemas.

Fuente: Elaboración propia

Actividad 15:

En las tablas LXVI y LXVII se encuentra el resultado de la actividad Verificar la configuración de herramientas e instalación, como lo describe la figura 78.

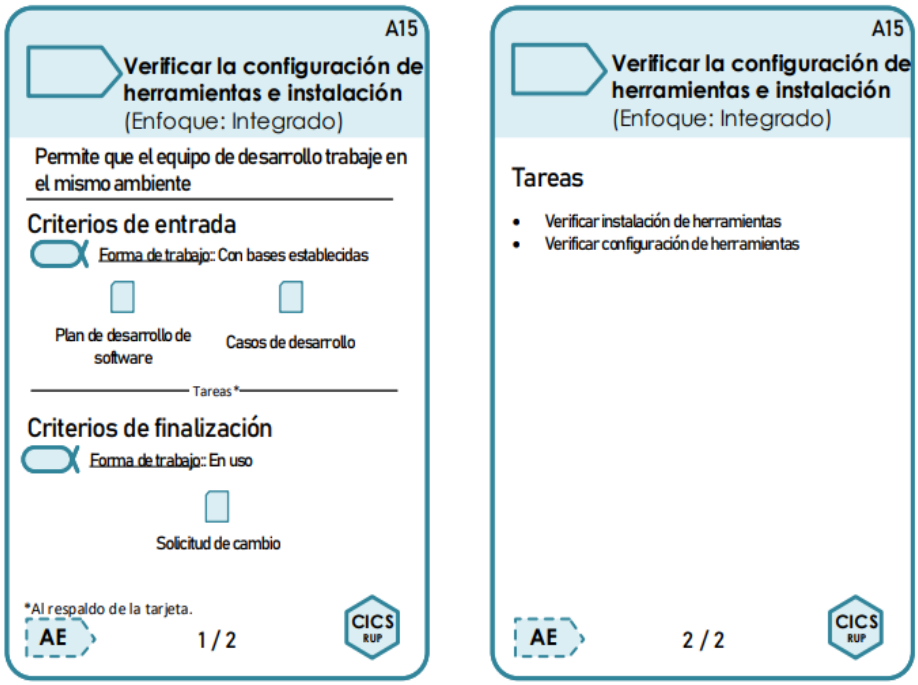


Fig. 78. Tarjeta de actividad Verificar la configuración de herramientas e instalación

Fuente: Elaboración propia

TABLA LXVI

RESULTADO DE LA ACIVIDAD VERIFICAR LA CONFIGURACIÓN DE HERRAMIENTAS E INSTALACIÓN (1)

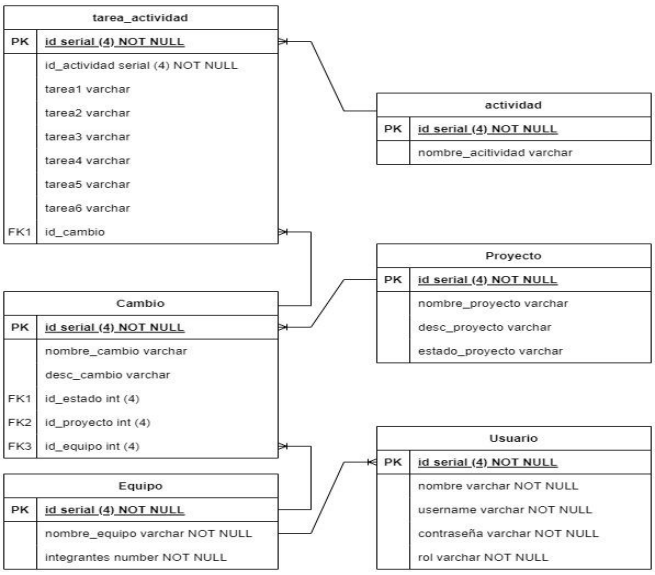
Producto de trabajo de entrada Actividad 15:
Plan de desarrollo de software
• Se requiere de dos desarrolladores, uno especializado en frontend y el otro en backend y un líder de equipo. • Se prioriza la integración de la nueva lógica y estabilidad del sistema software. • Al desarrollador frontend le corresponden los 4 artefactos de interfaz y el desarrollador backend de los 4 correspondientes a las peticiones. El líder mediará las reuniones semanales y estará encargado de guiar a los desarrolladores durante todo el proceso.

