Conciencia Tecnológica
ISSN: 1405-5597
contec@mail.ita.mx
Instituto Tecnológico de Aguascalientes
México

Propuesta de Metodología Híbrida y Base de Documentación para el Desarrollo de Software Actual

Santos-Romero, Martín Antonio; Escudero-López, Nahin Ezequiel
Propuesta de Metodología Híbrida y Base de Documentación para el Desarrollo de Software Actual
Conciencia Tecnológica, núm. 60, 2020
Instituto Tecnológico de Aguascalientes, México
Disponible en: https://www.redalyc.org/articulo.oa?id=94465715002

Artículos

Propuesta de Metodología Híbrida y Base de Documentación para el Desarrollo de Software Actual

Hybrid Methodology Proposal and Documentation Base for Current Software Development

Martín Antonio Santos-Romero
Universidad Tecnológica de Chetumal, México

Nahin Ezequiel Escudero-López
Universidad Tecnológica de Chetumal, México
nahin.escudero@utchetumal.edu.mx

Redalyc: https://www.redalyc.org/articulo.oa?id=94465715002

Recepción: 25 Septiembre 2020
Aprobación: 24 Noviembre 2020

Resumen:

La producción de software actual demanda un desarrollo ágil y entrega rápida, sin detenerse demasiado en la documentación. Sin embargo, cuando se requieren cambios y actualizaciones, el no contar con la documentación necesaria requiere inversiones mayores no presupuestadas en tiempo y recursos. El presente artículo propone una metodología híbrida con una base de documentación sólida que permita en un momento dado conocer y retomar lo que se hizo, sin incrementar los tiempos de desarrollo, con entregas parciales que representen versiones del software funcionando, que puedan ser adaptables a los cambios. La metodología propuesta se basa en la comparación e integración de un marco de trabajo ágil con un proceso de desarrollo tradicional, revisando sus fases y artefactos generados en cada una de ellas para integrarlos en una propuesta, lo que conlleva a definir las fases necesarias durante el proceso de desarrollo de software y una base de documentación que pueda realizarse a la par o posterior al desarrollo, que no implique sacrificar tiempos de entrega ágil, pero sí un esfuerzo mínimo de inversión simultánea para tener un proceso más completo que permita tener software ágil, actualizado y documentado.

Palabras clave: Scrum, RUP, documentación de software, proceso de desarrollo híbrido.

Abstract:

Today's software production demands agile development and fast delivery, without investing too much in documentation. However, when changes and updates are required, not having the necessary documentation requires major unbudgeted investments in time and resources. This article proposes a hybrid methodology with a solid documentation base that allows, at a given moment, to know and resume what was done, without increasing development times, with partial deliveries that represent versions of the software in use, which can be adaptable to the changes. The proposed methodology is based on the comparison and integration of an agile framework with a traditional development process, reviewing its phases and artifacts generated in each of them to integrate them into a proposal, which entails a definition of the necessary phases during the software development process and a documentation base that can be carried out alongside or after the development, which does not imply sacrificing agile delivery times, but a minimum effort of simultaneous investment to have a more complete process that allows having agile software , updated and documented.

Keywords: Scrum, RUP, software documentation, hybrid development process.

Cuando se desarrolla un software de manera profesional que va a ser usado por otras personas y que pueda ser modificado por otros ingenieros, se debe ofrecer información adicional al código del programa, como es la documentación del sistema. Actualmente, más del 70% de las organizaciones a nivel mundial han reportado usar metodologías ágiles en sus proyectos, de acuerdo al estudio Pulse of the Profession que realiza anualmente

Notas de autor

Contacto: nahin.escudero@utchetumal.edu.mx

el Project Management Institute (PMI) para evidenciar cuál es la tendencia del mercado en cuanto a la gestión de proyectos y la metodología para llevarlos al éxito [1]. Según datos del Business Agility Corporation (BAC), el 70% de negocios están utilizando las metodologías ágiles de forma regular [2]. Pressman [3] cita el "Manifiesto para el desarrollo ágil de software" de la "Alianza ágil", donde se afirma que la labor de desarrollar software debía valorarse por cuatro postulados: "*1. Los individuos y su interacción, por encima de los procesos y las herramientas. 2. El software que funciona, por encima de la documentación exhaustiva. 3. La colaboración con el cliente, por encima de la negociación contractual. 4. Responder al cambio, mejor que apegarse a un plan*".

Lo anterior implica que el desarrollo ágil y entrega rápida son por lo general el requerimiento fundamental de los sistemas de software actuales. Los negocios funcionan en un entorno cambiante y es casi imposible derivar un conjunto completo de requerimientos de software, el cómo un sistema afectará sus prácticas operacionales, cómo interactuará con otros sistemas y cuáles operaciones de usuarios se automatizarán. Es posible que sea sólo hasta después de entregar un sistema, y se adquiera experiencia con éste, cuando se aclaren los requerimientos reales. Razón por la cual se prefieren las metodologías ágiles y el software en producción y funcional, que la documentación exhaustiva [4].

Tal es el caso de la Universidad Chetumaleña (se omite el nombre real), una universidad pública del estado de Quintana Roo en México, cuenta con cuatro campus académicos donde se ofertan más de 20 carreras. Hoy tiene una plantilla de más de mil empleados entre profesores y personal administrativo. Su área de Tecnologías de la Información (TI) ha adoptado el marco de trabajo Scrum para sus proyectos y ha desarrollado un software de gestión de nómina para el departamento de Recursos Humanos, con el objetivo de potenciar la productividad de las personas encargadas de controlar los aspectos de sueldos y prestaciones laborales de sus empleados, teniendo en consideración la legislación fiscal y laboral vigente del país. Por la urgencia de contar con el software en producción, no se cuenta con una documentación completa sobre su diseño y desarrollo; y los ingenieros de software actuales no pueden realizar ajustes y actualizaciones de una manera rápida y eficiente, teniendo efectos negativos por inversión de tiempo, personal y dinero no presupuestado, pero sobre todo teniendo al software desactualizado y sin la funcionalidad adecuada.

