TESINA DE GRADO

LICENCIATURA EN SISTEMAS DE INFORMACIÓN
UNIVERSIDAD AUTÓNOMA DE ENTRE RÍOS
FACULTAD DE CIENCIA Y TECNOLOGÍA

AÑO 2025

GESTIÓN DE REQUERIMIENTOS EN PROYECTOS DE SOFTWARE PARA DESARROLLADORES NOVELES

Una guía práctica para apoyar actividades y productos del ciclo de vida del software

Autor: Sergio Ignacio Garbarino
Director: Ing. Gabriel Piccoli
Revisores: Ing. Vera Mónica, Lic. Bonadeo Pamela, Lic. Cabrera Ignacio

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

RESUMEN

La gestión de requerimientos constituye un componente crítico en el desarrollo de software y adquiere especial relevancia en contextos formativos, donde los equipos suelen contar con experiencia limitada, recursos acotados y plazos restringidos. Esta tesina analiza las prácticas de gestión de requerimientos en proyectos académicos realizados por desarrolladores noveles, identificando dificultades habituales vinculadas a la especificación, validación, trazabilidad y gestión de cambios. Como marco de referencia se adopta la norma ISO/IEC 29110, diseñada para pequeñas organizaciones y equipos de baja madurez en procesos, cuyo perfil básico proporciona actividades y productos de trabajo adaptables a entornos educativos.

A partir de un estudio empírico, se mapean los problemas detectados con los productos de la norma y se evalúan herramientas tecnológicas que favorecen su adopción. El resultado es una guía práctica que propone actividades mínimas, plantillas y herramientas accesibles para fortalecer la disciplina de requerimientos desde etapas tempranas de formación. La propuesta busca mejorar la documentación, comunicación y trazabilidad en los equipos estudiantiles, contribuyendo a elevar la calidad de los proyectos y a favorecer la transición hacia prácticas profesionales más maduras.

Palabras clave: gestión de requerimientos, norma ISO/IEC 29110, productos de trabajo, trazabilidad, buenas prácticas, validación.

INTRODUCCIÓN

Contexto

La ingeniería de software ha evolucionado como una disciplina clave en el desarrollo de soluciones informáticas eficientes, mantenibles y alineadas con los objetivos del usuario. Dentro de este campo, la gestión de requerimientos representa una actividad crítica, ya que define qué debe hacer un sistema antes de que se inicien el diseño y la implementación (Pressman & Maxim, 2021).

En el ámbito académico y de formación profesional, los desarrolladores noveles[1] que abordan sus primeros proyectos reales suelen enfocarse en la programación como actividad central, relegando la especificación, validación y documentación de requerimientos. Esta inclinación no implica desinterés, sino que obedece a la limitada experiencia técnica y metodológica propia de las etapas iniciales, así como al deseo de lograr resultados tangibles con rapidez.

Hunt y Thomas (2019), abordan esta problemática, al advertir que los requerimientos tienden a emerger de manera incompleta o ambigua desde el inicio, estando frecuentemente "enterrados bajo capas de suposiciones, malentendidos y política" (p. 202). Los autores subrayan que los desarrolladores noveles frecuentemente cometen el error de implementar directamente una solución basada en una declaración inicial del problema, sin explorar si esa necesidad representa realmente un requerimiento bien definido. Esta actitud puede conducir a soluciones técnicamente funcionales, pero alejadas de las verdaderas expectativas o necesidades del usuario.

Existen múltiples enfoques para organizar el desarrollo de software. Algunos modelos de procesos, como los tradicionales (cascada, modelo en V) o los más contemporáneos (*Scrum*, *XP*), difieren en su nivel de formalidad, documentación y trazabilidad. En muchos entornos de formación y en equipos pequeños, las metodologías ágiles son adoptadas con entusiasmo; sin embargo, la falta de experiencia práctica o un conocimiento superficial de sus principios pueden llevar a que prácticas esenciales, como la gestión sistemática de requerimientos o el mantenimiento de la trazabilidad, pierdan protagonismo frente a actividades más visibles como la codificación o la implementación de interfaces.

En este contexto, la norma ISO/IEC 29110 se presenta como una alternativa particularmente útil. Diseñada específicamente para pequeñas organizaciones y equipos

[1] En este trabajo se utiliza el término desarrolladores noveles para referirse a estudiantes o egresados recientes que abordan sus primeras experiencias prácticas en proyectos de *software*. Este perfil se caracteriza por una sólida formación teórica pero limitada experiencia aplicada, lo que genera desafíos específicos en la gestión de requerimientos y en la adopción de prácticas profesionales (McConnell, 2004; Hunt & Thomas, 2019; Pressman & Maxim, 2020)

con poca experiencia en procesos formales, esta norma propone un conjunto de actividades clave y productos asociados que permiten dar estructura al desarrollo sin imponer una carga desmedida. Su enfoque progresivo facilita la adopción de buenas prácticas en proyectos educativos y en contextos donde los recursos disponibles son inherentemente limitados, ofreciendo una guía adaptable y eficiente.

Objetivos

El presente trabajo persigue un objetivo general y un conjunto de objetivos particulares, que se detallan a continuación.

Objetivo General

Desarrollar una guía práctica para la gestión y trazabilidad de requerimientos en proyectos de desarrollo de software, destinada a programadores noveles.

Objetivos particulares

- Investigar cómo se describe la gestión de requerimientos en proyectos de software en la bibliografía y estándares relevantes.
- Investigar métodos y actividades que realizan los programadores noveles.
- Identificar problemas comunes y oportunidades de mejora en la gestión de requerimientos de proyectos de desarrollo de software que realizan los programadores noveles.
- Identificar las actividades y productos de trabajo de la Norma ISO/IEC 29110 que aborden los problemas comunes identificados en la gestión de requerimientos de proyectos de programadores noveles.
- Identificar y evaluar herramientas que soporten productos de trabajo de la Norma ISO/IEC 29110 relevantes para los problemas comunes señalados
- Desarrollar una guía práctica que integre herramientas junto con directrices relevantes de la Norma ISO/IEC 29110 para apoyar la gestión de requerimientos.

Organización de la tesina

El presente trabajo se estructura en siete capítulos, organizados de manera progresiva desde los fundamentos teóricos hasta la propuesta práctica final.

El Capítulo I introduce los procesos de desarrollo de software y presenta el problema de investigación, junto con su justificación y objetivos generales.

Los Capítulos II, III y IV conforman el marco teórico, abordando la ingeniería de requerimientos, la gestión de proyectos y el análisis de la norma ISO/IEC 29110. Este

bloque incluye también una revisión de antecedentes relevantes que sustentan la elección del marco normativo como referencia conceptual para esta investigación.

El Capítulo V describe el enfoque metodológico adoptado, incluyendo el diseño de la investigación, el instrumento de recolección de datos y el criterio de selección de herramientas tecnológicas.

El Capítulo VI, titulado Desarrollo, presenta los resultados obtenidos a partir del análisis empírico, organizados según ejes temáticos y dificultades comunes detectadas.

El Capítulo VII integra la guía práctica orientada a equipos estudiantiles. Se inicia con una fundamentación que explica su justificación, principios y criterios de diseño, y luego desarrolla las secciones prácticas: actividades mínimas, documentos y plantillas sugeridas, herramientas de apoyo y orientaciones de uso. Finalmente, se proponen estrategias para su aplicación y validación en contextos educativos.

Finalmente, la sección de Conclusiones y perspectivas, recupera los objetivos planteados, sintetiza los aportes generales del trabajo, reconoce sus limitaciones y propone líneas futuras de investigación.

CAPÍTULO I: PROCESOS DE DESARROLLO DE SOFTWARE

En este capítulo se abordan los fundamentos del desarrollo de software desde una perspectiva orientada a procesos. Se introducen las actividades típicas del ciclo de vida del software, los distintos modelos de procesos que guían su ejecución, y se plantea el concepto de desarrollo profesional como una forma de garantizar calidad, sostenibilidad y adaptabilidad en los productos generados. Esta base conceptual es esencial para comprender, en capítulos posteriores, la importancia de la gestión de requerimientos y el papel que desempeña en la formación de desarrolladores noveles.

1.1 Procesos del ciclo de vida del software

Los sistemas de software no surgen espontáneamente; su desarrollo implica un conjunto ordenado de actividades que van desde la concepción inicial del producto hasta su retiro definitivo. Estas actividades constituyen lo que se conoce como ciclo de vida del software (*Software Development Life Cycle*, SDLC), que permite estructurar y gestionar el desarrollo de forma controlada.

Sommerville (2011) define los procesos de software como “un conjunto estructurado de actividades requeridas para desarrollar un sistema de software”, y señala que dichas actividades incluyen desde la especificación de requerimientos hasta su validación, implementación, mantenimiento y evolución.

Las fases típicas del ciclo de vida incluyen:

- Especificación**:** definición y análisis de los requerimientos funcionales y no funcionales del sistema.
- Diseño y desarrollo: construcción de la arquitectura del sistema y codificación de los componentes.
- Validación**:** verificación de que el software cumpla con los requisitos establecidos.
- Evolución**:** mantenimiento, adaptación o mejora del sistema una vez entregado.

Pressman y Maxim (2020) destacan que organizar el desarrollo en fases no solo mejora la previsibilidad del proyecto, sino que también permite aplicar prácticas de aseguramiento de la calidad, control de riesgos y trazabilidad. De este modo, el SDLC actúa como una guía que contribuye a evitar problemas como la entrega tardía, el incumplimiento de requisitos o la baja calidad del producto final.

1.2 Modelos de procesos y metodologías

Los modelos de procesos de software proporcionan marcos conceptuales que organizan las actividades del ciclo de vida en esquemas comprensibles y repetibles. Sommerville (2015) y el *Project Management Institute* (2021) sostienen que no existe un

único modelo aplicable a todos los proyectos, ya que la elección depende del tipo de sistema, el entorno organizacional y la complejidad del proyecto. Además, cada modelo presenta fortalezas y limitaciones que deben considerarse al momento de su adopción (Wiegers & Beatty, 2013).

A continuación se explican los modelos tradicionales y enfoques ágiles.

1.2.1 Modelos tradicionales

De acuerdo con Sommerville (2015), los enfoques tradicionales o plan-driven se caracterizan por una planificación exhaustiva y estable al inicio del proyecto, lo que permite a las organizaciones grandes controlar tiempos, costos y entregables. En la terminología de la Guía de los fundamentos para la dirección de proyectos (Guía del PMBOK®, *Project Management Institute* [PMI], 2021), corresponden a los enfoques predictivos, que asumen estabilidad en los requisitos y altos costos de cambio. Estos modelos han sido aplicados históricamente en proyectos de gran escala y alta criticidad (Royce, 1970; Boehm, 1986; Balci, 1998).

Entre los enfoques predictivos, la literatura especializada destaca los siguientes modelos:

Modelo en cascada: propuesto por Royce (1970), establece una secuencia lineal de etapas. Su simplicidad lo hace fácil de comprender, aunque presenta rigidez ante cambios en los requerimientos una vez iniciadas las fases posteriores.

Modelo en espiral: introducido por Boehm (1986), incorpora iteración y gestión de riesgos, resultando adecuado para proyectos grandes y complejos.

Modelo en V: enfatiza la verificación y validación en cada etapa del ciclo, y se utiliza especialmente en contextos donde la calidad es crítica (Balci, 1998).

1.2.2 Enfoques ágiles

A partir de la publicación del Manifiesto Ágil (Beck et al., 2001), surgieron metodologías que promueven la entrega incremental, la interacción constante con el cliente y la flexibilidad ante el cambio. Métodos como *Scrum*, *Extreme* Programming (XP) o herramientas como *Kanban* han ganado popularidad, especialmente en contextos de innovación, desarrollo rápido o equipos pequeños.

No obstante, Sommerville (2011) advierte que la adopción de metodologías ágiles no debe implicar el abandono de prácticas fundamentales como la documentación, la trazabilidad o la gestión sistemática de requerimientos, ya que estas resultan esenciales para el mantenimiento y la evolución futura del software.

En entornos educativos, como los que involucran a desarrolladores noveles, las metodologías ágiles suelen ser valoradas por su flexibilidad y orientación práctica. Sin

embargo, su implementación puede ser parcial o interpretada de manera informal, lo que representa tanto una oportunidad de aprendizaje como un área que merece atención particular. Esta informalidad metodológica, si bien comprensible en un entorno de formación, puede afectar la gestión sistemática de los requerimientos, dificultando su trazabilidad, validación y evolución durante el desarrollo del proyecto.

1.3 El desarrollo profesional de software

El desarrollo profesional de software trasciende la programación individual, ya que persigue la construcción de sistemas confiables, mantenibles y útiles en contextos organizacionales. Como indica Sommerville (2014), “la ingeniería de software está destinada a apoyar el desarrollo profesional más que la programación individual”, ya que requiere aplicar procesos sistemáticos y técnicas rigurosas que aseguren tanto el funcionamiento del producto como su calidad a lo largo de su ciclo de vida.

Este enfoque se sustenta en tres pilares fundamentales. En los apartados siguientes se describen en detalle.

1.3.1 Equipos interdisciplinarios

La participación de profesionales con diferentes antecedentes (ingenieros, administradores, docentes o investigadores) enriquece la identificación de requerimientos y reduce la posibilidad de omitir necesidades críticas. Sommerville (2016) enfatiza que el diseño conceptual “debe ser siempre un proceso de equipo que involucre a personas con diferentes antecedentes” (p. 564), destacando la importancia del trabajo colaborativo para alcanzar soluciones integrales.

1.3.2 Planificación detallada

La planificación estructurada constituye un rasgo inherente al desarrollo profesional. En los enfoques predictivos, la mayor parte de las actividades se planifican al inicio, lo que reduce incertidumbre y facilita la gestión de riesgos cuando los requisitos son estables (PMI, 2021). En cambio, los enfoques ágiles (ver sección 1.2.2) priorizan una planificación incremental y continua, lo que permite adaptarse mejor a cambios en los requerimientos (Sommerville, 2016; Boehm & Turner, 2004). En la práctica, los proyectos profesionales suelen combinar ambos estilos para equilibrar previsibilidad y flexibilidad.

1.3.3 Prácticas y herramientas de soporte

El desarrollo profesional requiere la aplicación sistemática de metodologías y herramientas que garanticen la calidad y sostenibilidad del producto. Esto incluye definir un marco metodológico (tradicional, ágil o híbrido), usar sistemas de control de versiones

y gestión de configuración para asegurar trazabilidad, y adoptar integración continua y pruebas automatizadas. Asimismo, la documentación y la gestión del cambio cumplen un rol central para garantizar que el software evolucione de manera controlada y cumpla con las necesidades del usuario.

En conjunto, estas prácticas diferencian el desarrollo profesional de la programación informal, al promover la construcción de sistemas robustos, sostenibles y alineados con los objetivos organizacionales.

1.4 Aplicabilidad de los procesos en contextos reales y educativos

Aunque los procesos de desarrollo de software y sus modelos están ampliamente documentados y estandarizados en la literatura técnica, su implementación práctica suele presentar dificultades, sobre todo en entornos con recursos limitados, equipos pequeños o integrantes con poca experiencia (Pressman & Maxim, 2020). Como señalan los autores, “la ingeniería de software es tanto una ciencia como un arte”, y su ejecución efectiva requiere no solo conocimiento técnico, sino también juicio profesional, experiencia y adaptación al contexto. Esto implica que no basta con conocer las etapas del ciclo de vida o las actividades propuestas por un modelo determinado: también es necesario contar con la madurez organizacional y las condiciones operativas que posibiliten su implementación.

En el ámbito educativo, los desarrolladores noveles enfrentan múltiples desafíos simultáneos: deben aprender nuevas tecnologías, adaptarse al trabajo en equipo y comprender dominios de problemas que con frecuencia les resultan desconocidos, mientras intentan adoptar un enfoque disciplinado bajo restricciones de tiempo. McConnell (2004) señala que los graduados recientes suelen poseer “rico conocimiento teórico, pero pobre en el saber práctico que implica construir programas de producción” (p. 19), y que dicho conocimiento práctico suele adquirirse lentamente mediante la experiencia y la interacción con programadores más experimentados.

Esta brecha entre teoría y práctica se ve acentuada por la rápida evolución tecnológica: cada nuevo proyecto puede diferir significativamente de los anteriores, empleando técnicas o herramientas no familiares como servicios web o configuraciones de sistemas de aplicación, lo que dificulta aplicar efectivamente la experiencia previa (Sommerville, 2016). Además, la transición hacia una disciplina de ingeniería de software más rigurosa se ve obstaculizada por la inercia de prácticas anteriores y la necesidad de adaptación cultural, lo que exige procesos flexibles, mejores herramientas y una fuerte base educativa para evitar repetir errores del pasado (Pressman, 2014).

En este sentido, incluso con marcos teóricos robustos, la aplicación consistente de buenas prácticas en proyectos académicos puede resultar compleja. Wiegers y Beatty

(2013) afirman que, aun reconociendo la importancia de una adecuada gestión de requerimientos, muchas organizaciones encuentran dificultades para aplicarla de manera efectiva debido a limitaciones de tiempo, habilidades, cultura o comprensión. De igual manera, en el ámbito académico la implementación completa y sistemática de estos procesos se ve condicionada por factores propios de la formación inicial, tales como tiempos acotados, experiencia incipiente y la simultaneidad entre el aprendizaje técnico y la ejecución del proyecto. Tal como señalan McConnell (2004) y Sommerville (2016), los desarrolladores noveles enfrentan una brecha entre teoría y práctica que solo se supera gradualmente mediante experiencia guiada, lo que explica las dificultades para aplicar procesos formales de ingeniería de software durante sus primeras experiencias académicas.

En coherencia con lo expuesto en este capítulo respecto a los modelos de proceso y al desarrollo profesional, la formación en este ámbito requiere integrar fundamentos conceptuales con experiencias prácticas guiadas, promoviendo reflexión sobre problemas reales y adaptación progresiva. Este enfoque es coherente con el aprendizaje experiencial y progresivo descrito por McConnell (2004), y con la necesidad de adquirir competencias mediante práctica estructurada en contextos reales o simulados planteada por Sommerville (2016) y Pressman y Maxim (2020).

Comprender los procesos de desarrollo de software y los modelos que los estructuran permite a los desarrolladores noveles adoptar una visión más organizada de su trabajo. Esta perspectiva no solo mejora la calidad técnica de los productos generados, sino que también fortalece la formación profesional, al vincular la práctica con marcos metodológicos que facilitan la planificación, el control y la mejora continua de los proyectos. Esta base conceptual servirá de sustento para el estudio de la gestión de requerimientos, que será abordada en el próximo capítulo.

CAPÍTULO II: INGENIERIA DE REQUERIMIENTOS

El capítulo anterior abordó los fundamentos del desarrollo de software desde una perspectiva orientada a procesos, destacando la importancia de aplicar marcos metodológicos que organicen las actividades del ciclo de vida y promuevan la calidad del producto. Uno de los pilares más críticos y a la vez más desafiantes es la gestión adecuada de los requerimientos, dado que constituyen la base de todas las decisiones posteriores en el proyecto.

Sommerville (2011) define los requerimientos como “descripciones de lo que el sistema debe hacer, las restricciones bajo las que debe operar y las propiedades que debe poseer” (p. 83). En la práctica, los requerimientos indican qué debe hacer el sistema y bajo qué condiciones debe hacerlo, influyendo en el diseño, la implementación y la validación del producto.

Cuando los requerimientos se comprenden de forma incompleta o se gestionan de manera inadecuada, aumenta significativamente el riesgo de fallos, sobrecostos y retrasos.

Para abordar esta complejidad surge la ingeniería de requerimientos, disciplina que sistematiza el descubrimiento, documentación, validación y gestión de los requerimientos, asegurando que sean claros, trazables y comprensibles tanto para los clientes como para los desarrolladores (Wiegers & Beatty, 2013). Según Pressman y Maxim (2020), la ingeniería de requerimientos constituye un pilar esencial del desarrollo profesional de software, pues permite construir una visión compartida entre las partes involucradas y facilita la trazabilidad durante todo el ciclo de vida del sistema.

En los apartados siguientes se profundizará en los conceptos, actividades y productos asociados a la ingeniería de requerimientos, analizando su relevancia en la formación de desarrolladores noveles y en proyectos de pequeña escala, como los que constituyen el objeto de estudio de esta investigación.

2.1 Clasificación de los requerimientos

Un sistema puede presentar distintos tipos de requerimientos, los cuales se organizan según diferentes criterios, siendo los más comunes los siguientes (Sommerville, 2011):

Requerimientos funcionales: describen los servicios que el sistema debe proporcionar, las reacciones ante determinadas entradas y cómo debe comportarse en situaciones particulares.

Requerimientos no funcionales: especifican criterios que pueden usarse para juzgar el funcionamiento de un sistema, tales como rendimiento, usabilidad, confiabilidad

o seguridad. Muchas veces, estos requerimientos son más críticos que los funcionales, ya que definen las condiciones bajo las cuales el sistema será verdaderamente útil.

Requerimientos de dominio: son requerimientos específicos que emergen del conocimiento experto y las características propias del área de aplicación o sector de negocio donde operará el sistema. Pueden incluir regulaciones obligatorias, normativas legales, terminología específica del sector, políticas internas, reglas de negocio intrínsecas o flujos de trabajo particulares. Su correcto entendimiento es fundamental, ya que definen el contexto en el que el software debe funcionar y, a menudo, requieren una inmersión profunda en el conocimiento del usuario o del negocio.

Asimismo, en el campo de la ingeniería de requerimientos se distingue entre requerimientos explícitos e implícitos o tácitos.

Los primeros corresponden a necesidades expresadas de manera directa por los usuarios o partes interesadas, constituyendo la base inicial para la especificación formal del sistema.

En contraste, los requerimientos implícitos o tácitos no se enuncian de forma directa, pero se asumen como parte del sistema; suelen considerarse “obvios” o “dados por hecho” por los usuarios, aunque no siempre se verbalicen durante el proceso de captura de requerimientos.

La identificación de este tipo de requerimientos resulta crítica, dado que su omisión constituye una fuente frecuente de defectos y malentendidos en proyectos de software (Sommerville, 2011; Wiegers & Beatty, 2013)

Robertson y Robertson (2006) presentan el marco Volere[2], el cual extiende esta clasificación al incluir otras categorías relevantes para el éxito de un proyecto:

Restricciones de proyecto: condiciones impuestas por factores externos como presupuesto, plazos, recursos disponibles o regulaciones.

Restricciones de diseño: especificaciones que condicionan cómo deberá construirse el sistema, como el uso de ciertas tecnologías o plataformas.

Impulsores del proyecto: fuerzas estratégicas o comerciales que motivan el desarrollo del sistema, incluyendo metas del negocio o necesidades de los interesados.

Problemas del proyecto: factores de riesgo o incertidumbre que pueden afectar el desarrollo, y que deben considerarse al definir el alcance y la estrategia del sistema.

[2] Volere:es un marco de ingeniería de requerimientos que ofrece plantillas y guías para documentar necesidades, restricciones y factores estratégicos de un proyecto (Robertson & Robertson, 2006)

2.2 Obtención de requerimientos

La obtención de requerimientos (denominada en la literatura anglosajona *requirements elicitation*, a veces traducida como elicitación[3]) es el proceso mediante el cual se identifican las necesidades, deseos, objetivos y restricciones de las diversas partes interesadas (*stakeholders*) en un proyecto de software. Se trata de una fase crítica dentro de la ingeniería de requerimientos, dado que los errores y ambigüedades introducidos en este punto suelen manifestarse en etapas posteriores del ciclo de desarrollo y resultan significativamente más costosos de corregir (Sommerville, 2011; Pressman & Maxim, 2020).

Sommerville (2011) sostiene que la obtención no consiste simplemente en "recolectar" información, sino en comprender lo que los usuarios dicen, lo que quieren decir, y lo que realmente necesitan. Esto incluye no solo requerimientos explícitos, sino también necesidades implícitas, restricciones del entorno y expectativas que pueden no estar claramente formuladas.

Entre las técnicas más empleadas para la obtención de requerimientos se encuentran aquellas que combinan enfoques directos con métodos de análisis documental y exploración práctica.

Las entrevistas, tanto estructuradas como abiertas, permiten indagar en profundidad sobre las necesidades de los usuarios y otros *stakeholders* (Sommerville, 2011; Robertson & Robertson, 2012).

Los cuestionarios o encuestas son útiles para recopilar datos de un gran número de personas, aunque con menor nivel de detalle individual (Wiegers & Beatty, 2013).

Asimismo, los talleres de trabajo colaborativos, como las sesiones de *Joint Application Development* (JAD), facilitan la construcción compartida de requerimientos entre múltiples partes interesadas (Wiegers & Beatty, 2013). La metodología JAD consiste en reuniones estructuradas en las que usuarios finales, desarrolladores y analistas discuten conjuntamente los requerimientos y aspectos de diseño del sistema.

La observación directa del entorno de trabajo resulta especialmente valiosa para comprender procesos reales y detectar necesidades no verbalizadas (Sommerville, 2011).

De forma complementaria, la revisión de documentos existentes, manuales, reportes o formularios, aporta información sobre prácticas vigentes y restricciones del sistema (Pohl, 2010).

Finalmente, el prototipado de baja fidelidad y el uso de historias de usuario o escenarios favorecen la discusión temprana, la validación de supuestos y la obtención de

[3] En español, el término "elicitación" no está reconocido por la Real Academia Española, que lo considera un anglicismo innecesario (Real Academia Española, s. f.). Sin embargo, se ha difundido ampliamente en la literatura técnica como traducción de *requirements elicitation.*

retroalimentación rápida antes de definir requerimientos formales (Wiegers & Beatty, 2013).