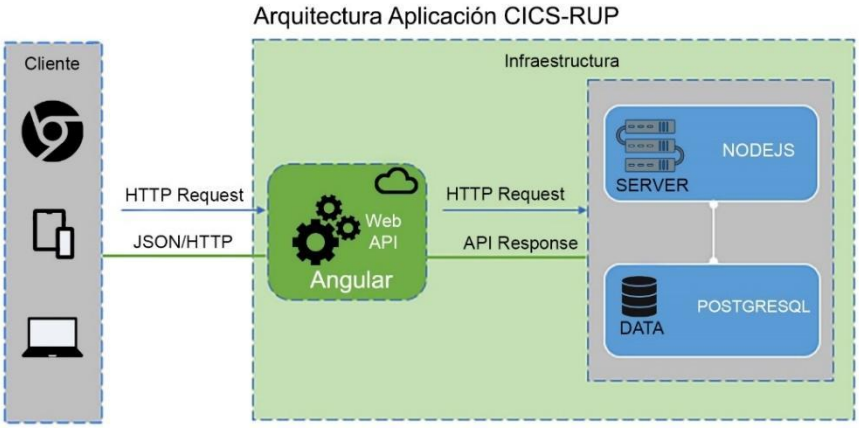
Fuente: Elaboración propia

TABLA LXVII

RESULTADO DE LA ACTIVIDAD VERIFICAR LA CONFIGURACIÓN DE HERRAMIENTAS E INSTALACIÓN (2)

Casos de desarrollo
Base de datos:





Producto de trabajo de salida Actividad 15:
Tarea 1: Verificar instalación de herramientas.
Tarea 2: Verificar configuración de herramientas.
Solicitud de cambio
Los desarrolladores se encuentran trabajando en un ambiente estable. No hay cambio

Fuente: Elaboración propia

Actividad 16:

En las tablas LXVIII y LXIX se encuentra el resultado de la actividad Presentar o actualizar solicitud de cambio, como lo describe la figura 79.

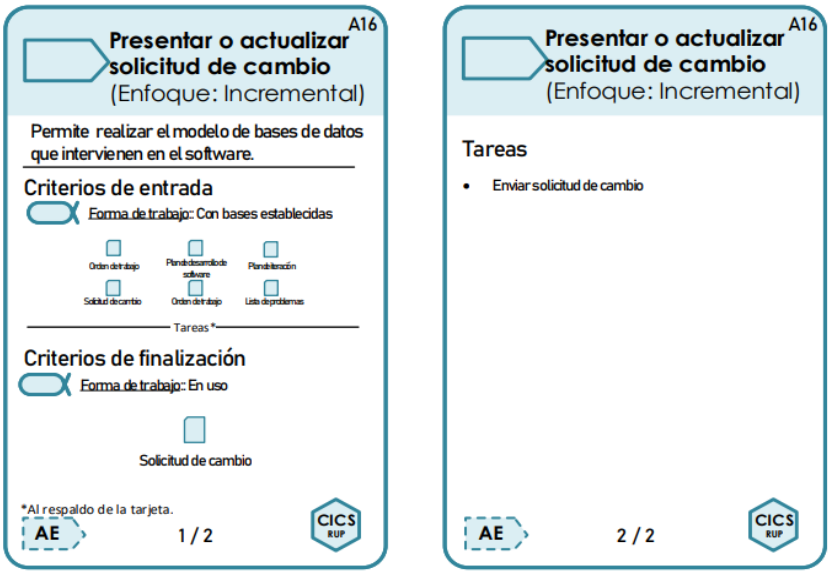


Fig. 79. Tarjeta de actividad Presentar o actualizar solicitud de cambio

Fuente: Elaboración propia

TABLA LXVIII

RESULTADO DE LA ACTIVIDAD PRESENTAR O ACTUALIZAR SOLICITUD DE CAMBIO (1)

Producto de trabajo de entrada Actividad 16:
Orden de trabajo

Numero orden	Cambio asociado
7	CICS-001
Encargado	
Especialidad	Jairo Arévalo
Desarrollador angular	
Entregables	
Artefacto	Tiempo estimado
Vista dashboard, nuevo elemento	8 hrs
Vista de página de proyectos	8 hrs
Vista de progreso del cambio del proyecto	8 hrs
Vista de actividad por proyecto	8 hrs
Detalles	
Verificar la versión de dependencias	
Asistir a las reuniones de control	
Verificar funcionalidad de cada cambio	

