



Análisis comparativo entre reglas de prioridad en la programación de proyectos y técnicas de priorización en contextos ágiles

A comparative analysis of project scheduling priority rules and prioritization techniques in agile contexts

María Paula Rodríguez Jerez ✉

Universidad Industrial de Santander, Bucaramanga, Colombia

maria2258071@correo.uis.edu.co

ORCID: 0000-0001-9908-8062

Néstor Raul Ortiz Pimiento

Universidad Industrial de Santander, Bucaramanga, Colombia

nortiz@uis.edu.co

ORCID: 0000-0003-0514-0021



<https://doi.org/10.36825/RITI.14.35.007>

Recibido: Junio 14, 2026

Aceptado: Octubre 02, 2026

Resumen: La literatura de desarrollo de software presenta una brecha teórica entre dos enfoques que abordan el mismo problema de decisión de forma aislada: las reglas de prioridad de la programación de proyectos (PM, del inglés Project Management) y las técnicas de priorización en contextos ágiles. Esta revisión sistemática de literatura, basada en el protocolo PRISMA 2020, las caracteriza y compara para identificar similitudes, diferencias y oportunidades de integración, analizando 72 estudios publicados entre 2018 y 2026, seleccionados de 807 registros en Scopus y Web of Science. Ningún estudio las compara directamente, y solo uno de los 72 estudios (1.4%) integra algorítmicamente reglas de prioridad de la programación de proyectos en un contexto ágil. Se identificaron posibles similitudes funcionales —WSJF y SPT, MoSCoW y EDD, WSJF+FCM y MINTWR—, solo la primera con sustento en las definiciones clásicas, y diferencias conceptuales: las técnicas ágiles seleccionan qué construir según el valor percibido por el cliente, mientras las reglas PM secuencian en qué orden ejecutar actividades bajo restricciones de recursos. Se plantea que la separación entre paradigmas responde a razones disciplinares más que técnicas, y se proponen recomendaciones de formación para *Product Owners* y un marco preliminar de traducción de métricas, lo que abre oportunidades de integración aún inexploradas en la literatura.

Palabras clave: Reglas de Prioridad, Priorización de Requisitos, Ágil, Programación de Proyectos, Desarrollo de Software.

Abstract: The software development literature presents a theoretical gap between two approaches that address the same decision problem in isolation: priority rules in Project Management (PM) and prioritization techniques in agile contexts. This systematic literature review, based on the PRISMA 2020 protocol, characterizes and compares them to identify similarities, differences, and integration opportunities, analyzing 72 studies published between 2018 and 2026, selected from 807 records in Scopus and Web of Science. No study compares them directly, and only one of the 72 studies (1.4%) algorithmically integrates project scheduling priority rules in an agile context.

Possible functional similarities were identified —WSJF and SPT, MoSCoW and EDD, WSJF+FCM and MINTWR—, only the first supported by classical definitions, along with conceptual differences: agile techniques select what to build based on perceived customer value, while PM rules sequence in what order to execute activities under resource constraints. It is argued that the separation between paradigms responds to disciplinary rather than technical reasons, and Product Owner training recommendations and a preliminary metric-translation framework are proposed, which opens integration opportunities still unexplored in the literature.

Keywords: *Priority Rules, Requirement Prioritization, Agile, Project Scheduling, Software Development.*

1. Introducción

El desarrollo de software enfrenta un problema de decisión recurrente: dada una capacidad finita, ¿qué construir y en qué orden? Dos comunidades investigadoras lo han abordado de forma aislada. La primera, de la Investigación de Operaciones, desarrolló las reglas de prioridad de la programación de proyectos (PM, del inglés *Project Management*): EDD, SP, LPT, LFT, MSLK, GRPW y sus variantes evolutivas, que optimizan *makespan* y tardanza bajo restricciones explícitas de recursos [1], [2], [3], [4]. La segunda, surgida del Manifiesto Ágil [5], produjo técnicas orientadas al valor del cliente —MoSCoW [6], [7], MCBRank [8], WSJF [9], Kano [10], PROMETHEE II+SMAA [11], ARRPM [12], Reprocolla [13], RAR-P [14], E-AHP [15], DA-EAHP [16], CSOP+RP [17], SAPC [18] y sus variantes—, respondiendo a qué construir primero según las prioridades del negocio [19], [20].

La separación entre paradigmas tiene consecuencias observables: los criterios que pesan en la priorización cambian según la etapa, pues antes de planificar influye el valor de negocio y, en la decisión de liberar, la prioridad del equipo tuvo un efecto muy grande (masa de probabilidad cercana al 70%) [21]; el 55% de los practicantes (38/69 encuestados) incorpora la complejidad técnica como criterio pese a que los marcos ágiles no la prescriben formalmente [22]; y casi la mitad de los *Product Owners* prioriza por intuición, sin un método estructurado que conecte valor percibido con capacidad real de ejecución [23]. Esta revisión sistemática (PRISMA 2020 [24]) analiza 72 estudios (2018–2026) para identificar similitudes, diferencias y oportunidades de integración.

Sus contribuciones son: (1) primera caracterización comparativa exhaustiva de ambas familias; (2) evaluación de la calidad metodológica de los estudios incluidos; (3) análisis de diferencias en cuatro planos analíticos con referencias a las 24 técnicas ágiles y 16 reglas PM identificadas; (4) síntesis explícita de los patrones que emergen del *corpus* y del tipo de evidencia empírica que los respalda; (5) verificación de que solo un estudio [25] (1,4%) integra algorítmicamente ambos paradigmas; y (6) recomendaciones contextualizadas para los distintos tipos de organización reportados en el *corpus*, incluyendo un marco preliminar de traducción de métricas y recomendaciones de formación para *Product Owners*.