En contextos reales, los requerimientos no son elementos fijos ni completamente conocidos desde el inicio, sino que se negocian, evolucionan y a menudo se descubren parcialmente en cada iteración (Wiegers & Beatty, 2013). Esta realidad contrasta significativamente con la visión de los modelos de desarrollo de software tradicionales, que a menudo asumen la posibilidad de una definición completa y estable de los requerimientos al comienzo del proyecto.

Por el contrario, la perspectiva contemporánea de la ingeniería de requerimientos subraya que la obtención no constituye una actividad puntual y cerrada, sino un proceso progresivo y colaborativo, en el que las necesidades y restricciones se aclaran, ajustan y priorizan a medida que avanza el proyecto (Sommerville, 2011; Wiegers & Beatty, 2013).

Este proceso requiere de iteraciones, validaciones y reajustes continuos, ya que los requerimientos evolucionan conforme se profundiza la comprensión del dominio, cambian las condiciones del entorno o emergen nuevas necesidades (Pohl, 2010).

En este enfoque dinámico, el rol del analista de requerimientos trasciende lo técnico y se vuelve también comunicacional y estratégico, dado que debe facilitar el diálogo entre usuarios, desarrolladores y demás partes interesadas, promoviendo una visión compartida del sistema (Wiegers & Beatty, 2013).

2.3 Documentación de requerimientos

Una vez identificados, los requerimientos deben ser documentados de manera clara, precisa y accesible para todos los actores involucrados en el proyecto.

La documentación cumple múltiples funciones: sirve como contrato entre las partes, guía el desarrollo del sistema, respalda las actividades de verificación y validación, y actúa como referencia para el mantenimiento futuro.

La verificación consiste en comprobar que el producto ha sido construido conforme a sus especificaciones, respondiendo a la pregunta: “¿Se construyó correctamente el producto?”.

En cambio, la validación asegura que el sistema final cumpla con las necesidades y expectativas de los usuarios, respondiendo a la pregunta: “¿Se construyó el producto correcto?” (Pressman & Maxim, 2020; Sommerville, 2011).

Según Pressman y Maxim (2020), la documentación eficaz de requerimientos debe ser comprensible y libre de ambigüedades, de modo que facilite estas actividades.

Sommerville (2011) agrega que la trazabilidad es un atributo clave: la posibilidad de seguir cada requerimiento desde su origen hasta su implementación y prueba, y

viceversa, lo que garantiza un mayor control sobre el producto durante todo su ciclo de vida.

En la práctica, el modo en que se documentan los requerimientos puede variar significativamente según el enfoque de desarrollo adoptado. En este sentido, se reconocen tres estilos predominantes:

1. Documentación completa anticipada: Este estilo es característico de enfoques de desarrollo basados en planificación detallada, como el modelo en cascada o el modelo en V. En estos casos, los requerimientos se documentan en su totalidad antes de comenzar con el diseño e implementación, generalmente a través de un documento formal de especificación de requerimientos (SRS[4]). Este documento suele organizarse siguiendo normas como IEEE 830-1998[5], e incluye tanto requerimientos funcionales como no funcionales, restricciones, supuestos y criterios de aceptación.

   Sommerville (2011) señala que este enfoque resulta adecuado cuando los requerimientos son estables y el dominio está bien comprendido, ya que permite una planificación rigurosa desde el inicio del proyecto y una especificación exhaustiva previa al desarrollo. Esta visión coincide con la perspectiva de Wiegers y Beatty (2013), quienes destacan que en estos contextos los requerimientos se documentan por completo antes de la construcción del sistema (p. 73).
2. Documentación progresiva o incremental: En contraste, los enfoques ágiles priorizan la adaptabilidad y la colaboración continua con el cliente. En estos casos, la documentación de requerimientos se caracteriza por ser concisa y elaborarse de manera incremental, reemplazando los documentos extensos por historias de usuario, tarjetas de *backlog*[6] y criterios de aceptación, que se elaboran únicamente cuando son necesarios para cada iteración de trabajo. Como señalan Pressman y Maxim (2020), “en los modelos ágiles, los requerimientos se representan típicamente como historias de usuario mantenidas en un *backlog* y se elaboran a medida que el desarrollo progresa” (p. 44). Este enfoque convierte la especificación en un proceso continuo (Sommerville, 2011, p. 120) y se alinea con la filosofía de requerimientos “justo a tiempo” (Wiegers & Beatty, 2013, p. 73).
3. Enfoques híbridos: Estos estilos no son mutuamente excluyentes. En la práctica, es habitual la adopción de esquemas híbridos, en los cuales los requerimientos generales

[4] Software Requirements Specification (SRS): documento formal que detalla los requerimientos funcionales y no funcionales de un sistema.

[5] IEEE 830-1998 es un estándar del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) que proporciona una guía para redactar especificaciones de requerimientos de software (SRS).

[6] Conjunto ordenado y priorizado de requerimientos, funcionalidades, mejoras y tareas pendientes que orientan el trabajo del equipo en proyectos gestionados con metodologías ágiles.

se establecen al inicio y se refinan progresivamente durante el desarrollo (Sommerville, 2016; Wiegers & Beatty, 2013).

Resulta fundamental que, independientemente del enfoque adoptado, la documentación preserve la trazabilidad, facilite la validación y sirva como guía efectiva para el desarrollo.

Documentación segmentada según tipo de lector: además de los enfoques anteriores, la documentación presenta variaciones según la audiencia destinataria. De este modo, un mismo requerimiento se expresa con diferentes grados de detalle o formatos, según las necesidades de cada rol:

- Usuarios finales o clientes: se privilegia el lenguaje natural, centrado en los objetivos del negocio y sin tecnicismos innecesarios.
- Diseñadores y desarrolladores: se requiere mayor precisión técnica, especificaciones formales o modelos que guíen la implementación.
- Testers o aseguradores de calidad: necesitan requerimientos verificables, con criterios de aceptación explícitos que permitan construir casos de prueba.

Wiegers y Beatty (2013) señalan que “la especificación adopta diferentes formas según la audiencia: diagramas, modelos, narrativas o historias de usuario” (p. 78). Esta segmentación no implica duplicar contenido, sino estructurarlo de forma que facilite su comprensión y uso según cada rol en el proyecto.

Entre los estilos mencionados, la Tabla 1 compara la documentación completa anticipada y la documentación progresiva o incremental, dejando a los enfoques híbridos como una alternativa intermedia que combina elementos de ambos según el contexto del proyecto.

Tabla 1

Comparación de documentación anticipada e incremental de requerimientos

Criterio	**Documentación anticipada**	**Documentación incremental**
Momento de elaboración	Al inicio del proyecto, antes del diseño e implementación	Durante el desarrollo, en cada iteración
Formato habitual	Documento formal (ej. SRS, IEEE 830-1998)	Historias de usuario, tarjetas de *backlog*, criterios de aceptación[7]
Nivel de detalle inicial	Alto; se busca especificación exhaustiva desde el principio	Bajo al inicio; se detalla progresivamente

(continúa en la siguiente página)

[7] Condiciones que debe cumplir una funcionalidad para considerarse completa y aceptada.

Tabla 1 (continuación)

Criterio	Documentación anticipada	Documentación incremental
Adaptabilidad al cambio	Limitada; requiere procedimientos formales para modificar	Alta; los cambios se integran con mayor flexibilidad
Adecuación	Proyectos con requisitos estables y bien conocidos	Proyectos dinámicos, entornos cambiantes, desarrollo ágil
Riesgos principales	Obsolescencia temprana, escasa adaptación a cambios	Ambigüedad, pérdida de trazabilidad si no se gestiona cuidadosamente
Ventajas clave	Claridad temprana, soporte a planificación rigurosa	Respuesta rápida a cambios, mejor alineación con expectativas del cliente

2.4 Administración y gestión del cambio

Los requerimientos rara vez permanecen inmutables a lo largo del ciclo de vida del software, y el cambio es considerado inevitable en proyectos de mediana y gran envergadura (Sommerville, 2011). Cambios en el entorno del negocio, la aparición de nuevas necesidades por parte de los usuarios o descubrimientos técnicos inesperados pueden generar modificaciones necesarias que deben ser gestionadas de forma proactiva.

La administración de requerimientos es una disciplina fundamental que asegura la cohesión y el control sobre el proyecto a medida que los requisitos evolucionan. Esta involucra un conjunto de actividades esenciales, como el control de versiones y la gestión de los cambios propuestos, garantizando que cada modificación sea trazable y aprobada. También abarca la priorización de requerimientos, lo que permite a los equipos enfocarse en lo más valioso en cada momento, y la evaluación de impacto de cada cambio, para comprender sus implicaciones en el diseño, desarrollo y pruebas. Finalmente, incluye la revisión y validación continua, asegurando que los requerimientos sigan siendo relevantes y alineados con las expectativas del negocio.

Como señalan Wiegers y Beatty (2013), una gestión eficaz de los requerimientos permite que el proyecto se mantenga flexible frente a cambios inevitables, sin perder el control sobre el alcance y la calidad del producto. Esta gestión es particularmente relevante en proyectos con duración prolongada o que involucran a múltiples actores, donde la comunicación y la alineación constante son clave para el éxito.

2.5 Desafíos en la práctica

Si bien la ingeniería de requerimientos cuenta con herramientas, metodologías y guías consolidadas, su aplicación práctica enfrenta múltiples desafíos, especialmente en contextos donde los recursos son limitados o los equipos cuentan con poca experiencia.

Pressman y Maxim (2020) reconocen que incluso en organizaciones maduras, la calidad de los requerimientos sigue siendo una fuente común de problemas. En entornos académicos o formativos, donde los desarrolladores están en proceso de aprendizaje, estos desafíos pueden verse acentuados por la necesidad de equilibrar conocimientos técnicos, habilidades blandas y la dinámica del trabajo colaborativo.

Esto refuerza la importancia de estudiar las prácticas reales de obtención, documentación y gestión de requerimientos, especialmente en proyectos educativos, para comprender mejor cómo se abordan estas actividades y qué oportunidades existen para fortalecerlas.

La ingeniería de requerimientos constituye un componente esencial del desarrollo de software, especialmente en contextos educativos donde la claridad, la trazabilidad y la validación son fundamentales para alcanzar resultados satisfactorios. Incorporar prácticas sistemáticas en la obtención, documentación y gestión del cambio no solo mejora la calidad del producto, sino que también promueve un aprendizaje más profundo y profesional en los equipos noveles.

Dado el peso que tiene la gestión de requerimientos en el éxito de un proyecto, en el siguiente capítulo se abordarán los principios generales de la gestión de proyectos de software, con especial atención a los contextos educativos donde estas prácticas se desarrollan y donde los desafíos de organización, planificación y seguimiento impactan directamente en la calidad del producto final.

CAPÍTULO III: GESTIÓN DE PROYECTOS

El capítulo anterior abordó la ingeniería de requerimientos como un pilar esencial del desarrollo de software, destacando la importancia de identificar, documentar y gestionar adecuadamente las necesidades del cliente para lograr productos de calidad.

Sin embargo, para que esos requerimientos se transformen en soluciones funcionales, deben ser integrados dentro de una estrategia organizativa más amplia: la gestión del proyecto en su totalidad.

Esta disciplina permite estructurar, coordinar y supervisar las actividades necesarias para alcanzar los objetivos del proyecto, articulando tanto los aspectos técnicos como los humanos. Su relevancia se acentúa en el contexto del desarrollo de software, donde la planificación, el seguimiento y la comunicación efectiva son fundamentales para minimizar riesgos y cumplir con las expectativas del cliente, especialmente en entornos educativos o de pequeña escala.

3.1 Definición y características generales

Un proyecto es “un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único” (PMI, 2021). Este esfuerzo se caracteriza por un inicio y un final definidos y se desarrolla bajo restricciones de tiempo, costo, recursos y alcance. Los proyectos se distinguen de las operaciones continuas por su carácter no repetitivo y transformador. Mientras que las operaciones tienden a mantener estructuras estables y resultados previsibles, los proyectos implican una transición desde un estado inicial hacia un resultado inédito, con grados significativos de incertidumbre y cambio (Kerzner, 2017).

La gestión de proyectos se entiende como la aplicación de conocimientos, habilidades, herramientas y técnicas para planificar, organizar, supervisar y controlar las actividades necesarias a fin de alcanzar los objetivos propuestos (PMI, 2021). Incluye la gestión del alcance, del cronograma, del presupuesto, de la calidad, de los recursos humanos, de los riesgos y de la comunicación. Su finalidad es garantizar que los entregables cumplan con los requisitos definidos, optimizar la utilización de los recursos y anticipar los problemas antes de que comprometan los resultados. Como señala McConnell (2004), “el proceso que se utiliza determina tanto la estabilidad de los requerimientos como la probabilidad de éxito del proyecto”.

Los proyectos presentan las siguientes características generales:

- Temporalidad: cada proyecto posee un inicio y un final definidos.

- Orientación a resultados únicos: busca generar un producto, servicio o resultado irrepetible, que lo diferencia de las operaciones rutinarias.
- Restricciones múltiples**:** se desarrolla dentro de límites de tiempo, costo, recursos y alcance, que condicionan su ejecución.
- Incertidumbre y riesgo: los proyectos se llevan a cabo en contextos donde las condiciones pueden variar, exigiendo capacidad de adaptación.
- Carácter transformador: los proyectos actúan como agentes de innovación y mejora dentro de las organizaciones, generando cambios en procesos, productos o estructuras (PMI, 2021).

Estas características se ven influenciadas por factores externos, como cambios en los requerimientos, la disponibilidad de personal o restricciones técnicas, que pueden alterar las condiciones iniciales del proyecto. Como se desarrolló en el Capítulo I, la naturaleza dinámica de los proyectos requiere apoyarse en marcos metodológicos y procesos definidos, capaces de estructurar las actividades, gestionar la incertidumbre y orientar los esfuerzos hacia el logro de los resultados previstos. En este sentido, una gestión organizada resulta indispensable para coordinar el trabajo del equipo y alinear las decisiones operativas con los objetivos del proyecto y las expectativas del cliente o patrocinador.

En síntesis, los proyectos constituyen una forma de organización del trabajo orientada a resultados, que exige enfoques adaptativos, liderazgo efectivo y capacidad para tomar decisiones en contextos inciertos. Cuando estas características se trasladan al desarrollo de software, adquieren particular relevancia debido a su naturaleza intangible, la complejidad técnica involucrada y la evolución frecuente de los requerimientos. Tal como se analizó en el Capítulo II, la gestión de requerimientos constituye un eje crítico en este tipo de proyectos, dado que su definición y control inciden directamente en la planificación, el alcance y la calidad del producto final. Estas condiciones hacen necesaria una visión de gestión específica para proyectos de software, que se examina a continuación.

3.2 Particularidades específicas de los proyectos de software

Los proyectos de desarrollo de software comparten las características generales de cualquier proyecto, pero también presentan rasgos distintivos que inciden directamente en su gestión. Comprender estas particularidades permite anticipar desafíos y adoptar enfoques más adecuados al tipo de producto que se busca construir.

Entre los aspectos más relevantes se encuentran:

Intangibilidad del producto: el software carece de una representación física observable o medible durante gran parte de su desarrollo. Esta falta de visibilidad dificulta

el monitoreo del progreso, ya que el avance es abstracto y no siempre se traduce en entregables concretos hasta etapas avanzadas del proyecto (Sommerville, 2015). Esto genera incertidumbre tanto en los equipos como en los clientes, ya que no siempre es evidente cuánto se ha avanzado ni qué funcionalidades están completas.

Complejidad inherente: el software moderno llega a alcanzar dimensiones extraordinarias, con millones de líneas de código y componentes interdependientes. Sommerville (2016) destaca que los sistemas actuales son “cien o incluso mil veces más grandes” que los de las décadas fundacionales de la ingeniería de software, y que existen sistemas recientes que superan incluso el umbral de los mil millones de líneas de código. Esta magnitud solo es posible gracias a la reutilización extensiva de componentes y la integración de subsistemas desarrollados por diferentes proveedores, conformando lo que se conoce como sistemas de sistemas (Sommerville, 2016). Estas características imponen desafíos significativos para la planificación, coordinación y validación continua del trabajo, ya que cualquier modificación tiene el potencial de producir efectos en cascada sobre otros módulos o servicios.

Cambios frecuentes en los requerimientos: resulta difícil identificar desde el inicio todas las necesidades del cliente o de los usuarios; con el avance del proyecto, el propio cliente aclara y ajusta sus expectativas. En consecuencia, los requerimientos evolucionan y la gestión del alcance y la planificación deben mantenerse flexibles y abiertas al cambio (Pressman & Maxim, 2020).

Dependencia de personas clave: a diferencia de los proyectos industriales, cuyo avance se apoya en procedimientos predecibles y estandarizados (PMI, 2021), el desarrollo de software depende en gran medida del intelecto humano y del conocimiento tácito del equipo. Como afirma McConnell (2004), el software es una de las actividades más puramente mentales, en la que “el material básico es el intelecto humano y la herramienta principal es el propio desarrollador” (p. 680). Esta naturaleza intensamente humana implica que las decisiones, percepciones y habilidades individuales influyen directamente en la calidad, los tiempos y los resultados del proyecto.

La productividad entre desarrolladores varía drásticamente (hasta un factor de 10 o más) según su experiencia y conocimientos acumulados (Pressman & Maxim, 2015). Por ello, equipos con miembros experimentados tienden a requerir menos esfuerzo total que aquellos conformados por noveles, aún utilizando las mismas herramientas. Además, el conocimiento tácito, difícil de documentar y transferir por medios formales, se transmite con mayor eficacia a través de la interacción directa entre las personas (Pressman & Maxim, 2015).

Estas condiciones explican por qué la gestión de proyectos de software exige competencias específicas que integren lo técnico con lo organizacional, contemplando tanto los aspectos estructurales como las dinámicas humanas del trabajo colaborativo.

La Tabla 2 sintetiza estas particularidades y sus implicancias habituales para la gestión.

Tabla 2

Particularidades de los proyectos de software *y su impacto en la gestión*

Particularidad	Descripción	Implicancias para la gestión
Intangibilidad del producto	El software no es visible ni medible físicamente hasta etapas avanzadas.	Dificulta el seguimiento del progreso y la validación temprana.
Complejidad inherente	Gran cantidad de componentes interdependientes y restricciones técnicas.	Requiere planificación detallada, control de versiones y pruebas frecuentes.
Cambios frecuentes en requerimientos	Las necesidades evolucionan durante el desarrollo.	Demanda flexibilidad en la planificación y control de alcance.
Dependencia de personas clave	El conocimiento técnico reside en individuos más que en procesos estructurados.	Incrementa el riesgo ante rotación del equipo; exige gestión activa del conocimiento.

3.3 Principios

Tal como se detalló previamente en el Capítulo 1, la Guía de los fundamentos para la dirección de proyectos (Guía del PMBOK®, Project Management Institute [PMI], 2021) constituye el estándar de referencia más difundido para la dirección de proyectos. En su Séptima Edición se introduce un cambio de enfoque significativo: la disciplina deja de organizarse en grupos de procesos y áreas de conocimiento, para estructurarse en dominios de desempeño, entendidos como esferas de práctica que agrupan actividades, responsabilidades y factores críticos de éxito que influyen directamente en los resultados. Entre ellos se incluyen la planificación, la entrega de valor, el trabajo en equipo, la gestión de la incertidumbre y la medición del progreso. Esta organización busca ofrecer una guía más flexible y adaptable a distintos contextos, reconociendo que los proyectos requieren tanto técnicas y procesos como competencias interpersonales y criterio situacional.

Sobre esta base, la gestión de proyectos de software se apoya en un conjunto de principios que orientan la planificación, ejecución, seguimiento y cierre de iniciativas, con el objetivo de entregar productos de calidad dentro de los plazos y presupuestos establecidos (Kerzner, 2017). Estos principios, lejos de ser prescriptivos, ofrecen

lineamientos generales que, al articularse con los dominios del PMBOK®, permiten adaptar la práctica a distintos enfoques y contextos (PMI, 2021). En el caso particular del desarrollo de software, cobran relevancia especial debido a la naturaleza compleja, intangible y dinámica del producto.

Sobre esta base, se detallan cinco principios de gestión que resultan especialmente relevantes en proyectos de software. Algunos de ellos provienen directamente de los dominios del PMBOK® Guide (PMI, 2021), mientras que otros se fundamentan además en aportes de autores como McConnell (2004) y Pressman y Maxim (2020).

- Planificación: consiste en definir los objetivos del proyecto, establecer el alcance, identificar las tareas, estimar tiempos y asignar recursos. Una planificación efectiva permite reducir la incertidumbre inicial, establecer expectativas realistas y facilitar la coordinación entre los actores involucrados (PMI, 2021; Pressman & Maxim, 2020). En el PMBOK® Séptima Edición, este principio se corresponde con el dominio de desempeño "Planificación", que organiza el trabajo del proyecto de manera coherente y adaptable a los cambios.
- Monitoreo y control: implica supervisar el progreso real frente a lo planificado, identificar desviaciones y aplicar acciones correctivas. En proyectos de software, esto requiere establecer métricas adecuadas que permitan medir avances en un producto de naturaleza intangible y compleja (Kerzner, 2017). Este principio se vincula con los dominios de "Trabajo del Proyecto" y "Medición", que aseguran la alineación continua entre los resultados esperados y los alcanzados (PMI, 2021).
- Gestión de riesgos: consiste en identificar eventos que puedan afectar negativamente al proyecto, evaluarlos y definir respuestas apropiadas. En el desarrollo de software, donde los cambios son frecuentes y los requisitos pueden evolucionar, anticiparse a los riesgos es clave para mantener el rumbo del proyecto (Pressman & Maxim, 2020). El dominio de desempeño "Incertidumbre" aborda específicamente estas prácticas.
- Gestión de calidad: se orienta a garantizar que el producto cumpla con los requisitos técnicos, las expectativas de los usuarios y los estándares del proyecto. Incluye tanto la calidad del producto final como la del proceso utilizado para desarrollarlo, lo cual resulta crítico en entornos donde los errores pueden propagarse rápidamente (Sommerville, 2011). Este principio se refleja en el dominio de "Entrega" del PMBOK, que se enfoca en la generación de valor a partir de los entregables.
- Gestión de recursos humanos: consiste en coordinar el trabajo del equipo, asignar roles de manera clara, fomentar la colaboración y resolver los conflictos que

puedan surgir durante el proyecto. En el desarrollo de software, donde el principal insumo es el conocimiento humano, la interacción entre las personas resulta decisiva para la calidad y los resultados alcanzados (McConnell, 2004; Pressman & Maxim, 2020). El dominio “Equipo” del PMBOK® Guide – Séptima Edición (PMI, 2021) incluye explícitamente estas prácticas, destacando la necesidad de construir un entorno de confianza y aprendizaje que favorezca tanto el alto desempeño como el liderazgo compartido. En contextos educativos, la asignación de roles, de forma explícita o mediante rotación, adquiere además un valor pedagógico particular, pues permite a los estudiantes experimentar distintas responsabilidades, comprender el ciclo de vida del software desde múltiples perspectivas y fortalecer competencias de colaboración (García, Treude, & Valentine, 2024).

Estos principios, aunque definidos en contextos profesionales, también resultan formativos y transferibles a proyectos de pequeña escala o entornos educativos, donde contribuyen a generar experiencias más estructuradas y alineadas con las prácticas del mundo laboral (PMI, 2021).

3.4 Gestión de proyectos de software en el ámbito educativo

En el contexto académico, como en proyectos realizados por estudiantes, la gestión de proyectos representa un aspecto central en la formación, aun cuando la atención se concentra en otras actividades más tangibles como el diseño o la codificación. Su incorporación resulta esencial para el desarrollo profesional de futuros egresados, tal como se introdujo en el Capítulo I al presentar la Guía del PMBOK® del Project Management Institute (PMI) como marco internacional para la dirección de proyectos, ya dichas guías destacan esta disciplina como una competencia transversal dirigida también a estudiantes y educadores, y la definen como la aplicación de conocimientos, habilidades y técnicas para cumplir con los resultados previstos (PMI, 2021).

En este sentido, el aprendizaje de prácticas como la planificación, la estimación de tiempos, el control de versiones, el trabajo colaborativo y el manejo de cambios permite a los estudiantes adquirir una visión más realista del proceso de desarrollo y prepararse para entornos laborales complejos. En proyectos donde los recursos y el tiempo son acotados, la gestión adecuada maximiza los resultados y reduce los riesgos de insatisfacción o entrega incompleta. Además, posibilita el seguimiento del trabajo en equipos pequeños, en los que la coordinación no siempre es formalizada, pero resulta igualmente crítica para cumplir con los compromisos asumidos ante terceros, como ocurre en experiencias con clientes reales.

La gestión de proyectos constituye un componente esencial del desarrollo de software profesional. Este enfoque se apoya en los fundamentos expuestos en el Capítulo I, donde se presentaron los procesos del ciclo de vida del software y la terminología del PMBOK® del *Project Management Institute* (PMI), y permite alinear el trabajo técnico con los objetivos del proyecto, anticipar problemas, coordinar recursos y garantizar que el producto final cumpla con los estándares esperados.

