Ciclo de Vida del Software: Modelo Espiral y Metodología Scrumban

Angel Luis Valdés Sánchez
Maestría en Ingeniería de Software
Universidad Tecnológica de la Mixteca
Huajuapan de León, Oaxaca
angelluis2605@gs.utm.mx

***Resumen*—El ciclo de vida del desarrollo de software (SDLC, por sus siglas en inglés) constituye un marco fundamental para la gestión sistemática de proyectos de software. Este reporte examina los fundamentos generales del SDLC y realiza un análisis comparativo detallado entre dos enfoques metodológicos distintos: el Modelo Espiral, como representante de las metodologías tradicionales orientadas al manejo de riesgos, y Scrumban, como ejemplo de las metodologías ágiles híbridas. A través del análisis de casos de estudio reales, incluyendo el proyecto NASA TReK y implementaciones empresariales de Scrumban, se evalúan las fortalezas, limitaciones y contextos de aplicación óptimos para cada metodología. Los resultados indican que mientras el Modelo Espiral aventaja en proyectos complejos con altos riesgos y requisitos inciertos, Scrumban ofrece flexibilidad superior para equipos que requieren la estructura de Scrum combinada con la fluidez de Kanban.**

I. Introducción

El desarrollo de software moderno enfrenta desafíos cada vez más complejos, caracterizados por requisitos cambiantes, plazos ajustados y la necesidad de entregar valor de manera continua [1]. En este contexto, la selección de un modelo de ciclo de vida del software adecuado se convierte en un factor crítico para el éxito del proyecto [2].

El concepto de ciclo de vida del desarrollo de software ha evolucionado significativamente desde sus inicios en la década de 1970. Mientras que los enfoques tradicionales como el modelo en cascada proporcionaban estructura y predictibilidad, las metodologías modernas han incorporado principios de adaptabilidad y gestión de riesgos [3]. Según estudios recientes, más del 50 % del software desarrollado nunca se utiliza, y más del 70 % no satisface los requisitos del usuario, lo que subraya la importancia de seleccionar el modelo SDLC apropiado [5].

Este reporte se enfoca en dos metodologías representativas: el Modelo Espiral, propuesto por Barry Boehm en 1986 como una respuesta orientada al manejo de riesgos [8], y Scrumban, una metodología híbrida que combina elementos de Scrum y Kanban para proporcionar flexibilidad en entornos ágiles [9]. El objetivo es proporcionar un análisis comparativo basado en evidencia que ayude a los profesionales a tomar decisiones informadas sobre la selección metodológica.

II. Fundamentos del Ciclo de Vida del Desarrollo de Software

II-A. Definición y Propósito del SDLC

El ciclo de vida del desarrollo de software se define como un proceso estructurado que describe las fases involucradas en el desarrollo de sistemas de información [6]. Este marco proporciona una metodología sistemática para planificar, crear, probar y desplegar sistemas de software de alta calidad [1].

Los componentes fundamentales del SDLC incluyen: análisis de requisitos, diseño del sistema, implementación, pruebas, despliegue y mantenimiento [4]. Cada fase tiene objetivos específicos, entregables definidos y criterios de finalización claros [6].

Según Sommerville [31], estas fases constituyen un marco genérico aplicable a la mayoría de los procesos de software:

- **Especificación de requisitos:** se identifican y documentan las necesidades del cliente y las restricciones del sistema.
- **Diseño y desarrollo:** se transforman los requisitos en una representación arquitectónica y detallada que guiará la construcción del software.
- **Implementación y validación:** se construye el sistema y se realizan pruebas para asegurar que cumple con los requisitos definidos.
- **Evolución o mantenimiento:** se realizan modificaciones para corregir errores, mejorar el rendimiento o adaptarse a cambios en el entorno y en los requisitos.

Este modelo genérico proporciona una base común sobre la cual se han desarrollado variantes más específicas como el modelo en cascada, el modelo espiral y metodologías ágiles.

II-B. Evolución de los Modelos SDLC

La evolución de los modelos SDLC refleja la maduración de la ingeniería de software como disciplina. Los modelos tradicionales, caracterizados por enfoques secuenciales y orientados a la documentación, han dado paso a metodologías más flexibles y adaptativas [7].