2. Estado del arte

2.1. Programación de proyectos: reglas de prioridad

El RCPS (Resource-Constrained Project Scheduling Problem, clasificado como NP-hard) minimiza el *makespan* bajo precedencias y restricciones de recursos [1], [2]. La investigación ha transitado desde métodos exactos hasta metaheurísticas, reglas de prioridad y aprendizaje automático [3], [4], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35]. Las reglas se estructuran en cuatro familias, detalladas con su fórmula completa en la Tabla 1. Las basadas en tiempo —EDD, SPT, LPT, LFT/SLFT, EFT— operan con un solo atributo de la actividad; LFT, SLFT y EFT lo obtienen del cálculo de la red (CPM). EDD no exige estimación de duraciones, haciéndola directamente transferible a contextos ágiles con compromisos contractuales. SLFT figura entre las de mejor desempeño ante alta variabilidad [26], [36]. Las basadas en red —MSLK, GRPW, MINTWR, MINLFT— incorporan estructura de dependencias. El esquema desacoplado MINTWR+MINLFT [3] separa la selección del proyecto (macro) de la actividad (micro): estructura jerárquicamente análoga a *Release Planning/Sprint Planning* ágil, analogía no establecida en el *corpus*. Las multi-criterio adaptativas —MLPR [27], [28], [37], VNSLSH [38], la combinación configurable de aspectos de decisión [39], reglas de portafolio [4], [40], [41], [42], [43], [44]— combinan atributos dinámicamente. Las evolutivas/IA(que combinan Programación Genética [GP] y Aprendizaje por Refuerzo Profundo, del inglés *Reinforcement Learning*, con las reglas de prioridad) —AGS+GP [45], [46], HAS+DQN

(*Deep Q-Network*) [47], ASPSTA+EPPO-GP [25], GPHH [26], [29], [30], [31], [32], [33], [34], diseño automático por regresión [35], CCBM [48], [49], MS-RCPSP [50], [51], modelo ARO [36], despacho con fallo de recurso [40], optimización NPV/riesgo [52], modelo de dos capas [51]— representan la frontera actual. ASPSTA, un motor RL+GP que recibe el valor de negocio como señal de recompensa, obtuvo la mayor recompensa media acumulada frente a diez algoritmos en las cuatro instancias evaluadas [25]. La Tabla 1 resume las 16 reglas identificadas.

Tabla 1. Reglas de prioridad en programación de proyectos.

Regla	Criterio y fórmula	Ventaja	Limitación	Refs.
SPT / SJF	<i>Shortest Processing Time</i> . Prioridad = $\min(p_j)$.	Minimiza el tiempo de finalización total en una máquina (caso no ponderado de WSPT) [2]	Puede relegar las tareas largas	[2], [27]
LPT	<i>Longest Processing Time</i> . Prioridad = $\max(p_j)$.	Garantía de peor caso para el <i>makespan</i> en máquinas paralelas [2]	Sin limitaciones reportadas	[2], [3], [50]
LFT / SLFT	<i>Latest Finish Time</i> : $\min(LF_j)$. SLFT con distribución de probabilidad.	SLFT rinde mejor ante alta variabilidad [26]	SLFT se calcula mediante simulación [26, 36]	[26], [36]
EFT	<i>Earliest Finish Time</i> . Prioridad = $\min(EF_j) = \min(ES_j + p_j)$.	Sin ventajas reportadas	Puede posponer tareas urgentes de larga duración	[27], [29]
MSLK	<i>Minimum Slack</i> . Prioridad = $\min(LS_j - t)$, t = instante actual.	Adecuada con variabilidad baja o media [4]	Requiere recalcular las holguras en cada decisión	[3], [4], [38]
GRPW	Greatest Rank Positional Weight. $GRPW_j = p_j + \sum p_k$ (k sucesor de j).	Considera el efecto de los sucesores de cada actividad	No considera los recursos	[31], [50]
MINTWR+ MINLFT (Desacoplado)	Macro: $MINTWR = \min(TWR_p)$; Micro: $MINLFT = \min(LF_j)$ dentro del proyecto elegido.	Entre las reglas desacopladas con mejor desempeño en APD	Con el objetivo de retraso del portafolio (PDEL), las reglas de proyecto único rinden mejor en los conjuntos existentes	[3]
MLPR	<i>Machine Learning Priority Rule</i> . Selecciona adaptivamente la regla de mejor desempeño histórico.	Compite favorablemente con 13 reglas de prioridad [27]	Sin limitaciones reportadas	[27], [37]
VNSLSH	<i>Variable Neighbourhood Search + Local Search</i> . Combina 6 reglas con perturbación.	Supera a los algoritmos existentes en instancias grandes (J120) [38]	Sin limitaciones reportadas	[38]
AGS + GP	<i>Activity Group Selection + Genetic Programming</i> . Lotes de actividades concurrentes.	Aplicable a problemas multimodo; supera a la selección individual de actividades [45]	Escala mal en problemas grandes [46]	[45], [46]
HAS + DQN	<i>Heuristic Adaptive Selection + Deep Q-Network</i> . Selección por Work Package (macro).	Vincula la selección macro (paquetes de trabajo) con la micro (actividades)	Mayor tiempo de cómputo (unos 10 s) [47]	[47]
ASPSTA+EP PO-GP	RL+GP: asigna historias a <i>sprints</i> , equipos, empleados. Valor de negocio = señal de recompensa.	Mayor recompensa media acumulada que diez algoritmos en las cuatro	No considera el costo de comunicación entre equipos ni la	[25]

		instancias (2,23 % sobre el segundo mejor)	incertidumbre del tiempo de las tareas	
GPHH	<i>Genetic Programming Hyper-Heuristics</i> . Evoluciona automáticamente funciones de prioridad no lineales combinando atributos de actividad, red y recursos.	Supera reglas manuales sin diseño experto; se adapta a la instancia	Alto costo computacional de entrenamiento [29]	[26], [29], [30], [31], [32], [33], [34]
CCBM	<i>Critical Chain Buffer Management</i> . <i>Buffer</i> = $f(\text{riesgo})$; acciones reactivas predefinidas.	Absorbe variabilidad	Sin limitaciones reportadas	[48], [49]
MS-RCPS	<i>Multi-Skill RCPSP</i> . Prioriza según criticidad de la habilidad requerida.	Modela la escasez de habilidades de los recursos	Sin limitaciones reportadas	[50], [51]
Reglas extendidas	NPV/semivarianza [52]; despacho con fallo de recurso [40]; modelo ARO dos etapas [36]; modelo de dos capas multi-habilidad [51]; cadena crítica [48, 49]; multi-proyecto distribuido [40, 41, 42, 43].	Abordan incertidumbre financiera, fallos y portafolios distribuidos	Sin limitaciones reportadas	[36], [40], [41], [42], [43], [44], [51], [52]

Fuente: Elaboración propia.