El objetivo general de este trabajo fue el desarrollar una propuesta de metodología híbrida de desarrollo de software y base de documentación integrando el marco de trabajo ágil Scrum con el proceso tradicional de desarrollo de software RUP (Rational Unified Process), que permita desarrollar de manera ágil software documentado combinando las ventajas de cada proceso. Para lo cual, se fijaron los siguientes objetivos específicos:

- Comparar las fases o etapas del proceso de desarrollo de software ágil Scrum y el proceso de desarrollo tradicional RUP.
- Identificar fases y disciplinas de desarrollo de software que se aplican actualmente a los proyectos de software.
- Identificar las fases o etapas donde se pueden integrar los artefactos de ambos procesos, definiendo una documentación base requerida.
- Proponer una metodología híbrida de desarrollo de software.

La metodología híbrida de desarrollo de software propuesta conserva la filosofía del manifiesto ágil de entregar software funcional en los tiempos marcados por estos procesos, pero propone la integración de documentación base para garantizar el manejo del cambio. Se realizó un comparativo de los diferentes artefactos entregables en cada una de las fases de un proceso tradicional, se analizó la necesidad de contar con ellos con base a las necesidades de los profesionistas actuales y su importancia para el entendimiento futuro del funcionamiento del sistema y se definió qué artefactos son los requeridos para estos casos. La base de documentación de software permite identificar claramente qué artefactos o documentos se requieren realizar para cualquier software que ayuden a los nuevos ingenieros y usuarios del sistema a entender claramente su diseño y funcionamiento para las futuras actualizaciones.

Fundamentos teóricos

De acuerdo con Pressman [3], el desarrollo de software implica llevar a cabo numerosas tareas que se agrupan en *fases o etapas*, al conjunto de estas etapas se le llama *proceso o ciclo de vida del desarrollo de software*. Cada fase puede implicar una o varias disciplinas. Una *disciplina* es una colección de actividades relacionadas con un área de atención dentro de todo el proyecto para lograr un resultado en particular. Los procesos de desarrollo de software se dividen en procesos ágiles y tradicionales. Los *procesos ágiles* se emplean en entornos complejos donde se necesita obtener resultados pronto, los requerimientos son cambiantes, la entrega del producto final se hace en módulos funcionales, se da más énfasis a las comunicaciones cara a cara en vez de la documentación. Caso contrario, los *procesos tradicionales*, buscan especificar por completo los requerimientos y luego, diseñar, construir y probar el sistema completo, ver Tabla 1.

TABLA 1.
Diferencias entre procesos ágiles y tradicionales [5].

Proceso ágil	Proceso tradicional
Proceso basado en heurísticas provenientes de prácticas de producción de código	Proceso basado en normas provenientes de estándares seguidos por el entorno de desarrollo
Preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos o entregables	Más artefactos (documentación)
Pocos roles en un proyecto	Más roles en un proyecto
Menos énfasis en la arquitectura del software	Arquitectura del software esencial y se expresa mediante modelos

Scrum es un marco de trabajo ágil empleado para proyectos de desarrollo de software y actualmente para cualquier tipo de proyecto. Un proyecto Scrum se ejecuta en iteraciones temporales cortas y fijas (de dos a cuatro semanas) llamadas *Sprint*. Cada sprint es un micro ciclo de desarrollo completo cuyo fin es la entrega de un resultado que represente un *incremento* del producto final. El proceso se inicia con una lista priorizada de requerimientos conocida como *Product Backlog* y definida por el cliente o *Product Owner* como *historias de usuario*. En el primer día de cada sprint se realiza una reunión de planificación del sprint o *Sprint Planning* donde se hace una selección de los requisitos más prioritarios a cubrir en esta iteración y se crea una lista de tareas necesarias para desarrollar los requisitos o *Sprint Backlog*. El objetivo del sprint o *Sprint Goal* es una meta establecida para el sprint que puede lograrse mediante la implementación del sprint backlog. Durante el periodo de cada sprint se realiza una reunión rápida diaria o *Daily Scrum* para conocer los avances del entregable y poder hacer las adaptaciones necesarias para cumplir con el compromiso, liderados por el *Scrum Master*. El último día de la iteración se realiza la reunión de revisión y retrospectiva del sprint o *Sprint Review and Retrospective* donde se presenta al cliente los requisitos completados y se acuerdan las adaptaciones necesarias de manera objetiva, replanificando el proyecto; de igual manera se analiza la forma de trabajo para eliminar obstáculos identificados [6], ver Figura 1.

El *Proceso Unificado Racional* (Rational Unified Process – RUP) es un proceso de producto, desarrollado y financiado por Rational Software. Aunque actualmente fue adquirido y es soportado por IBM [7].

RUP es un proceso tradicional que implica un solo ciclo completo dividido en cuatro fases: iniciación, elaboración, construcción y transición [8], ver Figura 2.

RUP es básicamente una guía para cómo usar efectivamente el Lenguaje Unificado de Modelado (Unified Modeling Language – UML).

Materiales y métodos

Esta sección presenta la metodología empleada para identificar las fases y disciplinas de los procesos de desarrollo de software y los artefactos que requieren los profesionistas de la ingeniería de software actual, esto para plantear la propuesta de metodología híbrida y la base de documentación de un proyecto de software.