Estudios contemporáneos indican que aproximadamente el 60 % de los *benchmarks* o criterios actuales se enfocan en la fase de desarrollo dentro del SDLC, mientras que la ingeniería de requisitos y el diseño de software reciben atención mínima con solo 5 % y 3 % respectivamente [4]. Esta disparidad

destaca la necesidad de modelos que equilibren todas las fases del desarrollo.

II-C. Criterios de Selección de Modelos

La selección de un modelo SDLC apropiado depende de múltiples factores, incluyendo la complejidad del proyecto, el nivel de incertidumbre, el tamaño del equipo y las restricciones organizacionales [2]. La investigación indica que diferentes enfoques son más efectivos según las características específicas del proyecto y el contexto organizacional [29].

III. Modelo Espiral: Un Enfoque Tradicional Orientado al Riesgo

III-A. Fundamentos Teóricos

El Modelo Espiral, introducido por Barry W. Boehm en 1986 [8], representa un enfoque evolutivo orientado al manejo de riesgos que combina elementos del modelo en cascada con desarrollo iterativo [8]. El modelo se caracteriza por su representación gráfica en espiral, donde el radio representa el costo acumulativo del proyecto y el progreso angular indica el avance a través del proceso de desarrollo [10].

Boehm diseñó el modelo como un meta-modelo que puede acomodar cualquier proceso de desarrollo apropiado, utilizando la evaluación de riesgos como mecanismo de selección [11]. Esta característica distintiva permite al modelo adaptarse a diferentes contextos de proyecto manteniendo un enfoque sistemático en la identificación y mitigación de riesgos [8].

III-B. Estructura y Fases del Modelo

El Modelo Espiral opera a través de cuatro cuadrantes principales [12]:

1. **Determinación de objetivos, alternativas y restricciones**: Los requisitos del sistema se definen con el mayor detalle posible, incluyendo artefactos para funcionalidad, rendimiento e interfaces [12].
2. **Identificación y resolución de riesgos**: Se evalúan todas las alternativas disponibles para desarrollar un proyecto costo-efectivo, identificando riesgos como falta de experiencia, tecnología nueva o cronogramas ajustados [13].
3. **Desarrollo y pruebas**: Se crea un prototipo del sistema basado en el diseño preliminar, siguiendo el patrón habitual de crear y revisar el diseño, codificar, inspeccionar código y probar el producto [12].
4. **Planificación**: Se desarrollan planes de proyecto, gestión de configuración, pruebas e instalación [13].

III-C. Fases del Ciclo de Vida en el Modelo Espiral

El Modelo Espiral organiza el desarrollo en ciclos sucesivos, donde cada iteración recorre actividades similares a las fases clásicas del SDLC, pero con un énfasis central en la identificación y mitigación de riesgos. La Figura 1 ilustra gráficamente estas fases y su relación con el progreso del proyecto.

- **Determinación de objetivos y requisitos:** en cada vuelta de la espiral se definen metas específicas, considerando funcionalidades a implementar, restricciones técnicas y alternativas viables.
- **Análisis de riesgos y planificación:** se identifican posibles amenazas técnicas, de costos o cronogramas, y se proponen estrategias de mitigación. Esta fase distingue al modelo espiral de otros enfoques secuenciales.
- **Desarrollo y validación:** se crean prototipos o incrementos de software, combinando diseño, implementación y pruebas para validar tempranamente con los interesados.
- **Revisión y preparación de la siguiente iteración:** al finalizar cada ciclo, se validan resultados, se ajusta la planificación y se define el alcance de la próxima vuelta de la espiral.

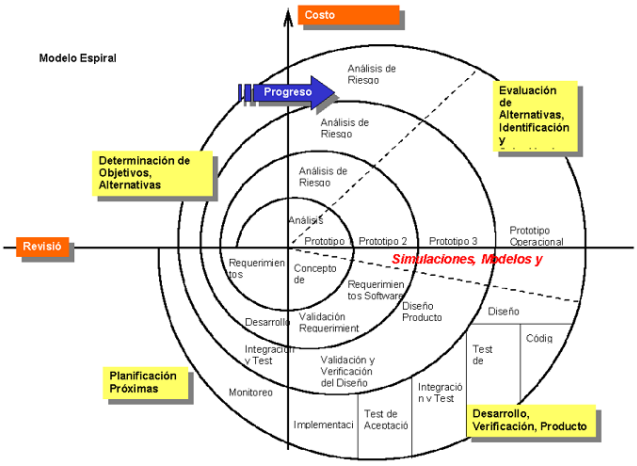

Figura 1. Representación gráfica del Modelo Espiral, mostrando sus fases iterativas orientadas al manejo de riesgos (Figura 2 del artículo de Olivera, 2021 [32]).