Numero orden	Cambio asociado
8	CICS-001
Encargado	
Especialidad	Nicolas Barrios
Desarrollador de Javascript	
Entregables	
Artefacto	Tiempo estimado
Controlador de dashboard	8 hrs
Controlador de proyectos	8 hrs
Controlador de progreso del cambio del proyecto	8 hrs
Controlador asignar usuarios a varios proyectos	8 hrs
Detalles	
Verificar la versión de dependencias	
Asistir a las reuniones de control	
Verificar funcionalidad de cada cambio	

Producto de trabajo de entrada Actividad 16:

Numero orden	Cambio asociado
9	CICS-001
Encargado	
Especialidad	Jairo Arévalo
Líder de equipo	Nicolas Barrios
Entregables	
Artefacto	Tiempo estimado
N/A	N/A
Detalles	
Organizar equipo	
Liderar reuniones en búsqueda de problemas	

Plan de desarrollo de software

- Se requiere de dos desarrolladores, uno especializado en frontend y el otro en backend y un líder de equipo.
- Se prioriza la integración de la nueva lógica y estabilidad del sistema software.
- Al desarrollador frontend le corresponden los 4 artefactos de interfaz y el desarrollador backend de los 4 correspondientes a las peticiones. El líder mediará las reuniones semanales y estará encargado de guiar a los desarrolladores durante todo el proceso.

Plan de iteración

La creación de interfaz y controladores se realiza en una iteración con duración total de un mes (160 horas), se requiere de dos desarrolladores quienes deben contar con las herramientas necesarias para llevar a cabo el cambio. Se realizarán 3 reuniones (25 minutos) semanales durante la duración de la iteración en las cuales se debe comunicar el avance realizado con el fin de identificar posibles problemas y oportunidades. Para ello, se identificaron 4 vistas nuevas. Así mismo, 4 peticiones para suplir la nueva necesidad.

Fuente: Elaboración propia

TABLA LXIX

RESULTADO DE LA ACTIVIDAD PRESENTAR O ACTUALIZAR SOLICITUD DE CAMBIO (2)

Solicitud de cambio
CICS-002: Se identifica la necesidad de controlar los cambios de diferentes proyectos, es necesario acceder a múltiples proyectos, asignar cambios y usuarios a estos.
Lista de problemas
No se encontraron problemas.
Producto de trabajo de salida Actividad 16:
Tarea 1: Enviar solicitud de cambio.
Solicitud de cambio
CICS-002: Se identifica la necesidad de controlar los cambios de diferentes proyectos, es necesario acceder a múltiples proyectos, asignar cambios y usuarios a estos.

Fuente: Elaboración propia

Actividad 17:

En la tabla LXX se encuentra el resultado de la actividad Gestionar pruebas de aceptabilidad, como lo describe la figura 80.