En el caso particular del software, su naturaleza intangible y cambiante exige un enfoque específico de gestión, con énfasis en la planificación flexible, la comunicación continua y la capacidad de adaptación. Estos fundamentos resultan particularmente relevantes en el marco de la norma ISO/IEC 29110 (ISO/IEC, 2011), y de los aportes de Laporte, Hébert & Mineau (2013), cuyo enfoque sobre la gestión de proyectos en entornos de pequeña escala será analizado en el próximo capítulo.

CAPÍTULO IV: NORMA ISO/IEC 29110

Las normas internacionales en ingeniería de software ofrecen marcos de referencia que buscan mejorar la calidad, la gestión y la eficiencia de los procesos de desarrollo. Sin embargo, estandares ampliamente utilizados en ingeniería de software, como ISO/IEC 12207, que establece un marco de ciclo de vida completo del software, o CMMI (*Capability Maturity Model Integration*), que define niveles de madurez para la mejora continua de procesos, fueron concebidos para contextos organizacionales complejos y de alta capacidad operativa, con equipos numerosos y estructuras altamente formalizadas. Esta orientación generó una brecha importante, especialmente frente a la realidad de las micro y pequeñas empresas, equipos reducidos o entornos educativos, donde los recursos, la experiencia formal y el grado de madurez organizacional suelen ser limitados (Laporte, Hébert & Mineau, 2013).

En respuesta a esta problemática, surge la norma ISO/IEC 29110, desarrollada por la Organización Internacional de Normalización (ISO) y la Comisión Electrotécnica Internacional (IEC), con el objetivo de proporcionar un marco simplificado y accesible para mejorar la calidad y la eficiencia en el desarrollo de software en *Very Small Entities* (*VSE, "organizaciones muy pequeñas"*), organizaciones de hasta 25 personas.

La ISO/IEC 29110 se centra en cubrir únicamente los procesos básicos y esenciales para una gestión y un desarrollo eficaz de proyectos de software en contextos de menor escala. A diferencia de normas más amplias y complejas, como ISO/IEC 12207 o CMMI, este enfoque permite una implementación progresiva, sin necesidad de grandes inversiones en tiempo, recursos técnicos o formación avanzada (ISO/IEC 29110-4-1, 2011; Laporte et al., 2012)

La norma se estructura en documentos diferenciados, que incluyen una guía de perfiles, informes técnicos (*Technical Reports*, TR) y especificaciones técnicas (*Technical Specifications*, TS). Estos componentes se organizan según un enfoque basado en perfiles, lo que permite adaptar el modelo según el nivel de madurez y las características de la organización.

En este capítulo se aborda la ISO/IEC 29110, con especial atención a su aplicación en contextos educativos y equipos de desarrollo noveles.

4.1 Origen y propósito

El origen de la norma ISO/IEC 29110 se remonta a la identificación de la necesidad concreta de las micro y pequeñas organizaciones de desarrollo de software con grandes dificultades para aplicar estándares internacionales como ISO/IEC 12207 o CMMI, concebidos para entornos corporativos de alta formalización. Esta problemática se

reflejó en estudios y encuestas realizadas en Europa, Asia y América Latina, que coincidieron en señalar que los marcos existentes resultaban demasiado costosos y complejos para equipos reducidos, con recursos y procesos aún incipientes (Laporte, Hébert & Mineau, 2013). Asimismo, muchas de estas organizaciones actuaban como proveedoras de componentes o subsistemas de soluciones mayores, lo que les exigía contar con un nivel mínimo de calidad y trazabilidad para insertarse en cadenas de valor globales.

Un antecedente importante fue MoProSoft, modelo de procesos desarrollado en México en el marco del programa PROSOFT (Secretaría de Economía, 2002). Tras su normalización como norma mexicana NMX-I-059-NYCE-2005 y su validación en micro y pequeñas empresas, demostró la posibilidad de elevar la capacidad de procesos en organizaciones de tamaño reducido (Oktaba, 2011). Estos resultados motivaron al Working Group 24 (WG24) del Subcomité 7 (SC7) del Joint Technical Committee 1 (JTC1) de ISO/IEC, responsable de la estandarización en ingeniería de software y sistemas, a adoptar MoProSoft como base para la elaboración de la norma ISO/IEC 29110 (Oktaba, 2011).

El trabajo impulsado por ISO y la IEC dio lugar a la primera publicación en 2011 de la ISO/IEC 29110-5-1-2 como Technical Report (TR), concebida como guía informativa para proyectos de pequeña escala. Su propósito central fue ofrecer un conjunto mínimo de procesos de gestión e ingeniería que pudiera adoptarse de manera progresiva, sin generar cargas burocráticas desproporcionadas. Con la revisión de 2025, la norma adquirió carácter de *International Standard* (IS), consolidando su reconocimiento como marco normativo formal, manteniendo el perfil básico como núcleo y alineándose más estrechamente con otros estándares (ISO/IEC/IEEE 12207, 15289), además de incorporar anexos informativos sobre temas actuales como pruebas, accesibilidad, seguridad y despliegue.

4.1.1 Perfiles

La norma ISO/IEC 29110 organiza sus recomendaciones en perfiles, entendidos como subconjuntos de procesos, actividades y productos de trabajo de normas más extensas (como ISO/IEC 12207 o ISO/IEC 15288), adaptados a las necesidades de las VSE descritas previamente. Esta estrategia permite ofrecer un camino de adopción gradual, evitando imponer a organizaciones pequeñas la complejidad completa de los estándares de referencia.

En el grupo genérico de perfiles para ingeniería de software se distinguen cuatro niveles principales (ISO/IEC TR 29110-1, 2016):

Perfil de Entrada: orientado a VSE de reciente creación (menos de tres años de operación) o a proyectos pequeños, con un esfuerzo estimado menor a seis persona-mes. Proporciona las prácticas mínimas para iniciar la formalización de procesos.

Perfil Básico: dirigido a *VSE* que desarrollan una única aplicación mediante un solo equipo de trabajo.

Perfil Intermedio: pensado para VSE que ejecutan más de un proyecto en paralelo, con varios equipos en funcionamiento.

Perfil Avanzado: destinado a organizaciones que buscan sostener y expandir sus capacidades, consolidándose como actores competitivos en el desarrollo de software.

Cada perfil incluye procesos, tareas y productos ajustados al grado de madurez esperado de la organización, facilitando así una evolución progresiva de las prácticas de gestión e ingeniería.

A continuación se desarrollará mayor detalle el Perfil Básico, por ser el más pertinente para equipos noveles y proyectos educativos de pequeña escala.

4.2 Procesos del Perfil Básico

Este perfil está dirigido a organizaciones que desarrollan software a medida con procesos aún poco formalizados y recursos muy limitados, la norma establece dos procesos fundamentales:

- Proceso de Gestión de Proyectos (PM)
- Proceso de Implementación de Software (SI)

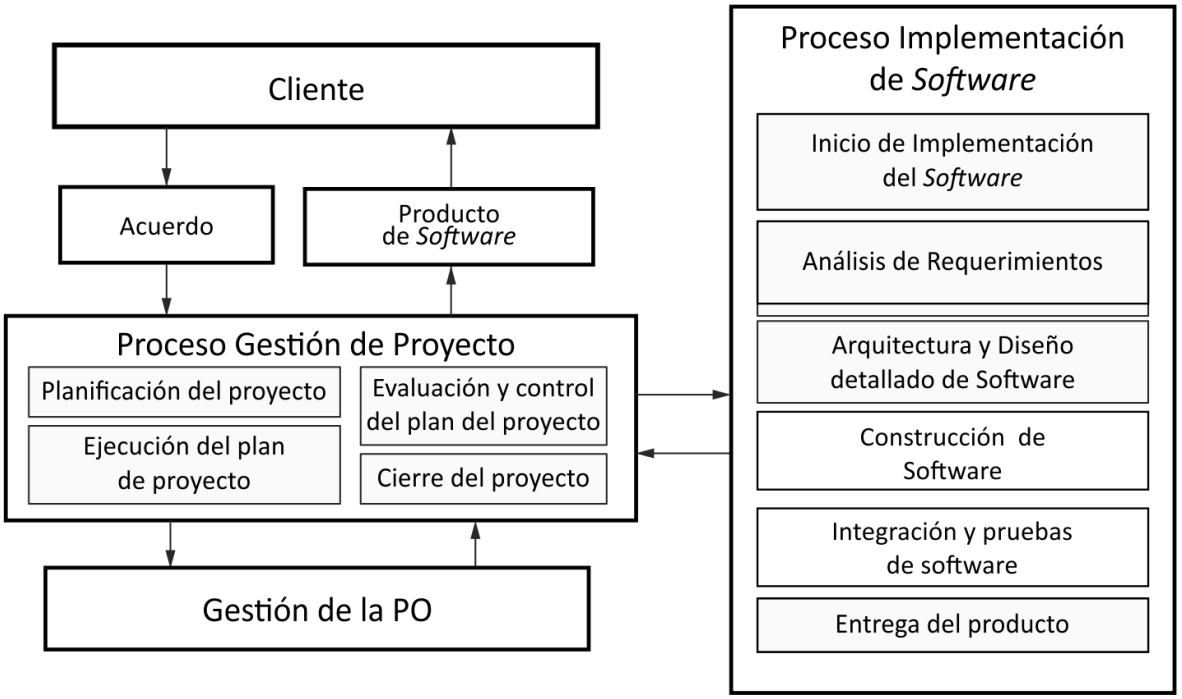
Ambos procesos cuentan con objetivos claramente definidos y se estructuran en distintos niveles: actividades, que representan las fases principales; tareas, que detallan las acciones a realizar dentro de cada actividad; y productos de trabajo, que son los artefactos resultantes. En conjunto, estos elementos orientan la planificación, ejecución, control y entrega del software, asegurando su calidad y adecuación a los requerimientos del cliente. La norma proporciona guías que indican quién realiza cada tarea, cuándo y cómo debe hacerse, junto con plantillas y ejemplos que facilitan su aplicación práctica. Además, se establecen productos de trabajo esperados (como listas de requerimientos, planes, informes de revisión, registros de validación o pruebas) que permiten organizar y documentar el trabajo de forma estructurada, manteniendo la trazabilidad y el control del proyecto sin imponer estructuras burocráticas.

La Figura 1 ilustra el esquema del Perfil Básico de la norma ISO/IEC 29110, donde se observa la interacción entre los procesos de Gestión de Proyecto e Implementación de Software. La representación muestra cómo la gestión organiza actividades de planificación, ejecución, evaluación y cierre, al tiempo que se articula con

la implementación técnica del software, que incluye fases como la iniciación, el análisis de requerimientos, el diseño arquitectónico, la construcción, las pruebas y la entrega del producto. Dentro de este flujo, el acuerdo de trabajo se establece como un producto clave que formaliza los compromisos con el cliente y alimenta la planificación del proyecto. La relación bidireccional entre gestión e implementación subraya que la planificación y el control deben retroalimentarse de los avances técnicos y, a su vez, orientar su desarrollo, garantizando que los resultados obtenidos se reflejen en la configuración final del software.

Figura 1

Esquema del Perfil Básico de la norma ISO/IEC 29110



Cabe destacar que la norma ISO/IEC 29110 detalla un conjunto extenso de tareas específicas asociadas a cada actividad de los procesos de Gestión de Proyectos e Implementación de Software. Estas tareas incluyen la definición de responsables, entradas, salidas y criterios de finalización, y abarcan acciones que van desde la elaboración y revisión de documentos hasta la coordinación con las partes interesadas y el aseguramiento de la calidad en cada fase.

En el presente trabajo no se reproducen todas las tareas debido a su extensión, aunque se reconoce su importancia como orientación práctica. En cambio, se prioriza la presentación de las actividades principales y los productos de trabajo asociados, por considerarse el nivel más pertinente para el análisis realizado y la construcción de la guía práctica. El listado completo de tareas puede consultarse directamente en la norma ISO/IEC 29110 y, de manera resumida, en el Anexo IV de este documento.

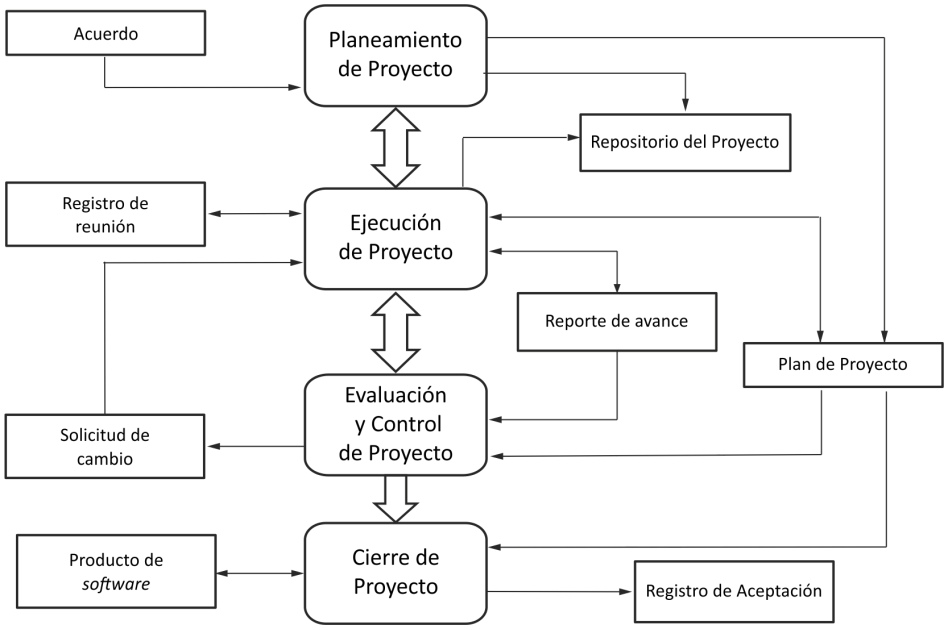
4.2.1 Proceso de Gestión de Proyecto

Este proceso tiene como propósito asegurar que los proyectos de software se lleven a cabo de forma controlada y efectiva, cumpliendo con los objetivos establecidos en cuanto a plazos, costos y calidad (ISO/IEC 29110-5-1-2, 2011).

A modo de síntesis, la Figura 2 presenta el esquema general del proceso, organizado en cuatro actividades principales que constituyen el núcleo de la gestión: planeación, ejecución, evaluación y control, y cierre.

Figura 2

Esquema del proceso de gestión de proyecto



Cada una de estas actividades se corresponde con las fases que orientan la organización del trabajo y la generación de productos de apoyo al proyecto.

Las principales actividades del PM incluyen:

- Inicio del proyecto: se definen los objetivos, el alcance, las restricciones y los criterios de aceptación.
- Planificación del proyecto: se elaboran cronogramas, se asignan recursos, se identifican riesgos y se definen las actividades de monitoreo.
- Ejecución y monitoreo: implica coordinar tareas, supervisar el avance, resolver desviaciones y mantener una comunicación fluida con los interesados.
- Cierre del proyecto**:** incluye la validación del producto final, la liberación de recursos y la documentación de lecciones aprendidas.

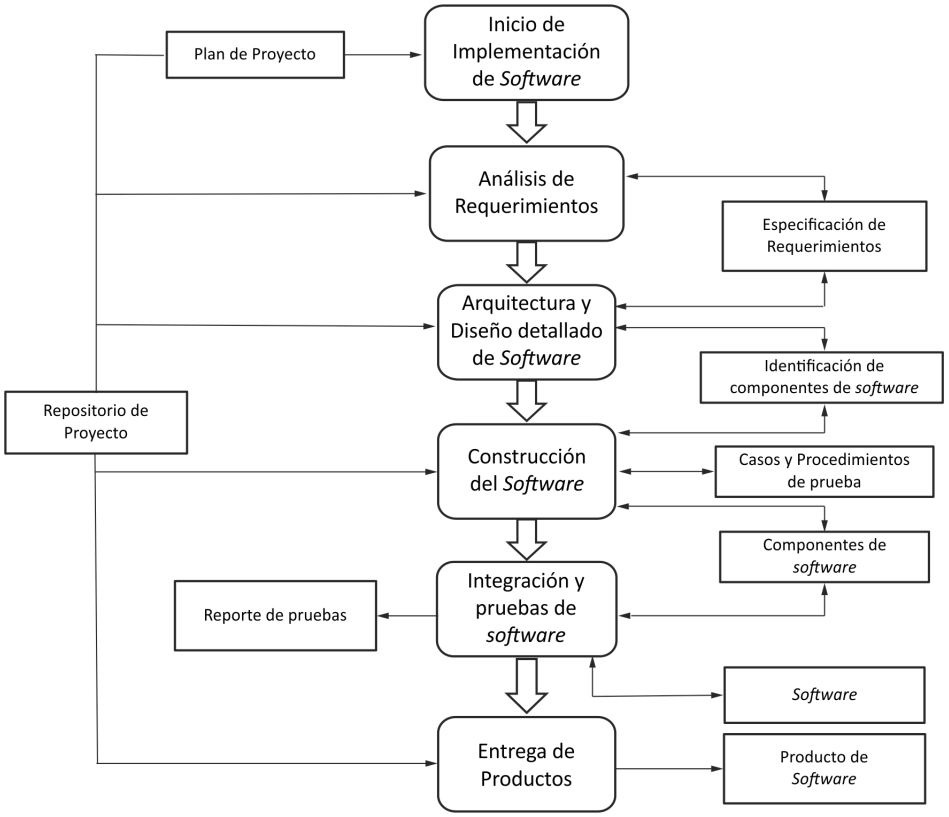
4.2.2 Proceso de implementación de software

El Proceso de Implementación de Software guía a los equipos a lo largo de las principales fases del ciclo de vida del software. Su objetivo es transformar los requerimientos del cliente en un producto funcional, verificable y mantenible.

La Figura 3 ilustra la secuencia de actividades que estructuran este proceso: análisis de requerimientos, diseño arquitectónico y detallado, construcción, integración y pruebas, y entrega del producto.

Figura 3

Esquema del proceso de implementación de software



Cada actividad genera productos de trabajo específicos, como especificaciones, casos de prueba, componentes o manuales, que aseguran la trazabilidad y la calidad en cada etapa..

Las actividades de este proceso incluyen:

- Análisis de requerimientos: recopilación, validación y análisis de las necesidades del cliente.
- Diseño del software: definición de la arquitectura, estructura y componentes del sistema.

- Construcción y pruebas: codificación, integración de componentes y ejecución de pruebas unitarias, de sistema y de aceptación.
- Entrega y mantenimiento: instalación del producto en el entorno de operación, capacitación a usuarios y atención de cambios o correcciones posteriores.

Este proceso promueve la trazabilidad desde los requerimientos hasta la entrega del producto, fortaleciendo el control sobre la calidad y facilitando la evolución posterior del sistema (Pressman & Maxim, 2020).

4.3 Roles en los procesos

La norma ISO/IEC 29110-5-1-2:2025 define un conjunto de roles fundamentales que intervienen en los procesos Gestión de Proyectos e Implementación de Software. Estos roles permiten asignar responsabilidades claras y asegurar que las tareas y productos de trabajo sean llevados a cabo de manera efectiva. Cada rol se asocia a un conjunto de competencias específicas que guían su participación en el proyecto:

4.3.1 Analista (*Analyst*)

El analista se especializa en la captura, especificación y análisis de requerimientos, incluyendo aspectos como el diseño de interfaces de usuario y criterios de usabilidad. Posee dominio de técnicas de revisión y edición de documentos, así como experiencia en desarrollo y mantenimiento de software. Su función es central en el proceso de SI, especialmente en la fase de análisis de requerimientos y validación con el cliente.

4.3.2 Cliente (*Customer*)

El cliente representa a los usuarios y es responsable de explicar, aprobar y priorizar los requerimientos. Debe contar con la autoridad suficiente para aceptar cambios y validar entregables. Su participación es crítica tanto en PM (definición de acuerdos y aceptación formal) como en SI (validación de requisitos y pruebas de aceptación).

4.3.3 Diseñador (*Designer*)

El diseñador aporta conocimientos en arquitectura de software y componentes, además de experiencia en planificación y ejecución de pruebas de integración. Domina técnicas de revisión y edición, y su rol resulta esencial en las fases de diseño arquitectónico y detallado del proceso de SI.

4.3.4 Programador (*Programmer*)

El programador es responsable de la codificación, integración y pruebas unitarias, además de participar en revisiones y mantenimiento del software. Su rol se concentra en la fase de construcción e integración del proceso de SI, garantizando que el producto cumpla con las especificaciones técnicas definidas.

4.3.5 Jefe de Proyecto (*Project Manager*)

El jefe de proyecto posee competencias de liderazgo, planificación, gestión de recursos humanos y técnicos, supervisión y control financiero. Es el principal responsable del proceso de PM, velando por el cumplimiento de los objetivos en plazo, costo y calidad. Coordina además la comunicación con el cliente y la toma de decisiones estratégicas.

4.3.6 Líder Técnico (*Technical Leader*)

El líder técnico combina experiencia en procesos de ingeniería de software y en el dominio técnico del proyecto. Su papel es de apoyo tanto en PM como en SI: colabora con la supervisión de actividades técnicas, asegura la aplicación de buenas prácticas y respalda al equipo en la resolución de problemas complejos.

4.3.7 Equipo de Trabajo (*Work Team*)

El equipo de trabajo está compuesto por los miembros que ejecutan actividades de diseño (DES), análisis (AN) y programación (PR). Sus competencias varían según el rol asumido, pero en conjunto aportan a la ejecución práctica de las tareas definidas en los procesos de PM y SI. Representan la base operativa del proyecto y aseguran la materialización de los productos de trabajo definidos por la norma.

4.4 Productos de trabajo

La norma ISO/IEC 29110 define los productos de trabajo (*work products*) como los artefactos gestionados por los procesos. Estos se generan, utilizan y actualizan a lo largo del ciclo de vida de un proyecto de software. Constituyen la base documental para demostrar lo realizado, facilitan la planificación y el seguimiento, apoyan la validación y la entrega del producto final, y aseguran la coherencia entre lo solicitado por el cliente, lo planificado y lo implementado.

Los productos de trabajo se asocian directamente con los dos procesos principales del perfil básico: Gestión de Proyectos e Implementación de Software. En este contexto se identifican tanto los insumos necesarios para iniciar actividades, como los

resultados que se entregan al cliente o a otros procesos, y los artefactos que se utilizan internamente dentro del equipo.

Entre los principales productos definidos por la norma se encuentran, en el proceso de gestión de proyectos:

- Registro de aceptación (WP.01 *Acceptance record*): documento que certifica formalmente la conformidad del producto con los criterios acordados, dejando constancia de la fecha de entrega, elementos recibidos y observaciones del cliente.
- Acuerdo (WP.02 *Agreement*): define el trabajo a realizar y los términos acordados con el cliente. Sustituye al Statement of Work de la versión 2011. Incluye propósito, alcance, entregables y condiciones de aceptación.
- Solicitud de cambio (WP.03 *Change request*): identifica problemas o mejoras deseadas en el software o en la documentación, registrando el propósito, el impacto estimado y el estado de aprobación.
- Registro de correcciones (WP.04 *Correction register*): detalla desviaciones respecto del plan, las acciones correctivas previstas, responsables y fechas de seguimiento.
- Acta de reunión (WP.07 *Meeting record*): registra acuerdos, participantes, temas tratados y compromisos asumidos en encuentros con el cliente o el equipo.
- Registro del estado de avance (WP.09 *Progress status record*): compara los resultados obtenidos con lo planificado en cuanto a tareas, recursos, costos y cronograma, identificando desvíos y riesgos.
- Plan del proyecto (WP.10 *Project plan*): es central para coordinar todas las actividades del proyecto. Describe objetivos, entregables, tareas y dependencias, recursos asignados, cronograma, riesgos, estrategia de control de versiones y mecanismos de respaldo.
- Repositorio del proyecto (WP.11 *Project repository*): almacén central de todos los productos relevantes, con control de acceso, recuperación de versiones y trazabilidad.
- Copia de respaldo del repositorio (WP.12 *Project repository backup*): medio de resguardo para garantizar la recuperación de la información en caso de fallos o pérdida de datos.

En el proceso de implementación de software se incluyen:

- Entorno de implementación (WP.05 *Implementation environment*): describe compiladores, librerías, herramientas de construcción y de prueba utilizados durante el desarrollo.

- Documentación de mantenimiento (WP.06 *Maintenance documentation*): detalla la configuración del software y del entorno para facilitar su futura modificación, incluyendo versiones y herramientas asociadas.
- Guía de operación del producto (WP.08 *Product operation guideline*): explica cómo instalar, configurar y utilizar el software en su entorno operativo, incluyendo condiciones de uso y advertencias relevantes.
- Especificación de requerimientos (WP.13 *Requirements specification*): es fundamental para guiar el diseño, la construcción y las pruebas del sistema. Recoge de manera clara y verificable las necesidades del cliente, abarcando funcionalidad, interfaces, restricciones y atributos de calidad.
- Software (WP.14 Software): código fuente y ejecutable integrado como producto final.
- Componentes de software (WP.15 *Software components*): unidades de código relacionadas, desarrolladas y probadas de forma independiente antes de su integración.
- Diseño de software (WP.16 *Software design*): incluye tanto la arquitectura general como el diseño detallado de los componentes, interfaces, estructuras de datos y consideraciones de rendimiento y seguridad.
- Producto de software (WP.17 *Software product*): configuración final que integra requerimientos, diseño, componentes, pruebas y documentación asociada, lista para ser entregada al cliente.
- Documentación de usuario (WP.18 *Software user documentation*): orienta al usuario final en la instalación, uso y resolución de problemas comunes.
- Casos y procedimientos de prueba (WP.19 *Test cases and test procedures*): describen entradas, condiciones, pasos y resultados esperados para verificar que el software cumple los requisitos.
- Informe de pruebas (WP.20 *Test report*): presenta los resultados obtenidos en la ejecución de pruebas, registrando defectos hallados y correcciones aplicadas.
- Registro de trazabilidad (WP.21 *Traceability record*): establece vínculos entre requerimientos, elementos de diseño, componentes, pruebas y entregables, asegurando cobertura y consistencia.
- Registro de validación (WP.22 *Validation record*): evidencia que el software cumple con las necesidades y expectativas del cliente en su contexto de uso.
- Registro de verificación (WP.23 *Verification record*): documenta que el software satisface las especificaciones técnicas y de diseño establecidas.