2.2. Priorización ágil: tres grupos de técnicas

El primer grupo comprende métodos cualitativos: MoSCoW [6], [7] clasifica en cuatro categorías sin orden interno y también se usa para ordenar las características de una hoja de ruta de producto [53]; MCBRank [8] añade clasificación ordinal por CBR (*Case-Based Ranking*), alcanzando kappa=0.60 frente a 0.067 de MoSCoW puro [8]; la Matriz Wieggers/PRIUS [54], [55] pondera con fórmula explícita $\text{Prioridad} = (\text{Beneficio} \times W_b + \text{Penalidad} \times W_p) / (\text{Costo} \times W_c + \text{Riesgo} \times W_r)$; RECTV [56], pensado para startups. El segundo grupo emplea modelos cuantitativos: WSJF [9], [20] —Prioridad = Costo del retraso/Tamaño—, con una variante para Scrum [57]; WSJF+FCM [9] corrige su asunción de independencia con Mapas Cognitivos Difusos, logrando 76% de correlación en el orden con los mejores resultados; Kano [10] distingue requisitos básicos (*must-be*), de desempeño y de deleite según su efecto en la satisfacción; las técnicas basadas en *velocity* formalizan la capacidad del sprint como restricción [19], [25]; AHP adaptado a ágil opera en cambios de requisitos [58], en NFRs [59] y en árboles de expansión mínima [60]; E-AHP [15] extiende AHP a cientos de requisitos reduciendo la carga computacional; un marco genérico para priorizar de forma continua [61]; CSOP+RP [17] modela selección de requisitos como problema de satisfacción de restricciones con presupuestos separados para FR y NFR; NFR-aware [62] prioriza considerando conflictos entre requisitos no funcionales; los métodos difusos de selección con dependencias de valor [63], [64] consideran las dependencias entre requisitos; el modelo costo-valor con perspectiva colaborativa [65] fusiona criterios del cliente y del desarrollador; el modelo de 360 grados+IFS [66] pondera *stakeholders* por retroalimentación jerárquica. El tercer grupo aplica IA: IFS [66] captura satisfacción μ , insatisfacción v y duda $\pi=1-\mu-v$; PROMETHEE II+SMAA [11] produce rankings estocásticos con zonas de ambigüedad explícitas; el operador IOWA lingüístico [67] agrega los juicios de varios expertos expresados con etiquetas lingüísticas difusas; Reprocolla [13] aplica FAHP(AHP difuso, del inglés Fuzzy AHP)+FTOPSIS (TOPSIS difuso, del inglés *Fuzzy Technique for Order Preference by Similarity to Ideal Solution*) bajo BOCR con siete conjuntos de datos (tres aplicaciones reales y cuatro de referencia); ARRPM [12] opera con 16 perfiles de riesgo; RAR-P [14] y SAPC [18] automatizan el triaje mediante NLP y agrupamiento semántico; el enjambre de partículas [68] ordena el backlog completo con un modelo lineal; DA-EAHP [16] combina AHP con análisis de dependencias semánticas ($F1=90\%$ en conjunto de datos real); enfoques de ML [69] clasifican y priorizan automáticamente usando SVM, *Decision Tree* y K-NN; la gamificación [55], [70] aumenta la participación de *stakeholders* en la priorización. Las revisiones [69], [71], [72], [73], [74], [75] consolidan el conocimiento sobre las técnicas de priorización, y un

modelo híbrido preselecciona la técnica según los factores críticos del proyecto [76]. La Tabla 2 resume las técnicas identificadas.

Tabla 2. Técnicas de priorización ágil.

Técnica	Descripción y criterio formal	Ventaja principal	Limitación	Refs.
MoSCoW	Clasifica en <i>Must/Should/Could/Won't</i> sin orden interno.	Comunicación directa; rápido	Sin ranking; subjetiva	[6], [7]
MCBRank	MoSCoW + CBR para orden interno. $\kappa=0,60$ vs $0,067$ de MoSCoW.	Añade orden interno a las categorías de MoSCoW	Validada con requisitos elegidos por los investigadores y en poco tiempo	[8]
Wieggers/PRIUS	Prioridad = $(\text{Beneficio} \times W_b + \text{Penalidad} \times W_p) / (\text{Costo} \times W_c + \text{Riesgo} \times W_r)$, escala 1-9.	Equilibra cliente y equipo	Pesos subjetivos	[54], [55]
RECTV	Puntúa valor, costo (esfuerzo), riesgo, factibilidad técnica, tiempo al valor (TtV) y nivel de confianza en una escala común.	Integra riesgo, factibilidad y tiempo al valor; pensada para startups	Confianza y riesgo dependen de opiniones personales	[56]
WSJF	Prioridad = $(UBV + TC + RR/OE) / \text{Tamaño}$. UBV: User Business Value, TC: Time Criticality, RR: Risk Reduction, OE: Opportunity Enablement; Tamaño = Job Size.	Cuantificable; urgencia y valor integrados	Asume independencia entre variables	[9], [20]
WSJF+FCM	WSJF con matriz de influencia difusa (FCM). 76% de correlación en el orden con los mejores resultados.	Captura interdependencias causales	Requiere calibración del mapa	[9]
Kano	Clasifica en básicos (must-be), de desempeño (one-dimensional) y de deleite (attractive) según su efecto en la satisfacción.	Revela qué esperan los clientes según su efecto en la satisfacción	No ordena los requisitos; se usa como marco complementario	[10], [56]
<i>Velocity/Planning</i>	Historias seleccionadas $\leq$ <i>velocity</i> promedio. Operacionaliza capacidad del sprint.	Usa capacidad real del equipo	Requiere historial suficiente	[19], [25], [77]
IFS	<i>Intuitionistic Fuzzy Sets</i> : μ (satisfacción) + ν (insatisfacción) + π (duda) = 1.	Modela indecisión del <i>stakeholder</i>	Sin limitaciones reportadas	[66]
PROMETHEE II+SMAA	<i>Outranking</i> estocástico. Tabla de aceptabilidad de rango (u4: rank-1 prob.=1,00).	Identifica historias no ordenables	Muchos parámetros y fórmulas; supone un equipo experimentado	[11]
Reprocolla	FAHP(Fuzzy AHP)+FTOPSIS(Fuzzy <i>Technique for Order Preference by Similarity to Ideal Solution</i>) bajo BOCR(Beneficios, Oportunidades, Costos, Riesgos). Validado en 7 conjuntos de datos: 3 aplicaciones reales y 4 de referencia (n=87).	Integra al cliente y al desarrollador con el enfoque costo-valor	Amenazas reconocidas: número de participantes y posible sesgo de selección	[13]
ARRPM	16 perfiles de riesgo de proyecto, definidos sobre cuatro tipos de	Prioriza los riesgos de	Panel Delphi reducido	[12]

