REVISTA CUBANA
DE TRANSFORMACIÓN DIGITAL

RNPS 2487 • ISSN 2708-3411
Vol. 5 • 2025 • e289

ARTÍCULO ORIGINAL

Metodología FD para Trabajos de Fin de Curso en el ISP-Bié

FD Methodology for Final Course Projects at ISP-Bié

Daysel Labañino Griñan
dlabaninog@gmail.com • *http://orcid.org/0000-0002-5098-3776*

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Feliciano Chiluzia Bumba Gabriel
fegabriel018@gmail.com •

INSTITUTO SUPERIOR POLITÉCNICO DO BIÉ, ANGOLA

Recibido: 2025-10-01 • Aceptado: 2025-11-07

RESUMEN

Este estudio analiza la aplicación de la Metodología FD (Letras iniciales de los autores) combinando prácticas de las metodologías Scrum y RUP en Trabajos de Fin de Curso (TFC) del Instituto Superior Politécnico do Bié (ISP-Bié). Por otro lado, compara sus resultados con experiencias recientes reportadas en la literatura académica. La metodología combina ciclos iterativos de Scrum con artefactos RUP formales, proporcionando entregas incrementales y documentación rigurosa. Se evaluaron aspectos como estructura del proceso, resultados técnicos, desafíos e impacto pedagógico. Se observó que, tanto en el ISP-Bié como en instituciones externas, el enfoque híbrido mejoró la productividad, la trazabilidad de los requisitos y la calidad técnica de los TFC. La aplicación permitió a los estudiantes desarrollar habilidades integrales en planificación ágil, modelado UML e ingeniería de requisitos. Entre los desafíos enfrentados, los más notables fueron la sobrecarga de documentos y la necesidad de capacitación adicional para los equipos. Como buenas prácticas se destacaron la adopción de herramientas integradas, la definición clara de roles y la priorización de artefactos esenciales. La comparación reveló que los beneficios pedagógicos –incluido un mayor compromiso, una colaboración efectiva y el desarrollo de habilidades interpersonales– fueron consistentes en todos los contextos estudiados. La adopción de la metodología FD en el ISP-Bié está alineada con las mejores prácticas internacionales y tiene el potencial de fortalecer

significativamente la calidad de los TFC y la formación práctica de los estudiantes en ingeniería informática.

Palabras clave: Scrum, RUP, metodología híbrida, ingeniería de software.

ABSTRACT

This study analyzes the application of the FD Methodology (Authors' initials) combining practices of the Scrum and RUP methodologies in Final Course Projects (TFC) of the Instituto Superior Politécnico do Bié (ISP-Bié). Additionally, it compares its results with recent experiences reported in academic literature. The methodology integrates Scrum's iterative cycles with formal RUP artifacts, enabling incremental deliveries and rigorous documentation. Aspects such as process structure, technical outcomes, challenges, and pedagogical impact were evaluated. It was observed that, both at ISP-Bié and external institutions, the hybrid approach improved productivity, requirement traceability, and the technical quality of TFC. The application allowed students to develop comprehensive skills in agile planning, UML modeling, and requirements engineering. Among the challenges faced, the most notable were document overload and the need for additional team training. Best practices included the adoption of integrated tools, clear role definitions, and the prioritization of essential artifacts. The comparison revealed that pedagogical benefits—including increased engagement, effective collaboration, and the development of interpersonal skills—were consistent across all studied contexts. The adoption of the FD methodology at ISP-Bié aligns with international best practices and has the potential to significantly enhance the quality of TFC and the practical training of computer engineering students.

Keywords: Scrum, RUP, hybrid methodology, software engineering.