La identificación de estos productos permite comprender qué tipo de información es útil registrar y conservar durante un proyecto. En contextos educativos, su uso como

referencia brinda a los equipos noveles un punto de apoyo para organizar su trabajo y reflexionar sobre sus prácticas, sin que ello implique adoptar una formalización estricta o ajena a su realidad.

4.5 Experiencias previas en entornos organizacionales y educativos

Laporte (2014), Mejía et al. (2019) y Vives, Meléndez y Dávila (2021) han explorado el uso de la norma ISO/IEC 29110 en contextos de pequeña escala, aportando evidencia empírica sobre su aplicabilidad tanto en entornos organizacionales como educativos. Estos antecedentes resultan especialmente valiosos para fundamentar su incorporación como marco de referencia en proyectos académicos desarrollados por estudiantes.

Uno de los aportes más significativos en esta línea es el trabajo de Laporte (2014), quien describe un enfoque innovador para la formación de profesionales en estándares a partir de la implementación práctica de ISO/IEC 29110 en proyectos reales. En la École de technologie supérieure[8] (*ÉTS*, Escuela Superior de Tecnología de Montreal), se promovió la participación de estudiantes de posgrado en proyectos industriales con empresas locales, adaptando la norma a sus necesidades y recursos.

Los resultados indicaron que este tipo de experiencias facilita la comprensión de los procesos y fortalece la competencia profesional en ingeniería de software, incluso en escenarios con escasa formalización. Además, se documentaron beneficios concretos en términos de calidad del producto, organización del trabajo y documentación generada.

Complementariamente, Vives, Meléndez y Dávila (2021) propusieron un conjunto de rutas metodológicas orientadas a asistir a las pequeñas organizaciones en la selección de herramientas que faciliten la adopción de la norma. Su estudio evidenció que uno de los principales obstáculos para la implementación efectiva de ISO/IEC 29110 reside en la falta de conocimiento sobre herramientas tecnológicas apropiadas. La propuesta incluyó una validación con una EMP real, lo que permitió constatar la utilidad de dichas rutas como estrategia de apoyo a la automatización y simplificación de tareas asociadas a los productos de trabajo.

Por su parte, Mejía et al. (2019) llevaron a cabo un análisis orientado a identificar herramientas que den soporte al perfil básico de la norma, focalizándose en su adecuación a proyectos de pequeña escala. Su investigación destacó que una selección cuidadosa de herramientas puede contribuir significativamente a mejorar la trazabilidad,

[8] La École de technologie supérieure (ÉTS) es una universidad pública de Montreal, Canadá, especializada en la enseñanza y la investigación aplicada en ingeniería y tecnología. Forma parte del sistema de la Université du Québec y se distingue por su estrecha vinculación con el sector industrial, promoviendo que los estudiantes participen en proyectos reales en colaboración con empresas.

la organización de la documentación y la gestión de requerimientos, todos aspectos centrales para la calidad de los productos de software.

En conjunto, estos antecedentes refuerzan la pertinencia de utilizar la norma ISO/IEC 29110 como marco flexible para la formación de desarrolladores noveles, y al mismo tiempo resaltan la necesidad de contar con recursos concretos (como herramientas tecnológicas adecuadas) para facilitar su implementación. Estas experiencias previas constituyen un punto de partida sólido para el presente trabajo, que se propone analizar prácticas reales de estudiantes y desarrollar una guía práctica que articule dichos elementos.

4.6 Aplicabilidad en contextos educativos y de equipos noveles

Una de las virtudes más destacadas de la norma ISO/IEC 29110 es su adaptabilidad a entornos con recursos limitados o escasa experiencia metodológica, como ocurre en contextos educativos, equipos de estudiantes o proyectos de pequeña escala. Esta norma fue desarrollada específicamente para atender las necesidades de las *VSE*, que suelen operar con pocos integrantes, sin procesos documentados y con baja adopción de estándares, los cuales suelen percibirse como pensados para organizaciones grandes y complejas (Laporte et al., 2015).

En el ámbito académico, la norma ha sido utilizada con éxito en cursos de grado y posgrado, donde los estudiantes no solo han logrado comprender e implementar sus procesos, sino que también han participado activamente en su evaluación y mejora (Laporte et al., 2015). Estas experiencias demuestran que el estándar facilita el aprendizaje de buenas prácticas sin imponer formalismos excesivos, promoviendo un enfoque activo, crítico y contextualizado de la formación en ingeniería de software.

ISO/IEC 29110 permite estructurar el trabajo en base a procesos definidos, entregables claros y prácticas internacionalmente reconocidas. El Perfil Básico, compuesto por los procesos de Gestión de Proyectos e Implementación de Software, proporciona una guía concreta con actividades, roles y productos de trabajo que pueden ser documentados, evaluados y trazados. Esto favorece la adquisición de competencias clave en planificación, seguimiento, documentación y control de calidad (Rocha & Travassos, 2021).

Asimismo, investigaciones han demostrado que su implementación en proyectos estudiantiles no obstaculiza la creatividad ni impone una carga innecesaria, sino que, por el contrario, contribuye a mejorar la calidad del desarrollo mediante prácticas como la trazabilidad de requerimientos, la revisión entre pares y la validación de productos (Laporte, O'Connor & Fanmuy, 2016). Para facilitar aún más su aplicación, se han desarrollado plantillas, guías y recursos interactivos que ayudan a comprender e

implementar los productos de trabajo definidos por la norma, incluso en entornos no profesionales.

Por último, cabe destacar que la flexibilidad del estándar también ha permitido su combinación con marcos ágiles. Un ejemplo de ello es *Q-Scrum*, un enfoque híbrido propuesto por Pasini, Esponda, Boracchia y Pesado (2021) que fusiona la estructura documental y los productos de la ISO/IEC 29110 con la dinámica iterativa de *Scrum*. Este tipo de aproximaciones resulta especialmente valioso en contextos educativos o en equipos pequeños que adoptan prácticas ágiles, ya que permite combinar la orientación a calidad y trazabilidad de la norma con la adaptabilidad del desarrollo incremental.

En suma, ISO/IEC 29110 representa una vía eficaz para introducir a los estudiantes en el uso de procesos reales, alineados con estándares internacionales, y fortalecer sus habilidades profesionales en gestión y desarrollo de software desde una perspectiva accesible y formativa.

En este trabajo, la norma ISO/IEC 29110 se adopta como marco de referencia conceptual y práctico, con el objetivo de identificar y adaptar los productos de trabajo y lineamientos más pertinentes al contexto de los equipos estudiantiles. Esta perspectiva promueve la transferencia gradual de buenas prácticas profesionales al ámbito educativo, contribuyendo a una formación integral que vincula teoría, práctica y estándares de la industria del software.

CAPÍTULO V: ENFOQUE METODOLÓGICO

En este capítulo se describe el enfoque metodológico adoptado para la investigación, y se presenta el diseño general del estudio y los instrumentos empleados para la recolección de datos. Se detalla la construcción del cuestionario, la conformación de la muestra y la aplicación del instrumento, así como los criterios de análisis utilizados para interpretar la información obtenida. Finalmente, se expone el proceso seguido para la selección de herramientas de software, vinculando las dificultades observadas en los proyectos estudiantiles con los productos definidos en la norma ISO/IEC 29110.

5.1 Diseño general de la investigación

Esta investigación adopta un enfoque metodológico mixto, combinando métodos cualitativos y cuantitativos con el fin de proporcionar un análisis integral de la gestión de requerimientos en proyectos de software desarrollados por estudiantes del tercer año de las carreras de Analista de Sistemas y Licenciatura en Sistemas de Información en la Facultad de Ciencia y Tecnología (FCyT), sede Oro Verde.

El propósito general es identificar buenas prácticas y oportunidades de mejora en la forma en que los estudiantes gestionan los requerimientos a lo largo del ciclo de vida de sus proyectos, así como evaluar herramientas que podrían apoyar dichas prácticas en el marco de la norma ISO/IEC 29110.

Dado el carácter exploratorio del estudio, el análisis de los datos se centró en un tratamiento descriptivo simple, basado en frecuencias, promedios y observaciones cualitativas generales, sin aplicar técnicas estadísticas avanzadas.

5.2 Instrumentos de recolección de datos

Se diseñó un cuestionario alineado con los procesos y productos definidos en la norma ISO/IEC 29110 (ver Capítulo IV), con el objetivo de relevar información empírica sobre cómo los estudiantes gestionan los requerimientos en proyectos académicos. El instrumento completo se presenta en el Anexo II, como parte de la documentación metodológica del trabajo. El instrumento combinó preguntas cerradas y abiertas, permitiendo un análisis mixto (cuantitativo-descriptivo y cualitativo-exploratorio).

El cuestionario se estructuró en seis ejes temáticos principales, cada uno orientado a indagar aspectos clave de la gestión de requerimientos y su articulación con la práctica estudiantil:

- Eje 1: Especificación de requerimientos. Indagó la claridad con la que los estudiantes comprendieron el problema inicial, los mecanismos de representación empleados

(lenguaje natural, diagramas de casos de uso, historias de usuario, tablas, etc.) y las dificultades encontradas para delimitar el dominio del problema.

- Eje 2: Validación de requerimientos. Relevó la frecuencia y profundidad de las instancias de validación con el cliente o docente, así como la utilización de prototipos, demostraciones o revisiones incrementales como mecanismos de verificación.
- Eje 3: Gestión de cambios. Analizó cómo los equipos registraron y controlaron las modificaciones a los requerimientos, distinguiendo entre prácticas informales (listas, mensajes) y mecanismos formales (actas, solicitudes de cambio).
- Eje 4: Seguimiento y trazabilidad. Indagó el uso de herramientas o técnicas para seguir el avance, identificar desvíos y establecer relaciones entre requerimientos, diseño, código y pruebas.
- Eje 5: Uso de herramientas. Registró los tipos de herramientas utilizadas, tanto específicas (gestores de requerimientos, control de versiones) como generalistas (procesadores de texto, planillas, pizarras colaborativas), su grado de adopción y limitaciones.
- Eje 6: Experiencia y cierre del proyecto. Reunió percepciones generales sobre la organización del trabajo, la existencia de actividades de cierre (revisión de logros, documentación final, retroalimentación) y los aprendizajes obtenidos.

El cuestionario fue aplicado tanto de forma asincrónica como en entrevistas virtuales grupales, en las que se utilizó el mismo instrumento con acompañamiento conversacional. Este formato permitió profundizar en ciertos aspectos y obtener mayor claridad sobre el contexto de los trabajos desarrollados. Cada entrevista incluyó una breve introducción sobre el proyecto del estudiante, repasando aspectos formales del desarrollo de software para contextualizar las respuestas.

5.3 Muestra

La muestra estuvo conformada por estudiantes que realizaron su proyecto final en el Taller de Integración entre los años 2018 y 2025, según información proporcionada por los docentes de la cátedra. Este espacio curricular corresponde al tercer año y posee carácter transversal, ya que integra conocimientos técnicos, metodológicos y de gestión adquiridos en asignaturas previas. Se trató de una muestra de sujetos voluntarios, personas que eligieron participar en el estudio por decisión propia, sin haber sido seleccionadas al azar.

A partir del listado de proyectos disponibles, se identificaron los potenciales participantes y se los invitó a colaborar mediante el envío de un formulario en línea, acompañado de una breve explicación sobre los objetivos del estudio. En todos los

casos, se procuró resguardar la confidencialidad de las respuestas y el anonimato de los participantes.

5.4 Aplicación del instrumento

La recolección de datos se llevó a cabo principalmente mediante un formulario en línea, enviado por correo electrónico a los integrantes de la muestra. Este instrumento incluyó preguntas cerradas y abiertas, organizadas según los ejes temáticos definidos previamente (ver sección 5.2).

Adicionalmente, algunos de los estudiantes que accedieron a participar fueron invitados a entrevistas virtuales grupales, en las que se aplicó el mismo cuestionario bajo un formato conversacional asistido. Este enfoque permitió profundizar en ciertas respuestas, aclarar dudas y obtener mayor información contextual sobre los proyectos desarrollados.

Cada entrevista comenzó con una breve introducción centrada en el proyecto del estudiante, repasando aspectos formales del desarrollo de software, lo que permitió contextualizar mejor las respuestas obtenidas.

5.5 Análisis de datos

El análisis de los datos se realizó mediante técnicas descriptivas, entendidas como procedimientos básicos de la estadística orientados a resumir y organizar la información sin realizar inferencias ni generalizaciones. En este estudio se aplicaron frecuencias absolutas y relativas, así como medidas simples de tendencia central (por ejemplo, promedios), con el fin de representar de manera clara las respuestas obtenidas en las preguntas cerradas del cuestionario. Las respuestas abiertas, en cambio, fueron tratadas a través de un análisis cualitativo exploratorio, identificando patrones y temas recurrentes que permitieran reconocer buenas prácticas, dificultades comunes y oportunidades de mejora en la gestión de requerimientos.

5.6 Selección de herramientas

La selección de herramientas de software estuvo guiada por los problemas detectados en la gestión de requerimientos de proyectos estudiantiles y por los productos de trabajo definidos en la norma ISO/IEC 29110 (ver Capítulo IV). Con el fin de asegurar un proceso sistemático y adecuado al contexto educativo, se definió un procedimiento que contempló: (a) vincular las dificultades detectadas con los productos de trabajo de la norma, (b) establecer categorías funcionales de herramientas y fuentes de búsqueda, (c) definir criterios de evaluación, y (d) analizar alternativas mediante una matriz herramienta-producto.

Las etapas se describen a continuación.

5.6.1 Vinculación con los productos de la norma ISO/IEC 29110

Se analizaron los problemas relevados y se estableció su correspondencia con los productos de trabajo definidos por la norma. Se priorizaron aquellos cuya implementación resulte más relevante para mitigar las dificultades identificadas en el contexto académico.

5.6.2 Estrategia de identificación de alternativas

Se definieron categorías funcionales de herramientas potencialmente aplicables, tales como editores colaborativos, gestores visuales de tareas, repositorios con control de versiones y herramientas de diagramación. La identificación de alternativas dentro de cada categoría consideró documentación técnica, literatura especializada y prácticas reportadas por los estudiantes en el relevamiento.

La descripción y análisis de las herramientas seleccionadas se presenta posteriormente, una vez aplicado este procedimiento.

5.6.3 Criterios de evaluación

Los criterios de evaluación fueron propuestos por el tesista en función de los objetivos del estudio, los productos de trabajo priorizados y las condiciones reales del contexto académico. En particular se consideraron:

- Cobertura funcional: grado en que la herramienta permite implementar los productos de trabajo priorizados.
- Accesibilidad: disponibilidad gratuita, licencias educativas o facilidad de acceso.
- Usabilidad: facilidad de aprendizaje y uso para estudiantes sin experiencia avanzada.
- Entorno técnico amigable: compatibilidad básica con sistemas operativos habituales y sin requerimientos técnicos complejos para su instalación o uso.
- Gestión de cambios y trazabilidad: capacidades para registrar modificaciones, mantener versiones y relacionar requerimientos con otros elementos del proyecto.
- Resguardo y exportación de datos: posibilidad de almacenar, proteger y reutilizar la información generada.
- Condiciones de uso reales: restricciones prácticas que pueden afectar su implementación, como límites de tiempo, funciones bloqueadas o dificultades para exportar contenidos.

Las herramientas fueron evaluadas considerando cobertura funcional, accesibilidad, usabilidad, entorno técnico amigable, trazabilidad, resguardo y exportación

de datos, y condiciones de uso reales, otorgando mayor peso a los criterios de cobertura y trazabilidad por su relación directa con la aplicabilidad práctica y la alineación con los productos de la norma.

5.6.4 Diseño de análisis funcional herramienta-producto

Se diseñó una matriz herramienta–producto para contrastar, por cada alternativa, el soporte a los productos de trabajo priorizados. La valoración se realizó con una escala ordinal (Alta/Parcial/Limitada/Nula), complementada con observaciones cualitativas sobre trazabilidad, exportación y condiciones reales de uso. Este diseño permitió aplicar consistentemente los criterios definidos en 5.6.3.

Queda, entonces, especificado el procedimiento de selección y evaluación: vinculación con productos de la norma, estrategia de identificación de alternativas, criterios de evaluación y diseño de la matriz herramienta–producto. Este procedimiento fue el que se aplicó en el estudio.

CAPÍTULO VI: RESULTADOS, ANÁLISIS Y SÍNTESIS

Este capítulo presenta los resultados obtenidos a partir del cuestionario aplicado a estudiantes del espacio curricular Taller de Integración, común a las carreras de Analista de Sistemas y Licenciatura en Sistemas de Información de la Facultad de Ciencia y Tecnología (FCyT) de la Universidad Autónoma de Entre Ríos (UADER), sede Oro Verde. Este espacio curricular corresponde al tercer año y posee carácter transversal, ya que integra conocimientos técnicos, metodológicos y de gestión adquiridos en asignaturas previas. Se incluyen además observaciones derivadas de entrevistas virtuales grupales y del análisis funcional de herramientas tecnológicas relevantes para la gestión de requerimientos, así como su mapeo con los productos de trabajo definidos por la norma ISO/IEC 29110.

Los contenidos se estructuran en dos partes: primero se exponen los datos relevados, organizados según los ejes definidos en el diseño metodológico presentado en el capítulo anterior , y luego se analizan e interpretan a la luz del marco teórico y de las recomendaciones de dicha norma. Finalmente, se presenta una síntesis interpretativa que integra los hallazgos y sus implicancias para el contexto educativo. Estas secciones sirven de base para la construcción de una propuesta práctica que será desarrollada en el capítulo siguiente.

6.1 Resultados obtenidos

Los resultados derivados del cuestionario aplicado a los estudiantes permiten describir de manera integral la experiencia de los equipos en el Taller de Integración.

El análisis se organizó considerando dos dimensiones complementarias: por un lado, las dificultades enfrentadas durante el desarrollo de los proyectos; por otro, las fortalezas identificadas en las prácticas que los equipos lograron consolidar.

Esta doble perspectiva ofrece una visión equilibrada, que reconoce tanto los obstáculos que limitaron la gestión de requerimientos como los avances y aprendizajes que fortalecieron la práctica.

6.1.1 Caracterización general de los proyectos

Los estudiantes encuestados trabajaron en proyectos del espacio curricular Taller de Integración, correspondiente al tercer año de las carreras de Analista de Sistemas y Licenciatura en Sistemas de Información de la FCyT-UADER (sede Oro Verde), estuvieron dirigidos a clientes reales y buscaron resolver necesidades concretas en organizaciones locales.

En cuanto a los rubros de los mandantes, se observó una diversidad temática: salud, educación, tecnología e informática fueron los sectores más frecuentes. Esto sugiere que los proyectos enfrentaron problemáticas reales y variadas, lo que implicó distintos niveles de complejidad técnica y organizacional.

Los equipos fueron predominantemente reducidos: 37,5% estuvieron conformados por un único integrante, 37,5% por dos integrantes y 25% por tres. En consecuencia, más de tres cuartas partes de los proyectos se desarrollaron con menos de tres integrantes, lo que refleja la escala reducida y el carácter formativo de las experiencias.

En cuanto al estado de avance, el 81,25% de los proyectos fueron finalizados y aprobados, el 6,25% se encontraba aún en desarrollo y el 12,5% restante fue abandonado. Este dato indica que la gran mayoría de los equipos logró alcanzar un resultado funcional, aun con limitaciones.

Respecto de las metodologías empleadas, se observa una fuerte presencia de enfoques ágiles: *Scrum* (37,5%), *Extreme Programming* –XP– (25%) y *Kanban* (18,8%). También se registró un 12,5% de equipos que aplicaron cascada y un 6,2% que no especificó metodología. En conjunto, estos resultados confirman una preferencia por prácticas iterativas y adaptativas, aun cuando su aplicación haya sido parcial.

6.1.2 Fortalezas identificadas en las respuestas del cuestionario

Además de las dificultades enfrentadas por los equipos durante el desarrollo de sus proyectos, el análisis también permitió reconocer un conjunto de fortalezas que resultan igualmente relevantes para comprender la experiencia de los estudiantes. Estas fortalezas se reflejan en las prácticas positivas que los equipos lograron incorporar durante el desarrollo de sus proyectos, y fueron extraídas de las respuestas abiertas y de los patrones observados en las preguntas estructuradas del cuestionario:

- Metodología de desarrollo: La adopción de marcos ágiles, ya mencionada en la caracterización general de los proyectos (ver Capítulo II), constituye aquí una fortaleza en tanto refleja una apropiación inicial de enfoques estructurados para organizar el trabajo, aun cuando no se aplicaron de manera estricta.
- Prueba continua: Ejecución de pruebas funcionales a lo largo del desarrollo, no limitándose al cierre, lo que evidencia preocupación por la calidad del producto y una validación progresiva del funcionamiento.
- Registro documental: Documentación básica de acuerdos de reuniones y de defectos detectados, aun con herramientas como Editor de Texto o Planillas de cálculo, lo que muestra cierta conciencia sobre la importancia del seguimiento.

- Estrategia de comunicación: Incluso en equipos unipersonales, los estudiantes buscaron mecanismos para coordinar con los clientes o entre pares, utilizando canales como WhatsApp o correo electrónico. Esta adaptación al contexto operativo muestra capacidad de resolución y flexibilidad.
- Comprensión de requerimientos: Capacidad de los equipos para interpretar y delimitar de manera adecuada los requerimientos y objetivos iniciales planteados por el cliente. Esta comprensión favoreció la organización del trabajo posterior, aunque en ciertos casos fue necesario realizar aclaraciones adicionales para alcanzar una definición más precisa.
- Reflexión crítica: Manifestación de una actitud autocrítica por parte de los equipos, expresada en comentarios y propuestas orientadas a la mejora de aspectos como la planificación y la formalización de prácticas. Esta disposición evidencia una orientación hacia el aprendizaje continuo y la consolidación de competencias profesionales.

6.1.3 Hallazgos sobre prácticas y dificultades en los proyectos encuestados

A partir de los datos recolectados mediante el cuestionario diseñado en el Capítulo V (sección 5.2), se presentan a continuación los resultados organizados en siete ejes temáticos, que agrupan los aspectos centrales de la gestión de requerimientos en los proyectos estudiantiles. Los seis primeros ejes corresponden a las dimensiones previstas en el diseño metodológico, mientras que el séptimo se incorporó durante el análisis cualitativo, a partir de observaciones recurrentes vinculadas a la estimación del esfuerzo. En cada uno de estos siete ejes se expone información cuantitativa (frecuencia de respuestas) y cualitativa (observaciones, patrones y dificultades).

6.1.3.1 Eje 1: Especificación de los requerimientos

En relación con la comprensión inicial del problema, los resultados muestran una distribución heterogénea. El 68,75% de los equipos indicó que el planteamiento del cliente fue comprensible en general, aunque requirió aclaraciones puntuales; un 6,25% lo percibió como muy claro, mientras que un 12,5% lo consideró bastante confuso y un 6,25% algo confuso. Finalmente, un 6,25% de los equipos no aplicó esta pregunta, probablemente por la naturaleza particular del proyecto. En conjunto, puede afirmarse que tres de cada cuatro equipos lograron comprender de manera adecuada el problema, aunque más de una cuarta parte reportó ambigüedades relevantes.

En cuanto a la forma de documentar los requerimientos, el 62,5 % usó descripciones en lenguaje natural, y el 31,2 % recurrió a historias de usuario. A su vez, el 87,5% empleó algún tipo de representación gráfica (como casos de uso, diagramas de

flujo, modelos entidad–relación, prototipos o *wireframes*) para complementar la descripción textual.

Este abanico de recursos refleja un esfuerzo de diversificación y búsqueda de apoyos visuales que resulta valioso para la comprensión compartida entre cliente y equipo. No obstante, la aplicación de estas representaciones fue heterogénea y sin un marco metodológico unificado, lo que dificulta la trazabilidad y limita la posibilidad de reutilización sistemática. Estas características se corresponden con los problemas identificados como ambigüedad inicial en los requerimientos (P1) y falta de conocimiento de procesos formales (P6).

6.1.3.2 Eje 2: Validación de los requerimientos

El proceso de validación con los clientes se desarrolló de manera parcial y heterogénea. El 56,2% de los equipos realizó reuniones con el cliente para validar los requerimientos, lo que evidencia un esfuerzo inicial por sostener la comunicación directa. A su vez, un 25% complementó esta instancia mediante la presentación de prototipos o representaciones visuales, recurso que favorece la detección temprana de inconsistencias.

Sin embargo, el 18,8% de los equipos reconoció no haber validado de forma explícita o haber tenido dificultades para comprender completamente las necesidades del cliente. Esto revela que las prácticas de validación tienden a concentrarse en intercambios verbales o textuales, sin apoyarse en técnicas formales ni en criterios sistemáticos de aceptación.