1. Comparativa del marco de trabajo ágil Scrum vs el proceso de desarrollo tradicional RUP. La comparativa de ambos procesos se resume en la Tabla 2.

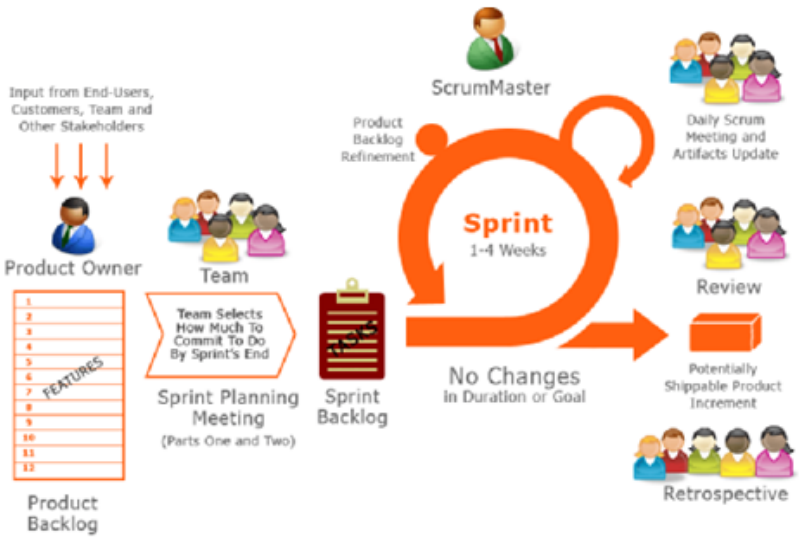

FIGURA 1.
Proceso Scrum [9].

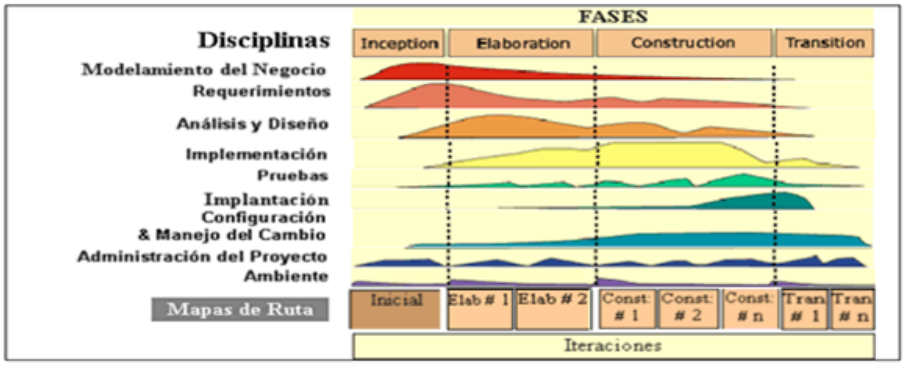

FIGURA 2.
Proceso RUP [10].

TABLA 2.
Entre Scrum y RUP (Aportación personal con base en [7], [8], [10], [11]).

	SCRUM	RUP
Enfoque	Iterativo	Iterativo
Ciclos	Cada Sprint es un ciclo completo	Un solo ciclo formal definido en cuatro fases, las cuales pueden ser concurrentes
Fases	La única fase formal es el Sprint, con duración de 2 a 4 semanas, aunque se tiene una fase previa de planificación donde se define el Sprint Backlog y la lista de Sprints.	1. Incepción o iniciación. 2. Elaboración. 3. Construcción. 4. Transición.
Planeación	No hay un plan de proyecto de principio a fin. Cada plan de la siguiente iteración se determina al final de la iteración actual. El Product Owner determina cuando el proyecto se terminó.	Se utiliza un plan de proyecto formal, asociado con múltiples iteraciones. El plan define una fecha final y tiene hitos intermedios.
Alcance	Scrum utiliza el product backlog, el cual es evaluado al final de cada iteración (Sprint).	El alcance es definido antes del inicio del proyecto y documentado en el artefacto del Alcance del Proyecto. El alcance puede revisarse durante el proyecto, como se van aclarando los requerimientos, de modo controlado.
Artefactos o documentos	Al inicio se define el Product Backlog, como simple documento usado por los desarrolladores. El único artefacto formal es el software operacional entregado al finalizar cada Sprint y el proyecto completo.	1. Incepción o iniciación: documento de visión/alcance, modelado inicial de proceso de negocio, diagrama de actividades. 2. Elaboración: Modelo completo del proceso de negocio, modelos de casos de uso, diagrama y especificación de casos de uso, lista de requerimientos funcionales y no funcionales, modelo de datos, diagrama de clases, diagrama de secuencia, diagrama de arquitectura del software y diseño de las interfaces. 3. Construcción: diagrama de estados, diagrama de colaboración y manual técnico. 4. Transición: diagrama de componentes, diagrama de despliegue, plan de pruebas, manual de implementación y manual de usuario.
Tipo de proyecto/producto	Recomendado para proyectos rápidos y organizaciones que no dependen de una fecha límite.	Recomendado para proyectos a nivel empresarial, grandes, a largo plazo y de complejidad media a alta.

2. Definición de disciplinas necesarias en las fases de un proceso de desarrollo de software. Se realizó una encuesta a 50 profesionistas del área de desarrollo de software provenientes de diferentes oficinas públicas del gobierno del estado y del sector empresarial, para conocer su opinión sobre la necesidad del desarrollo ágil y tradicional, la documentación del software, así como, conocer las principales disciplinas del proceso de desarrollo de software que aplican en sus proyectos actuales. Los resultados obtenidos se muestran en la Tabla 3.