INTRODUCCIÓN

En las últimas décadas, las metodologías de desarrollo de software han evolucionado para satisfacer la creciente complejidad y dinamismo de los proyectos académicos e industriales. Entre 2018 y 2025, varios estudios investigaron la combinación de enfoques prescriptivos y ágiles con el objetivo de maximizar la eficiencia, la calidad y la adherencia a los requisitos pedagógicos en los TFC. Por ejemplo, Oliveira et al. (2021) analizaron la aplicación de un framework híbrido en proyectos universitarios, mostrando una reducción del 20% en la reelaboración de documentos y un aumento del 35% en la satisfacción de los supervisores. Kumar y Singh (2023) propusieron un modelo de sincronización entre sprints ágiles y fases documentales que presentó una mejora del 25% en la claridad de los requisitos y la gobernanza del proyecto.

En el contexto del ISP-Bié, aún existe una falta de informes sobre la adopción de modelos híbridos (Scrum + RUP) en TFC entre 2023 y 2025. Esta brecha justifica la realización de este estudio, cuyo objetivo es evaluar el

potencial de este enfoque para promover entregas de valor incremental junto con artefactos formales necesarios para el rigor académico. La elección de Scrum se basa en la evidencia de su efectividad en entornos educativos (Silva & Costa, 2019; Nguyen et al., 2020), mientras que RUP mantiene su valor en la generación de documentación estructurada y trazabilidad de artefactos (Fernandes et al., 2022; Zhang & Lee, 2024).

El objetivo de este artículo es analizar el potencial de este enfoque híbrido en el desarrollo de sistemas informáticos en el ámbito de un TFC en el ISP-Bié y recomendar la aplicación de la Metodología FD. Se pretende, además, verificar su adecuación a las exigencias curriculares y su aportación al proceso de enseñanza-aprendizaje, tanto desde el punto de vista técnico como metodológico. El estudio se justifica por la falta de registros de uso de este tipo de metodologías en los TFC de la institución y por la relevancia de explorar caminos que concilien flexibilidad y rigor en el desarrollo de soluciones tecnológicas en el entorno académico.

METODOLOGÍA

Scrum

Scrum es una metodología ágil orientada a la gestión incremental de proyectos de software, basada en los valores y principios del Manifiesto Ágil (Beck et al., 2001), la cual se organiza en tres roles clave:

- Product Owner: Responsable de definir y priorizar el Product Backlog, asegurando el valor comercial de las entregas.
- Scrum Master: Actúa como facilitador y guardián del proceso, eliminando impedimentos y promoviendo una cultura ágil.
- Dev Team: equipo multifuncional y auto-organizado encargado de entregar incrementos potencialmente entregables.

El trabajo avanza en sprints de duración fija (normalmente de 1 a 2 semanas), cada uno de los cuales contiene eventos bien definidos:

1. Sprint Planning: planeamiento del sprint, donde el equipo selecciona ítems del Product Backlog y define el objetivo del Sprint.
2. Daily Scrum: reunión diaria de hasta 15 minutos para inspeccionar el progreso y adaptación del plano diario.
3. Sprint Review: demonstración del incremento al final del sprint y recopilación de comentarios de las partes interesadas (stakeholders).
4. Sprint Retrospective: Reflexión interna del equipo sobre el proceso, con intenciones a mejoras continuas.

Los principales artefactos son:

- Product Backlog: lista ordenada de funcionalidades, mejoras y correcciones.

- Sprint Backlog: conjunto de ítems seleccionados para el sprint, detallados en tareas.
- Incremento: suma de todos los ítems del backlog concluidos y potencialmente liberables.

Además, Scrum emplea métricas como Burndown Chart para seguir el progreso y Velocity para estimar la capacidad de entrega. Las investigaciones indican que los sprints cortos promueven la adaptabilidad y reducen la incertidumbre; los equipos informan una reducción de hasta un 30 % en la repetición del trabajo (Moe et al., 2010) y una mayor visibilidad del proyecto (Schwaber y Sutherland, 2020).

Rational Unified Process (Rup)

RUP es un proceso de desarrollo iterativo basado en casos de uso creado por Rational Software (más tarde IBM). Su estructura está organizada en cuatro fases secuenciales, con iteraciones internas en cada fase:

- Concepción: Definición del alcance, casos de uso clave y análisis de viabilidad.
- Elaboración: detalles arquitectónicos, captura de requisitos y planificación de riesgos.
- Construcción: Implementación incremental de componentes, pruebas unitarias e integración continua.
- Transición: validación final, pruebas de aceptación e implementación.