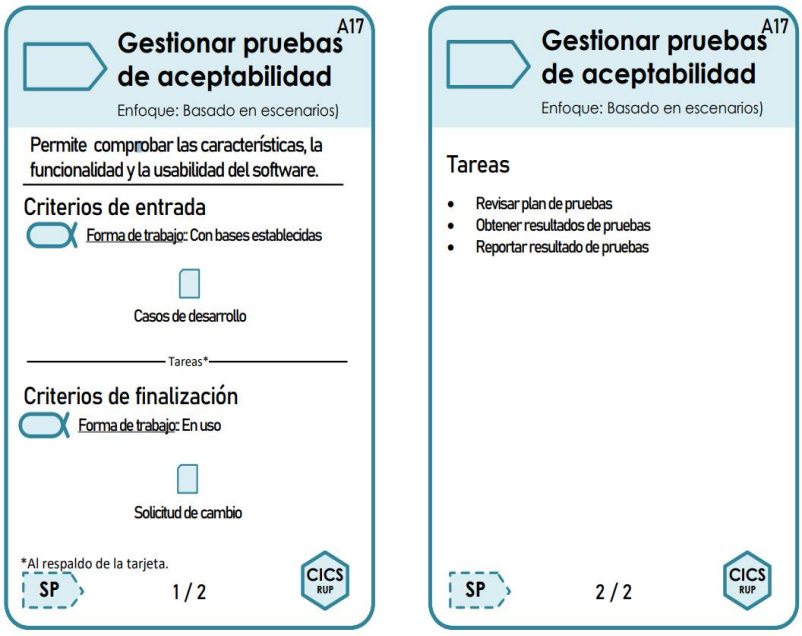


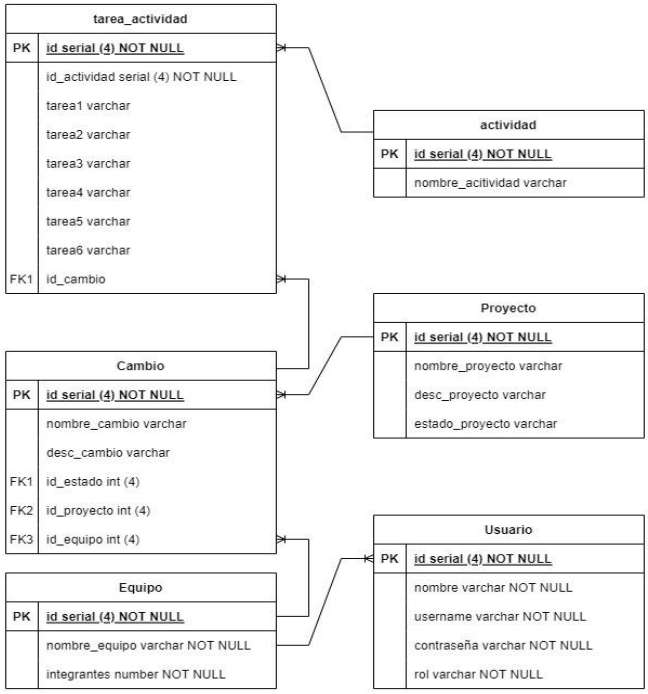
Fig. 80. Tarjeta de actividad Gestionar pruebas de aceptabilidad

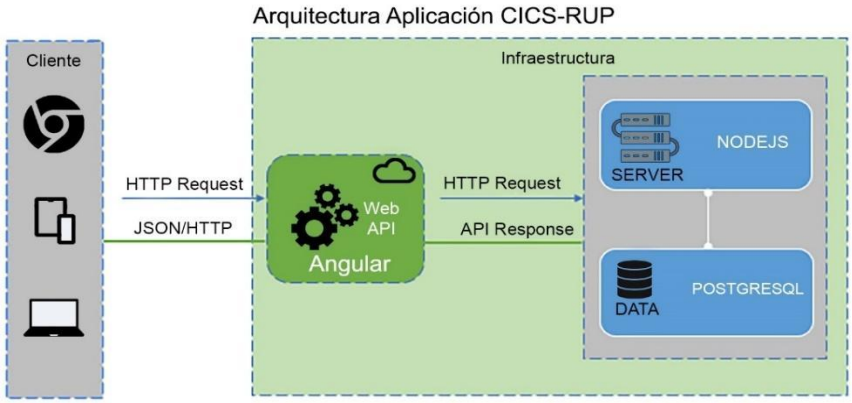
Fuente: Elaboración propia

TABLA LXX

RESULTADO DE LA ACTIVIDAD GESTIONAR PRUEBAS DE ACEPTABILIDAD

Producto de trabajo de entrada Actividad 17:
Casos de desarrollo
Base de datos:





Producto de trabajo de salida Actividad 17:
Tarea 1: Revisar plan de pruebas.
Tarea 2: Obtener resultados de pruebas.
Tarea 3: Reportar resultado de pruebas.
Solicitud de cambio
No se identifican cambios

Fuente: Elaboración propia

Actividad 18:

En la tabla LXXI se encuentra el resultado de la actividad Revisar aceptación del cambio, como lo describe la figura 81.