	riesgo de requisitos: identidad, volatilidad, complejidad e integridad.	requisitos del proyecto	(generalización limitada)	
NFR-aware	Prioriza FRs según prioridad y conflicto de los NFRs asociados; orden parcial ponderado.	Evita retrabajo por conflictos NFR no detectados	Umbral de conflicto depende del juicio del desarrollador	[62]
Costo-valor colaborativo	Primera propuesta de Reprocolla: el cliente pondera el valor y el desarrollador el costo con FAHP; las alternativas se ordenan con FTOPSIS.	Considera la perspectiva del cliente (valor) y del desarrollador (costo)	Modelo propuesto sin validación empírica (la validación se reporta en Reprocolla)	[65]
E-AHP	AHP escalado a cientos de requisitos agrupando por puntaje. Más rápido que ReDCCahp.	Escala a backlogs masivos; mantiene consistencia	Pierde velocidad y exactitud si las puntuaciones están muy dispersas	[15]
DA-EAHP	AHP + análisis semántico de dependencias (NLP). Precisión=92%, recall=93% en conjunto de datos real.	Detecta dependencias implícitas y explícitas	Escalabilidad más allá de 500 requisitos y dependencia de una referencia anotada por expertos	[16]
CSOP+RP	Satisfacción de restricciones con presupuestos separados FR/NFR. Validado en PCCS.	Gestiona simultáneamente FR y NFR con restricción de presupuesto	Muy sensible a sus parámetros	[17]
AHP ágil (cambios/NFR)	AHP para priorizar solicitudes de cambio [58] y NFRs en Scrum [59] mediante comparación por pares.	Considera múltiples criterios al priorizar cambios y NFRs	Subjetividad inicial en comparaciones	[58], [59]
RAR-P / SAPC / ML	NLP+TF-IDF+ <i>K-means</i> (RAR-P); Sentence-BERT+PSO+XGBoost (SAPC); SVM/DT/KNN (ML classify); PSO+modelo lineal sobre el backlog completo.	Reduce sesgo humano; automatiza y escala la priorización	Requiere datos textuales de calidad; posibles sesgos de los modelos	[14], [18], [68], [69]
Gamificación (PRIUS/juego)	Puntos, insignias, rankings para aumentar participación en la priorización del backlog.	Mejora participación de <i>stakeholders</i>	Prototipo sin elementos avanzados de seguridad; no usa todos los elementos de gamificación	[55], [70]
BeCoMe	Compromiso de grupo con números difusos triangulares; calcula el número ideal de tareas del sprint mediante centroides y medianas de las estimaciones expertas.	Reduce la subjetividad en la planificación del sprint	Evaluated en un solo caso (un equipo Scrum)	[78]
Selección difusa con dependencias de valor	Optimización difusa con presupuesto dado [63] o selección parcial de requisitos [64], que consideran las dependencias de valor entre requisitos.	Reduce el riesgo de pérdida de valor por dependencias	Resultados iniciales; validación limitada	[63], [64]

IOWA lingüístico (grupo)	Agrega con un operador IOWA lingüístico los juicios de varios expertos sobre múltiples criterios expresados con etiquetas lingüísticas difusas.	Captura la imprecisión del juicio de cada experto; pesos por experto y por criterio	Demostrado solo con un conjunto de datos de prueba	[67]
Selección híbrida de técnica	Preselecciona la técnica de priorización según factores críticos del proyecto cuantificados.	Adapta la técnica al proyecto	Validación limitada a dos casos de estudio	[76]

Fuente: Elaboración propia.

3. Metodología

Se adopta el protocolo PRISMA 2020 [24] con preguntas PICO (*Population, Intervention, Comparison, Outcome*), utilizadas para formular las ecuaciones de búsqueda. Las preguntas de investigación son: (1) ¿Qué técnicas ágiles existen? (2) ¿Qué reglas PM existen? (3) ¿Cuáles son sus similitudes y diferencias? (4) ¿Qué estudios identifican convergencias? (5) ¿En qué contextos se aplican empíricamente?

3.1. Búsqueda y selección

Se realizaron dos iteraciones de búsqueda en las bases de datos Scopus y Web of Science (WoS) para el período 2018–2026. La primera iteración, utilizando términos generales de priorización combinados con desarrollo de software, ágil y *scheduling*, produjo 148 registros (Scopus: 85; WoS: 63). La segunda iteración, mediante nombres específicos de técnicas y filtros disciplinares, generó 659 registros adicionales (Scopus: 370; WoS: 289), lo que totalizó 807 estudios identificados.

Tras la identificación, se procedió a eliminar 285 registros duplicados, dejando un total de 522 estudios para la fase de tamizaje por título y resumen, de los cuales 297 fueron excluidos. Del conjunto restante, se intentó la recuperación de textos completos para 225 estudios; de estos, 21 no fueron recuperados y 9 fueron excluidos por idioma.

Posteriormente, se evaluó la elegibilidad de los 195 estudios restantes mediante lectura íntegra a texto completo aplicando los criterios de exclusión (E1–E8), lo que resultó en la exclusión de 123 estudios adicionales. Culminado este proceso, se consolidó el *corpus* final de 72 estudios incluidos para el análisis cualitativo y cuantitativo de la revisión.

3.2. Extracción y síntesis

Para cada uno de los 72 estudios seleccionados se sistematizaron 38 dimensiones analíticas. La síntesis empleó estadística descriptiva y análisis temático, cuyos resultados se presentan en la Figura 1. Se consideró que hay integración cuando un estudio aplica reglas de prioridad de la programación de proyectos dentro del flujo ágil; solo uno de los 72 lo cumple (ASPSTA [25], que asigna historias de usuario a *sprints*, equipos y empleados con reglas de prioridad generadas por programación genética); los demás estudios PM no se aplican en contextos ágiles y los de técnicas ágiles no emplean reglas de prioridad de programación de proyectos. La distribución temporal del *corpus* confirma la transición bimodal en la literatura: el clúster asociado a 2018-2022 consolida el paradigma tradicional mediante términos como *priority rules*, *scheduling* y *genetic programming*, mientras que el clúster asociado a 2024-2026 concentra la tendencia actual hacia la priorización basada en valor con *requirements prioritization* y *software design*. La baja densidad de conexión en los nodos puente como *models* y *decision making* evidencia que la integración operativa entre ambos paradigmas representa una oportunidad exploratoria aún no consolidada en la investigación actual.

En síntesis, la revisión de literatura demuestra un nivel de calidad metodológica global de moderado a alto, respaldado por la aplicación rigurosa de los criterios PRISMA. No obstante, 9 estudios obtienen nivel bajo en relevancia técnica (C8) y otros 9 en transparencia de métricas (C9), lo que revela limitaciones en la validación empírica y el reporte de métricas. Este hallazgo subraya la necesidad de que futuras investigaciones fortalezcan la validación empírica y la operabilidad matemática, consolidando así la integración entre los paradigmas ágiles y la gestión tradicional de proyectos.