Con base en estos resultados de la encuesta se puede observar que los proyectos actuales de desarrollo de software exigen una entrega funcional lo más ágil posible, donde la necesidad de documentación del proceso de desarrollo es simple y básica y solo un 20% lo requiere. Al aplicar un marco de desarrollo ágil, el 100% le da mayor importancia a tener una lista de requerimientos funcionales y no funcionales bien definida, solo el 70% lo complementa con un correcto modelado del proceso de negocio. En cuanto al análisis y diseño, el 85% lo realiza, hacen el diseño del modelo de datos, identificación de las clases y objetos, sin tanto detalle y documentación, argumentan que hacen un diseño único como plantilla para aplicar a todas sus interfaces. La implementación y plan de pruebas si es necesario para entregar un buen producto, así como, realizar los ajustes y cambios que surjan en esta disciplina. La documentación que realizan es simple y rápida en su mayoría.

3. Definición de artefactos necesarios en el desarrollo de software. De igual manera se aplicó una encuesta a los mismos 50 profesionistas para definir cuáles artefactos y diagramas son para ellos imprescindibles contar cuando se desarrolla y termina un proyecto y cualquier ingeniero pueda comprender y manejar el cambio. Se les presentó la lista de artefactos con base a las disciplinas de RUP para que indiquen cuáles consideran como documentación mínima necesaria, obteniendo los resultados de la Tabla 4, donde se

pudo observar que los artefactos con los que se requiere contar son los que en un 80% o más son señalados como sí necesarios. Ninguno de los encuestados está dispuesto a aumentar los tiempos de entrega ágiles de software funcional, pero si estarían en un 80% dispuestos a invertir recursos y tiempos a la par y adicionales para garantizar la documentación necesaria que permita un manejo del cambio sin problemas que impliquen mayores costos en el futuro.

TABLA 3.
Resultados de encuesta Proceso de Desarrollo y Disciplinas Aplicadas.

Proceso de desarrollo y documentación	**Sí**	**No**
Necesidad de software rápido y funcional	100%	0%
Necesidad de documentación del desarrollo del software	20%	80%
Disciplinas aplicadas en proyectos de software	**Sí**	**No**
Modelado del negocio	70%	30%
Requerimientos	100%	0%
Análisis y diseño	85%	15%
Implementación	100%	0%
Plan de pruebas	100%	0%
Configuración y manejo del cambio	100%	0%
Documentación simple y rápida de todas las disciplinas	95%	5%
Documentación completa y a detalle de todas las disciplinas	5%	95%

TABLA 4.
Resultados de encuesta Documentación Mínima del Desarrollo de Software.

Documento requerido para comprensión y manejo del cambio de un proyecto de desarrollo de software	**Sí**	**No**
Modelo del proceso de negocio (diagrama PAD)	90%	10%
Modelo de casos de uso (diagramas de casos de uso y especificación de casos de uso)	100%	0%
Requerimientos funcionales y no funcionales	100%	0%
Modelo de datos (diseño de la base de datos)	100%	0%
Diagrama de clases	100%	0%
Diagrama de secuencia	60%	40%
Diagrama de arquitectura de software	85%	15%
Diseño de las interfaces (Mockups)	25%	75%
Diagrama de estados	25%	75%
Diagrama de colaboración	25%	75%
Diagrama de componentes	15%	85%
Diagrama de despliegue	25%	75%
Plan de pruebas	10%	90%
Manual de implementación	85%	15%
Manual de usuario	90%	10%
Manual técnico	25%	75%
Incrementar los tiempos de entrega de software funcional para documentar	0%	100%
Invertir tiempo y recursos adicionales a la documentación, a la par del desarrollo y posterior a la entrega del software	80%	20%

Resultados y discusión

Propuesta de integración de las fases y artefactos Scrum-RUP.

Las etapas o fases que contendrá el proceso híbrido se describen a continuación:

1. Modelo del proceso de negocio. Partir de la definición de un proceso de negocio, para comprender las diferentes actividades que se llevan a cabo en las organizaciones hacia los clientes, es la clave para el éxito.

Para Hitpass [12] un proceso de negocio es "un conjunto de actividades que toman uno o más tipos de inputs y crean un output que es de valor para el cliente". La Gestión de Procesos de Negocio (BPM, en inglés Business Process Management), es "un conjunto de herramientas, tecnologías, técnicas, métodos y disciplinas de gestión para la identificación, modelización, análisis, ejecución, control y mejora de los procesos de negocio. Las mejoras incluyen tanto cambios de mejora continua como cambios radicales". El Modelo y Notación de Procesos de Negocio (en inglés, Business Process Modeling Notation, BPMN), permite el modelado de proceso de negocios. Por otra parte, Bizagi [13], define que "BPMN es una notación grafica que describe la lógica de los pasos de un proceso de negocio". La documentación de un modelado de negocio se presenta a través de diferentes diagramas. Los Diagramas de Arquitectura de Procesos (PAD) son la carta de presentación de la empresa y en ellos, de manera detallada se dan a conocer las actividades que se realizan en dicha empresa. Un Diagrama Rol Actividad (RAD) es representado desde el punto de vista de roles, actividades (son las acciones realizadas por el rol) e interacciones [14].

Para el caso de estudio del sistema de gestión de nómina de la Universidad Chetumaleña, se muestra en la Figura 3 un diagrama PAD, requerido para que los ingenieros tengan una visión general de los procesos y funcionalidades principales que cubre el sistema.

2. Requerimientos (Product Backlog). La disciplina de levantamiento de requerimientos es fundamental para cualquier proyecto. El siguiente documento indispensable que se debe generar es el Product Backlog. El product backlog es una lista ordenada de los requerimientos del producto. Enumera todas las características, funcionalidades, requisitos, mejoras y correcciones que constituyen cambios a realizarse sobre el producto para entregas futuras [6]. En la Figura 4 se observa una fracción del product backlog del software del sistema de nómina de la Universidad Chetumaleña, realizado para que los ingenieros actuales puedan entender lo que se requería del sistema.