Este enfoque asegura que cada fase del ciclo de vida no se aborda de manera rígida y secuencial, sino como parte de un proceso iterativo en el que se reduce la incertidumbre progresivamente [8].

III-D. Casos de Estudio: NASA TReK y TRW-SPS

III-D1. NASA TReK Project: El proyecto TReK de la NASA representa una aplicación exitosa del Modelo Espiral en un entorno de alta complejidad técnica [14]. El proyecto enfrentó numerosos desafíos incluyendo restricciones presupuestarias, limitaciones temporales y el fenómeno de "*requirements creep*" o desviación incontrolable del alcance [14].

La implementación del Modelo Espiral permitió al equipo identificar y mitigar riesgos críticos de manera temprana, resultando en una entrega exitosa a pesar de las restricciones significativas. El enfoque iterativo facilitó la adaptación a cambios en los requisitos regulatorios y técnicos durante el desarrollo [14].

III-D2. TRW Software Productivity System: Boehm utilizó el Modelo Espiral para desarrollar el Sistema TRW-SPS, un proyecto complejo que resultó en 1,300,000 instrucciones de computadora [15]. Se aplicó un estudio de factibilidad llamado "*Round Zero*" para determinar si el proyecto TRW-SPS representaba valor significativo para la organización [15].

El estudio de factibilidad indicó que el proyecto era viable (bajo riesgo), permitiendo que el proyecto se expandiera hacia afuera en la espiral. Los entregables de rondas posteriores incluyeron múltiples pasos repetidos, incluyendo desarrollo de prototipos y, más importante, análisis de riesgos [15].

III-E. Ventajas y Limitaciones

Ventajas del Modelo Espiral:

- Manejo sistemático de riesgos a través de análisis continuo [16]
- Flexibilidad para adaptarse a proyectos con requisitos inciertos [8]
- Capacidad para incorporar elementos de otros modelos según sea necesario [11]
- Enfoque iterativo que permite refinamiento continuo [17]

Limitaciones:

- Complejidad de gestión que puede ser excesiva para proyectos pequeños [16]
- Tiempo considerable requerido para evaluación de riesgos y prototipado [13]
- Necesidad de experiencia especializada en análisis de riesgos [18]
- Potencial para espirales indefinidas sin criterios de finalización claros [12]

IV. Scrumban: Una Metodología Ágil Híbrida

IV-A. Fundamentos Conceptuales

Scrumban emerge como una metodología híbrida que combina la estructura organizacional de Scrum con la flexibilidad de flujo de trabajo de Kanban [19]. Esta aproximación híbrida fue inicialmente desarrollada como un mecanismo de transición para equipos que migraban entre metodologías, pero ha evolucionado hacia un sistema maduro capaz de manejar proyectos complejos y continuos [19].

La metodología se fundamenta en principios clave que incluyen la visualización del flujo de trabajo, limitación del trabajo en progreso (WIP, por sus siglas en inglés), flujo continuo de trabajo y mejora iterativa [20]. Estos principios permiten a los equipos mantener la predictibilidad de Scrum mientras aprovechan la adaptabilidad inherente de Kanban [21].

IV-B. Características Distintivas

Scrumban incorpora elementos específicos de ambas metodologías padre [22]:

De Scrum:

- Reuniones diarias de seguimiento
- Retrospectivas para análisis de desempeño
- Roles definidos incluyendo el *Scrum Master*
- *Sprints* con duraciones flexibles (típicamente no superiores a dos semanas) [22]

De Kanban:

- Tableros visuales para gestión de flujo de trabajo
- Límites de trabajo en progreso
- Flujo continuo sin restricciones temporales estrictas
- Métricas de flujo y tiempo de ciclo [23]

IV-C. Fases del Ciclo de Vida en Scrumban

En contraste con los modelos secuenciales, Scrumban gestiona el ciclo de vida de manera continua y dinámica. Sus fases no son estrictamente lineales, sino que coexisten dentro del flujo de trabajo visualizado en los tableros Kanban. La Figura 2 ilustra este ciclo, donde la planificación, ejecución, revisión y mejora continua se integran de manera iterativa.