Estas situaciones se relacionan con los problemas de escasa validación formal (P2) y comunicación ineficiente con el cliente (P7).

6.1.3.3 Eje 3: Gestión de cambios

En relación con la gestión de cambios, los resultados muestran que se trata de una práctica habitual, aunque poco sistematizada. El 100% de los equipos que respondieron reconoció haber experimentado modificaciones en los requerimientos, ya sea por solicitudes de la cátedra o el cliente, por mejoras propuestas internamente o por la detección de ambigüedades durante la validación. Esto confirma que, en contextos educativos, la adaptación de los requerimientos es la norma más que la excepción.

Sin embargo, solo el 12,5% de los equipos reportó haber utilizado algún mecanismo formal para documentar y controlar los cambios, como el uso de repositorios con control de versiones. En la mayoría de los casos, las modificaciones se gestionaron de manera informal, mediante acuerdos verbales o ajustes directos sobre el código y la documentación existente.

Este panorama revela una alta exposición a cambios, pero también una escasa capacidad para registrar y trazar sus efectos, lo cual debilita la consistencia de los proyectos.

La ausencia de registros sistemáticos refleja una debilidad importante en el manejo del alcance a lo largo del proyecto, la cual se refleja en los códigos P3 (falta de consistencia y trazabilidad) y P4 (ausencia de control de cambios en los requerimientos).

6.1.3.4 Eje 4: Seguimiento y trazabilidad

Las actividades de seguimiento y trazabilidad resultaron limitadas y, en la mayoría de los casos, informales.

Solo el 31,2% de los equipos indicó haber utilizado algún mecanismo básico de control del avance (como cronogramas, listas de tareas o tableros digitales), mientras que el 68,8% restante no realizó un seguimiento estructurado de las actividades.

Asimismo, no se registró la aplicación de matrices de trazabilidad ni de plantillas que permitieran mantener alineados los requerimientos con las tareas de desarrollo, las pruebas o los entregables. La ausencia de estos artefactos metodológicos muestra que, aunque algunos equipos intentaron organizar el progreso, el control de la consistencia y la cobertura de los requerimientos quedó mayormente desatendido.

Este panorama revela una gestión poco sistemática del progreso, dificultando la detección temprana de desvíos y afectando la consistencia general del proyecto y la calidad del desarrollo. Estas carencias se reflejan en los problemas P3 (trazabilidad débil) y P5 (falta de planificación y control).

6.1.3.5 Eje 5: Uso de herramientas

El uso de herramientas tecnológicas fue amplio, aunque con distintos niveles de aprovechamiento. Todos los equipos utilizaron herramientas de ofimática (*Google Docs, Word, Excel*) para redactar documentos y el 81,2% utilizó sistemas de control de versiones, lo que representa una práctica extendida y valiosa para equipos noveles, ya que favorece la colaboración y el resguardo del código fuente. En menor medida, se emplearon herramientas de gestión de tareas como Jira o Trello (18,8%), diagramadores o pizarras digitales como Draw.io o Miro (31,2%), y recursos multimedia como grabaciones de reuniones (12,5%).

No obstante, estos resultados muestran que, si bien hay interés en incorporar herramientas, su uso aún no se articula con una gestión formal de requerimientos. Esta falta de apropiación metodológica se corresponde con el problema P6 (falta de conocimiento en procesos estructurados).

6.1.3.6 Eje 6: Cierre del proyecto y reflexión final

Las etapas de cierre mostraron un nivel limitado de formalización. Aunque en la mayoría de los casos la entrega se concretó de manera funcional, no siempre incluyó instancias estructuradas de validación con el cliente ni documentación de aprendizajes.

Según los datos, el 62,5% de los equipos afirmó haber recibido una aceptación explícita por parte del cliente, ya sea mediante revisiones, entregas validadas o registros formales. Este indicador refleja la existencia de prácticas de validación final en términos operativos, aunque sin un marco metodológico que asegure su sistematicidad.

En ningún caso se identificaron mecanismos de reflexión final ni registros de lecciones aprendidas, lo que limita la posibilidad de capitalizar la experiencia y consolidar buenas prácticas.

Estas debilidades se relacionan con los problemas P2 (validación informal) y P5 (falta de cierre planificado).

6.1.3.7 Eje 7: Estimación del esfuerzo y planificación de tareas

Finalmente, se identificó una dificultad transversal mencionada en varias respuestas abiertas: la falta de herramientas o criterios claros para estimar el esfuerzo requerido por las tareas del proyecto. el 43,8% de los equipos manifestó explícitamente no saber cómo dimensionar el trabajo, ya sea subestimando la duración de tareas o acumulando actividades críticas hacia el final del cronograma, el resto no reportó dificultades específicas en ese aspecto, aunque la evidencia cualitativa sugiere que la planificación del esfuerzo fue en general, poco estructurada y documentada.

Esta situación derivó en cronogramas poco realistas, acumulación de trabajo hacia las fechas de entrega y dificultades para identificar desvíos a tiempo. También se observó que en equipos unipersonales o de baja coordinación, el seguimiento se realizaba de manera informal, sin registros que permitieran evaluar avances o redistribuir esfuerzos de forma planificada.

Estas observaciones permiten identificar un nuevo problema, que será codificado como P8 – Dificultad para estimar tareas y esfuerzo. Este problema afecta directamente a la planificación inicial y al seguimiento posterior del proyecto, y se vincula con productos clave del proceso de Gestión de Proyectos, tales como el Plan del Proyecto, el Cronograma y el Registro del estado de avance. Su abordaje será considerado en las recomendaciones del capítulo siguiente y en los recursos prácticos provistos en el capítulo VII.

6.1.4 Sistematización de problemas comunes detectados

A lo largo de los siete ejes temáticos analizados se identificaron múltiples dificultades recurrentes. En esta sección se sistematizan dichas observaciones mediante una codificación (P1 a P8), que permitirá organizar los hallazgos de manera más estructurada para facilitar el diseño de propuestas posteriores. La Tabla 3 resume estas dificultades, indicando para cada código el problema identificado.

Estas dificultades están alineadas con hallazgos previos en la literatura, que indican que los desarrolladores noveles tienden a subestimar las actividades de documentación y control, priorizando el avance técnico del desarrollo (Rocha & Travassos, 2021; Laporte et al., 2015).

Tabla 3

Codificación de las dificultades identificadas

Código	Dificultad identificada
P1	Ambigüedad en la definición inicial de requerimientos.
P2	Escasa validación formal con el cliente.
P3	Dificultades para mantener consistencia y trazabilidad a lo largo del proyecto.
P4	Falta de registro y control frente a cambios en los requerimientos.
P5	Escasa planificación inicial del alcance y seguimiento de tareas.
P6	Falta de conocimiento o experiencia en procesos estructurados de desarrollo.
P7	Comunicación ineficiente con el cliente.
P8	Dificultad para estimar tareas y esfuerzo.

Nota: La codificación P1–P8 se emplea como referencia en las secciones siguientes.

Las dificultades relevadas en este estudio afectan principalmente la especificación, validación, seguimiento y documentación de los requerimientos, así como el uso de herramientas que podrían facilitar dichas prácticas. La comunicación con el cliente también fue señalada como un aspecto débil, reforzando la importancia de formalizar los mecanismos de validación y seguimiento. Estas observaciones sirven como base para proponer recomendaciones concretas en el capítulo siguiente.

6.1.5 Correspondencia entre dificultades y productos de la norma ISO/IEC 29110

Los resultados muestran que las dificultades sistematizadas en la sección 6.1.4 (P1 - P8) encuentran un correlato en los productos de trabajo definidos por la norma ISO/IEC 29110, presentados previamente en el Capítulo IV. Estos productos derivados de prácticas normativas orientadas a asegurar la calidad y el control en proyectos de pequeña escala, permiten establecer una relación directa entre los problemas identificados en contextos educativos y las soluciones sugeridas por el marco normativo.

En este estudio se optó por focalizar el análisis en los productos de trabajo, dado que representan los artefactos concretos que los equipos pueden generar y utilizar, y permiten establecer de manera más directa la relación con las dificultades detectadas en contextos educativos. A continuación, la Tabla 4 presenta la correspondencia entre los problemas (P1 a P8) sistematizados en la sección anterior y los productos de trabajo pertinentes que pueden contribuir a abordarlos.

Tabla 4

Correspondencia entre dificultades y productos de la norma

Cód	Producto de la norma	Justificación
P1	Especificación de Requerimientos	Formaliza el contenido, formato y alcance de los requerimientos.
P2	Registro de Aceptación; Actas de Reunión	Documentan acuerdos, validaciones y conformidad del cliente.
P3	Matriz de Trazabilidad; Repositorio del Proyecto	Permiten relacionar requerimientos con pruebas, tareas y entregables.
P4	Solicitud de Cambio; Registro de cambios	Aseguran seguimiento formal de modificaciones.
P5	Plan del Proyecto; Cronograma	Organizan el trabajo, anticipan riesgos, permiten seguimiento y coordinación.
P6	Guía de proceso interna; Plantillas, Repositorio estructurado	Proveen estructura y orientación a equipos sin experiencia.
P7	Actas de Reunión; Registro de decisiones	Favorecen la comunicación y registro de compromisos con el cliente.
P8	Plan del Proyecto; Cronograma, Registro de estado de avance	Permiten estimar el esfuerzo, descomponer tareas y registrar avances de forma sistemática.

Este artefacto permite establecer y visualizar las relaciones entre los distintos elementos del ciclo de vida (requerimientos, tareas de desarrollo, pruebas y entregables),

asegurando que cada requerimiento esté correctamente implementado, validado y vinculado a los artefactos correspondientes. Su utilización facilita el control de cambios y la verificación de la cobertura funcional del sistema, contribuyendo a mejorar la consistencia y la calidad del desarrollo. Las actividades y tareas completas asociadas a cada producto se encuentran detalladas en la norma ISO/IEC 29110 y, de manera resumida, en el Anexo IV de este documento.

6.1.6 Productos excluidos

Si bien la norma ISO/IEC 29110 propone un conjunto amplio de productos de trabajo, en el presente estudio se priorizaron aquellos directamente vinculados con los problemas más frecuentes detectados en los proyectos estudiantiles.

Esta priorización responde al objetivo de adaptar las recomendaciones normativas al entorno educativo, focalizando en las prácticas que pueden tener mayor impacto inmediato y que resultan viables de aplicar en proyectos de baja escala, sin experiencia formal previa.

En consecuencia, algunos productos no fueron considerados en el análisis funcional, ya sea porque no atienden directamente a las dificultades identificadas, o porque su aplicación resulta más pertinente en etapas avanzadas del proyecto o en contextos profesionales más estructurados. La Tabla 5 detalla cuáles son estos productos y expone la justificación de su exclusión.

Tabla 5

Productos de la norma excluidos

Producto de la norma	Motivo de exclusión
Lista de riesgos	Requiere mayor madurez metodológica; puede incorporarse en etapas futuras.
Plan de verificación	No identificado como una necesidad por los equipos.
Registro de pruebas	Aporta valor, pero no directamente vinculado a los problemas P1–P8.
Lecciones aprendidas	Producto opcional según la ISO/IEC 29110. Pertinente para el cierre, pero no clave para la gestión de requerimientos.

6.1.7 Resultados de la evaluación de herramientas de soporte

En esta sección se presentan los resultados de la evaluación de herramientas orientadas a sostener los productos del Perfil Básico de la ISO/IEC 29110 en contextos educativos. La evaluación se realizó conforme al procedimiento y criterios establecidos en el Capítulo V. A continuación se exponen: (i) una descripción funcional de las herramientas consideradas, (ii) la cobertura alcanzada respecto de los productos de la norma, (iii) la evaluación técnica por criterios y (iv) la selección final propuesta.

6.1.7.1 Descripción funcional de las herramientas

A continuación se describen las herramientas seleccionadas, evaluadas explícitamente según los siete criterios de evaluación definidos en la sección 5.6.3, y se detalla su vinculación con los productos de trabajo de la norma ISO/IEC 29110. La evaluación se basa en la revisión de documentación oficial, pruebas piloto del autor y la alineación con las prácticas reales de los estudiantes.

Google[9] Workspace (Docs, Sheets, Drive, Calendar, Meet)

Google Workspace es un conjunto integrado de aplicaciones en la nube que incluye procesador de textos (Docs), hojas de cálculo (Sheets), almacenamiento (Drive), calendario (Calendar) y videoconferencias (Meet). Está diseñado para la colaboración en tiempo real, la gestión centralizada de documentos y la comunicación sincrónica y asincrónica. Es ampliamente utilizado en entornos educativos y empresariales por su simplicidad, accesibilidad y robustez.

Documentación oficial:

https://workspace.google.com/

Evaluación por criterios:

- Cobertura funcional: Alta. Permite implementar de forma directa productos como Especificación de Requerimientos, Actas de Reunión, Plan del Proyecto (en Sheets), Cronograma y Registro de Decisiones. Su uso estructurado permite abordar P1, P2, P5, P7 y P8.

[9] Google LLC es una empresa multinacional estadounidense fundada en 1998 y subsidiaria de Alphabet Inc. Ofrece servicios y productos relacionados con Internet, *software* y computación en la nube.

- Accesibilidad: Excelente. Disponible gratuitamente con una cuenta de Google; las instituciones educativas suelen contar con planes institucionales (Google Workspace for Education) con mayor cuota de almacenamiento y funciones avanzadas.
- Usabilidad: Muy alta. Interfaz intuitiva, ampliamente familiar para los estudiantes, con curva de aprendizaje casi nula.
- Entorno técnico amigable: Totalmente web, sin instalación. Compatible con cualquier sistema operativo y dispositivo con navegador.
- Gestión de cambios y trazabilidad: Limitada. Ofrece historial de versiones automático y comentarios, pero no permite enlaces cruzados automáticos entre documentos (trazabilidad manual). Mitigable mediante nomenclatura estandarizada y uso de hipervínculos.
- Resguardo y exportación de datos: Alta. Permite exportar en múltiples formatos (PDF, DOCX, XLSX) y respaldar en Google Drive con control de permisos granular.
- Condiciones reales de uso: El plan gratuito incluye 15 GB compartidos (puede ser limitante en proyectos con muchos diagramas o archivos multimedia), pero los planes educativos eliminan esta restricción. La dependencia de internet se mitiga con modo offline activable.

Trello[10]

Trello es una aplicación de gestión visual de tareas basada en tableros Kanban. Permite organizar el trabajo en listas (por ejemplo: “Por hacer”, “En progreso”, “Hecho”) y tarjetas que representan tareas individuales. Es ideal para equipos que necesitan una vista clara del flujo de trabajo sin complejidad técnica.

Documentación oficial:

https://trello.com/guide

Evaluación por criterios:

- Cobertura funcional: Media. Soporta visualmente el Plan del Proyecto, Solicitud de Cambio y el seguimiento del estado de requerimientos (P3, P4, P5). No sustituye documentación formal.

[10] Trello es una aplicación de gestión de proyectos creada en 2011 y actualmente propiedad de Atlassian, empresa australiana de *software* conocida por productos como Jira y Confluence.

- Accesibilidad: Alta. Versión gratuita robusta, aunque con límites (10 tableros, 1 Power-Up por tablero).
- Usabilidad: Muy alta. Interfaz Kanban visual e intuitiva, ideal para equipos noveles.
- Entorno técnico amigable: Web y apps móviles; no requiere instalación ni configuración técnica.
- Gestión de cambios y trazabilidad: Parcial. Permite registrar cambios y estados, pero la trazabilidad es manual. Se mejora con el Power-Up de GitHub (vinculación con issues y pull requests), pero este consume el único Power-Up permitido en el plan gratuito.
- Resguardo y exportación de datos: Media. Permite exportar en JSON, pero no en formatos estándar para auditoría (PDF, Excel). La información crítica debe complementarse en Docs/Sheets.
- Condiciones reales de uso: El límite de 1 Power-Up y la brecha de seguridad reportada en enero de 2024 (exposición de datos de usuarios) son riesgos a considerar. Para trazabilidad robusta, se recomienda complementar con GitHub/GitLab.

GitHub[11]

GitHub es una plataforma de desarrollo basada en Git que permite el control de versiones del código fuente, la gestión de incidencias (issues), revisiones de código (pull requests) y la automatización de pruebas mediante CI/CD (GitHub Actions). Es el estándar de facto en la comunidad de software libre y se ha convertido en una plataforma central para el desarrollo colaborativo.

Documentación oficial:

https://docs.github.com/

Evaluación por criterios

- Cobertura funcional: Alta. Soporta Repositorio del Proyecto, Solicitud de Cambio (Issues), Registro de Cambios (Pull Requests), Matriz de Trazabilidad (vía enlaces entre Issues, commits y PRs) y documentación técnica (Wiki). Aborda P3, P4 y P6.
- Accesibilidad: Excelente. Repositorios privados ilimitados y 2.000 minutos/mes de CI/CD en el plan gratuito.

[11] GitHub es una plataforma fundada en 2008 y adquirida por Microsoft en 2018. Es ampliamente reconocida como el principal repositorio de proyectos de *software* libre y colaborativo.

- Usabilidad: Media. Requiere conocimientos básicos de Git, pero su interfaz web y la familiaridad de los estudiantes reducen la barrera.
- Entorno técnico amigable: Web + CLI. Compatible con todos los sistemas operativos. La integración con el entorno de desarrollo (IDEs) es fluida.
- Gestión de cambios y trazabilidad: Alta. Ofrece trazabilidad nativa mediante la vinculación automática entre Issues, commits, ramas y PRs. Ideal para mantener consistencia y auditar cambios (P3, P4).
- Resguardo y exportación de datos: Alta. Todo el contenido es versionado y exportable mediante clonación o API.
- Condiciones reales de uso: Los 2.000 minutos de CI/CD son suficientes para la mayoría de los proyectos académicos. El GitHub Student Developer Pack ofrece beneficios adicionales.

GitLab[12]

GitLab es una plataforma DevOps "todo en uno" que integra control de versiones, gestión de incidencias, CI/CD, wiki y tableros de seguimiento en una sola aplicación. Su enfoque integrado reduce la necesidad de múltiples herramientas y centraliza todo el ciclo de vida del software en un único entorno.

Documentación oficial:

https://docs.gitlab.com/

Evaluación por criterios:

- Cobertura funcional: Alta. Similar a GitHub, con soporte integrado para Repositorio, Issues, CI/CD, Wiki y Boards. Cubre los mismos productos y problemas que GitHub.
- Accesibilidad: Buena. Plan gratuito con repositorios privados ilimitados, pero solo 400 minutos de CI/CD/mes y 10 GiB/proyecto.
- Usabilidad: Media. Interfaz más densa que GitHub, pero todo en una sola plataforma.
- Entorno técnico amigable: Web + CLI. Ofrece también versión autogestionada, aunque no necesaria en contexto académico.
- Gestión de cambios y trazabilidad: Alta. Igual que GitHub, con trazabilidad nativa entre artefactos.
- Resguardo y exportación de datos: Alta. Exportación vía clonación o API.

[12] GitLab Inc. es una empresa de software fundada en 2014, con sede en San Francisco. Ofrece una plataforma DevOps integrada bajo un modelo de código abierto.

- Condiciones reales de uso: El límite de CI/CD es restrictivo, pero el plan GitLab Ultimate es gratuito para instituciones educativas, eliminando esta limitación.

Draw.io[13] (diagrams.net)

Draw.io (ahora diagrams.net) es una herramienta de diagramación gratuita y de código abierto que permite crear diagramas UML, BPMN, ER, flujos de proceso y mockups. Está diseñada para ser simple, versátil y compatible con múltiples plataformas de almacenamiento, incluyendo Google Drive, GitHub y GitLab.

Documentación oficial:

https://www.diagrams.net/doc/

Evaluación por criterios:

- Cobertura funcional: Específica. Soporta la creación de Representaciones Visuales (diagramas UML, BPMN, flujos), clave para la Especificación de Requerimientos y la validación temprana (P1, P2).
- Accesibilidad: Excelente. Totalmente gratuito, de código abierto, sin registro obligatorio.
- Usabilidad: Alta. Bibliotecas predefinidas, arrastrar y soltar, y plantillas para diagramas comunes.
- Entorno técnico amigable: Disponible como app web, escritorio o integración en Google Drive/GitHub. No requiere instalación si se usa en el navegador.
- Gestión de cambios y trazabilidad: No nativa. Pero al guardarse en Google Drive o en repositorios Git (en formato .drawio.svg), hereda la trazabilidad del entorno de almacenamiento.
- Resguardo y exportación de datos: Alta. Exporta a PNG, SVG, PDF y XML. El formato .drawio.svg permite “visual diffs” en Git, reduciendo drásticamente la tasa de conflictos.
- Condiciones reales de uso: No presenta limitaciones funcionales. La única consideración es el formato de archivo para evitar conflictos en Git.

[13] diagrams.net (antes Draw.io) es un proyecto de software libre desarrollado por la empresa JGraph Ltd., con sede en Londres, especializado en herramientas de diagramación en la nube.

6.1.7.2 Cobertura de productos de la norma por herramienta

La Tabla 6 resume el grado de soporte por herramienta para los principales productos del Perfil Básico de la ISO/IEC 29110, con la escala: Alta / Parcial/ Limitada/ Nula.

Tabla 6

Soporte de herramientas para productos de la norma ISO/IEC 29110

Producto de la norma	Google Docs	Trello	GitHub	Google Sheets	Google Calendar	Draw.io
Especificación de Requerimientos	Alta	Parcial	Limitada	Alta	Nula	Parcial
Actas de Reunión / Registro de decisiones	Alta	Limitada	Limitada	Parcial	Parcial	Nula
Solicitud de Cambio / Registro de cambios	Limitada	Alta	Alta	Limitada	Nula	Nula
Plan del Proyecto	Limitada	Parcial	Nula	Alta	Parcial	Nula
Cronograma	Nula	Parcial	Nula	Parcial	Alta	Nula
Matriz de Trazabilidad / Repositorio	Limitada	Nula	Alta	Parcial	Nula	Nula
Repositorio del Proyecto / Configuración	Nula	Nula	Alta	Limitada	Nula	Nula
Representaciones visuales / Mockups	Limitada	Nula	Nula	Nula	Nula	Alta

Nota. Escala: Alta = soporte nativo y completo; Parcial = soporte indirecto o con configuración simple; Limitada = registro manual sin vínculos automáticos o configuraciones ad hoc; Nula = no aplicable.

6.1.7.3 Evaluación por criterios técnicos

Las herramientas fueron analizadas funcionalmente, evaluando según los criterios previamente establecidos. La Tabla 7 muestra los resultados obtenidos.

Tabla 7

Evaluación técnica de herramientas según criterios establecidos

Herramienta	CF	AC	US	ET	GT	RE	CU	Comentario general
Google Docs	A	A	A	A	L	A	P	Excelente para documentación y acuerdos; carece de enlaces cruzados automáticos.

Tabla 7 (continuación)

Herramienta	CF	AC	US	ET	GT	RE	CU	Comentario general
Google Sheets	P	A	A	A	L	A	P	Útil para cronogramas y matrices simples; trazabilidad manual.
Trello	P	A	A	A	L	P	P	Visual e intuitivo; ideal para seguimiento de tareas, no sustituye repositorio formal.
GitHub	A	A	M	A	A	A	A	Potente para código y trazabilidad; requiere conocimientos básicos de Git.
GitLab	A	P	M	A	A	A	P	Similar a GitHub, más integral pero con límites en el plan gratuito.
Draw.io	P	A	A	A	L	A	A	Muy útil para representaciones visuales; depende de repositorio externo para trazabilidad.
Google Calendar	P	A	A	A	L	A	A	Adecuado para planificación y coordinación de reuniones.

Nota. Abreviaturas: CF = Cobertura funcional; AC = Accesibilidad; US = Usabilidad; ET = Entorno técnico amigable; GT = Gestión de cambios y trazabilidad; RE = Resguardo / exportación de datos; CU = Condiciones de uso reales.
Escala: A = Alta; P = Parcial; L = Limitada; M = Media.

6.1.7.4 Selección final y justificación

Las herramientas analizadas ofrecen cobertura parcial y complementaria de los problemas identificados (P1–P8) y de los productos de trabajo del Perfil Básico de la norma ISO/IEC 29110. Para lograr una cobertura más completa y efectiva es necesario utilizarlas en conjunto, estableciendo convenciones operativas claras y un flujo de trabajo integrado. Esta coordinación permite abordar de forma coherente, sostenible y de bajo costo las dificultades más críticas en contextos académicos.

La selección final propone el siguiente ecosistema mínimo viable:

Google Workspace (Docs, Sheets, Drive, Calendar, Meet) como pilar documental y repositorio central.

Trello como tablero visual para la gestión de tareas, cambios y seguimiento del estado de los requerimientos.

GitHub como plataforma principal para el control de versiones del código, la trazabilidad entre artefactos y la ejecución automatizada de pruebas.

Draw.io (diagrams.net) como herramienta de diagramación integrada con el repositorio y el entorno colaborativo.

Tanto GitHub como GitLab son plataformas viables para la gestión del código fuente y la trazabilidad de artefactos a lo largo del ciclo de vida del software. En este contexto, se recomienda GitHub como opción predeterminada para la mayoría de los equipos estudiantiles. Esta elección se fundamenta, en primer lugar, en la generosidad de su plan gratuito, que al momento de la redacción de este trabajo (verificado en la documentación oficial de la plataforma en abril de 2025) ofrece 2.000 minutos mensuales de ejecución automatizada de pruebas (*GitHub Actions*) para repositorios privados, frente a los 400 minutos que ofrece el plan gratuito de GitLab. Este recurso es crítico para la verificación sistemática de la calidad del software y representa una ventaja significativa para proyectos académicos con ciclos de integración frecuentes.