3. Análisis y diseño. En esta disciplina se realiza el modelo de casos de uso, modelo de datos, diagrama de clases, diagrama de arquitectura.

Modelo de casos de uso - Diagrama de casos de uso general. Este documento modela el comportamiento del sistema de gestión de nómina, mostrando un conjunto de casos de uso, actores y sus relaciones, esto se observa en la Figura 5.

Modelo de casos de uso - Descripción de casos de uso de cada módulo A continuación, se describen los casos de uso y sus escenarios, a través de las plantillas de Coleman o plantillas IEEE 830-1998.

Diagrama de clases. Este documento de sirve para modelar las clases persistentes, sus atributos y sus relaciones, y sirve para identificar los objetos del sistema, como se muestra en la Figura 7.

Diagrama de arquitectura. La arquitectura del funcionamiento del sistema se muestra en la Figura 8 para el caso de estudio del sistema de gestión de nómina.

Modelo de datos. Para la documentación del diseño de la base de datos, se basó en el modelo relacional, que es definido en [15] como: "Una vista unificada de los datos, centrándose en la estructura lógica y abstracta de los datos, como representación del mundo real, con independencia de consideraciones de flujo físico", ver Figura 9.

Se recomienda realizar un *diccionario de datos* para tener un buen entendimiento de la base de datos se necesita conocer los detalles de las tablas y sus tipos de datos en cada campo para su correcto funcionamiento.

FIGURA 3.
Proceso general de elaboración de nómina (Recursos Humanos Universidad Chetumaleña).

Product Backlog

Proyecto: Sistema de gestión de nómina de la Universidad Chetumaleña

Identificador (ID) de la Historia	Enunciado de la Historia	Alias	Estado	Dimensión / Esfuerzo	Iteración (Sprint)	Priorid
SGN-01	Como administrador, necesito poder gestionar usuarios, con la finalidad de asignar roles y permisos dentro del sistema.	Gestión de usuarios	Terminado	4 días	5	Baja
SGN-02	Como administrador, necesito poder gestionar empleados, con la finalidad de registrar a todos los empleados de la universidad.	Gestión de empleados	Terminado	7 días	1	Alta
SGN-03	Como administrador, necesito poder gestionar un catálogo de puestos, con la finalidad de registrar todos los puestos del organigrama de la universidad.	Gestión de puestos	Terminado	2 días	4	Media
SGN-04	Como administrador, necesito poder gestionar un catálogo de áreas, con la finalidad de registrar todas los áreas del organigrama de la universidad.	Gestión de áreas	Terminado	2 días	4	Media
SGN-05	Como administrador, necesito poder gestionar un catálogo de niveles de puestos, con la finalidad de poder registrar todos los niveles que se pueden tener en un puesto.	Gestión de niveles de puestos	Terminado	2 días	4	Media
SGN-06	Como jefe de Recursos Humanos, necesito poder gestionar un catálogo de prestaciones a empleados, con la finalidad de poder registrar todas las prestaciones que puede recibir un empleado.	Gestión de prestaciones a empleados	Terminado	2 días	4	Media
SGN-07	Como jefe de Recursos Humanos, necesito poder gestionar un catálogo de impuestos a empleados, con la finalidad de poder registrar todos los tipos de impuestos que puede tener un empleado.	Gestión de tipos de impuestos	Terminado	2 días	2	Alta
SGN-08	Como jefe de Recursos Humanos, necesito poder calcular el pago quincenal de los empleados, con la finalidad de tener los ingresos netos del empleado cada quincena.	Cálculo de pagos quincenales	Terminado	7 días	3	Alta

FIGURA 4.
Product Backlog sistema de gestión de nómina.

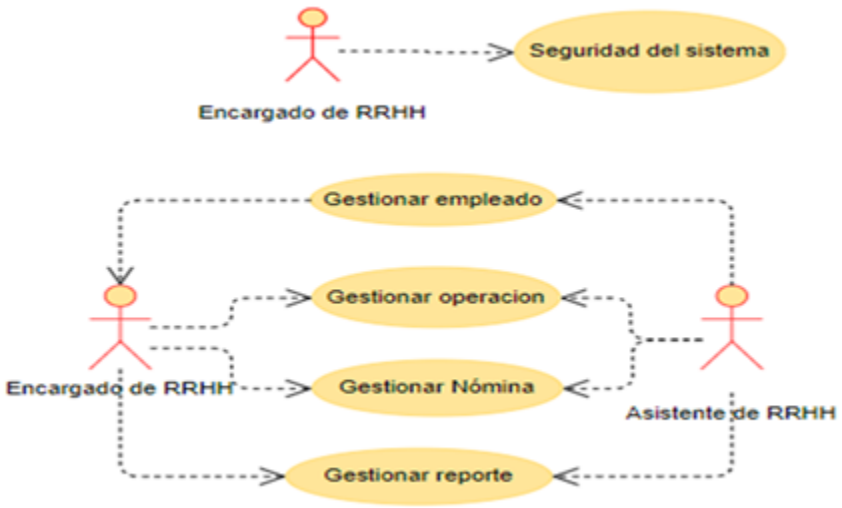

FIGURA 5.
Diagrama de casos de uso sistema gestión de nómina.