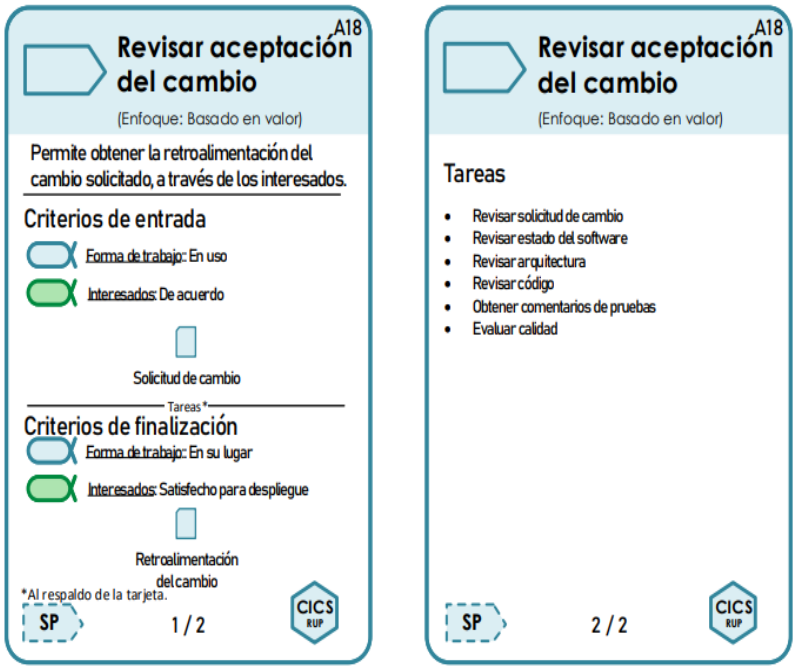


Fig. 81. Tarjeta de actividad Revisar aceptación del cambio

Fuente: Elaboración propia

Tabla LXXI

Resultado de la actividad Revisar aceptación del cambio

Producto de trabajo de entrada Actividad 18:
Solicitud de cambio
CICS-002: Se identifica la necesidad de controlar los cambios de diferentes proyectos, es necesario acceder a múltiples proyectos, asignar cambios y usuarios a estos.
Producto de trabajo de salida Actividad 18:
Tarea 1: Revisar solicitud de cambio.
Tarea 2: Revisar estado del software.
Tarea 3: Revisar arquitectura.
Tarea 4: Revisar código.
Tarea 5: Obtener comentarios de pruebas.
Tarea 6: Evaluar calidad.

Producto de trabajo de entrada Actividad 18:
Retroalimentación del cambio
• Software: Funcional. • Pruebas: Aprobadas satisfactoriamente. • Calidad: Aceptable. • Los problemas encontrados durante la ejecución del cambio se solucionaron de manera exitosa. • El cambio se ha realizado exitosamente, los artefactos fueron finalizados en su totalidad y entregados a tiempo.

Fuente: Elaboración propia

Actividad 19:

En la tabla LXXII se encuentra el resultado de la actividad Evaluar iteración, como lo describe la figura 82.