En segundo lugar, la amplia adopción de GitHub en la comunidad de desarrollo de software facilita el acceso a una abundante oferta de tutoriales, ejemplos de configuración y soporte informal, lo que reduce la curva de aprendizaje para estudiantes noveles.

Aunque GitLab también ofrece una alternativa sólida (especialmente en su modalidad autogestionada, que permite a una organización desplegar su propia instancia), esta opción requiere disponer de infraestructura y personal técnico para su instalación, mantenimiento y actualización, un escenario que excede las capacidades y los objetivos de los proyectos del Taller de Integración. Por ello, en el contexto académico se considera únicamente la oferta de la plataforma como servicio (SaaS). GitLab puede considerarse como una alternativa viable en contextos donde la institución disponga del plan educativo Ultimate (gratuito para universidades), o cuando se priorice un entorno más integrado y autocontenido, que incluye en una sola aplicación funciones de documentación, seguimiento de tareas y automatización.

En cualquier caso, se recomienda a los equipos revisar la documentación oficial de la plataforma elegida para profundizar en aspectos específicos como las estrategias de ramificación o la configuración de flujos de integración. Estos elementos, aunque relevantes para la consolidación del proceso, exceden el alcance del presente análisis y se consideran oportunidades de mejora para etapas posteriores de madurez metodológica.

6.2 Síntesis interpretativa

Los hallazgos presentados en los ejes temáticos (sección 6.1.3) evidencian que las principales dificultades en la gestión de requerimientos guardan estrecha relación con los conceptos y fundamentos expuestos en el Capítulo II (Ingeniería de Requerimientos) y con los productos de proceso de la norma ISO/IEC 29110 presentados en el Capítulo IV. Sobre esta base, el análisis de resultados evidencia que, si bien los estudiantes lograron completar sus proyectos, la gestión de requerimientos presenta debilidades estructurales que limitan la calidad del proceso y el producto desarrollado.

En la especificación de requerimientos (Eje 1), predominó el uso de lenguaje natural y estructuras informales, lo que dio lugar a ambigüedades o contradicciones (P1) y dificultades para comprender el dominio del problema (P6). La falta de plantillas o modelos estandarizados afectó la claridad inicial, obligando en muchos casos a reinterpretaciones tardías.

En cuanto a la validación (Eje 2), si bien el 60% de los equipos realizó alguna instancia con el cliente, solo una minoría incorporó prototipos o criterios de aceptación. Esta validación parcial o informal se tradujo en una comunicación ineficiente (P7) y en una escasa capacidad para detectar errores o redefinir requerimientos (P2).

Respecto a la gestión de cambios (Eje 3), pocos equipos documentaron modificaciones de manera sistemática. En varios casos, los cambios fueron incorporados sin registro (P4), lo que dificulta la trazabilidad y el análisis posterior del proceso. Esta situación refleja la ausencia de procedimientos explícitos y un desconocimiento general sobre cómo abordar el control de cambios en proyectos reales.

La falta de seguimiento y trazabilidad (Eje 4) fue otro aspecto crítico. La mayoría de los equipos no realizó un control formal del avance ni registró los desvíos respecto a lo planificado (P5). Tampoco se evidenció el uso de matrices de trazabilidad o herramientas que permitan relacionar los requerimientos con otras actividades del ciclo de vida, lo que afecta directamente la consistencia del desarrollo (P3).

En relación al uso de herramientas (Eje 5), se constató una alta dependencia de herramientas generalistas (procesadores de texto, hojas de cálculo, pizarras digitales), mientras que el uso de herramientas específicas para gestión de requerimientos fue marginal (P6). Esta elección parece responder más a la familiaridad que a criterios funcionales, lo que refuerza una gestión centrada en lo operativo, con escasa formalización.

Finalmente, en el cierre del proyecto (Eje 6), la falta de instancias de revisión formal y de aceptación documentada refleja debilidades en el cierre del ciclo de vida. Pocos estudiantes indicaron haber realizado una reflexión sistemática sobre los aprendizajes obtenidos, lo que limita la retroalimentación para futuros proyectos (P2, P5).

En conjunto, estas observaciones permiten identificar un patrón de informalidad que atraviesa todos los ejes analizados.

Algunos casos particulares permiten ilustrar esta tendencia general a la informalidad. Por ejemplo, un equipo que adoptó prácticas de desarrollo ágil evitó el uso de presentaciones formales y recurrió a mostrar directamente el sistema en funcionamiento durante las reuniones con los solicitantes, permitiendo así validar requerimientos de forma inmediata. Si bien esta estrategia promovió una retroalimentación ágil, también generó modificaciones espontáneas y sin registro sistemático, lo que refuerza las debilidades observadas en la trazabilidad y en la gestión de cambios. En el otro extremo, varios equipos unipersonales simplificaron su dinámica de trabajo al punto de omitir actividades clave de planificación o validación, confiando en interpretaciones personales del problema sin instancias de verificación externa.

Estos casos reflejan cómo la falta de estructura puede derivar tanto en decisiones improvisadas como en simplificaciones excesivas, con impacto negativo sobre la calidad de la gestión de requerimientos. La presencia sistemática de informalidad, la escasa documentación estructurada y la limitada utilización de herramientas específicas sugieren que los equipos podrían beneficiarse de orientaciones concretas, adaptadas al contexto formativo.

El conjunto de evidencias recogidas confirma la necesidad de contar con orientaciones prácticas que faciliten la aplicación de los conceptos normativos y mejoren la formalización del trabajo en los proyectos estudiantiles. Este diagnóstico constituye la base para diseñar acciones específicas de apoyo metodológico dentro del mismo marco de investigación

CAPÍTULO VII: GUÍA PRÁCTICA PARA LA GESTIÓN DE REQUERIMIENTOS EN PROYECTOS ESTUDIANTILES

Los resultados obtenidos y las conclusiones derivadas del análisis anterior evidenciaron la necesidad de contar con orientaciones prácticas que faciliten la aplicación de los conceptos normativos y mejoren la formalización del trabajo en los proyectos estudiantiles.

En respuesta a esa necesidad, el presente capítulo integra los fundamentos, principios orientadores, criterios de diseño y la guía práctica para la gestión de requerimientos en proyectos estudiantiles de software. La propuesta se apoya en tres pilares: (i) los hallazgos del Capítulo VI, en particular las dificultades P1–P8 (sección 6.1.4; Tabla 3); (ii) los productos de trabajo definidos por la norma ISO/IEC 29110 para el Perfil Básico; y (iii) la evaluación de herramientas tecnológicas según criterios de viabilidad académica (cobertura funcional, usabilidad, trazabilidad, exportación y compatibilidad).

A partir de estos insumos se plantea un conjunto básico de prácticas y artefactos esenciales, concebidos para ser incorporados de manera progresiva y flexible, con el objetivo de mejorar la claridad, la validación y la trazabilidad de los requerimientos sin imponer cargas desproporcionadas en contextos educativos.

7.1 Fundamentos

7.1.1 Justificación y propósito

La elaboración de una guía práctica para la gestión de requerimientos en proyectos estudiantiles se justifica a partir de los hallazgos del Capítulo VI y de la necesidad de contar con un instrumento que facilite la incorporación de prácticas sistemáticas en contextos de formación. Los resultados mostraron que los equipos noveles tienden a concentrarse en la implementación técnica del software, relegando actividades como la documentación, la validación o la trazabilidad de los requerimientos. Esta situación, aunque comprensible en experiencias iniciales, limita la calidad de los productos y restringe las oportunidades de aprendizaje en relación con estándares profesionales.

En este escenario, la norma ISO/IEC 29110, presentada en el Capítulo IV como marco de referencia para organizaciones de pequeña escala y equipos con experiencia limitada, resulta particularmente pertinente. Sus procesos y productos de trabajo ofrecen una base adecuada, aunque su aplicación directa puede resultar compleja en entornos educativos, donde el tiempo, los recursos y la madurez metodológica son reducidos. De

allí surge la necesidad de una adaptación práctica, que traduzca los lineamientos normativos en actividades concretas, herramientas accesibles y plantillas simples, ajustadas a las características de los equipos estudiantiles.

La propuesta que se desarrolla en este capítulo busca ofrecer un marco de aplicación gradual, que acerque los lineamientos de la norma ISO/IEC 29110 (presentada en el Capítulo IV) a la práctica educativa. Su propósito principal es fortalecer la gestión de requerimientos y favorecer el aprendizaje de estándares profesionales a través de una guía didáctica, accesible y ajustada a las condiciones reales de los proyectos estudiantiles.

7.1.2 Principios orientadores

La guía práctica se sustenta en un conjunto de principios que articulan los resultados empíricos obtenidos en el Capítulo VI, los lineamientos conceptuales de la norma ISO/IEC 29110 y aportes de la literatura especializada en ingeniería de software y gestión de requerimientos. Estos principios establecen criterios de orientación que favorecen la coherencia, pertinencia y aplicabilidad de la propuesta en contextos educativos.

Progresividad. La obtención y especificación de requerimientos debe abordarse de forma iterativa e incremental, acompañando el entendimiento del dominio y la evolución del proyecto. Sommerville (2011) subraya que este proceso no ocurre en un único momento, mientras que Wiegers y Beatty (2013) destacan que los requerimientos se descubren, refinan y priorizan de manera continua, lo cual resulta especialmente pertinente en equipos noveles.

Simplicidad y adaptabilidad. Las prácticas propuestas deben poder aplicarse con recursos limitados, evitando burocracia excesiva y privilegiando instrumentos accesibles y de bajo costo. La norma ISO/IEC 29110 fue diseñada precisamente para *VSE*, proponiendo un marco simplificado y adaptable frente a estándares más complejos (Laporte, Hébert & Mineau, 2013).

Trazabilidad. Mantener vínculos claros entre requerimientos, tareas, pruebas y entregables es fundamental para sostener la coherencia del proyecto. La ausencia de trazabilidad, frecuente en los equipos estudiantiles, se reconoce en la literatura como una fuente crítica de defectos y costos adicionales (Pressman & Maxim, 2020; Sommerville, 2011). Por ello, es importante conservar identificadores consistentes para los requerimientos a lo largo de los distintos artefactos del proyecto.

Validación sistemática. Cada fase debe incluir instancias de revisión y validación explícita con clientes o usuarios, dejando evidencia verificable. Wiegers y Beatty (2013)

resaltan que esta práctica reduce ambigüedades y desviaciones tempranas, y en el ámbito educativo potencia además competencias de comunicación y negociación.

Principio pedagógico. Más allá de su utilidad técnica, la guía se concibe como un recurso de aprendizaje. Promueve la rotación de roles, el desarrollo de una visión integral del ciclo de vida del software y la apropiación de prácticas profesionales, reforzando así el valor formativo de la gestión de requerimientos.

En conjunto, estos principios configuran el marco orientador de la propuesta, asegurando que las actividades, herramientas y plantillas respondan tanto a necesidades técnicas como a objetivos educativos.

7.1.3 Criterios de diseño de la guía práctica

El diseño de la guía práctica traduce los principios orientadores en un conjunto de decisiones concretas que buscan garantizar su coherencia normativa, su pertinencia técnica y su usabilidad pedagógica en contextos estudiantiles.

En primer lugar, se priorizó la alineación con los productos de proceso de la norma ISO/IEC 29110, presentados en el Capítulo IV. La norma define un conjunto reducido de artefactos cuya producción resulta clave para asegurar comunicación y trazabilidad en proyectos de pequeña escala (ISO/IEC, 2011; Laporte et al., 2013). A partir de los hallazgos del Capítulo VI, se seleccionaron aquellos directamente vinculados con las dificultades más frecuentes (como la Especificación de Requerimientos, la Solicitud de cambio y el Registro de aceptación), evitando una cobertura exhaustiva que resultaría poco realista en el ámbito educativo.

El segundo criterio fue la integración de actividades mínimas y herramientas accesibles. La literatura enfatiza que en equipos pequeños la adopción de buenas prácticas depende de instrumentos de bajo costo y fácil aprendizaje (Wiegers & Beatty, 2013; Pressman & Maxim, 2020). Por ello, se privilegiaron aplicaciones familiares a los estudiantes (documentos colaborativos, planillas en línea y tableros visuales), complementadas con repositorios de control de versiones (GitHub/GitLab) para favorecer la trazabilidad técnica.

En tercer lugar, se atendió a la usabilidad pedagógica. Como señalan Sommerville (2011) y McConnell (2004), la incorporación de prácticas no depende solo de su utilidad técnica, sino también de la capacidad de los equipos para comprenderlas y mantenerlas en el tiempo. La guía, en consecuencia, se organiza en secciones claras y autocontenidas, acompañadas de ejemplos y plantillas listas para usar, de modo que los estudiantes puedan aplicarlas sin necesidad de recurrir constantemente a otras fuentes.

Por último, se buscó un equilibrio entre formalidad y flexibilidad. Aunque la ISO/IEC 29110 ofrece un marco simplificado, su implementación literal puede resultar exigente para equipos noveles. Por ello, la guía propone una adopción gradual: introduce

prácticas básicas que pueden ampliarse según la madurez metodológica alcanzada por cada grupo (Laporte et al., 2012; ISO/IEC, 2015).

En conjunto, estos criterios permitieron construir una guía normativamente coherente, técnicamente viable y pedagógicamente accesible, que traduce los hallazgos empíricos y la literatura especializada en recursos concretos (actividades, herramientas, roles y plantillas) listos para su aplicación en proyectos estudiantiles.

7.2 Desarrollo de la guía práctica

GUÍA PRÁCTICA PARA GESTIÓN DE REQUERIMIENTOS EN PROYECTOS ESTUDIANTILES

La presente guía reúne un conjunto de recursos destinados a equipos estudiantiles que desarrollan proyectos de software. Ofrece actividades mínimas, plantillas, ejemplos y esquemas sirven como apoyo para organizar el trabajo, validar requerimientos, documentar acuerdos y mantener la trazabilidad durante todo el proyecto.

Está organizada en bloques temáticos que pueden consultarse de manera independiente o en conjunto, de modo que cada equipo disponga de referencias claras y aplicables en las distintas etapas del proyecto.

7.2.1 Modo de uso

La guía está concebida como un recurso de aplicación inmediata. No reemplaza las metodologías que los equipos decidan emplear, sino que las complementa con un conjunto de prácticas básicas que favorecen la claridad, la validación y la trazabilidad de los requerimientos.

Los materiales (actividades, plantillas, matrices y roles) pueden ajustarse al tipo de proyecto, al contexto del cliente y a las características del equipo, siempre que se mantengan los elementos esenciales requeridos para la validación académica: objetivos claros, acuerdos documentados, entregables definidos y trazabilidad del trabajo.

Para utilizarla, se sugiere:

- Comenzar por lo esencial: aplicar primero las actividades mínimas y la secuencia sugerida, incorporando luego herramientas, roles y plantillas a medida que avanza el proyecto.
- Usar cada bloque como referencia puntual: por ejemplo, recurrir a las plantillas cuando sea necesario documentar acuerdos o revisar la matriz de trazabilidad al cierre de una iteración.

- Adaptar los contenidos al contexto: cada grupo puede seleccionar las secciones más útiles según sus necesidades y plazos del proyecto, integrándolas en la metodología de trabajo que ya tenga definida.

En los apartados siguientes, la guía presenta primero una secuencia de actividades mínimas, luego herramientas sugeridas, ejemplos de matrices, roles y responsabilidades, plantillas, y finalmente, una síntesis con pautas de diagnóstico rápido.

7.2.2 Documentos esenciales

Antes de entrar en las actividades, se presentan los documentos mínimos que permiten dar trazabilidad y formalidad al proyecto:

- Especificación de requerimientos: listado claro con identificador, descripción y criterios de aceptación.
- Registro de aceptación: evidencia de validación con el cliente o docente.
- Solicitud de cambio: registro breve para controlar modificaciones.
- Matriz de trazabilidad: tabla que relaciona requerimientos con tareas, pruebas y entregables.
- Cronograma/plan de tareas: esquema visual o tabular de actividades, responsables y plazos.
- Acta de inicio (opcional): acuerdos iniciales, objetivos, restricciones y criterios de éxito.

Para facilitar la trazabilidad, cada requerimiento debe contar con un identificador único y estable. Este código debe mantenerse constante en todos los documentos del proyecto, de modo que pueda reconocerse fácilmente a lo largo de todo el ciclo de desarrollo.

7.2.3 Actividades mínimas, fases y evidencia

Para facilitar la organización del trabajo y el seguimiento de buenas prácticas, se proponen actividades mínimas agrupadas en cinco fases: Inicio, Relevamiento y especificación, Validación y gestión de cambios, Construcción y trazabilidad, y Cierre y entrega.

Estas fases reflejan la dinámica real de los proyectos estudiantiles y ofrecen una guía sencilla para planificar, documentar y verificar el avance del equipo.

Cada fase reúne acciones básicas que orientan desde la definición inicial del proyecto hasta la validación final.

En cada actividad se indica el momento sugerido de aplicación y la evidencia mínima esperada, priorizando el uso de herramientas accesibles y registros colaborativos.

La Tabla 8 resume esta secuencia como una guía operativa y lista de verificación que los equipos pueden usar para controlar su progreso y mantener la coherencia en la documentación.

Tabla 8

Actividades mínimas por fase y documentos asociados

Fase	Actividad mínima	Momento de aplicación	Documentos o evidencias asociadas
Inicio	Definir objetivos, alcance y acuerdos básicos	Primera reunión con cliente/docente	Acta de inicio compartida
	Estimar esfuerzo y planificar tareas iniciales	Al comenzar el proyecto	Cronograma o plan de tareas
Relevamiento y especificación	Registrar requerimientos iniciales con identificadores únicos	Tras la primera entrevista	Especificación de requerimientos (borrador) con IDs y criterios de aceptación
	Refinar y estructurar requerimientos	Al inicio de cada iteración	Especificación actualizada (versión controlada)
Validación y gestión de cambios	Validar requerimientos con el cliente/docente	Al menos una vez por iteración	Registro de aceptación parcial o por ítem
	Revisar con prototipos o demostraciones	Durante iteraciones	Prototipo + registro de acuerdos
	Registrar y aprobar cambios	Ante cualquier modificación	Solicitud de cambio + ER o cronograma actualizados
Construcción y trazabilidad	Vincular tareas a requerimientos	Al planificar cada iteración	Matriz de trazabilidad actualizada
	Actualizar estado de cumplimiento	Al cierre de cada ciclo	Matriz con estado (“implementado”, “pendiente”)
Cierre y entrega	Validar funcionalmente los requerimientos	Antes de la entrega final	Registro de aceptación final
	Documentar lecciones aprendidas (opcional)	Al finalizar el proyecto	Documento o notas de reflexión

Lista de verificación para seguimiento

Al finalizar una iteración o antes de una entrega, el equipo puede utilizar esta lista para confirmar que se han cubierto las actividades esenciales:

- ☐ ¿Se documentaron los requerimientos con identificadores y criterios de aceptación?
- ☐ ¿Existe un único documento accesible con la versión vigente de los requerimientos?
- ☐ ¿El cliente revisó y validó los requerimientos documentados?
- ☐ ¿Todo cambio fué registrado y aprobado antes de implementarse?
- ☐ ¿Hay una matriz que vincule requerimientos con tareas y entregables?
- ☐ ¿Se obtuvo una validación final antes de la entrega?

7.2.4 Herramientas sugeridas y productos de trabajo

La Tabla 9 conecta cada documento clave con herramientas que pueden utilizarse para elaborarlo y mantenerlo actualizado. Se priorizan aplicaciones gratuitas o de uso extendido, conocidas por estudiantes y docentes, que facilitan su adopción en proyectos académicos.

Tabla 9

Herramientas propuestas por producto de trabajo

Producto de trabajo	Herramienta sugerida	Uso mínimo recomendado
Especificación de requerimientos	Google Docs / LibreOffice Writer	Registrar requerimientos con identificador, descripción y criterio de aceptación.
Registro de aceptación	Google Sheets / Formularios en línea	Documentar validaciones del cliente con fecha y observaciones.
Solicitud de cambio	Trello / GitLab Issues	Registrar cambios propuestos, con motivo, impacto estimado y estado.
Matriz de trazabilidad	Google Sheets / Libreoffice Calc	Vincular requerimientos, tareas, pruebas y entregables.
Cronograma de actividades	Trello / Google Sheets / Google Calendar	Planificar tareas, responsables y plazos.
Repositorio de proyecto	GitHub / GitLab	Mantener versiones de código y documentos clave.

Para configuraciones avanzadas, como automatización de flujos o estrategias de ramificación en GitHub/GitLab, se recomienda consultar la documentación oficial de cada plataforma.

7.2.5 Plantillas básicas

Para facilitar la aplicación de las actividades y el uso de las herramientas presentadas en las secciones anteriores, se ofrecen plantillas simples para los documentos básicos de gestión. Estas plantillas permiten mantener un formato consistente y aseguran que la información esencial quede registrada de manera clara y verificable.

El conjunto incluye documentos iniciales, como el acta de inicio, y otros de seguimiento, como la Especificación de requerimientos, el Registro de aceptación o la Matriz de trazabilidad. En todos los casos se priorizan formatos accesibles (documentos o planillas compartidas) que pueden ser replicados y adaptados por los equipos según sus necesidades.

Las tareas mínimas de un proyecto pueden registrarse directamente en el Cronograma de actividades, que funciona como listado de control de responsables y plazos. Para equipos que busquen un mayor nivel de detalle, la guía ofrece además una plantilla de desglose de tareas y estimación de esfuerzo (ver Sección 7.2.8). La Tabla 10 sintetiza las plantillas básicas recomendadas y el contenido mínimo que debe contemplar cada una.

Tabla 10

Plantillas básicas recomendadas

Documento	Formato sugerido	Contenido mínimo
Acta de inicio	Documento breve	Objetivos, alcance inicial, interesados clave, restricciones, criterios de exito, acuerdos básicos
Especificación de requerimientos	Documento estructurado	ID, título, descripción, fuente, prioridad, criterio de aceptación
Registro de aceptación	Tabla o formulario	ID del requerimiento, responsable, fecha, evidencia/medio de validación (firma, comentario, correo, acuerdo), conformidad (aceptado / observaciones)
Registro de reuniones	Tabla o documento breve	Fecha, participantes, temas tratados, acuerdos o decisiones, responsables, próximos pasos.
Solicitud de cambio	Formulario o tabla	Fecha, descripción, motivo, impacto estimado, responsable, estado

continúa en la siguiente página)

Tabla 10 (continuación)

Documento	Formato sugerido	Contenido mínimo
Matriz de trazabilidad	Planilla	Relación entre requerimientos, tareas, pruebas y entregables; estado del requerimiento; observaciones de seguimiento
Cronograma de actividades	Planilla o tablero visual	Listado de tareas con responsables, plazos y dependencias

7.2.6 Plantilla de inicio de proyecto

Esta plantilla se propone como guía para la primera reunión formal entre el equipo, el docente y el cliente. Su propósito es asegurar que todos los actores partan de una visión común sobre los objetivos, entregables, restricciones y acuerdos clave antes de iniciar el trabajo técnico.

Además del registro de los elementos fundamentales (objetivos, interesados y criterios de éxito), la plantilla cumple una función preventiva al asegurar que aspectos clave del inicio del proyecto no queden omitidos, tales como:

- políticas de comunicación (frecuencia, canales, roles),
- herramientas de colaboración acordadas,
- términos o lenguaje común,
- restricciones tecnológicas o de tiempo,
- acuerdos específicos con el docente y con el cliente.

Se recomienda completar esta plantilla al comienzo del proyecto y revisarla cada vez que surjan cambios significativos. Debe guardarse en un repositorio compartido (por ejemplo, una carpeta en la nube) para que todos los integrantes puedan consultarla, actualizarla y utilizarla como referencia durante todo el ciclo del proyecto.

A su vez, la plantilla contribuye a alinear expectativas desde el inicio y constituye evidencia formal de los acuerdos iniciales, útil para el seguimiento del equipo y la validación académica.

La Tabla 11 presenta una cuadrícula para ser utilizada como plantilla de trabajo para completar en proyectos reales.

Tabla 11

Plantilla de inicio de proyecto

Elemento a definir	Responsable/s	Fecha acuerdo	Observaciones
Objetivos del proyecto			
Interesados clave (clientes, docentes, usuarios)			
Comprensión del dominio (¿Necesitamos capacitación?)			
Estudio de factibilidad (tecnológico, tiempo, equipo)			
Adaptación de la metodología al contexto			
Criterios de éxito / aceptación			
Entregables mínimos (producto esperado, funcionalidades)			
Restricciones del proyecto (tiempo, tecnología, etc.)			
Políticas de comunicación (frecuencia, medios, roles)			
Herramientas acordadas (documentación, control de tareas, validación)			
Términos clave / lenguaje común			
Acuerdos específicos con docentes			
Acuerdos específicos con el cliente			
Evidencia del acuerdo (acta, documento compartido)			

Con los acuerdos iniciales definidos, el siguiente paso es organizar al equipo y distribuir responsabilidades.

Esto no solo facilita la coordinación, sino que también fortalece la rendición de cuentas y la trazabilidad del trabajo. En contextos educativos, la rotación explícita de roles tiene además un valor pedagógico, ya que permite a los estudiantes experimentar distintas funciones del ciclo de vida del software.