Caso de Uso	**Generar Nómina**				
Definición	**Permite generar la Nómina de un determinado periodo**				
Prioridad	o	**Vital**	✓	**Importante**	o **Conveniente**
Urgencia	o	**Inmediata**	✓	**Necesario**	o **No urgente**
Actores					
Nombre	**Definición**				
Representante de y Asistente de RRHH	**Usuarios que tienen acceso al sistema de nómina**				
Servidor de Base de datos	**Se usa la base de datos del servidor para generar la nóminas**				
Escenario					
Nombre:	**Generar Nómina**				
Precondiciones:	**Debe existir al menos un empleado en el sistema**				
Iniciado por:	**Usuario**				
Finalizado por:	**Sistema**				
Post-condiciones:	**La nómina se almacena en el sistema**				
	1.- Acceder a la interfaz del sistema				

FIGURA 6.
Descripción de casos de uso plantilla Coleman.

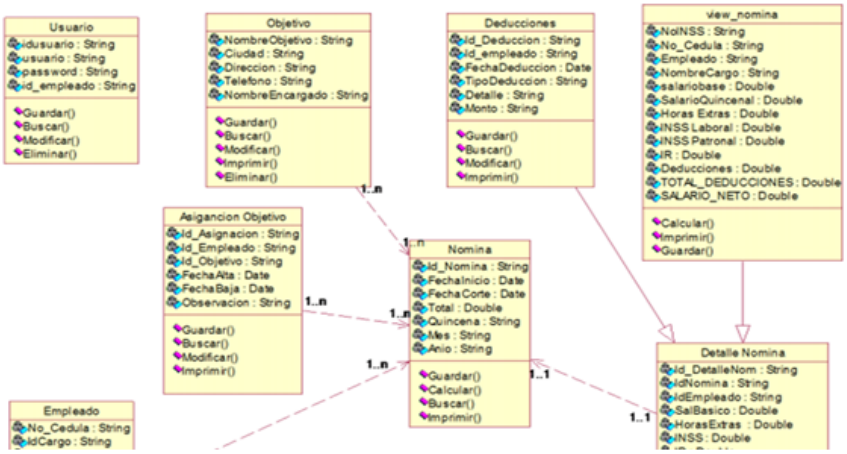

FIGURA 7.
Diagrama de clases del sistema de gestión de nómina.

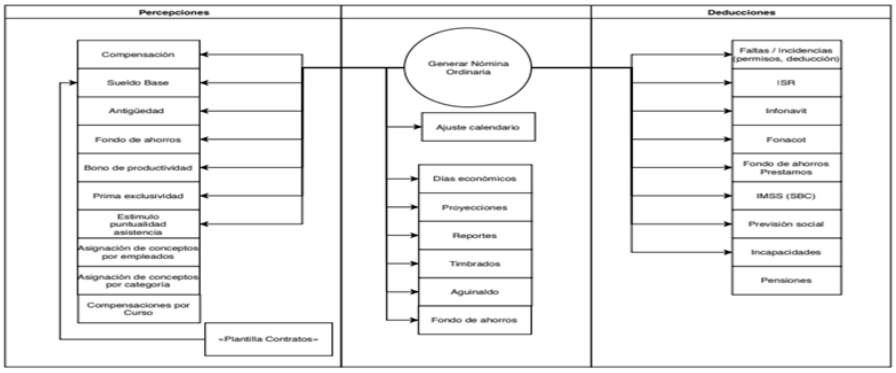

FIGURA 8.
Diagrama de arquitectura del sistema de gestión de nómina.

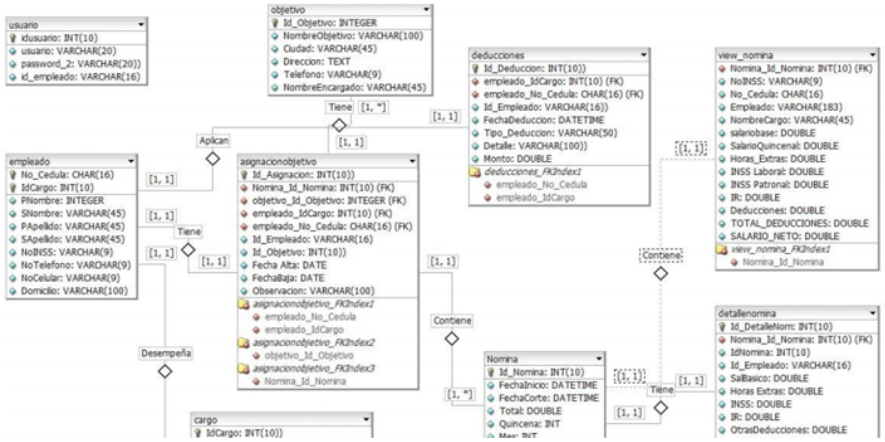

FIGURA 9.
Modelo relacional del sistema de gestión de nómina.

4. Iteración o Sprint. Se toma como base un Sprint de Scrum para el desarrollo de un módulo funcional y entregable del software, donde una vez que se ha definido junto con el cliente el *product backlog* se hace una selección de los requisitos más prioritarios a cubrir en cada iteración y se crea una lista de tareas necesarias para desarrollar estos requisitos o *sprint backlog* y conseguir el *sprint goal* [6]. Esta etapa es la principal de Scrum donde se integra las etapas de RUP y sus artefactos. Las etapas del RUP que se integran al Sprint de Scrum son:

Fase de Iniciación. Se comienza definiendo el sprint backlog que representará el Documento de Requerimientos Funcionales para un determinado módulo entregable, ver Figura 10.

Fase de elaboración. Basados en el sprint backlog del entregable se documenta esta fase, generando los diagramas de secuencia. Un *diagrama de secuencia* es un tipo de diagrama de interacción porque describe cómo —y en qué orden— un grupo de objetos funcionan en conjunto. Tanto los desarrolladores de software como los profesionales de negocios usan estos diagramas para comprender los requisitos de un sistema nuevo o documentar un proceso existente. A los diagramas de secuencia en ocasiones se los conoce como diagramas de eventos o escenarios de eventos, ver Figura 11.