- **Planificación ligera:** se definen objetivos inmediatos y se priorizan las tareas a través de un *backlog* flexible.
- **Ejecución iterativa:** las tareas se mueven a través del tablero visual, limitando el trabajo en progreso (WIP) para mantener un flujo sostenible.
- **Revisión y retroalimentación:** al finalizar cada conjunto de tareas o mini-*sprint*, el equipo evalúa el progreso y ajusta prioridades.
- **Mejora continua:** las retrospectivas y métricas de flujo permiten refinar procesos, reducir desperdicios y mejorar la calidad del producto.

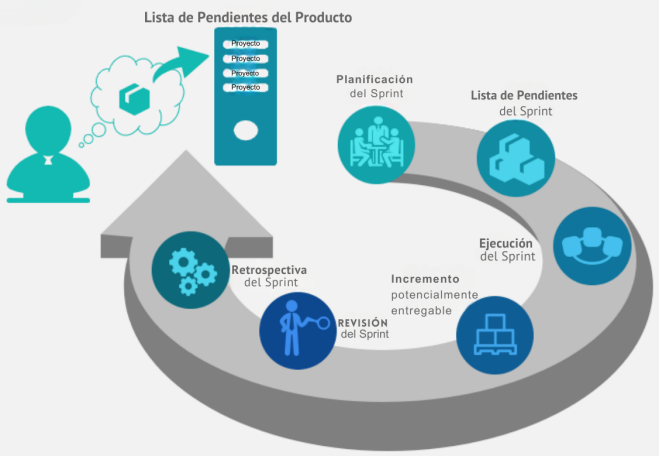

Figura 2. Representación de las fases iterativas en Scrumban (elaboración propia adaptada de recursos en línea).

Estas fases no tienen límites temporales rígidos; más bien, se integran en un ciclo continuo que favorece la adaptabilidad y la entrega constante de valor [19], [20].

IV-D. Casos de Estudio de Implementación

IV-D1. Transición Scrum a Scrumban en Empresa Tecnológica: Un estudio de caso documentó es la transición de Scrum a Scrumban en una empresa de desarrollo de software vietnamita [24]. La investigación utilizó un enfoque de investigación-acción para analizar el proceso de transición, los cambios implementados en el proceso de desarrollo y las mejoras logradas [24].

Los resultados mostraron mejoras significativas en la gestión del flujo de trabajo, reducción en el tiempo de entrega y mayor satisfacción del equipo. La flexibilidad adicional proporcionada por Scrumban permitió al equipo adaptarse mejor a cambios en los requisitos sin comprometer la entrega continua [24].

IV-D2. Implementación con Gamificación en Turquía: Un caso de estudio empírico realizado en una empresa mediana en Turquía demostró la implementación exitosa de Scrumban combinado con elementos de gamificación [25]. El estudio involucró a 30 profesionales trabajando en el mismo proyecto dentro de un centro de investigación tecnológica [25].

Los resultados preliminares indicaron que la integración de elementos de juego con la metodología híbrida Scrumban mejoró la productividad individual y organizacional. El análisis estadístico mostró una diferencia de 0.5 puntos en la escala Likert entre las mediciones pre y post implementación, con un valor t calculado (6.12) superior al valor crítico de tabla (1.79) [25].

IV-D3. Estudios de Formación de Scrumban: La investigación sobre la formación de prácticas Scrumban revela que los equipos ágiles familiarizados con tanto Scrum como Kanban pueden combinar y beneficiarse más de Scrumban [26]. El estudio identifica factores que asisten a los miembros del equipo ágil en la formación exitosa de Scrumban mediante la combinación apropiada de prácticas de Kanban y Scrum [21].

Los hallazgos indican que Scrumban resulta más apropiado que Scrum o Kanban individualmente para ahorrar tiempo, mejorar calidad y minimizar desperdicios [21].