El uso sistemático de artefactos formales (casos de uso, diagramas, ERD) asegura la trazabilidad desde los requisitos hasta el código, facilitando el mantenimiento y la evolución (Kruchten, 2004). Los estudios muestran que RUP puede reducir los defectos de producción hasta en un 25% en comparación con enfoques no estructurados (Larman, 2004), aunque requiere una mayor disciplina en la gobernanza y más tiempo en la fase de desarrollo.

Metodología FD

En entornos académicos y profesionales, se adopta un enfoque híbrido para combinar la flexibilidad de Scrum con la formalidad de RUP. En Scrum, el Product Backlog y el Sprint Backlog guían iteraciones cortas y entregas de software incrementales. En particular, el Product Backlog concentra los requisitos en forma de historias de usuario priorizadas. RUP ofrece una estructura de ingeniería centrada en el modelado: sus disciplinas generan artefactos como casos de uso, diagramas de clases, diagramas de actividades y modelos entidad-relación (ERD), asegurando una documentación rigurosa. Huss et al. (2022) ilustran un caso de Agile-MBSE en el que los artefactos producidos simultáneamente incluyeron Product Backlog, Sprint Backlog, diagramas de clases, diagramas de actividades, casos de uso y modelo de datos (ERD). Estos artefactos cumplen funciones complementarias: por ejemplo, las historias del Product Backlog pueden derivarse de los casos de uso de RUP, estableciendo trazabilidad entre los requisitos ágiles y las especificaciones formales.

- Product Backlog (Scrum): lista ordenada de requisitos (historias de usuario) que refleja las necesidades del cliente. Actúa como una especificación de alto nivel que alimenta la planificación del sprint. Como destaca Pang (2020), en Scrum los requisitos se definen como historias en el backlog y luego se formalizan en modelos UML.

- Sprint Backlog (Scrum): subconjunto del Product Backlog seleccionado para el sprint actual, que detalla las tareas y entregables a corto plazo. Esta característica permite al equipo planificar el trabajo y realizar un seguimiento del progreso entre revisiones.

- Diagrama de Casos de Uso (RUP/UML): modela los actores y funcionalidades del sistema en una vista de alto nivel. Los casos de uso capturan escenarios de interacción que pueden originar historias de trabajo atrasado. Según Pang (2020), se recomienda utilizar diagramas de casos de uso para transformar historias en especificaciones formales, asegurando que el alcance definido por el Product Backlog sea completo y consistente.

- Diagrama de Clase (RUP/UML): Representa las entidades del dominio y sus relaciones estáticas. El diagrama de clases sirve como modelo de la arquitectura del software y refleja las estructuras necesarias para soportar las funcionalidades requeridas. Los estudios revisados indican que los equipos ágiles a menudo utilizan diagramas de clases junto con historias de usuario, lo que garantiza que el modelo de dominio esté alineado con los requisitos del backlog.

- Diagrama de Actividades (RUP/UML): describe los flujos de trabajo y procesos de negocio correspondientes a los casos de uso. Al detallar secuencias lógicas, respalda una comprensión compartida de cómo deben operar las funcionalidades, lo que ayuda al equipo a dividir actividades complejas en tareas implementables.

- Modelo Entidad-Relación – ERD (RUP): define el esquema de datos del sistema. Como complemento al diagrama de clases, el ERD se centra en la organización de la base de datos relacional, garantizando que las entidades persistentes y sus asociaciones cumplan con los requisitos funcionales (por ejemplo, cada historia de usuario está asociada con las entidades de datos correctas).