Fase de Construcción. En esta fase se procede ya a la codificación de cada módulo, realizando la documentación del código, en el cual se pueden seguir estándares de calidad. Así como, un plan de pruebas del software desarrollado, de tal manera que cuando llegue el día de la *reunión de revisión y retrospectiva*, el módulo esté libre de errores técnicos y probablemente libre de errores funcionales. Se puede documentar usando los

diagramas de componentes y con los casos de prueba. El *diagrama de componentes* representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes, ver Figura 12.

Fase de Transición. En esta fase se realiza la documentación a través del manual de implementación y diagrama de despliegue. El *diagrama de despliegue* es la configuración de los módulos de procesamiento en tiempo de ejecución y los componentes que interactúan entre sí. Muestra qué elementos de software se implementan mediante qué elementos de hardware, ilustran el procesamiento en tiempo de ejecución para el hardware y proporcionar una vista de la topología del sistema, ver Figura 13.

SPRINT 1: Gestión de empleados

SPRINT	INICIO	DURACIÓN
1	04 de noviembre de 2019	7

	Inicio	Miércoles	Jueves	Viernes	Sábado	Domingo	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado	D
Cantidad de Tareas Pendientes	6	5	4	4	3		2	1	1				
Horas de Trabajos Pendientes	56	49	42	35	28		21	14	7				

PILA DEL SPRINT					ESFUERZO
Product Backlog ID	Tarea/Actividades	Tipo	Estado	Responsable	
SGN-02	Submodulo Registro de empleados		Terminado	Programador 1	
SGN-02.1	Diseñar formulario para registro de empleados	Diseño	Terminado	Programador 1	X
SGN-02.2	Crear formulario para registro de empleados	Desarrollo	Terminado	Programador 1	X X
SGN-02.3	Validar formulario para registro de empleados	Desarrollo	Terminado	Programador 1	X
SGN-02.4	Query para enviar los datos ingresados a la base de datos	Desarrollo	Terminado	Programador 1	
SGN-02.5	Respuesta de la base de datos	Desarrollo	Terminado	Programador 1	X
SGN-02.6	Pruebas de funcionamiento	Pruebas	Terminado	Tester 1	X X

FIGURA 10.
Sprint Backlog de Sprint Registro de Empleados.

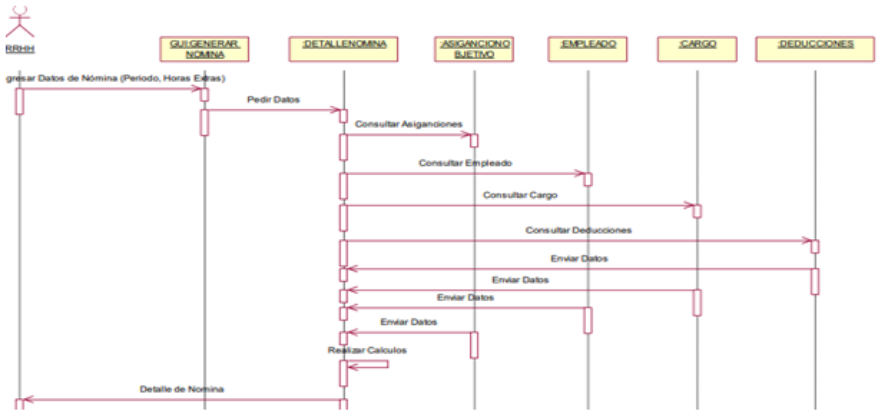

FIGURA 11.
Diagrama de secuencia escenario generar nómina.

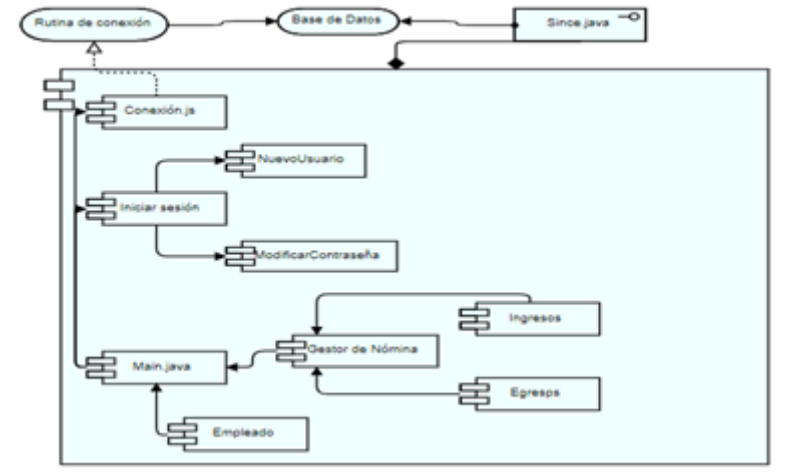

FIGURA 12.
Diagrama de componentes del sistema de Nóminas.

FIGURA 13.
Diagrama de despliegue del sistema de Nóminas.

Propuesta de documentación base para el desarrollo de software. Complementado lo anterior, se propone una base de documentación que incluye los documentos identificados y su ubicación en las fases del proceso de desarrollo híbrido, que se considerar necesarios generar en cualquier software ya desarrollado o con avances en su proceso de desarrollo. Se mantuvo la base del Scrum, y se consideró cada Sprint como un proyecto independiente, de manera que cada Sprint se convierte en único ciclo donde se pudo integrar el proceso RUP para tener un proceso más organizado y documentado.