4. Resultados

4.1. Análisis comparativo entre paradigmas

Las técnicas ágiles determinan qué construir —seleccionan funcionalidades por valor percibido del cliente, midiendo éxito en satisfacción, ROI y *velocity*, en horizontes de 2–4 semanas [77]— y las reglas PM determinan en qué orden ejecutar —secuencian actividades definidas bajo restricciones duras de recursos, midiendo *makespan*, tardanza y utilización—. La diferencia más profunda radica en los supuestos: las técnicas ágiles asumen incertidumbre alta sobre los requisitos y recursos implícitamente disponibles, mientras las reglas PM modelan recursos como restricciones duras. Sin embargo, variantes estocásticas (SLFT [26]) y robustas (ARO [36], CCBM [48], [49]) reducen esta diferencia a una cuestión de grado.

Un hallazgo estructural no reconocido en el *corpus* es que ambas familias separan una decisión macro de una micro —con la estructura desacoplada MINTWR/MINLFT [3] análoga a *Release/Sprint Planning* ágil—. De los 72 estudios, solo uno (1.4%) integra algorítmicamente reglas de prioridad PM en un contexto ágil [25]. La evidencia cuantitativa más sólida proviene de: ASPSTA [25] —mayor recompensa media acumulada frente a diez algoritmos en las cuatro instancias—; WSJF+FCM [9] —76% de correlación en el orden con los mejores resultados—; BeCoMe [78] —determina el número ideal de tareas para el sprint (9,1 de un rango entre 5 y 15) mediante el juicio agregado de 10 expertos, con un error máximo de estimación de 0.27—; MCBRank [8] — $\kappa=0,60$ frente a 0.067 de MoSCoW—; DA-EAHP [16] —precisión=92%, *recall*=93% en detección de dependencias—; y Reprocolla [13] —validado con siete conjuntos de datos (tres aplicaciones reales y cuatro de referencia) y 87 participantes, con la mayor precisión frente a la referencia de verdad (correlación=0.09 frente a 0.05 de FAHP y 0.04 de FTOPSIS), el menor tiempo promedio de uso (18 min frente a 28 de FTOPSIS y 35 de FAHP) y la mayor satisfacción global reportada (78% frente a 77% de FTOPSIS y 71% de FAHP). Ninguno de los otros 29 estudios PM menciona términos ágiles y ninguno de los 42 estudios de técnicas ágiles cita reglas de prioridad.

Al contrastar estos hallazgos con la evaluación de calidad metodológica (Tabla 3), emergen dos patrones consistentes que la sola descripción de los estudios no revela. Primero, las técnicas con mayor evidencia empírica —ASTPA [25], WSJF+FCM [9], BeCoMe [78] y *Decoupled Scheduling* [3]— comparten validación empírica con casos o instancias reales o de referencia y detalle algorítmico suficiente para su replicación (criterio C1); esto sugiere que la solidez de la evidencia, y no solo la novedad algorítmica, debería guiar la adopción práctica. Segundo, entre las reglas PM, SLFT rinde mejor ante alta variabilidad [26], mientras que EDD y MSLK son adecuadas con variabilidad baja o media [4]; en el lado ágil, IFS [66] y PROMETHEE II+SMAA [11] modelan la incertidumbre de los interesados, aunque sin evidencia comparativa de su desempeño. En conjunto, estos patrones sugieren que, al elegir una técnica, pesa más el grado de incertidumbre del contexto que su etiqueta disciplinar (ágil o PM): con alta variabilidad conviene una técnica que la modele explícitamente y con variabilidad baja o media bastan reglas simples, una relación contexto-técnica que aún requiere validación empírica.

4.2. Diferencias en cuatro planos de análisis

Dentro del paradigma ágil las diferencias son de rigor formal entre grupos de técnicas. MoSCoW clasifica nominalmente sin orden interno; WSJF produce un ranking continuo asumiendo independencia entre variables, limitación que WSJF+FCM [9] corrige. Kano [10] introduce la distinción más importante —requisitos básicos (*must-be*), de desempeño y de deleite según su efecto en la satisfacción— que ninguna regla PM modela. PROMETHEE II+SMAA [11] añade tratamiento estocástico de la ambigüedad. E-AHP [15], DA-EAHP [16] y CSOP+RP [17] operacionalizan dimensiones técnicas (escalabilidad, dependencias, presupuesto) ausentes en las técnicas del primer grupo. ARRPM [12] e IFS [66] formalizan el riesgo y la incertidumbre de los *stakeholders*.

Los estudios de práctica [7], [22], [23] documentan que se usan sobre todo técnicas simples —escalas numéricas y MoSCoW— y que las más elaboradas, como Kano, se usan poco [22].

Dentro del paradigma PM las diferencias son de complejidad algorítmica y alcance temporal. EDD minimiza el retraso máximo sin estimación de duraciones; SPT minimiza el tiempo de flujo generando inanición de las tareas largas; GRPW considera el impacto global mediante la suma ponderada de sucesores. La jerarquía MINTWR/MINLFT del *Decoupled Scheduling* [3] es la distinción más relevante: MINTWR (macro) y MINLFT (micro) operan en dos capas análogas a *Release/Sprint Planning*. Las reglas evolutivas —MLPR [27], [28], [37], AGS+GP [45], [46], HAS+DQN [47], ASPSTA [25], GP hiper-heurístico [26], [29], [30], [31], [32], [33], [34]— superan a las heurísticas manuales con que se compararon. Las reglas de portafolio —despacho con fallo de recurso [40], modelo ARO [36], transferencias de recursos [44], multi-proyecto distribuido [40], [41], [42], [43], NPV/riesgo [52]— amplían el alcance más allá del *makespan*.

Entre paradigmas las diferencias reflejan supuestos distintos sobre qué optimizar. WSJF y SPT priorizan trabajo corto de alto impacto, pero WSJF incorpora el costo del retraso mientras SPT solo minimiza el tiempo de flujo sin valor: ante una historia crítica de larga duración, WSJF la ejecutaría primero; SPT haría lo contrario [2], [9]. La similitud con sustento en las definiciones clásicas es la de WSJF y WSPT (la versión ponderada de SPT), que ordenan por la razón entre un peso y un tiempo [2], [20]; las equivalencias entre MoSCoW y EDD y entre WSJF+FCM y MINTWR no tienen fuente en el *corpus*. El *Sprint Buffer* ágil y el Project Buffer CCBM [48], [49] absorben variabilidad con orígenes distintos. Por analogía, los requisitos básicos (*must-be*) de Kano y EDD priorizan ambos ítems cuya ausencia tiene consecuencias críticas, con diferencia de dominio (satisfacción vs urgencia del plazo) pero lógica subyacente idéntica.