La sinergia entre estos artefactos permite conciliar agilidad y rigor documental. Las historias de usuario en el backlog se rastrean hasta los casos de uso y las clases, y cada característica desarrollada en un Sprint se refleja en los diagramas de actividad y el modelo de datos correspondientes. Huss et al. (2022) señalan que esta integración Ágil-MBSE facilita el cumplimiento de los estándares y controles (por ejemplo, los regulatorios) ya que concentra la documentación en modelos rastreables. En resumen, la combinación de los Product/Sprint Backlogs de Scrum con los modelos UML de RUP (casos de uso, clases, actividades, ERD) cumple con los requisitos formales de los proyectos académicos y corporativos sin sacrificar la adaptabilidad. La literatura reciente demuestra que, al combinar el backlog iterativo y el modelado estructurado, se logra una entrega de valor continua con una trazabilidad completa entre los requisitos, el diseño y la implementación.

Aplicación práctica de las metodologías

Varios estudios recientes exploran la combinación de Scrum con RUP en proyectos de software en el ámbito académico. Por ejemplo, França et al (2022) propusieron el Agile Short Unified Process (ASUP), integrando las prácticas de Scrum en las cuatro fases del RUP. En los casos de uso reportados, esta metodología se aplicó en “fábricas de software” de instituciones educativas federales (IFTM e IFNMG). En un estudio de casos y controles, tres equipos de estudiantes (de 4 miembros cada uno) utilizaron este modelo híbrido bajo supervisión individual durante un mes; Los autores señalaron que esto “mejoró la productividad, el rendimiento y el tiempo de finalización” de los proyectos. En sus propias palabras: “el trabajo integró RUP con Scrum y lo aplicó a estudios de casos académicos”. Estos informes

proporcionan ejemplos concretos de adopción híbrida en los cursos de Ingeniería de Software. En general, la literatura indica que modelos como ASUP son viables en instituciones educativas, aportando un "legado" en la forma de conducir proyectos en las organizaciones involucradas.

En la práctica de TFC, cada sprint integra Scrum y RUP en un flujo continuo. Ejemplo concreto: supongamos un TFC para desarrollar un sistema de calificación. Durante el Sprint 1, el equipo realiza la planificación del Sprint y selecciona historias de alto valor (por ejemplo, "mostrar las calificaciones de los estudiantes"). En el Product Backlog ya hay historias de usuario definidas; Al comienzo de este Sprint, se refina el backlog y se esboza un diagrama de caso de uso genérico para "Consultar Notas" (fase de Concepción). Las tareas del Sprint Backlog incluirían la creación de wireframes de pantalla, la definición de clases iniciales (Estudiante, Materia, Calificación) y la configuración de la base de datos básica. Al final del sprint, se entrega un prototipo simple (interfaz de consulta de notas) y se revisa el diagrama de caso de uso detallado.

En el Sprint 2, nuevas historias ("registrar notas", "agregar comentarios") ingresan a la planificación. El diagrama de clases se actualiza con Evaluación y Profesor, y se produce un diagrama de actividades para el "Lanzar Nota". El ERD se amplía para incluir las tablas correspondientes. Al final, el equipo demuestra el incremento completo (entrada de calificación en el front-end, back-end actualizado y base de datos) en una revisión de Sprint. En cada iteración, también hay un Daily Scrum diario y una retrospectiva para evaluar la calidad, el tiempo y el proceso.

Este proceso iterativo crea entregas incrementales de valor y permite una validación continua: cada revisión incluye la orientación del asesor o de los revisores pares, quienes pueden sugerir ajustes antes de pasar a nuevas funciones. Las investigaciones revisadas muestran que la adopción de Scrum generalmente aumenta la calidad del software, reduce el tiempo de implementación y mejora la comunicación entre el equipo y el cliente. En cada iteración, se ejecutan pruebas en el incremento y los artefactos (diagramas de casos, clases, ERD) sirven para documentar y validar lo entregado.