7.2.7 Registro de reuniones

Durante la ejecución del proyecto, se recomienda registrar las reuniones del equipo de trabajo, con el cliente o con el docente. El objetivo es dejar constancia de acuerdos clave, responsables y próximos pasos, fortaleciendo la trazabilidad y evitando malentendidos. Para mantener la simplicidad operativa, se sugiere utilizar una tabla breve o documento compartido adaptable a cualquier plataforma colaborativa, priorizando un formato liviano por encima de un detalle exhaustivo.

La Tabla 12 presenta la estructura mínima sugerida para este registro.

Tabla 12

Estructura sugerida para registro de reuniones

Fecha	Participantes	Temas tratados	Decisiones y responsables	Próximos pasos

.

7.2.8 Roles y responsabilidades

Se proponen funciones mínimas, orientadas a la rotación pedagógica y adecuadas para equipos pequeños. En proyectos unipersonales, se recomienda asumir estas funciones de manera secuencial por fase (análisis → construcción → validación).

La Tabla 13 presenta una descripción sintética de los roles sugeridos, sus responsabilidades principales y recomendaciones de aplicación.

Tabla 13

Roles y responsabilidades

Rol/Función	Responsabilidades principales	Recomendaciones
Responsable de cliente	Agenda reuniones; centraliza consultas; confirma validaciones y acuerdos.	Mantener registro de decisiones.
Documentación	Mantiene la especificación de requerimientos y su historial; gestiona versiones.	Usar control de cambios y comentarios.

(continúa en la siguiente página)

Tabla 13 (continuación)

Rol/Función	Responsabilidades principales	Recomendaciones
Trazabilidad	Actualiza la matriz; enlaza req ↔ tareas ↔ pruebas ↔ entregables.	Revisar al cierre de cada iteración.
Calidad/pruebas (opcional)	Define casos de prueba básicos; verifica criterios de aceptación.	Registrar resultados y evidencias.

7.2.9 Plantilla de desglose de tareas y estimación de esfuerzo

La plantilla de desglose de tareas y estimación de esfuerzo permite organizar el trabajo en actividades concretas, asignar responsables y anticipar la carga de dedicación requerida. Además, facilita el registro del tiempo real invertido y el análisis de posibles desvíos, ofreciendo insumos valiosos para la reflexión y la mejora en proyectos educativos.

Se recomienda utilizarla desde las primeras etapas de planificación y actualizarla periódicamente durante el desarrollo. Puede funcionar tanto como guía de organización inicial como instrumento de seguimiento en reuniones internas o instancias de validación con el docente o el cliente. Para garantizar su utilidad, es conveniente mantenerla en un formato colaborativo y de fácil acceso (por ejemplo, planillas compartidas en línea), lo que favorece la actualización continua, el control de avances y la transparencia en la distribución de tareas.

Recomendaciones prácticas para su uso

Esta plantilla resulta especialmente útil para equipos que:

- no saben por dónde empezar con la planificación.
- estiman “a ojo” o improvisan sobre la marcha, y
- desean tener una visión clara del avance real.

Al aplicarla se sugiere:

- desglosar el proyecto en tareas concretas: no más de una o dos por día/persona.
- estimar el esfuerzo en horas: no en días.
- registrar quién se encarga de cada tarea
- actualizar el tiempo real invertido conforme avanza el trabajo.
- anotar observaciones (dificultades, cambios, reestimaciones).

La Tabla 14 presenta el modelo base de la plantilla, con datos mínimos sugeridos.

Tabla 14

Plantilla de desglose de tareas y estimación de esfuerzo

ID	Tarea	Responsable	Estimación (hs)	Tiempo real (hs)	Observaciones

Para equipos que requieren un control más detallado o desean detectar desvíos tempranos, la Tabla 15 ofrece como ejemplo una versión extendida que agrega columnas de estado y porcentaje de avance, con datos para que sirva de ilustración..

Tabla 15

Ejemplo ilustrativo de desglose de tareas y estimación de esfuerzo

.

ID	Tarea o Actividad	Responsa-ble	Estimación (hs)	Tiempo real (hs)	Estado	% Avance	Observaciones
1.1	Análisis de requerimientos	Laura	4	5	Finalizada	100%	Se extendió por cambios del cliente
1.2	Validación inicial	Marcos	2	2	Finalizada	100%	Sin cambios
2.1	Maqueta pantalla principal	Laura	3	3.5	En revisión	80%	Requiere rediseño posterior

De manera opcional, puede incluirse una columna adicional para el tipo de tarea (análisis, codificación, validación, documentación) a fin de asegurar una distribución equilibrada del esfuerzo. Se recomienda actualizar esta plantilla al menos una vez por semana o por iteración, y utilizarla como insumo para reuniones internas y para la validación con el docente o el cliente.

7.2.10 Matriz de trazabilidad

La trazabilidad permite verificar que cada requerimiento esté vinculado con tareas de implementación, entregables y evidencia de validación. Este artefacto facilita el control del avance y la alineación con las expectativas de las partes interesadas responsables de la solicitud y validación.

En primer lugar la Tabla 16 presenta la estructura mínima sugerida y luego, la Tabla 17 muestra un ejemplo completo para facilitar su comprensión y aplicación práctica.

Tabla 16

Estructura sugerida para la Matriz de trazabilidad

Req	Descripción	Tareas vinculadas	Pruebas	Estado

Tabla 17

Ejemplo ilustrativo de Matriz de trazabilidad

Req	Descripción breve	Historia Caso	Tarea	Entregable/ Archivo	Prueba Caso	Estado validación	Observaciones
R01	Alta de paciente	HU01	Formulario “Alta”	alta.vue	TC-01	Aprobado	Validado con prototipo
R02	Búsqueda por DNI	HU02	Barra de búsqueda	svc/pacientes /search.ts	TC-02	En revisión	Ajustar mensajes de error
R03	Exportar listado de pacientes	HU03	Botón “Exportar CSV”	svc/export/cs v.ts	TC-03	Aprobado	Confirmar formato de columnas

Uso recomendado:

- Actualizar al cerrar cada iteración.
- Utilizar como insumo en reuniones de validación.
- Añadir columnas según necesidad: prioridad, impacto del cambio, sprint, responsable.

7.2.11 Dificultades frecuentes y respuestas prácticas

En esta sección se sintetizan las dificultades más habituales identificadas durante el análisis de proyectos estudiantiles y en la literatura especializada. Para cada problema se propone una acción concreta que puede ser llevada a cabo por equipos noveles, junto con un recurso práctico que facilita su implementación. La Tabla 18 ofrece una herramienta de diagnóstico rápido que vincula síntomas frecuentes en proyectos de software con respuestas operativas y soportes mínimos, orientando a los equipos hacia prácticas sistemáticas desde etapas tempranas de formación.

Tabla 18

Dificultades comunes, acciones y recursos prácticos recomendados

Síntoma frecuente en proyectos de software	Acción sugerida	Recurso práctico recomendado
Requerimientos ambiguos o poco claros. Las descripciones iniciales no incluyen criterios de aceptación, lo que genera interpretaciones distintas y contradicciones posteriores.	Redactar requerimientos mínimos con criterios de aceptación antes de comenzar a desarrollar.	Plantilla de especificación de requerimientos con campos mínimos (ID, descripción, fuente, criterios de aceptación).
Validación "de palabra". Los acuerdos con el cliente quedan en conversaciones informales, sin evidencia documentada, lo que provoca desacuerdos en las entregas.	Registrar y confirmar acuerdos y aprobaciones inmediatamente después de cada reunión o demostración.	Registro de aceptación sencillo (tabla o formulario compartido) que documente cada validación con fecha y responsable.
Cambios aplicados sin control. Se incorporan ajustes directamente, sin registro ni análisis de impacto, lo que lleva a pérdida de coherencia y retrasos.	Documentar cada cambio y evaluar impacto antes de aplicarlo.	Formulario de solicitud de cambio con motivo, impacto estimado y estado de aprobación antes de aplicarlo.
Avance poco claro. Sin cronograma ni seguimiento, el trabajo se acumula al final y no se puede visualizar el progreso real del proyecto.	Planificar tareas y registrar avances regularmente para anticipar desvíos.	Cronograma básico y registro iterativo de progreso (en planilla o tablero visual).
Ausencia de trazabilidad. Los requerimientos no se vinculan con tareas, pruebas o entregables, dificultando verificar que todo lo solicitado se implementó.	Relacionar cada requerimiento con tareas, evidencia de validación y pruebas.	Matriz de trazabilidad simple en hoja de cálculo que relacione requerimientos ↔ validación ↔ tareas ↔ pruebas.
Estimaciones poco realistas. Se subestima el esfuerzo requerido, lo que produce cronogramas optimistas y sobrecarga hacia el cierre.	Desglosar tareas y estimar esfuerzo antes de iniciar cada iteración.	Plantilla de desglose de tareas y estimación de esfuerzo (horas estimadas vs. reales).

(continúa en la siguiente página)

Tabla 18 (continuación)

Síntoma frecuente en proyectos de software	Acción sugerida	Recurso práctico recomendado
Comunicación informal. Los acuerdos se dispersan en mensajes sueltos, generando malentendidos y expectativas desalineadas.	Centralizar acuerdos y responsabilidades por escrito y compartirlos con el equipo.	Acta de inicio y registro de acuerdos en documento compartido, con responsables claros.

7.3 Estrategia de aplicación sugerida

La guía práctica se concibe como un recurso de apoyo flexible para espacios curriculares orientados a proyectos integradores, de carácter transversal, y puede aplicarse de manera gradual, ajustada al contexto de cada cohorte. Las siguientes pautas están dirigidas a docentes, coordinadores o tutores, quienes desempeñan un rol clave en la implementación y acompañamiento de la propuesta. El objetivo es maximizar su valor pedagógico, reforzando aprendizajes significativos sin imponer rigidez ni sobrecargar responsabilidades.

7.3.1 Presentación de la guía

Se recomienda introducir la guía al inicio del curso o módulo, destacando su carácter de recurso de apoyo y no de requisito rígido. Resulta pertinente explicar cómo puede contribuir a prevenir problemas recurrentes en proyectos de software (ambigüedad en los requerimientos, validación informal, retrabajo) y a mejorar la calidad de los resultados alcanzados en cada cohorte.

7.3.2 Adopción gradual y focalizada

La guía puede implementarse de manera progresiva, comenzando por tres ejes críticos: especificación de requerimientos, validación con el cliente y trazabilidad básica. Una vez afianzados, se sugiere incorporar prácticas adicionales como el control de cambios, la estimación de esfuerzo y el seguimiento iterativo. Cada equipo conserva autonomía para decidir qué componentes aplicar según su madurez y contexto, con la orientación de docentes o tutores.

7.3.3 Fase inicial estructurada

Antes de iniciar el desarrollo técnico, se propone dedicar una breve etapa a alinear expectativas y establecer condiciones mínimas de trabajo. Con la participación de docentes/tutores y clientes, esta instancia debería abordar: comprensión general del

dominio del problema, acuerdos sobre canales y frecuencia de comunicación, definición de criterios de éxito y aceptación, establecimiento de un lenguaje común e identificación de restricciones relevantes (tiempo, recursos, conocimientos).

La evidencia de estos acuerdos puede registrarse en un Acta de Inicio o documento compartido (ver plantilla en la Sección 7.2.6). Esta práctica se alinea con los dominios de desempeño Planificación y Equipo definidos en el PMBOK® Guide – Séptima Edición (PMI, 2021).

7.3.4 Planificación docente y entregables intermedios

Se recomienda integrar la guía al cronograma académico de cada cohorte, vinculando los hitos previstos con evidencias mínimas, tales como:

- una versión preliminar de la Especificación de Requerimientos (ER v1);
- un primer registro de aceptación validado con el cliente o con el docente/tutor;
- una matriz de trazabilidad actualizada al cierre de cada iteración.

7.3.5 Repositorio compartido

El uso de un espacio digital centralizado (carpeta en la nube, tablero colaborativo o repositorio de código) favorece la transparencia y el acceso equitativo a los artefactos del proyecto. Allí deberían concentrarse documentos, planillas, evidencias de validación y versiones de software, disponibles tanto para estudiantes como para docentes/tutores.

7.3.6 Revisión entre pares

Es conveniente promover instancias de evaluación informal entre equipos, utilizando como referencia la Tabla 8 presentada en la Sección 7.2.3. Este documento ofrece un marco claro para verificar que las prácticas esenciales se estén aplicando en cada fase del proyecto. Como complemento, puede emplearse la lista de verificación para seguimiento incluida en la misma sección, a fin de facilitar una revisión rápida al cierre de cada iteración.

Este mecanismo fortalece el aprendizaje colaborativo, refuerza la apropiación de buenas prácticas y complementa la labor de seguimiento de los docentes. Se sugiere realizar revisiones breves por iteración (10 a 15 minutos), entre equipos cruzados, registrando observaciones y acuerdos de mejora en los documentos compartidos del proyecto.

7.3.7 Curaduría de ejemplos reales

Finalmente, se sugiere recopilar, anonimizar y compartir ejemplos de plantillas o artefactos completados en diferentes cohortes. Estos insumos enriquecen la guía, ofrecen

modelos concretos a futuros equipos y permiten a docentes e investigadores identificar patrones de uso y oportunidades de mejora.

7.4 Estrategias de validación de la guía

La pertinencia de la guía práctica para la gestión de requerimientos deberá ser validada en contextos reales de enseñanza-aprendizaje. Su aplicación inicial se proyecta como piloto en asignaturas integradoras, como el Taller de Integración, donde los equipos desarrollan software para clientes reales. La validación se orienta principalmente a docentes e investigadores, quienes podrán evaluar la utilidad de la propuesta como recurso pedagógico y como insumo para la mejora continua de prácticas formativas.

Las estrategias sugeridas incluyen:

- Aplicación piloto en cátedra. Implementar la guía en una comisión y acompañar su uso durante todo el proyecto.
- Encuestas breves al cierre. Relevar la percepción de utilidad, barreras encontradas y sugerencias de mejora por parte de los equipos.
- Revisión docente. Realizar una reunión de cierre para detectar secciones confusas o poco utilizadas y proponer ajustes.
- Rúbricas de referencia. Incorporar criterios de especificación, validación y trazabilidad de requerimientos en las evaluaciones formativas.
- Evidencia objetiva. Analizar si el uso de la guía se correlaciona con indicadores concretos: menor retrabajo, más acuerdos documentados, trazabilidad completa.

Como apoyo a estas estrategias, se destacan dos herramientas ya incluidas en la guía:

- Plantilla de inicio de proyecto. Facilita la formalización de objetivos, restricciones, canales de comunicación, criterios de éxito y acuerdos básicos.
- Plantilla de estimación y seguimiento. Permite desglosar tareas, estimar y registrar esfuerzo real, monitorear estado y porcentaje de avance.

Estas acciones conjuntas permitirán consolidar la utilidad práctica y pedagógica de la guía, a la vez que habilitarán su adaptación progresiva en función de la retroalimentación recibida de los equipos, docentes e investigadores.

CONCLUSIONES

El presente trabajo tuvo como objetivo principal analizar la manera en que los desarrolladores noveles gestionan los requerimientos en proyectos de software realizados en contextos educativos e identificar oportunidades de mejora a partir de un enfoque alineado con la norma ISO/IEC 29110. La investigación permitió relevar prácticas efectivas, reconocer dificultades recurrentes y elaborar una propuesta concreta: la guía práctica para la gestión de requerimientos en proyectos estudiantiles, diseñada para ajustarse a las condiciones de equipos con experiencia incipiente.

Los resultados mostraron que, aunque los estudiantes reconocen la importancia de los requerimientos, enfrentan limitaciones para administrarlos de forma sistemática y trazable. Estas limitaciones se vinculan más con la falta de experiencia, el uso restringido de herramientas específicas y la ausencia de modelos accesibles, que con la falta de compromiso. En este sentido, la adaptación de la norma ISO/IEC 29110 se reveló como un marco apropiado para introducir prácticas ordenadas sin generar sobrecarga en los equipos.

El aporte principal de la investigación reside en haber vinculado estas dificultades con productos normativos y herramientas de uso común, lo que permitió construir una guía práctica sustentada en evidencia empírica y concebida tanto como recurso pedagógico para asignaturas aplicadas, como insumo de apoyo en proyectos finales. Este enfoque ofrece a estudiantes y docentes un instrumento para fortalecer la documentación, la validación y la trazabilidad de los requerimientos, contribuyendo al desarrollo de competencias cercanas al ejercicio profesional.

Al mismo tiempo, los hallazgos abren líneas de continuidad que merecen ser exploradas en futuras investigaciones. Entre ellas, cabe destacar: la replicación del estudio en otros entornos académicos para evaluar la generalización de los resultados; la aplicación de pruebas controladas sobre las herramientas sugeridas; la extensión del análisis a otros procesos del ciclo de vida del software; el análisis comparativo de diferentes configuraciones y combinaciones de herramientas para la gestión de requerimientos y su impacto en la trazabilidad y la calidad del proceso; y la elaboración de materiales de apoyo que faciliten el rol docente en la implementación de buenas prácticas.

En conjunto, este trabajo propone una aproximación situada a la gestión de requerimientos en proyectos estudiantiles que combina lineamientos internacionales con prácticas reales y accesibles. La guía desarrollada se concibe como un recurso inicial, adaptable y en evolución, que podrá enriquecerse con la experiencia acumulada de

sucesivas cohortes, consolidándose como un instrumento pedagógico flexible y de valor duradero para la enseñanza y el aprendizaje de la disciplina.

REFERENCIAS

Al-Rawas, A., & Easterbrook, S. (1996). *Communication problems in requirements engineering: a field study. In Proceedings of the First International Conference on Requirements Engineering* (ICRE'94) (pp. 317–323). IEEE. https://doi.org/10.1109/ISRE.1996.491465

Atlassian. (s. f.). *Trello guide*. Atlassian. https://trello.com/guide

Diagrams.net. (s. f.). *User guide*. diagrams.net. https://www.diagrams.net/doc/

Balci, O. (1998). *Verification, validation, and accreditation*. In J. Banks (Ed.), *Handbook of simulation: Principles, methodology, advances, applications, and practice* (pp. 335–393). Wiley.

García, R., Treude, C., & Valentine, A. (2024). *Application of collaborative learning paradigms within software engineering education: A systematic mapping study. In Proceedings of the 55th ACM Technical Symposium on Computer Science Education* V.1 (SIGCSE 2024) (pp. 1–7). ACM. https://doi.org/10.1145/3626252.3630780

GitHub. (s. f.). *GitHub documentation*. GitHub. https://docs.github.com/

GitLab. (s. f.). *GitLab documentation*. GitLab. https://docs.gitlab.com/

Google. (s. f.). *Google Workspace*. Google. https://workspace.google.com/

ISO/IEC. (2011). ISO/IEC 29110-4-1:2011 *Software engineering — Lifecycle profiles for very small entities (VSE*) *— Management and engineering guide* — Part 4-1: *Basic profile*. International Organization for Standardization.

Hernández Sampieri, R. (2014). *Metodología de la investigación: Las rutas cuantitativa, cualitativa y mixta* (6.ª ed.). McGraw-Hill / Interamericana Editores, S.A. de C.V.

ISO/IEC. (2025). *ISO/IEC 29110-5-1-2:2025 Systems and software engineering — Life cycle profiles for very small entities* (VSE) *— Part 5-1-2: Software engineering guidelines for the generic Basic profile*. International Organization for Standardization.

Kerzner, H. (2017). *Project Management: A Systems Approach to Planning, Scheduling, and Controlling* (*12th ed.*). Wiley.

Laporte, C. Y. (2016). *Implementing ISO/IEC 29110 in Educational Settings. Software Engineering Education and Training* (CSEE&T).

Laporte, C. Y., Hébert, C., & Mineau, C. (2013). *Development of a New ISO/IEC Software Engineering Standard for Very Small Entities. International Journal of Information Technologies and Systems Approach*, 6(1), 1–22.

Laporte, C. Y., et al. (2012). *An Overview of the ISO/IEC 29110 Standard*.

Laporte, C. Y., et al. (2015). *Implementing ISO/IEC 29110 in Academia*.

Oktaba, H. (2011). *Introducción a ISO/IEC 29110: Perfil básico para empresas muy pequeñas de desarrollo de software* [Presentación]. México: Universidad Nacional Autónoma de México.

McConnell, S. (2004). *Code complete* (2.ª ed.). Microsoft Press.

Mejía, J., Bonilla, E., Villanueva, E., Serna, E., Faustino, I., & Milán, A. (2020). *Identificación de herramientas como soporte en la implementación de la norma ISO/IEC 29110 (perfil básico).* Revista Ibérica de Sistemas y Tecnologías de Información, (36), 15–33. https://doi.org/10.17013/risti.36.15-33

Mendicoa, G. E. (2003). Sobre tesis y tesistas: Lecciones de enseñanza-aprendizaje (1.ª ed.). Espacio Editorial.

Muñoz, M., & Montoya-Méndez, P. (2021). *Rutas de apoyo a las EMPs para la selección de herramientas que faciliten la implementación del estándar ISO/IEC 29110.* Revista Ibérica de Sistemas y Tecnologías de Información, (45), 3–23. https://doi.org/10.17013/risti.45.3-23

NYCE. (2005). NMX-I-059-NYCE-2005. *Tecnología de la Información – Software – Modelos de procesos y evaluación para desarrollo y mantenimiento de software (MoProSoft)*. Normalización y Certificación Electrónica (NYCE), México.

Pasini, A., Esponda, S., Boracchia, M., & Pesado, P. (2021). *Q-Scrum: una fusión de Scrum y el estándar ISO/IEC 29110*. Instituto de Investigación en Informática LIDI, Facultad de Informática, UNLP. https://sedici.unlp.edu.ar/handle/10915/32421

Pohl, K. (2010). *Requirements engineering: Fundamentals, principles, and techniques*. Springer. https://doi.org/10.1007/978-3-642-12578-2

Pressman, R. S., & Maxim, B. R. (2020). *Software* Engineering: A Practitioner's Approach (9th ed.). McGraw-Hill.

Project Management Institute. (2021). *A guide to the project management body of knowledge* (PMBOK® Guide) – Seventh edition. Project Management Institute.

Real Academia Española. (s. f.). *Elicitar.* En Diccionario panhispánico de dudas. Recuperado el 17 de septiembre de 2025, de https://www.rae.es/dpd/elicitar

Rocha, A. R., & Travassos, G. H. (2021). *Application of the ISO/IEC 29110 standard in computer science student projects: a systematic review*. Journal of *Software* Engineering Research and Development, 9(1), 1–27.

Sommerville, I. (2015). *Software* Engineering (10th ed.). Pearson.

Wiegers, K. & Beatty, J. (2013). *Software Requirements* (3rd ed.). Microsoft Press.

Thomas, D., & Hunt, A. (2019). *The pragmatic programmer: Your journey to mastery (20th anniversary ed.)*. Addison-Wesley.

Anexo I: Glosario

Acta de reunión

Documento que registra lo discutido y acordado durante una reunión, incluyendo decisiones, tareas asignadas y puntos pendientes.

Acuerdo de trabajo (Work Agreement)

Documento que describe el alcance, objetivos y condiciones del proyecto.

Adaptación

Ajuste de prácticas, herramientas o procesos para adecuarse a las características de un contexto específico, como equipos noveles o entornos educativos.

Actividad

Unidad de trabajo dentro de un proceso. Puede descomponerse en tareas.

Artefacto

Documento, modelo, código u otro elemento producido durante el proceso de desarrollo.

***Backlog* (Lista de pendientes)**

Lista priorizada de requerimientos, funcionalidades o tareas pendientes, mantenida y organizada a lo largo del proyecto.

Caso de uso

Descripción del comportamiento esperado del sistema desde la perspectiva del usuario, expresando cómo interactúan con el sistema para lograr un objetivo.

Control de versiones

Práctica que permite registrar, rastrear y administrar los cambios realizados en los archivos de un proyecto a lo largo del tiempo.

Criterios de aceptación

Condiciones que debe cumplir un requerimiento o historia de usuario para considerarse completado y aceptado por el cliente.

Cliente

Parte interesada que solicita el sistema y valida los requerimientos y entregables.

Configuración del *software*

Conjunto completo de artefactos del sistema que incluye requerimientos, código, documentación y resultados de prueba.

Cronograma

Planificación temporal de actividades y entregables del proyecto.

Desviación

Diferencia entre lo planificado y lo efectivamente realizado durante el desarrollo del proyecto.

Entregable

Producto o resultado concreto generado durante el proyecto, que debe ser entregado al cliente o usuario.

Especificación de Requerimientos

Documento que formaliza el contenido, formato y alcance de los requerimientos funcionales y no funcionales del sistema.

Estándar

Conjunto de normas o directrices reconocidas internacionalmente, como ISO/IEC 29110.

Funcionalidad

Componente de un sistema que describe lo que este debe hacer.

Gestión de Cambios

Práctica para registrar, evaluar y aplicar modificaciones a los requerimientos o planes del proyecto.

Gestión de Proyectos (PM)

Proceso destinado a planificar, ejecutar, monitorear y cerrar un proyecto cumpliendo objetivos de calidad, tiempo y costo.

Gestión de Requerimientos

Disciplina que incluye la obtención, documentación, validación, mantenimiento y seguimiento de los requerimientos del sistema.

Guía de Proceso Interna

Documento que define las prácticas estándar que un equipo adopta para gestionar sus proyectos.