Entre los planos macro y micro la separación es transversal a ambos paradigmas: *Release Planning* (épicas, trimestre) es análogo a MINTWR (portafolio, macro) y *Sprint Planning* (historias, 2–4 semanas) a MINLFT (actividad, micro) [3]. La diferencia es que en PM esta jerarquía está formalizada matemáticamente y en ágil opera por consenso de equipo.

4.3. Contextos de aplicación empírica

En cinco empresas de software tailandesas, de 4 a 400 empleados, los equipos usaron un método propio o el enfoque costo-valor [79]; en equipos ágiles distribuidos, la priorización estructurada con MoSCoW y WSJF favorece la transparencia y la consistencia [80]. Los experimentos más prometedores se dan en contextos de distinto tamaño: BeCoMe [78] (un equipo Scrum de 10 integrantes en una empresa de software de 126 empleados), Reprocolla [13] (tres aplicaciones reales y cuatro proyectos de referencia), WSJF+FCM [9] (uno de los cinco mayores bancos de Ecuador, en transformación digital). En el lado PM, las aplicaciones sobre proyectos reales son escasas y provienen de la construcción: 17 reglas de prioridad aplicadas a un portafolio de proyectos de construcción en aguas profundas [42] y la selección adaptativa de reglas en un proyecto modular de cinco edificios [47]. Estudios cualitativos sobre el *Product Owner* [23] documentan que el 65% involucra al equipo en el refinamiento del backlog, el 79% refina antes de involucrar al equipo completo, y que casi la mitad prioriza por preferencia individual o intuición. Se señala además que el *Product Owner* suele convertirse en un cuello de botella en la gestión del backlog [81].

5. Discusión

La separación entre paradigmas es generalizada en la literatura: ninguno de los 29 artículos PM puros cita literatura ágil, y los artículos ágiles rara vez referencian la literatura RCPSP. Cuatro factores la explican: diferencia en la pregunta fundamental (qué vs. en qué orden); diferencia en los supuestos de incertidumbre; inconmensurabilidad aparente de las métricas (valor vs. *makespan*); y sesgo de citación disciplinar entre Investigación de Operaciones e Ingeniería de Software.

La baja adopción de técnicas formales documentada en el *corpus* [22], [23] admite varias explicaciones; un análisis más crítico de los estudios que la reportan permite distinguir al menos tres causas posibles, en lugar de limitarse a constatar su existencia. La primera es epistemológica: los equipos ágiles conciben el valor como una propiedad emergente y negociada con el cliente, mientras que las reglas PM presuponen que el valor ya fue determinado y solo resta secuenciar su ejecución; adoptar una regla PM se percibe, por tanto, como una regresión hacia la planificación predictiva que el Manifiesto Ágil buscó superar [5]. La segunda es de incentivos

organizacionales: los equipos ágiles son evaluados por *velocity* y satisfacción del cliente, métricas que no se alinean directamente con el *makespan* o la tardanza que optimizan las reglas PM, por lo que no existe una razón instrumental inmediata para adoptarlas. La tercera es formativa: los currículos de formación ágil (*Certified ScrumMaster*, PSPO, SAFe) rara vez incluyen contenidos de Investigación de Operaciones, de modo que los *Product Owners* carecen del vocabulario técnico para evaluar si una regla PM podría mejorar su proceso de priorización; en efecto, solo el 26,5% de los *Product Owners* neerlandeses usa un método estructurado de priorización [23]. Estas tres causas —epistemológica, de incentivos y formativa— son complementarias antes que excluyentes, y explican por qué la evidencia técnica de que las herramientas funcionan no se traduce automáticamente en adopción práctica.

Sin embargo, la incompatibilidad percibida es más un constructo disciplinar que una barrera técnica: la integración algorítmica de ASPSTA [25] y la validación de cada familia por separado —aunque mayormente en entornos controlados— indican que las herramientas funcionan. Los estudios sobre el rol del *Product Owner* y de práctica ágil sugieren que influyen factores organizacionales y de capacitación más que limitaciones computacionales.

En la planificación ágil, el alcance del *release* se define a partir del valor, que influye en la priorización antes de planificar [21], y cada sprint se llena con las historias de mayor valor hasta agotar la capacidad del equipo [77]. La asignación de historias a *sprints* también se ha abordado como un problema de optimización, como en ASPSTA [25]. Con base en los dos niveles de decisión análogos (macro y micro), cabe plantear que, en la segunda etapa (el sprint), las reglas PM secuencien las historias bajo restricciones de capacidad, dependencias y plazos, y contrastar su desempeño con el de técnicas de valor y con el de la optimización de la capacidad. Para ello, los puntos de historia, que los equipos ya estiman y miden [77], pueden servir como medida de tamaño, previa calibración histórica del equipo. Ningún estudio del *corpus* ha probado esta comparación experimentalmente. Para su aplicación en la práctica se sugiere: en startups, MoSCoW o RECTV; en organizaciones grandes en transformación digital, WSJF+FCM para selección; con compromisos contractuales, EDD/LFT para secuenciar el sprint; con varios equipos que comparten recursos, el esquema desacoplado; en DevOps, SPT para triaje; con muchos requisitos, E-AHP o DA-EAHP.

5.1. Hacia un marco de traducción de métricas

Como respuesta directa a la ausencia de mecanismos de traducción entre métricas ágiles y de PM (brecha identificada en el análisis anterior y reforzada por la heterogeneidad de métricas señalada entre las limitaciones), se esboza un marco preliminar de tres pasos. En la práctica ágil ya existe una traducción parcial: la *velocity* convierte *story points* en iteraciones al planificar un *release* [77]; lo que no se encontró en el *corpus* es un marco que alimente reglas de prioridad PM con estas métricas. Paso 1 (calibración histórica): para cada equipo, se estima una función de conversión *story point* → duración estimada ajustando una regresión lineal simple entre los *story points* asignados a historias completadas en los últimos *n sprints* y su duración real registrada (en días o en iteraciones, pues [77] advierte que la conversión directa de *story points* a horas es inexacta), obteniendo un coeficiente de conversión específico del equipo. Paso 2 (ajuste por incertidumbre): suponiendo que el desvío estándar de la duración real crece con el tamaño de la historia, se recomienda aplicar un factor de holgura proporcional al percentil 75 del error histórico, análogo al tratamiento que SLFT da a la variabilidad estocástica. Paso 3 (mapeo inverso de *velocity* a *throughput*): la *velocity* promedio del equipo (*story points* por *sprint*) se convierte en un *throughput* equivalente (tareas por unidad de tiempo) dividiendo por el tamaño promedio de historia, permitiendo alimentar directamente reglas basadas en tiempo (EDD, SPT) o basadas en red (MSLK, GRPW) sin abandonar la unidad de medida familiar al equipo ágil. Este marco requiere validación empírica, ausente en los 72 estudios revisados.