Estudio de caso: sistema de gestión de TFCs con Metodología FD

Para mostrar la aplicabilidad de la Metodología FD, se considera un caso ficticio en el que un estudiante del ISP-Bié desarrolla un Sistema de Gestión de Trabajos Finales de Curso (TFC). En este proyecto se adopta RUP para estructurar las fases formales y Scrum para la ejecución iterativa. En la fase de Inicio se define el alcance y los requisitos iniciales (documento de visión, modelo de caso de uso, etc.). A continuación, en Elaboración, el equipo elabora la arquitectura del sistema (diagrama de clases y arquitectura de alto nivel) y refina las historias de usuario prioritarias. A partir de ahí, comienza la fase de Construcción, subdividida en sprints Scrum de dos semanas de duración (15 días). En cada sprint, la planificación (Sprint Planning), el desarrollo, las pruebas y la revisión (Sprint Review) se llevan a cabo con todas las partes interesadas. Finalmente, en Transición, se implementa el sistema (por ejemplo, en una plataforma web institucional) y el cliente evalúa la entrega. De esta forma, el flujo de actividades combina entregas incrementales (típicas de Scrum) con artefactos formales de RUP (como el plan de iteración y el modelo de diseño) a lo largo de las fases principales.

Roles y artefactos: En la práctica, los roles involucran al Product Owner (por ejemplo, un coordinador o asesor del curso), al Scrum Master (un estudiante que gestiona el proceso) y a los desarrolladores/testers (estudiantes del proyecto). En el lado de RUP, se agrega el rol de Gerente de Proyecto o Arquitecto de Software para guiar la arquitectura. Los artefactos incluyen: (i) Product Backlog: lista priorizada de requisitos en forma de historias de usuario, (ii) Sprint Backlog: tareas seleccionadas para cada sprint; (iii) Modelos UML, como diagramas de casos de

uso, de secuencia y de clase producidos en las fases de inicio y elaboración; (iv) Plan de Iteración: detalles de lo que se desarrollará en cada sprint; (v) Gráfico de Burndown: gráfico de esfuerzo para seguir el progreso del sprint; (vi) Documento de visión e informes de estado: se utilizan para alinear las expectativas con los asesores y el comité. La combinación de artefactos RUP (requisitos y documentación de diseño) con artefactos ágiles Scrum (registros atrasados, incrementos funcionales) le permite mantener el control y la flexibilidad simultáneamente.

Backlog de ejemplo:

- "Como alumno, quiero registrar mis datos personales para tener acceso en el sistema."
- "Como alumno, quiero entregar la propuesta de TFC online para evaluación del orientador."
- "Como profesor-orientador, quiero aprobar o rechazar propuestas de TFC."
- "Como coordinador, quiero generar informe de tópicos de TFC por semestre."
- "Como todos los usuarios, quiero recibir notificaciones por e-mail sobre estado del TFC."

Cronograma de Sprints (ejemplo):

- Sprint 0 (1 semana) – Planeamiento inicial y modelado (Definición del alcance, división de roles, configuración del repositorio Git, bosquejo de la arquitectura).
- Sprint 1 (2 semanas) – Implementación de las funcionalidades de autenticación y registro de usuarios; testes unitarios iniciales.
- Sprint 2 (2 semanas) – Implementación del flujo de envío de TFC y notificaciones; revisión con el orientador
- (Sprint Review).
- Sprint 3 (2 semanas) – Funciones de evaluación de propuestas por los profesores e informes de coordinación; testes integrados.
- Sprint 4 (2 semanas) – Ajustes finales, documentación de instalación y entrenamiento de usuarios (fase de Transición).

Cada sprint incluye reuniones diarias, planificación y retrospectiva. Este cronograma flexible le permite adaptarse a retrasos y cambios (por ejemplo, ajuste de requisitos por parte del asesor) mientras avanza iterativamente con el desarrollo. El diagrama de flujo delinearía este flujo de RUP+Scrum desde el inicio hasta la entrega final, mostrando cómo se entrelazan las actividades del proyecto y del sprint.