IV-E. Ventajas y Desafíos

Ventajas de Scrumban:

- Flexibilidad superior para adaptarse a cambios de requisitos [20]
- Combinación efectiva de estructura y fluidez [19]
- Reducción del tiempo de ciclo de desarrollo [30]
- Mejora en la transparencia y comunicación del equipo [25]
- Capacidad para manejar trabajos de mantenimiento y desarrollo simultáneamente [27]

Desafíos:

- Requiere experiencia previa en ambas metodologías (Scrum y Kanban) [21]
- Complejidad inicial en la configuración del proceso híbrido [28]
- Necesidad de adaptación cultural organizacional [25]
- Riesgo de inconsistencia en la aplicación sin guías claras [27]

V. Análisis Comparativo y Casos de Estudio

V-A. Análisis Comparativo Metodológico

La comparación entre el Modelo Espiral y Scrumban revela diferencias fundamentales en filosofía, aplicación y contextos óptimos de uso. Mientras el Modelo Espiral prioriza la mitigación sistemática de riesgos a través de análisis exhaustivo y prototipado iterativo [8], Scrumban enfatiza la adaptabilidad continua y la entrega de valor incremental [19].

Gestión de Riesgos: El Modelo Espiral incorpora análisis de riesgos como actividad central en cada iteración [8], mientras que Scrumban maneja riesgos a través de entregas frecuentes y feedback continuo del cliente [20]. Estudios empíricos muestran que el Modelo Espiral es superior para proyectos con riesgos técnicos altos y requisitos inciertos [17], mientras que Scrumban excele en entornos con cambios frecuentes de prioridades [29].

Flexibilidad vs. Estructura: El análisis estadístico de implementaciones muestra que Kanban (componente de Scrumban) supera ligeramente a Scrum en el factor de cronograma [29]. Sin embargo, el Modelo Espiral proporciona mayor predictibilidad en proyectos de largo plazo debido a su enfoque de planificación exhaustiva [14].

V-B. Contextos de Aplicación Óptimos

Modelo Espiral - Contextos Apropiados:

- Proyectos de alta complejidad técnica (sistemas espaciales, software crítico) [14]
- Duraciones de 6 meses a 2 años [12]
- Presupuestos significativos (mayor a $3M) [12]
- Equipos con experiencia en análisis de riesgos [18]
- Requisitos con alta incertidumbre inicial [8]

Scrumban - Contextos Apropiados:

- Equipos de desarrollo ágil con experiencia previa [21]
- Proyectos con requisitos cambiantes frecuentemente [20]
- Entornos que requieren entrega continua [19]
- Organizaciones que buscan balance entre estructura y flexibilidad [25]
- Proyectos de mantenimiento y desarrollo simultáneo [27]

VI. Conclusión

Este análisis comparativo del Modelo Espiral y Scrumban revela que ambas metodologías ofrecen ventajas distintivas según el contexto del proyecto. El Modelo Espiral demuestra superioridad en proyectos de alta complejidad y riesgo, como evidenciado por su aplicación exitosa en proyectos como NASA TReK y TRW-SPS. Su enfoque sistemático en la gestión de riesgos y la capacidad para adaptarse a requisitos inciertos lo convierten en la opción preferida para proyectos críticos de gran escala.

Por el contrario, Scrumban ofrece la flexibilidad necesaria para entornos de desarrollo modernos caracterizados por cambios frecuentes de requisitos y la necesidad de entrega continua. Los casos de estudio analizados demuestran mejoras significativas en satisfacción del equipo, tiempo de ciclo y adaptabilidad organizacional.

La selección entre estas metodologías debe basarse en una evaluación cuidadosa de factores como complejidad del proyecto, tolerancia al riesgo, experiencia del equipo y características organizacionales. Futuras investigaciones deberían enfocarse en desarrollar marcos de decisión cuantitativos que faciliten la selección metodológica basada en características específicas del proyecto.

Los hallazgos de este estudio contribuyen al cuerpo de conocimiento en ingeniería de software al proporcionar evidencia empírica sobre la aplicabilidad contextual de diferentes

modelos SDLC, facilitando decisiones informadas para profesionales y organizaciones en la selección de metodologías de desarrollo de software.

REFERENCIAS

[1] K. L. Dhiman, "Reassessment of Software Development Life Cycle Models," *International Journal of Research in Engineering, Science and Management*, vol. 7, no. 2, pp. 4–10, Feb. 2024.