Herramienta

Aplicación o plataforma empleada para facilitar tareas como la documentación, trazabilidad, validación, planificación, etc.

Historia de usuario

Formato breve y centrado en el usuario para expresar un requerimiento o necesidad, común en metodologías ágiles.

Ingeniería de Requerimientos

Disciplina que sistematiza el tratamiento de requerimientos desde su descubrimiento hasta su validación.

Iteración

Unidad de trabajo cíclica en métodos ágiles en la que se construye una parte funcional del producto.

Join Application Development (JAD)

Metodología de talleres estructurados que reúnen a usuarios, clientes y desarrolladores para definir requerimientos y aspectos clave del sistema de manera colaborativa y rápida. Se utiliza para mejorar la calidad de los requerimientos y reducir retrabajos.

Línea Base

Versión aprobada de un producto de trabajo que solo puede cambiarse mediante procedimientos de control formal.

Matriz de Trazabilidad

Herramienta que relaciona los requerimientos con los productos del proyecto, como tareas, pruebas o entregables.

***Product Owner* (Propietario del producto)**

Rol en metodologías ágiles responsable de maximizar el valor del producto, priorizando y gestionando el backlog.

Producto de trabajo

Artefacto asociado con la ejecución de un proceso.

[ISO/IEC TR 29110-1:2016]

Proceso

Conjunto estructurado de actividades diseñadas para alcanzar un objetivo.

Proyecto

Esfuerzo temporal con objetivos definidos, que produce un producto, servicio o resultado único.

Prototipado

Técnica que consiste en construir representaciones preliminares del sistema (diagramas, pantallas, modelos funcionales) para explorar y validar requerimientos.

Prototipo

Representación visual o funcional de una parte del sistema que permite validar ideas con el cliente.

Recurso

Elemento necesario para la ejecución del proyecto, incluyendo personas, tiempo, herramientas y presupuesto.

Regla de negocio

Declaración que define o restringe un aspecto del negocio, con el objetivo de influir en su comportamiento.

Repositorio

Lugar físico o virtual donde se almacenan los artefactos de un proyecto: documentos, código, pruebas, etc.

Repositorio del proyecto

Espacio (físico o digital) donde se almacenan los productos de trabajo del proyecto de manera estructurada.

Requerimiento

Condición o capacidad que el sistema debe cumplir o poseer para satisfacer una necesidad. Se clasifica en funcional, no funcional y de dominio.

Requerimiento funcional

Describe los servicios que el sistema debe proporcionar.

Requerimiento no funcional

Define atributos de calidad como rendimiento, seguridad, usabilidad o confiabilidad.

Requerimiento de dominio

Requerimiento derivado de la naturaleza específica del sector o dominio de aplicación del sistema.

Retroalimentación *(feedback)*

Información recibida del cliente, usuario o equipo, que permite ajustar el producto o proceso para mejorar resultados.

Rol

Función asignada a una persona dentro del equipo, con responsabilidades específicas (por ejemplo, responsable de documentación, validador, etc.).

Solicitud de Cambio

Documento o registro que inicia el proceso de evaluación y aprobación de modificaciones en el proyecto.

Sprint

Iteración corta y fija en metodologías ágiles (por lo general de 1 a 4 semanas), en la cual se desarrollan funcionalidades incrementales del producto.

Stakeholder

Persona o grupo que tiene interés o influencia en el proyecto o en el producto resultante.

Trazabilidad

Capacidad de relacionar cada requerimiento con su origen y con los elementos que lo satisfacen (diseño, código, pruebas, etc.).

Validación

Proceso que asegura que el producto final cumple con las necesidades y expectativas del cliente o usuario. Responde a la pregunta: “¿Se construyó el producto correcto?”.

Verificación

Proceso que evalúa si el producto fue construido conforme a sus especificaciones. Responde a la pregunta: “¿Se construyó correctamente el producto?”.

Anexo II: Cuestionario
Gestión de Requerimientos para Desarrolladores Noveles

1. Nombre del proyecto*
2. ¿A qué rubro o sector pertenece el mandante de tu proyecto?
3. Descripción del mandante y su contexto
4. Cuántas personas integraban el equipo?
   1,2,3,4,6…
5. Estado actual del proyecto
   Finalizado o Aprobado, En implementación, En fase de pruebas, En desarrollo, En diseño, Detenido
6. Si el proyecto está detenido: ¿cuál fue el motivo principal?

Requerimientos más grandes o complejos de lo esperado, Falta de madurez técnica o experiencia del equipo, Cancelación por parte del cliente, Motivos ajenos al proyecto

Metodología y tecnología

7. ¿Qué metodología de desarrollo utilizaron?*
   Scrum, Kanban, XP (Extreme programming), Cascada / Tradicional, Otro
8. ¿Realizaron alguna adaptación a la metodología para aplicarla en su proyecto?
9. ¿Qué tecnologías utilizaron en el proyecto?
10. ¿Utilizaron un sistema de control de versiones?
    Si, No
11. ¿El equipo ya tenía experiencia con las tecnologías utilizadas?*
    Sí, Parcialmente (algunos integrantes), No, fue aprendizaje durante el proyecto

Dimensión del desafío del proyecto

12. ¿Qué tan difícil fue implementar las funcionalidades con los conocimientos y herramientas que tenían?
    Nada desafiante (Sabíamos cómo hacerlo sin mayores dificultades), Poco desafiante (Presentó algunos retos menores), Moderadamente desafiante (Requirió esfuerzos técnicos y toma de decisiones), Bastante desafiante (Hubo varios obstáculos importantes), Muy desafiante (Fue difícil en casi todas las etapas del desarrollo)
13. ¿Qué tan claro fue el problema planteado por el cliente (requerimientos y objetivos)?
    Muy claro (Comprendimos todo desde el inicio sin mayores dudas), Bastante claro (Algunos puntos necesitaron aclaración, pero no fueron graves), Algo confuso

(Nos llevó tiempo entender lo que se esperaba), Bastante confuso (Hubo muchas dudas y malentendidos con el cliente), Muy confuso (No logramos tener una visión clara del problema ni de los requerimientos)

14. ¿Qué tan difícil fue para el equipo comprender el dominio del problema?
    Muy fácil (Ya conocíamos el dominio o era muy intuitivo)
    Bastante claro(Hubo pocos conceptos nuevos y los comprendimos rápidamente), Moderado (Tuvimos que estudiar o investigar un poco para entender bien), Difícil (Requirió bastante esfuerzo comprender el dominio del problema), Muy difícil (El dominio era complejo o no logramos comprenderlo del todo)
15. ¿Pueden describir alguna funcionalidad o aspecto del proyecto que haya resultado especialmente desafiante para el equipo? ¿Por qué lo fue?

Documentación y planificación

16. ¿Qué información registraron al planificar y documentar el proyecto?
    Propósito general del sistema, Objetivos del sistema, Alcance (qué se incluye y qué no), Entregables esperados del producto, Requerimientos, Especificación de las tareas a realizar, Cronograma de trabajo o fechas estimadas, Dependencias entre tareas (orden o condiciones para avanzar), Roles y responsabilidades dentro del equipo (quién se ocupa de qué tarea), Otro:
17. ¿Cómo planificaron el proyecto?
    Definimos tareas y plazos desde el inicio, Planificamos sobre la marcha; ajustando tareas y plazos, No hicimos planificación formal, pero definimos algunas tareas, No definimos tareas ni plazos
18. ¿Qué recursos dejaron asentados para el desarrollo del sistema?
    Hardware (PCs de desarrollo, equipos para pruebas o despliegue local), Software (editores, frameworks, sistemas operativos, etc.), Licencias (de software o servicios utilizados), Tiempo y disponibilidad del equipo de desarrollo. Ej.: cantidad de personas que podían trabajar, tiempo semanal estimado, horarios en los que se coordinaban, Capacitación técnica previa o durante el proyecto (cursos, investigación, tutoriales)
19. ¿El plan fue revisado?
    Verificado por el equipo, Revisado por pares, Validado con el cliente, No fue revisado
20. ¿Qué dificultades encontraron en la planificación?
    Dificultad para estimar tareas o tiempos, Problemas para repartir el trabajo en el equipo, Cambios que alteraron la planificación, Falta de herramientas o conocimiento para hacerlo, Otro

Requerimientos: obtención y registro

21. ¿Qué tipos de representaciones usaron para registrar los requerimientos?
    Descripción en lenguaje natural, Diagramas de contexto o flujo, Casos de uso, Historias de usuario, Diagramas de estado, Wireframe o prototipo, Otro:
22. ¿Qué herramientas o soporte utilizaron para registrar los requerimientos?
    Herramientas de ofimática (Word; Excel; Google Docs; etc.), Herramientas de software especializadas (Ej.: Jira; Trello; Figma; etc.), Diagramadores o pizarras digitales (Draw.io; Miro; etc.), Grabaciones de audio o video, Otro:
23. ¿Utilizaron plantillas para describir y documentar los requerimientos?
    Sí, No
24. ¿Qué tipos de requerimientos identificaron y documentaron?
    Funcionales (acciones que debe realizar el sistema), No funcionales (rendimiento; usabilidad; etc.), De dominio o de negocio (reglas, cálculos, normas específicas del rubro o de la organización), Técnicos (infraestructura, bases de datos, lenguajes, comunicación con otros sistemas), Implícitos o tácitos (necesidades no mencionadas explícitamente, pero detectadas por el equipo),
25. ¿Qué dificultades encontraron en el análisis de requerimientos?
    No se comprendieron bien las necesidades del cliente, Hubo ambigüedad o contradicciones, Se identificaron tarde algunos requerimientos importantes, Dificultad para entender el dominio, Otro

Requerimientos: validación y seguimiento

26. ¿Validaron los requerimientos antes de implementarlos?¿Cómo?
    No se realizó una actividad de validación, Presentamos prototipos al cliente, Revisamos que no hubieran contradicciones entre requerimientos (análisis de consistencia), Desarrollamos una versión mínima funcional (MVP) para validar con el cliente, Tuvimos reuniones con el cliente específicamente para repasar los requerimientos, Otro
27. ¿Qué tan fácil te resulta identificar el bloque de código que implementa un requerimiento o funcionalidad específica?
    Muy difícil – No hay ningún tipo de registro; debo buscar manualmente en el código, Difícil – A veces se puede inferir, pero requiere mucho esfuerzo, Intermedio – Depende del caso o de la funcionalidad; en algunos es claro, en otros no, Fácil – Generalmente tengo forma de ubicar el código, aunque no siempre está documentado, Muy fácil – Contamos con trazabilidad o registros claros que permiten identificarlo directamente.

Seguimiento del avance

28. ¿Cómo hicieron el seguimiento del avance del proyecto?
    No se hizo seguimiento, Seguimiento informal (por reuniones, conversaciones, revisión verbal), Seguimiento comparando lo planificado con lo logrado (cronograma, tareas, entregables)
29. ¿El equipo usó alguna forma para evaluar si el proyecto avanzaba bien?
    No, Sí, con seguimiento del porcentaje de avance, Sí, con control del tiempo dedicado, Sí, con entregables parciales / hitos, Sí, con revisión regular en equipo, Otro
30. ¿Cómo se registraron o gestionaron esos desvíos?
    No se registraron, Se registraron con herramientas de uso general (Word, Excel, Google Docs), Otro
31. ¿Qué desafíos enfrentaron al realizar el seguimiento del proyecto?
    Dificultad para medir el avance real, Falta de herramientas adecuadas, Poca claridad en los objetivos, Cambios frecuentes en los requerimientos, Otro

Gestión de cambios

32. ¿Hubo cambios en los requerimientos durante el proyecto?
    El cliente solicitó nuevas funcionalidades o modificaciones una vez iniciado el desarrollo., El equipo propuso mejoras o detectó necesidades no previstas., Hubo malentendidos o ambigüedades que se identificaron tarde, cuando ya había partes del sistema desarrolladas, Otro
33. ¿Cómo se gestionaron estos cambios?
    Ajustamos los requerimientos documentados (agregamos, modificamos o quitamos requerimientos, ya sea por decisión del equipo o por acuerdos con el cliente), Rehicimos funcionalidades que ya estaban implementadas, lo que implicó retrabajo para el equipo, Se modificó el alcance de las funcionalidades (postergar, descargar o modificar requerimiento), Otro

Comunicación y coordinación

34. ¿Realizaron reuniones con los clientes?
    No, Sólo al principio, Esporádicamente, Frecuentemente, Sólo sobre el final
35. ¿Se registraron los acuerdos tomados en reuniones?
    No se registraron, Si, se grabaron las reuniones, Si, con herramientas de uso general (Word, Excel, Google Docs), Otro

36. ¿Cómo se organizaban para comunicarse dentro del equipo?

Pruebas

37. ¿Cuándo hicieron pruebas al sistema o partes del mismo?

    Solo al final del proyecto, En el cierre de cada módulo, Durante todo el proceso de desarrollo, No hicimos pruebas

38. ¿Qué tipo de pruebas realizaron?

    Pruebas funcionales ejecutadas manualmente, Pruebas con usuarios o el cliente, Pruebas automatizadas, Documentamos los pasos para probar, No hicimos pruebas específicas

39. ¿Registraron los defectos detectados?

    No se registraron, Si, con herramientas de uso general (Word, Excel, Google Docs), Otro

40. ¿Utilizaron alguna documentación para realizar las pruebas?

Cierre y entrega del producto

41. ¿Cómo se dio por cerrado el proyecto?

    Se realizó una revisión final con el cliente (con o sin correcciones posteriores), El proyecto se dio por cerrado al finalizar las tareas planificadas, El proyecto se cerró porque se alcanzó un plazo límite, aunque no todas las tareas se completaron, Se dejó registro de la aceptación final del sistema (por acta, correo, mensaje u otro medio)

Retrospectiva

42. ¿Qué mejorarías si tuvieras que volver a hacerlo?

Anexo III: Resultados del Cuestionario

Instrumento y trazabilidad

El instrumento utilizado corresponde al Anexo II: Cuestionario de esta tesina, donde se encuentran transcritas las preguntas aplicadas a los equipos estudiantiles. Las respuestas procesadas permiten establecer la trazabilidad entre los hallazgos presentados en el Capítulo VI y la evidencia empírica provista por los participantes.

Tratamiento de datos

Se procesaron las respuestas completas provistas por los participantes. Para preguntas cerradas se presentan tablas de frecuencias y porcentajes calculados sobre el total de casos (n=16); para selección múltiple se calcularon ocurrencias por opción; para ítems numéricos se informan estadísticos descriptivos (media, desvío, cuartiles) y distribución; para texto abierto se incluyen las respuestas literales y un conteo orientativo de términos.

Alcance de las preguntas incluidas

Este anexo no reproduce en detalle la totalidad de las preguntas del cuestionario, sino aquellas que resultan directamente relevantes para los objetivos de la investigación y para los hallazgos presentados en el Capítulo VI.

Se incluyen:

- Datos generales: tamaño del equipo, estado del proyecto.
- Gestión de requerimientos: claridad inicial, formas de representación, validación, cambios y tipología de cambios.
- Gestión del proyecto: seguimiento, uso de herramientas, cierre del proyecto, estimación del esfuerzo.

Con este criterio, se busca asegurar que el anexo aporte evidencia suficiente para la trazabilidad, manteniendo al mismo tiempo una extensión adecuada.

Notas de lectura

Los porcentajes pueden no sumar 100% por redondeo. Los textos abiertos se muestran sin edición para preservar fidelidad.

Estructura del anexo

(a) Resumen general cantidad de respuestas por pregunta; (b) Una sección por cada pregunta incluida

Cantidad de respuestas por pregunta

Nro.	Pregunta	Con respuesta	Sin respuesta	Valores únicos
4	¿ Cuántas personas integraban el equipo?	16	0	3
5	¿ Cual es el estado actual del proyecto ?	16	0	3
8	¿Realizaron alguna adaptación a la metodología para aplicarla en su proyecto?	13	3	12
10	¿Utilizaron un sistema de control de versiones?	15	1	2
13	¿Qué tan claro fue el problema planteado por el cliente (requerimientos y objetivos)?	15	1	4
21	¿Qué tipos de representaciones usaron para registrar los requerimientos?	16	0	13
22	¿Qué herramientas o soporte utilizaron para registrar los requerimientos?	16	0	8
26	¿Validaron los requerimientos antes de implementarlos?¿Cómo?	16	0	11
28	¿Cómo hicieron el seguimiento del avance del proyecto?	16	0	2
32	¿Hubo cambios en los requerimientos durante el proyecto?	14	2	8
41	¿Cómo se dio por cerrado el proyecto?	12	4	6
42	¿Qué mejorarías si tuvieras que volver a hacerlo?	14	2	14

Tabla AIII.1 – Cantidad de respuestas por pregunta

Resultados detallados

Pregunta Nº 4: ¿Cuántas personas integraban el equipo?

Cantidad de miembros	Frecuencia	% sobre total
1 integrante	6	37,50
2 integrantes	6	37,50
3 integrantes	4	25,00

Tabla AIII.2 – Tamaño de los equipos

Pregunta Nº 5 ¿Cuál es el estado actual del proyecto?

Estado	Frecuencia	% sobre total
Finalizado / Aprobado	13	81,25
En desarrollo	2	12,5
Detenido	1	6,25

Tabla AIII.3 – Estado actual de los proyectos

Pregunta Nº 8: ¿Realizaron alguna adaptación a la metodología?

De las 16 respuestas, 3 no consignaron información. Entre las 12 respuestas válidas, 3 equipos declararon no haber realizado adaptaciones metodológicas y 9 describieron algún tipo de ajuste (el 56,25%). Las adaptaciones identificadas se vinculan principalmente con la simplificación de prácticas (reducción de ceremonias, historias más pequeñas), la flexibilización de tiempos (reuniones menos frecuentes, ciclos de duración variable), la combinación de enfoques (Scrum con XP o Kanban) y la redefinición de roles y responsabilidades (rotación de funciones, priorización hecha por el desarrollador).

Pregunta Nº 10 ¿Utilizaron un sistema de control de versiones?

Uso de control de versiones	Frecuencia	% sobre total
Sí	13	81,25
No	2	12,5
< sin respuesta >	1	6,25

Tabla AIII.4 – Uso de sistemas de control de versiones

Pregunta Nº 13 ¿Qué tan claro fue el problema planteado por el cliente (requerimientos y objetivos)?

Respuesta	Frecuencia	% sobre total
Bastante claro (Algunos puntos necesitaron aclaración, pero no fueron graves)	11	68,8
Bastante confuso (Hubo muchas dudas y malentendidos con el cliente)	2	12,5
Muy claro (Comprendimos todo desde el inicio sin mayores dudas)	1	6,2

Algo confuso (Nos llevó tiempo entender lo que se esperaba)	1	6,2
< sin respuesta >	1	6,2

Tabla AIII.5 – Claridad inicial del problema

Pregunta Nº 21 ¿De qué manera documentaron o representaron los requerimientos?

Representación	Frecuencia	% sobre total
Historias de usuario	5	31,2
Lenguaje natural	10	62,5
Representaciones gráficas	14	87,5

Tabla AIII.6 – Representaciones de requerimientos

Pregunta Nº 22 ¿Qué herramientas emplearon durante el proyecto?

Herramienta/Tipo	Frecuencia	% sobre total
Ofimática	16	100
Trello/Jira	3	18,8
Diagramadores/Pizarras	6	37,5
Grabaciones multimedia	3	18,8

Tabla AIII.7 – Otras herramientas de apoyo

Pregunta Nº 26 ¿Realizaron alguna validación de los requerimientos antes de implementarlos? ¿Cómo?

Práctica de validación	Frecuencia	% sobre total
MVP / versión mínima	5	31,2
Prototipos / visuales	5	31,2
Reuniones con el cliente	9	56,2
Sin validación explícita	5	31,2

Tabla AIII.8 – Validación de requerimientos

Pregunta Nº 28 ¿Cómo hicieron el seguimiento del avance del proyecto?

Respuesta	Frecuencia	Porcentaje
Seguimiento formal (al menos básico)	5	31,2
Sin seguimiento formal	11	68,8

Tabla AIII.9 – Seguimiento y trazabilidad

Pregunta Nº 32 ¿Hubo cambios en los requerimientos durante el proyecto?

Respuesta	Frecuencia	Porcentaje
Sí	13	81,2
NA	3	18,8

Tabla AIII.10 – Cambios en los requerimientos (Sí/No/NA)

Pregunta Nº 41 ¿Cómo se realizó el cierre del proyecto y la aceptación por parte del cliente?

Modalidad de cierre	Frecuencia	% sobre total
Aceptación explícita del cliente (revisión final, registro)	10	62,5
Cierre por cumplimiento interno (tareas o plazos)	3	18,8

Tabla AIII.11 – Formas de cierre del proyecto

Categoría derivada del análisis cualitativo ¿Encontraron dificultades para estimar las tareas y el esfuerzo requerido?

Esta categoría surge del análisis de respuestas abiertas en las preguntas sobre dificultades encontradas y sugerencias de mejora, donde varios equipos mencionaron problemas para estimar tareas y esfuerzo.

Respuesta	Frecuencia	% sobre total
Menciona dificultades de estimación	7	43,8
No menciona dificultades de estimación	9	56,2

Tabla AIII.12 – Dificultades en la estimación del esfuerzo

Nota: esta categoría no proviene de una pregunta específica del cuestionario, sino del análisis de respuestas abiertas en las secciones de dificultades y sugerencias de mejora.

Anexo IV: Actividades y tareas de los procesos del Perfil Básico (ISO/IEC 29110)

A. Proceso de Gestión de Proyectos (PM)

- Actividad 1: Inicio del proyecto
    - Tarea 1.1: Identificar al cliente y las partes interesadas.
    - Tarea 1.2: Definir objetivos, alcance, entregables y restricciones.
    - Tarea 1.3: Elaborar el acuerdo con el cliente (Agreement).
    - Tarea 1.4: Designar al responsable del proyecto y definir el equipo de trabajo.

- Actividad 2: Planificación del proyecto
    - Tarea 2.1: Elaborar el plan del proyecto, incluyendo cronograma, recursos y dependencias.
    - Tarea 2.2: Identificar riesgos y definir estrategias de mitigación.
    - Tarea 2.3: Establecer el repositorio del proyecto y los mecanismos de respaldo.
    - Tarea 2.4: Definir los criterios de aceptación y control de calidad.

- Actividad 3: Ejecución y control
    - Tarea 3.1: Coordinar las actividades del equipo.
    - Tarea 3.2: Monitorear el progreso frente al plan (registro de estado de avance).
    - Tarea 3.3: Gestionar cambios y correcciones.
    - Tarea 3.4: Mantener comunicación con el cliente y registrar reuniones.
    - Tarea 3.5: Gestionar la configuración de productos intermedios.

- Actividad 4: Evaluación y cierre
    - Tarea 4.1: Revisar el cumplimiento de los objetivos del proyecto.
    - Tarea 4.2: Formalizar la aceptación del producto por parte del cliente.
    - Tarea 4.3: Documentar lecciones aprendidas y resultados.
    - Tarea 4.4: Liberar recursos y archivar la documentación.

B. Proceso de Implementación de Software (SI)

- Actividad 1: Análisis de requerimientos
    - Tarea 1.1: Recopilar y documentar los requerimientos.
    - Tarea 1.2: Validar los requerimientos con el cliente.
    - Tarea 1.3: Revisar consistencia y trazabilidad.

- Actividad 2: Diseño del software
    - Tarea 2.1: Definir la arquitectura general.
    - Tarea 2.2: Elaborar el diseño detallado de componentes.
    - Tarea 2.3: Especificar interfaces, estructuras de datos y restricciones técnicas.
    - Tarea 2.4: Revisar y validar el diseño.

- Actividad 3: Construcción e integración
    - Tarea 3.1: Implementar componentes de software.
    - Tarea 3.2: Integrar y compilar el sistema.
    - Tarea 3.3: Ejecutar pruebas unitarias e integración.
    - Tarea 3.4: Registrar defectos y correcciones.

- Actividad 4: Pruebas y validación
    - Tarea 4.1: Elaborar casos y procedimientos de prueba.
    - Tarea 4.2: Ejecutar pruebas de sistema y aceptación.
    - Tarea 4.3: Documentar resultados en informes de prueba.
    - Tarea 4.4: Validar el producto con el cliente.

- Actividad 5: Entrega y cierre técnico
    - Tarea 5.1: Preparar la configuración final del software.
    - Tarea 5.2: Instalar el producto en el entorno operativo.
    - Tarea 5.3: Elaborar documentación de usuario y de mantenimiento.
    - Tarea 5.4: Capacitar a usuarios y transferir el sistema al cliente.

Aval del director de tesis – FCyT 2025

Estudiante/s: Sergio Ignacio Garbarino

Titulo del trabajo de investigación: GESTIÓN DE REQUERIMIENTOS EN PROYECTOS DE SOFTWARE PARA DESARROLLADORES NOVELES

Programa de estudios: **Licenciatura en Sistemas de Información**

Fecha de entrega: 9/12/2025

Por medio de la presente, me dirijo a ustedes en calidad de Director de Tesis a fin de dar consentimiento respecto al cumplimiento de los requisitos de escritura y calidad académica del trabajo final del cual presido, en conformidad con lo establecido en la "Resolución CD FCyT Nro. 503/23".

Ademas, quiero dejar en claro que tanto los objetivos propuestos en el pre-proyecto aprobado en el año lectivo 2024 fueron respetados y llevados a cabo en todo el trabajo de investigación desarrollado.-

Ing. Gabriel Piccoli

piccoli.gabriel@uader.edu.ar