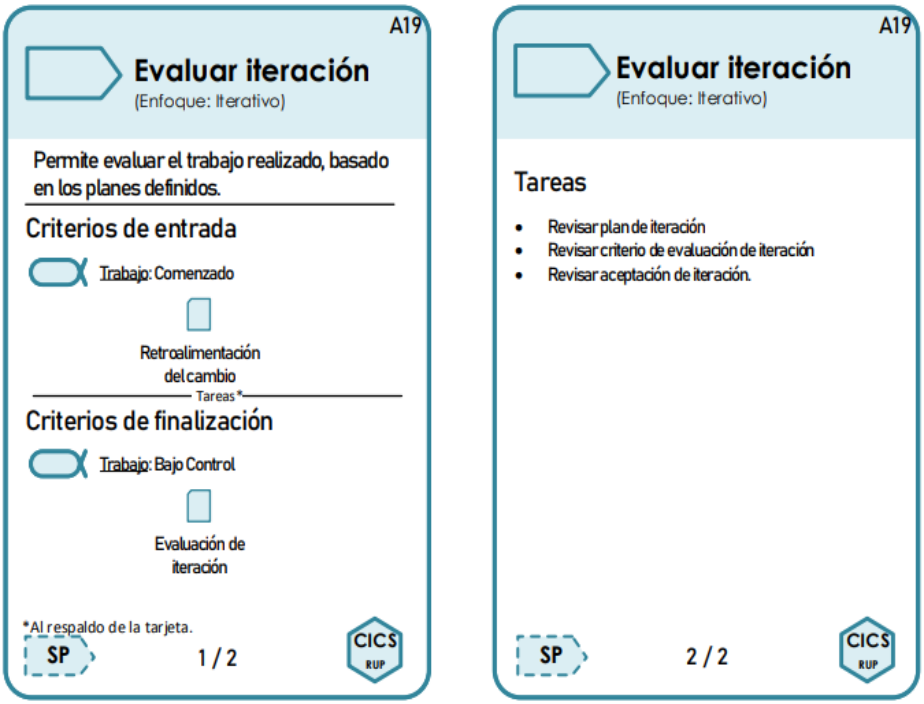


Fig. 82. Tarjeta de actividad Evaluar iteración

Fuente: Elaboración propia

TABLA LXXII

RESULTADO DE LA ACTIVIDAD EVALUAR ITERACIÓN

Producto de trabajo de entrada Actividad 19:
Retroalimentación del cambio
• Software: Funcional. • Pruebas: Aprobadas satisfactoriamente. • Calidad: Aceptable. • Los problemas encontrados durante la ejecución del cambio se solucionaron de manera exitosa. • El cambio se ha realizado exitosamente, los artefactos fueron finalizados en su totalidad y entregados a tiempo.
Producto de trabajo de salida Actividad 19:
Tarea 1: Revisar plan de iteración. Tarea 2: Revisar criterio de evaluación de iteración. Tarea 3: Revisar aceptación de iteración.
Evaluación de iteración
• Artefactos identificados/Artefactos entregados: 8/8. • Tiempo estimado/Tiempo contabilizado: 160h/160h. • Problemas encontrados/Problemas solucionados en la iteración: 0. • Nuevos cambios encontrados/Cambios tratados: 0. Comentarios: Cambio realizado satisfactoriamente.

Fuente: Elaboración propia

Actividad 20:

En la tabla LXXIII se encuentra el resultado de la actividad Finalizar iteración, como lo describe la figura 83.

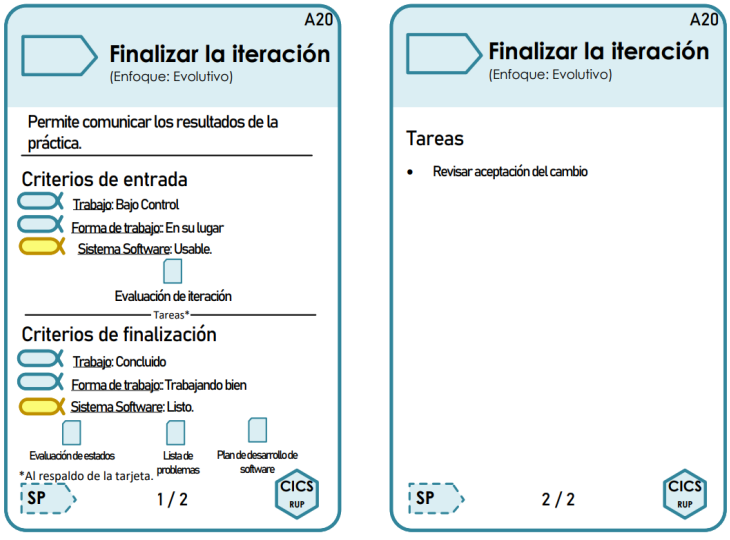


Fig. 83. Tarjeta de actividad Finalizar iteración

Fuente: Elaboración propia

TABLA LXXIII

RESULTADO DE LA ACTIVIDAD FINALIZAR ITERACIÓN

Producto de trabajo de entrada Actividad 20:
Evaluación de iteración
● Artefactos identificados/Artefactos entregados: 8/8. ● Tiempo estimado/Tiempo contabilizado: 160h/160h. ● Problemas encontrados/Problemas solucionados en la iteración: 0. ● Nuevos cambios encontrados/Cambios tratados: 0. Comentarios: Cambio realizado satisfactoriamente.
Producto de trabajo de salida Actividad 20:
Tarea 1: Programar trabajo. Tarea 2: Asignar trabajo.
Evaluación de estados
● Artefactos entregados 8 (100%) ● Tiempo utilizado: 160hs (100%) ● Estado final: Iteración exitosa.
Lista de problemas
No se encontraron problemas.

Fuente: Elaboración propia

CONCLUSIONES

- En este trabajo de grado se presentó el proceso de esencialización de CCS-RUP aplicando el Modelo para la Definición de Prácticas en Ingeniería de Software (Barón, 2019). La práctica CCS-RUP esencializada se denomina CICS. En la práctica CICS se identifica y se define los elementos que componen la práctica bien formada y bien nombrada. De esta manera permite a los practicantes entender, aplicar y evaluar la práctica en contextos reales.
- La RSL que guio el estudio de la práctica CCS-RUP, constituye una estrategia eficaz para identificar y analizar de manera estructurada e imparcial los estudios relevantes sobre cómo, la comunidad de ingeniería de software define la práctica CCS-RUP, así mismo reportar los resultados del estudio.
- El Modelo para la Definición de Prácticas en ingeniería de Software (Barón, 2019), se constituyó en un mecanismo adecuado para definir la práctica CICS como una práctica bien formada y bien nombrada. El Modelo permitió identificar y definir los elementos que componen la práctica CICS, de esta manera es fácil para los profesionales entender, aplicar y evaluar la práctica CICS e contextos reales.
- Essence se constituyó en el marco de trabajo que, integrado al Modelo para la Definición de Prácticas en Ingeniería de Software (Barón, 2019), permite identificar y definir los componentes de la práctica CICS empleando los elementos del núcleo. De esta manera se garantiza una práctica esencializada que facilita a los practicantes entender aplicar y evaluar la práctica en contextos reales.
- 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 CICS 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.