[2] "Software Development Life Cycle Models-A Comparative Study," *ResearchGate*, July 2020. [Online]. Available: https://www.researchgate.net/publication/346819120_Software_Development_Life_Cycle_Models-A_Comparative_Study

[3] S. S. Shylesh, "A Study of Software Development Life Cycle Process Models," *SSRN*, June 2017. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2988291

[4] K. Wang et al., "Software Development Life Cycle Perspective: A Survey of Benchmarks for Code Large Language Models and Agents," *arXiv preprint arXiv:2505.05283*, May 2025.

[5] "Comparative Analysis of Software Life Cycle Models," *IEEE Conference Publication*, IEEE Xplore. [Online]. Available: https://ieeexplore.ieee.org/document/9362931/

[6] "Research on Contemporary Software Development Life Cycle Models," *American Journal of Computer Science and Technology*, Science Publishing Group, Mar. 2023.

[7] "A Comprehensive Review of Software Development Life Cycle methodologies: Pros, Cons, and Future Directions," *Iraqi Journal for Computer Science and Mathematics*, July 2023.

[8] B. W. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, May 1988.

[9] W. Zayat and O. Senvar, "Framework Study for Agile Software Development Via Scrum and Kanban," *ResearchGate*, July 2020.

[10] "What is the Spiral Model and How is It Used?," *TechTarget*. [Online]. Available: https://www.techtarget.com/searchsoftwarequality/definition/spiral-model

[11] B. W. Boehm, "A spiral model of software development and enhancement," *Semantic Scholar*, 1986.

[12] "The Spiral Model - The Ultimate Guide to the SDLC," *Ultimate SDLC*, 2023.

[13] D. T. Gordon, "Spiral Model in Software Engineering with Case Study," *SlideShare*.

[14] "NASA's TReK project: a case study in using the spiral model of software development," *Communications of the ACM*, vol. 45, no. 4, 2002.

[15] "Spiral Model - an overview," *ScienceDirect Topics*.

[16] "What is Spiral Model in Software Engineering?," *GeeksforGeeks*, Mar. 2018.

[17] "Spiral Model in Software Development: Guide to Risk-Driven Development," *Teaching Agile*, Apr. 2023.

[18] H. C. Nelson, "Applying the spiral model: A case study in small project management," *Wiley Online Library*, Dec. 1998.

[19] "Scrumban: Mastering Two Agile Methodologies," *Atlassian*, Apr. 2025.

[20] "Scrumban: Hybrid Methodology of Scrum and Kanban," *SixSigma.us*, July 2024.

[21] "An Empirical Study of Scrumban Formation based on the Selection of Scrum and Kanban Practices," *ResearchGate*, Dec. 2018.

[22] "Comparison of agile methods: Scrum, Kanban, and Scrumban," *ResearchGate*, Oct. 2016.

[23] "Kanban vs. scrum: which agile are you?," *Atlassian*, Apr. 2025.

[24] "From Scrum to Scrumban: a case study of a process transition," *ResearchGate*, June 2012.

[25] M. Yilmaz and R. V. O'Connor, "A Scrumban Integrated Gamification Approach To Guide Software Process Improvement: A Turkish Case Study," *ResearchGate*, Feb. 2016.

[26] "An Empirical Study of Scrumban Formation based on the Selection of Scrum and Kanban Practices," *ResearchGate*, Dec. 2018.

[27] "Scrumbanfall: An Agile Integration of Scrum and Kanban with Waterfall in Software Engineering," *ResearchGate*, Feb. 2020.

[28] "Scrumban: An Agile Integration of Scrum and Kanban in Software Engineering," *ResearchGate*, Feb. 2020.

[29] F. Ganjeizadeh et al., "A statistical analysis of the effects of Scrum and Kanban on software development projects," *ScienceDirect*, Dec. 2015.

[30] "Agile Marketing Examples & Case Studies," *AgileSherpas*.

[31] I. Sommerville, *Ingeniería de Software*, 10ª ed., Pearson, 2016.

[32] L. C. Olivera, "Modelos de Desarrollo de Software," *Revista de Investigación 3783*, Redalyc, 2021, figura 2. Disponible: https://www.redalyc.org/journal/3783/378366538003/html/