5.2. Recomendaciones de formación para Product Owners

En cuanto a los factores humanos, los hallazgos de esta sección sugieren tres recomendaciones específicas de formación para *Product Owners*, coherentes con una separación entre paradigmas de origen más cultural y formativo que técnico. Primera, incorporar en los programas de certificación ágil (CSPO, PSPO, SAFe PO/PM) un módulo introductorio sobre restricciones de recursos y su efecto en la secuenciación, de modo que el *Product Owner* comprenda el vocabulario mínimo de la Investigación de Operaciones sin requerir formación matemática

avanzada. Segunda, promover un ejercicio piloto de la secuencia en dos etapas descrita en esta sección: definir el alcance del *release* con una técnica ágil adecuada al contexto y secuenciar las historias dentro del sprint con una regla PM simple, contrastando el resultado con la práctica habitual del equipo. Tercera, formalizar la comunicación entre el *Product Owner* y el *Scrum Master/Project Manager* mediante un artefacto compartido (por ejemplo, un tablero que muestre simultáneamente el ranking por valor (ágil) y el ranking por urgencia de recursos (PM)) para hacer visible la complementariedad entre ambos paradigmas en lugar de forzar al *Product Owner* a elegir uno de los dos. Dado que ningún estudio del *corpus* ha puesto a prueba experimentalmente la integración, estas recomendaciones se fundamentan en la evidencia sobre la práctica del *Product Owner* y en la complementariedad entre ambos conjuntos de técnicas, y su validación empírica queda como línea de trabajo futuro.

5.3. Limitaciones

Cinco limitaciones: (1) cobertura 2018–2026 con fundamentos clásicos incorporados por referencias seminales; (2) sesgo de publicación; (3) heterogeneidad de métricas que limita la síntesis cuantitativa: el *corpus* mezcla indicadores de eficiencia (*makespan*, tardanza, utilización de recursos) con indicadores de valor percibido (satisfacción del cliente, ROI, correlación con ranking óptimo), sin una unidad de medida común que permita un meta-análisis estadístico formal; esta heterogeneidad es, precisamente, la motivación central del marco de traducción de métricas propuesto y dificulta una comparación cuantitativa directa entre ambos paradigmas, que ningún estudio del *corpus* reporta; (4) predominio del contexto académico: los estudios PM se validan casi siempre con instancias de referencia, generadas o ilustrativas, y solo dos [42], [47] aplican a proyectos reales; y (5) sesgo idiomático (inglés y español).

6. Conclusiones

Las técnicas ágiles y reglas PM responden preguntas complementarias (qué construir vs. en qué orden ejecutarlo) que la literatura trata de forma aislada. Solo un estudio (1,4%) las integra algorítmicamente, sin que ninguno las compare directamente como alternativas. Los casos más robustos son ASPSTA, WSJF+FCM, BeCoMe y el *Decoupled Scheduling*.

Estas diferencias operan en cuatro planos: dentro del paradigma ágil (grupos con formalización creciente; Kano introduce una clasificación por efecto en la satisfacción no modelable con métricas PM); dentro del PM (reglas simples vs. evolutivas; jerarquía MINTWR/MINLFT); entre paradigmas (criterio de optimización y supuesto sobre recursos); y entre planos macro/micro (estructura análoga en ambos paradigmas, formalizada matemáticamente en PM, operada por consenso en ágil). Se identifican además posibles similitudes funcionales —WSJF con SPT (en su versión ponderada, WSPT), MoSCoW con EDD y WSJF+FCM con MINTWR—, de las cuales solo la primera tiene sustento en las definiciones clásicas; las otras dos son analogías sin fuente en el *corpus*.

Adicionalmente, se plantea que la separación es primordialmente disciplinar y social, más que técnica. Las herramientas existen y cada familia se ha validado por separado, mayormente en entornos controlados, con una evaluación de calidad metodológica que respalda esta afirmación y una identificación explícita de los patrones que sostienen esta síntesis. En este sentido, se sugiere a la academia: diseñar estudios comparativos directos implementando técnicas de ambos paradigmas en el mismo contexto empírico (brecha más evidente del *corpus*: cero representantes en 72 artículos); desarrollar marcos de traducción (*story points* a duraciones, *velocity* a *throughput*), para lo cual esta revisión ofrece una propuesta preliminar; e investigar las condiciones de contorno que determinan cuándo la integración agrega valor. Y en el ámbito práctico, se recomienda: adoptar la técnica según el contexto y progresar desde técnicas ágiles en entornos volátiles hasta esquemas desacoplados en entornos con compromisos formales, acompañando esta transición con la formación específica de *Product Owners* propuesta en la discusión.

7. Referencias

- [1] Graham, R. L., Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5, 287-326. [https://doi.org/10.1016/S0167-5060\(08\)70356-X](https://doi.org/10.1016/S0167-5060(08)70356-X)