- Como validación de la práctica CICS, se aplicó en un estudio de caso orientado a la construcción del software Practica-CICS, en donde se aplicaron las actividades y flujo de actividades obteniendo los resultados. Esto permite demostrar que la práctica CICS en un proceso de control de cambios de software es fácil de entender aplicar y evaluar en contextos reales.
- Como trabajo futuro de esta investigación, se plantea el siguiente:
  - ✓ Finalizar la construcción del software Práctica-CICS que soporta entender, aplicar y evaluar la práctica CICS.
  - ✓ Extender el uso de la práctica CICS en contextos reales, compartiendo los resultados de la investigación con empresas de desarrollo de software y practicantes.

REFERENCIAS

[1] A. Barón, "Modelo para la Definición Unificada de la Práctica como Constructo Teórico en Ingeniería de Software", tesis de doctorado, Universidad Nacional, 2019.

[2] A. Barón and C. Zapata, "Conceptual synthesis of practice as a thorerical construct in Software Engineering", En C. Zapata, Durango, & W. Perdomo, *Software Engineering: Methods, Modelling, and Teaching*, vol. 4, pp. 244-267, 2017.

[3] Beck et al. "Manifesto for Agile Software Development," 2001. [Online]. Available: http://agilemanifesto.org/

[4] Berzal. "El ciclo de vida de un sistema de información", 2004. [En línea]. Disponible en: http://flanagan.ugr.es/docencia/2005-2006/2/apuntes/ciclovida.pdf

[5] R. Castro. "Estructura básica del proceso unificado", 2004. [En línea]. Disponible en: Estructura básica del proceso unificado de desarrollo de software, https://www.icesi.edu.co/contenido/pdfs/rcastro_estructura-bas-puds.pdf

[6] D. Cerrón, "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", trabajo de fin de grado, Escuela profesional de ingeniería de sistemas e informática, Lima, 2017.

[7] Durango y Zapata. "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, 24-37, 2017.

[8] C. Durango, "Definición de buenas prácticas de desarrollo de sistemas de información geográfica utilizando el núcleo de Semat", tesis de doctorado, Universidad Nacional, Medellín, 2019.

[9] R. Gacitúa Bustos, "Métodos de desarrollo de software: El desafío pendiente de la estandarización", 2003. [En línea]. Disponible en: https://www.redalyc.org/articulo.oa?id=29901203

[10] F. García y A. García, *Ingeniería de Software I. Tema 5: Introducción al Proceso Unificado.* Universidad de Salamanca, Departamento de Informática y Automática, Salamanca, 2018.

[11] A. Hernández, "Aplicación del Proceso Unificado de Desarrollo a proyectos de software". Universidad Tecnológica de la Habana, José Antonio Echeverría, La Habana, 2004. [En línea]. Disponible en: https://www.researchgate.net/profile/Anaisa_Hernandez_Gonzalez/publication/312656269_Aplicacion_del_Proceso_Unificado_de_Desarrollo_a_proyectos_de_software/links/58878a87a6fdcc6b791ec281/Aplicacion-del-Proceso-Unificado-de-Desarrollo-a-proyectos-de-softwa

[12] *Guide to the Software Engineering Body of Knowledge.* IEEE Computer Society Press, 2014.

[13] *Norma Internacional ISO 9001,* International Organization for Standardization - 2015. [En línea]. Disponible en: http://www.itvalledelguadiana.edu.mx/ftp/Normas%20ISO/ISO%209001-2015%20Sistemas%20de%20Gesti%C3%B3n%20de%20la%20Calidad.pdf

[14] I. Jacobson, G. Booch y J. Rumbaugh, *El Proceso Unificado de Desarrollo de Software*, Addison-Wesley, 2001.