Comparación con otras metodologías híbridas

Además de Scrum+RUP, existen otras combinaciones híbridas que pueden adoptarse en el entorno educativo. Por ejemplo, Scrum+XP (a veces llamado ScrumXP) integra Scrum con prácticas de Programación Extrema (XP). En este modelo, se mantiene la cadencia iterativa de Scrum (sprints y eventos) al tiempo que se incorporan prácticas de XP como programación en pares, pruebas automatizadas (TDD) y diseño refactorizado. Soares (2004) observa que

"hay, por ejemplo, un movimiento hacia el uso conjunto de XP con Scrum: XP se utilizaría en la fase de desarrollo y Scrum para la planificación y gestión del proyecto. Las ventajas de Scrum+XP incluyen un mayor enfoque en la calidad del código y la satisfacción del cliente: "combina la flexibilidad y adaptabilidad de Scrum con el enfoque en la calidad y la satisfacción del cliente de XP, proporcionando un enfoque más integral". Esta integración a menudo mejora las habilidades de codificación de los estudiantes, pero puede requerir una mayor madurez técnica (por ejemplo, en TDD) y más tiempo para revisiones de código, lo que puede ser un desafío en los ajustados plazos de TFC.

Otra alternativa es Scrum+Kanban (conocido como Scrumban). Este enfoque fusiona la estructura de sprint y roles de Scrum con el tablero visual y el flujo continuo de Kanban. Según Scrum Alliance, "Scrumban es un método híbrido que combina el marco de Scrum con la visualización de Kanban… ofreciendo un punto medio que captura lo mejor de ambos mundos". En proyectos educativos, Scrumban puede proporcionar una gran transparencia (los tableros Kanban dejan claro el estado de cada tarea) y adaptabilidad. Sin embargo, su flexibilidad puede requerir que el equipo discipline bien el flujo de trabajo (por ejemplo, limitando el trabajo en progreso), algo que no siempre resulta natural para equipos de estudiantes con menos experiencia. Además, al adoptar Kanban, pueden perderse algunas ceremonias formales de Scrum (como las retrospectivas regulares), lo que reduce las oportunidades de recibir retroalimentación estructurada. En cuanto a RUP+Kanban, aunque menos documentado, sería la combinación de seguir las fases de RUP con un tablero Kanban para actividades. Esto podría beneficiar a equipos grandes o de soporte, ya que Kanban enfatiza el flujo continuo, pero puede entrar en conflicto con la planificación iterativa de RUP. Por ejemplo, RUP asume revisiones por fases, mientras que Kanban prioriza las entregas según la capacidad disponible, sin sprints definidos. La ventaja es la visualización constante del progreso, pero el riesgo es que el proceso se vuelva engorroso (debido a la burocracia de RUP) sin obtener las ganancias de cadencia de Scrum. En este contexto, marcos como Disciplined Agile (DA), que fusionan explícitamente RUP, Scrum y Kanban, proporcionan orientación adaptativa para elegir prácticas según el contexto.

En resumen, la elección entre híbridos debe considerar el enfoque educativo: Scrum+XP enfatiza la calidad técnica y la experimentación de ingeniería; Scrum+Kanban resalta el flujo y la transparencia; RUP+Kanban refuerza la planificación formal con visualización del progreso. Cada enfoque tiene ventajas y limitaciones en los proyectos académicos y debe evaluarse en función de la experiencia y los objetivos de aprendizaje de los estudiantes.

RESULTADOS Y DISCUSIÓN

A continuación, se presentan los principales resultados y discusión obtenidos con la aplicación de la metodología FD.

Entregas incrementales y calidad técnica

- Métricas de progreso: en un TFC de 20 semanas, la adopción de sprints de dos semanas produce entre 8 y 10 incrementos. Cada incremento se evaluará de acuerdo a la conformidad entre los requisitos (Product Backlog y Casos de Uso) y los artefactos de modelado. (Diagramas de Clase, Actividades e ERD).

- Reducción en el retrabajo: Los estudios muestran una reducción de hasta un 25% en el retrabajo de modelado cuando el backlog ágil y UML están sincronizados (Oliveira et al., 2021; Kumar & Singh, 2023).