El proceso híbrido junto con su base de documentación se resume a continuación.

Fase de incepción o iniciación:

1. Documento de modelo de proceso de negocio.
2. Product Backlog o requerimientos funcionales.

Fase de elaboración:

1. Documento modelo de casos de uso.
2. Documento modelo de datos.
3. Documento de diseño: diagrama de clases, de secuencia y de arquitectura, lista de Sprint o módulos funcionales.

Por cada uno de los Sprint o módulos:

1. Sprint Backlog o requerimientos del sprint
2. Diagrama de componentes.
3. Plan de pruebas
4. Diagrama de despliegue o implementación.

Al concluir, se puede integrar todo como un manual técnico y adicionalmente integrar las guías de usuario en un manual de usuario para todo el sistema.

Para mantener un proceso ágil es muy importante no sacrificar los tiempos de entrega, pero si invertir en recursos adicionales para documentadores a la par del desarrollo y posteriormente a la entrega e implementación, donde es común que se hagan los últimos ajustes a los requerimientos y funcionalidades del software y se vayan realizando los ajustes a los documentos.

La documentación generada para el software del sistema de gestión de nómina de la Universidad Chetumaleña permitió tener un software documentado, de modo que los nuevos ingenieros pueden conocer cómo está estructurado y diseñado el sistema, cuál debe ser su correcto funcionamiento y hacer las

actualizaciones y modificaciones necesarias para estar alineados a las nuevas reformas fiscales y laborales del país.

Conclusiones

La metodología híbrida propuesta y la base de documentación para procesos de desarrollo de software que actualmente demandan entregas ágiles y funcionales, es el resultado del análisis y comparativa de un marco de trabajo ágil y un proceso de desarrollo de software tradicional, que permitió integrar el marco de trabajo ágil Scrum con el proceso tradicional de desarrollo de software RUP, combinando las ventajas de cada proceso. Conocer lo que los proyectos actuales de software demandan en los sectores público y privado a través de la opinión de sus ingenieros expertos en el área, permitió determinar la necesidad de aplicar un proceso ágil con entrega de software funcional y lo más rápido posible, sin embargo, cuando se requieren cambios y actualizaciones, el no contar con la documentación necesaria conlleva a inversiones mayores no presupuestadas en tiempo y recursos, por lo que también consideran la necesidad de realizar esta inversión a la par y posterior al desarrollo. Se mantuvo la base del Scrum, y se consideró cada Sprint como un proyecto independiente, de manera que cada Sprint se convierte en único ciclo donde se pudo integrar el proceso RUP para reforzar el Sprint con un proceso más organizado y documentado. Esto es para generar la información necesaria para poder ser adaptables a los cambios y además cuenten con una base de documentación sólida en cada entrega que permita en un momento dado conocer lo que se hizo y cómo se hizo, de manera que los cambios solicitados en la retroalimentación son más fáciles de retomar, y al finalizar cada proyecto se tiene un conjunto de entregables adicionales que puedan servir para mantenimientos y actualizaciones futuras del software. La documentación generada para el software del sistema de gestión de nómina de la Universidad Chetumaleña permitió tener un software funcional y documentado, al realizar lo definido en los resultados de este trabajo como una base de documentación con el que todo software debe contar. De modo que, al finalizar un proyecto de desarrollo de software ágil, se obtiene además del producto final, la documentación necesaria para sus futuros mantenimientos y actualizaciones.

Referencias

[1] Project Management Institute. (2018). Pulse of the profession. Recuperado de https://www.pmi.org//media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2018.pdf

[2] Business Agility Corporation. (2018). BACometro. Recuperado de http://businessagilitycorp.com/bacometro/

[3] Pressman, Roger. (2010). Ingeniería de Software. Un enfoque práctico. Mc-Graw Hill, Séptima edición.

[4] Sommerville, Ian. (2011). Ingeniería del software. Addison Wesley. 9a edición.

[5] Montoya, Lina & Sepulveda, Jorge & Ramos, Luisa. (2016). Análisis comparativo de las metodologías ágiles en el desarrollo de software aplicadas en Colombia. https://www.researchgate.net/publication/317840767_Analisis_comparativo_de_las_metodologias_agiles_en_el_desarrollo_de_software_aplicadas_en_Colombia

[6] Schwaber, Ken & Sutherland, Jeff. (2017). La guía definitiva de Scrum: las reglas del juego. Recuperado de https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Spanish-SouthAmerican.pdf

[7] Jaramillo, Wendy. (2016). Tesis: 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. Pontificia Universidad Católica de Ecuador.

[8] Kruchten, P. (2004). The rational unified process: an introduction. Addison-Wesley Professional.

[9] European Scrum. (2020). Guía de formación Scrum. Recuperado de https://www.europeanscrum.org/formacion-scrum.html

[10] Martínez, A., & Martínez, R. (2014). Guía a rational unified process. Escuela Politécnica Superior de Albacete– Universidad de Castilla la Mancha.

[11] Jacobson, I., Booch, G., & Rumbaugh, J. (2000). The unified software development process (No. 004.41). Pearson Educación.

[12] Hitpass, B. (2012). BPM: Business Process Management Fundamentos y Conceptos de Implementación. Santiago de Chile: BHH Ltda.

[13] Bizagi. (2014). BPMN 2.0. Desconocida: Bizagi Suite. Recuperado de http://resources.bizagi.com/docs/BPMNbyExampleSPA.pdf

[14] Pacheco, J. (2017). Arquitectura de procesos. Recuperado de https://www.heflo.com/es/blog/bpm/arquitectura-procesos/

[15] Peraire, C. & Edwards, M. Et. al. (2017). The IBM Rational Unified Process for System z. IBM.