[15] I. Jacobson, P. McMahon, I. Spence y S. Lidman., "The Essence of Software Engineering: The SEMAT Kernel", Communications of the ACM, vol. 55, no.12, 42-49, 2012.

[16] W. Jaramillo, "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", trabajo de fin de grado, Pontificia Universidad Católica del Ecuador, Quito, 2016.

[17] L. Jiménez, "Representación en el Núcleo de SEMAT de Prácticas de Métodos de Desarrollo Basados en Planes", tesis de Maestría, Universidad Nacional, Medellín, 2016.

[18] B. Kitchenham y P. Brereton., "A systemic review of systematic review process research in software engineering", Information and Software Technology, vol. 55, no. 12, 2049-2075, 2013.

[19] B. Kitchenham y S. Charters, *Guidelines for performing systematic literature.* EBSE, Technical report, 2007.

[20] P, Kroll y P. Kruchten, *The Rational Unified Process Made Easy - A Practitioner's Guide to the RUP.* Addison Wesley object technology series, 2003.

[21] P. Kruchten, *The Rational Unified Process: An Introduction.* Boston, USA: Addison-Wesley, 2004.

[22] P. Monteiro, P. Borges, R. Machado y P. Ribeiro, "A Reduced Set of RUP Roles to Small Software Development Teams", Zurich, 2012. [En línea]. Disponible en: http://www3.dsi.uminho.pt/rmac/privatefiles/papers/2012_ICSSP_MonteiroBorgesMachadoRibeiro-ieee.pdf

[23] M. Moreno Espino, A. Rosarte Suárez, A. S. Cuevas, R. Valdéz González, E. Leyva Pérez, A. García Fernández. *Ingeniería de software orientada a agentes: Roles y metodologías*. Universidad Tecnológica de la Habana, José Antonio Echeverría, La Habana, 2006.

[24] *Kernel and Language for Software Engineering*, Object Management Group, 2018.

[25] L. Pérez-Queiruga y D. Vallespir, "Adaptación de RUP para PSP: experiencia en el PIS. Universidad de la República", Montevideo, 2009. [En línea]. Disponible en: https://www.colibri.udelar.edu.uy/jspui/bitstream/20.500.12008/3434/1/TR0915.pdf

[26] W. Pozo, "Metodología para el desarrollo de software escalable para el departamento de pensiones del IESS", tesis de maestría, Universidad Central del Ecuador, Quito, 2015.

[27] R. Pressman. *Ingeniería del Software. Un enfoque práctico.* New York, USA: McGraw- Hill, 2010.

[28] *Rational Unified Process: Best practices for software development teams.* Rational Software Company, 2001. [En línea]. Disponible en: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

[29] S. M. Saqib, S. Ahmand, S. Hussain, B. Ahmad y A. Bano. "Improvement in RUP Project Management via Service Monitoring: Best Practice of SOA". Journal of computing, vol. 2, no. 2, 38-43, 2010. [En línea]. Disponible en: https://arxiv.org/ftp/arxiv/papers/1002/1002.3996.pdf

[30] *A Guide to the Scrum Body of Knowledge*. Scrum Study, 2013. [En línea]. Disponible en: http://www.scrumstudy.com/overview-of-sbok.asp

[31] *CMMI for Development, Version 1.3. Technical Report, Carnegie Mellon Institute.* Software Engineering Institute, Pittsburgh, USA, 2010.

[32] I. Somerville, "Sotware Engineering" Boston, Massachusetts, USA: Addison-Wesley, 2011.

[33] A. Soriano, "RUP: Manejo de Cambios y Configuraciones", 2015. [En línea]. Disponible en: http://carolina.terna.net/ingsw3/datos/Semana5_CM.pdf

[34] L. F. Tabares Bedoya, "Personalización de RUP para proyectos académicos de desarrollo de software", trabajo de fin de grado, Universidad EAFIT, Medellín, 2011. [En línea]. Disponible en: https://repository.eafit.edu.co/xmlui/bitstream/handle/10784/2754/TabaresBedoya_LuisFelipe_2011.pdf;jsessionid=5E9CC18D105147CFA46B10199867781B?sequence=1

[35] C. Zapata y L. F. Castro. "Software Engineering: Methods. modeling and teaching. En M. González, C. Zapata, & L. González", *Toward a standardized representation of RUP best practices of project*, Medellín, pp 47-52, 2014.