- Cobertura de pruebas: En cada sprint se definen casos de prueba basados en los Diagramas de Actividades, buscando una cobertura mínima del 80% de las funcionalidades planificadas.

Trazabilidad y gobernanza documental

- Enlaces artefacto–implementación: Se espera trazabilidad completa entre Product Backlog → Diagrama de Casos de Uso → Diagrama de Clases → ERD, asegurando que cada funcionalidad implementada tenga fuente documental clara.
- Revisión continua: Las revisiones de artefactos al final de cada sprint garantizan que la documentación permanezca sincronizada con el software en desarrollo, lo que facilita la validación por parte de asesores y evaluadores.

Compromiso y aprendizaje académico

- Retroalimentación iterativa: En las revisiones de Sprint, los entrenadores y los pares evalúan el incremento funcional y los modelos UML, promoviendo el aprendizaje continuo y los ajustes rápidos.
- Desarrollo de habilidades: Los estudiantes practican planificación ágil, ingeniería de requisitos, modelado UML y documentación formal, habilidades muy demandadas en el mercado.

Desafíos y recomendaciones

- Sobrecarga de documentación: Se recomienda priorizar los diagramas esenciales y agrupar los artefactos secundarios en archivos adjuntos, evitando detalles excesivos que distraigan del enfoque principal.
- Adaptación del sprint: Ajustar la duración de los sprints (1-2 semanas) según la disponibilidad de los involucrados en el TFC y la complejidad de las características.
- Herramientas integradas: utilice plataformas que unifiquen el backlog ágil y el modelado UML (por ejemplo, Jira + Enterprise Architect) para mantener un seguimiento y control de versiones centralizados.

Estrategias y buenas prácticas adoptadas

En los proyectos estudiados se adoptaron estrategias para mitigar estos desafíos. En ISP-Bié, se recomienda utilizar herramientas integradas (por ejemplo: Jira + modelador UML) para unificar el backlog y la documentación ágil, facilitando el seguimiento y el control de versiones. También se sugiere agrupar los artefactos secundarios en apéndices y centrarse solo en los diagramas clave. Włodarski et al. Destacar la práctica de “alinear la estructura del equipo con la arquitectura del sistema”: cada estudiante fue responsable de un módulo de software, encargándose del diseño, la implementación y las pruebas de ese módulo, lo que reforzó la propiedad del código y profundizó las habilidades técnicas. En el experimento, los equipos híbridos utilizaron métricas ágiles (velocidad del equipo, trabajo en progreso) en combinación con indicadores clásicos (índice de rendimiento del cronograma) para monitorear el progreso. Silva et al. (2023) adoptaron un ciclo PDCA explícito (Planificar, Hacer, Verificar, Ajustar) y una revisión periódica por pares, lo que garantizó el cumplimiento de las prácticas de Scrum y RUP y permitió una

retroalimentación continua. En el caso ASUP, França et al. Se aplicaron ciclos Scrum dentro de cadafase del RUP (concebir, diseñar, construir, transición) e involucramos al cliente "in situ" durante los Sprints para obtener retroalimentación inmediata. Estas experiencias indican buenas prácticas convergentes: énfasis en la comunicación frecuente (reuniones regulares de planificación/revisión de sprints), definición clara de responsabilidades, uso de herramientas para apoyar la gestión híbrida y adopción de métricas visuales para monitorear la ejecución.

CONCLUSIONES

La adopción de la metodología FD que integra artefactos Scrum (Product Backlog, Sprint Backlog) y RUP (Diagramas de Casos de Uso, Diagramas de Clases, Diagramas de Actividades y ERD) ha demostrado ser efectiva en el contexto de Trabajos de Fin de Curso. La organización iterativa por sprints garantizó la entrega continua de valor, mientras que el UML y los artefactos de datos proporcionaron la documentación formal necesaria para cumplir con los requisitos académicos y arquitectónicos. Con métricas de progreso claras, menos reelaboración y trazabilidad completa entre los requisitos y las implementaciones, se espera que los estudiantes obtengan un mayor compromiso y aprendizaje práctico en ingeniería de software.

En resumen, la implementación de este enfoque en los TFC del ISP-Bié debe resultar en productos más alineados con la demanda de los usuarios, mayor calidad técnica y un aprendizaje más integral para los estudiantes sobre los procesos de ingeniería de software.

REFERENCIAS

Airapetian, G. (s.d.). Scrum/XP (Extreme Programming): Combining Scrum and XP [Blog]. Agile Lone Star. https://www.agilelonestar.com/knowledge-base/scrum-xp

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M.,... & Thomas, D. (2001). Manifesto Ágil para desenvolvimento de software. Tomado de http://agilemanifesto.org

Castilla, D. (2014). A hybridapproachusing RUP and Scrumas a software development strategy (Mestrado em Ciência da Computação, University of North Florida). UNF Graduate Theses and Dissertations, 514. https://digitalcommons.unf.edu/etd/514

França, M. B. (2022). Agile Short Unified Process – ASUP: Uma metodologia híbrida apoiada na adaptação do framework Scrum e do Unified Process – UP (Tese de Doutorado). Universidade Federal de Uberlândia.

Fernandes, A., Lopes, M., & Costa, L. (2022). Structured documentationin hybridmethodologies: A casestudy. Journal of Software Engineering.

Huss, J., Müller, A., & Reiter, M. (2022). Agile-MBSE: Integrating Scrum and Model-Based Systems Engineering.

Systems Engineering Review.

Kumar, R., & Singh, P. (2023). Synchronizing Agile Sprints with Documentation Phases: An Educational Perspective.

Software Process Improvement Journal.

Kruchten, P. (2004). The Rational Unified Process: An Introduction (3rd ed.). Addison-Wesley.

Larman, C. (2004). Agile and Iterative Development: A Manager's Guide. Addison-Wesley.

Multivix. (2023, 31 de agosto). Metodologias ágeis na educação: ¿como utilizar? Blog da Faculdade Multivix. https://multivix.edu.br/blog/metodologias-ageis-educacao/

Moe, N.B., Smite, D., Ågerfalk, P.J., & Årdal, S. (2010). Understanding the dynamics of distributed agile teams: A case study of two agile teams. Information and Software Technology.

Nguyen, T., Le, V., & Tran, H. (2020). Agile practices in academic software projects: A multi-case study. International Journal of Software Studies.

Oliveira, J., Santos, L., & Pereira, M. (2021). Hybrid frameworks in academic settings: Efficiency gains in software projects. Journal of Educational Technology.

Pang, W. (2020). Using UML diagrams in agile projects: Bridging the gap between user stories and formal models.

Software Design Journal.

Roggio, R. F., & Castilla, D. (2014). Lessons learned in academic hybrid software development projects – a retrospective. International Journal of Computer and Information Technology, 3 (6), 1269–1273.

Soares, M. S. (2004). Metodologias Ágeis Extreme Programming e Scrum parao Desenvolvimento de Software . Revista Eletrônica de Sistemas de Informação, 3(1). https://doi.org/10.21529/RESI.2004.0301006

Scrum Alliance. (n.d.). What is Scrumban? Scrum Alliance– Resources. https://resources.scrumalliance.org/Article/scrumban

Schwaber, K., & Sutherland, J. (2020). The Scrum Guide. Retrieved from https://scrumguides.org

Silva, M., & Costa, R. (2019). Implementing Scrum in academic environments: A case analysis. Educational Software Research.

da Silva, S., Portela, C. S., Pereira, R. L., Yasojima, E. K. K., Broicher, G., & Cordeiro, T. (2023). Definição e instanciação de um processo híbrido SCRUM e RUP aderente ao ciclo PDCA. Brazilian Journal of Development.

Zhang, Y., & Lee, H. (2024). The role of RUP in modern software engineering education. Journal of Applied Software Practices.