- [2] Pinedo, M. L. (2016). *Scheduling: Theory, algorithms, and systems* (5ta Ed.). Springer.
<https://doi.org/10.1007/978-3-319-26580-3>
- [3] Van Eynde, R., Vanhoucke, M. (2020). Resource-constrained multi-project scheduling: benchmark datasets and decoupled scheduling. *Journal of Scheduling*, 23 (3), 301-325. <https://doi.org/10.1007/s10951-020-00651-w>
- [4] Chen, H., Ding, G., Zhang, J., Qin, S. (2019). Research on priority rules for the stochastic resource constrained multi-project scheduling problem with new project arrival. *Computers & Industrial Engineering*, 137. <https://doi.org/10.1016/j.cie.2019.106060>
- [5] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D. (2001). *Manifesto for agile software development*. Agile Alliance.
<https://agilemanifesto.org>
- [6] Clegg, D., Barker, R. (1994). *CASE method fast-track: A RAD approach*. Addison-Wesley.
<https://folio.caval.edu.au/Record/c000181229>
- [7] Vijayakumar, S., Prasad, K., Holla, R. (2024). Assessing the Effectiveness of MoSCoW Prioritization in Software Development: A Holistic Analysis across Methodologies. *EAI Endorsed Transactions on Internet of Things*, 10, 1-9. <https://doi.org/10.4108/eetiot.6515>
- [8] Ahmad, S., Rizawanti, R., Woodings, T., Jalil, I. E. A. (2022). MCBRank Method to Improve Software Requirements Prioritization. *International Journal of Advanced Computer Science and Applications*, 13 (7), 215-222. <https://doi.org/10.14569/ijacsa.2022.0130728>
- [9] Zambrano Manzur, B. N., Espinoza Bazán, F. A., Fernandez, Y., Cruz Corona, C. (2025). Solving the Interdependence of Weighted Shortest Job First Variables by Applying Fuzzy Cognitive Mapping. *Information*, 16 (11), 1-20. <https://doi.org/10.3390/info16110944>
- [10] Kano, N., Seraku, N., Takahashi, F., Tsuji, S. (1984). Attractive quality and must-be quality. *Journal of the Japanese Society for Quality Control*, 14 (2), 147-156. https://doi.org/10.20684/quality.14.2_147
- [11] Pereira, J., Vera Rivera, F. H., Astudillo, H., Contreras, P., Almeida, J. (2020). *A multi-criteria stochastic approach to prioritize user-stories in Agile Software Development*. Proceedings - International Conference of the Chilean Computer Science Society (SCCC), Coquimbo, Chile.
<https://doi.org/10.1109/SCCC51225.2020.9281257>
- [12] Tuunanen, T., Vartiainen, T., Kainulainen, S., Ebrahim, M. (2023). Development of an Agile Requirements Risk Prioritization Method: A Design Science Research Study. *Communications of the Association for Information Systems*, 52, 609-637. <https://doi.org/10.17705/1CAIS.05226>
- [13] Amelia, T., Mohamed, R. (2024). Reprocolla: Requirements Prioritisation Model with Collaboration Perspectives Based on Cost-Value Approach. *Journal of Information and Communication Technology (Malaysia)*, 23 (2), 211-252. <https://doi.org/10.32890/jict2024.23.2.3>
- [14] Izhar, R., Cosh, K., Bhatti, S. N. (2024). *Enhancing Agile Software Development: A Novel Approach to Automated Requirements Prioritization*. Proceedings - 21st International Joint Conference on Computer Science and Software Engineering (JCSSE), Phuket, Thailand.
<https://doi.org/10.1109/JCSSE61278.2024.10613648>
- [15] Mohamed, N., Mazen, S., Helmy, W. (2022). E-AHP: An Enhanced Analytical Hierarchy Process Algorithm for Priortizing Large Software Requirements Numbers. *International Journal of Advanced Computer Science and Applications*, 13 (7), 190-199. <https://doi.org/10.14569/ijacsa.2022.0130725>
- [16] Mohamed, N., Helmy, W., Mazen, S. (2025). A Novel AI-based Dependency-aware Algorithm for Prioritizing Software Requirements in Large-scale Projects. *Mesopotamian Journal of Big Data*, 5 (1), 394-414. <https://doi.org/10.58496/MJBD/2025/024>
- [17] Afraz, S., Rashidi, H., Mikaeilvand, N. (2021). CSOP plus RP: a novel constraints satisfaction model for requirements prioritization in large-scale software systems. *Turkish Journal of Electrical Engineering and Computer Sciences*, 29 (7), 3074-3091. <https://doi.org/10.3906/elk-2102-139>
- [18] Radwan, A. M., Abdel-Fattah, M. A., Mohamed, W. (2025). Smart Agile Prioritization and Clustering: An AI-Driven Approach for Requirements Prioritization. *IEEE Access*, 13, 127335-127350.
<https://doi.org/10.1109/ACCESS.2025.3589959>
- [19] Schwaber, K., Sutherland, J. (2020). *The Definitive Guide to Scrum: The Rules of the Game*.
<https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf>

- [20]Leffingwell, D. (2018). *SAFe 4.5 reference guide: Scaled agile framework for lean enterprises* (2da Ed.). Addison-Wesley Professional. <https://dl.acm.org/doi/10.5555/3265203>
- [21]Svensson, R. B., Torkar, R. (2024). Not all requirements prioritization criteria are equal at all times: A quantitative analysis. *Journal of Systems and Software*, 209, 1-16. <https://doi.org/10.1016/j.jss.2023.111909>
- [22]Jarzebowicz, A., Sitko, N. (2020). Agile requirements prioritization in practice: Results of an industrial survey. *Procedia Computer Science*, 176, 3446-3455. <https://doi.org/10.1016/j.procs.2020.09.052>
- [23]van Daalen, E., van Solingen, R. (2024). The Current State of Operationalizing Value by Dutch Product Owners in Agile Software Development. *Lecture Notes in Business Information Processing*, 512, 89-106. https://doi.org/10.1007/978-3-031-61154-4_6
- [24]Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D.,..., Moher, D. (2021). The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *BMJ*, 372, 1-9. <https://doi.org/10.1136/bmj.n71>
- [25]Shen, X., Shi, J., Zhang, G., Wang, Z. (2026). Agile software project scheduling using reinforcement learning with genetic programming. *Engineering Applications of Artificial Intelligence*, 176. <https://doi.org/10.1016/j.engappai.2026.114694>
- [26]Zhang, H., Li, L., Bai, S., Zhang, J. (2024). Co-evolution genetic programming-based hyper-heuristics for the stochastic project scheduling problem with resource transfer and idle costs. *Swarm and Evolutionary Computation*, 90. <https://doi.org/10.1016/j.swevo.2024.101678>
- [27]Adamu, P. I., & Aromolaran, O. T. (2018). Machine Learning Priority Rule (MLPR) for solving resource-constrained project scheduling problems. *Lecture Notes in Engineering and Computer Science*, 2. <https://www.iaeng.org/publication/IMECS2018/>
- [28]Adamu, P. I., Agarana, M. C., Okagbue, H. I. (2018). *Machine learning heuristic for solving multi-mode resource-constrained project scheduling problems*. International MultiConference of Engineers and Computer Scientists, Hong Kong. https://www.iaeng.org/publication/IMECS2018/IMECS2018_pp23-28.pdf
- [29]Luo, J., Vanhoucke, M., Coelho, J. (2023). Automated design of priority rules for resource-constrained project scheduling problem using surrogate-assisted genetic programming. *Swarm and Evolutionary Computation*, 81. <https://doi.org/10.1016/j.swevo.2023.101339>
- [30]Chand, S., Huynh, Q., Singh, H., Ray, T., Wagner, M. (2018). On the use of genetic programming to evolve priority rules for resource constrained project scheduling problems. *Information Sciences*, 432, 146-163. <https://doi.org/10.1016/j.ins.2017.12.013>
- [31]Dumic, M., Sisejkovic, D., Coric, R., Jakobovic, D. (2018). Evolving priority rules for resource constrained project scheduling problem with genetic programming. *Future Generation Computer Systems*, 86, 211-221. <https://doi.org/10.1016/j.future.2018.04.029>
- [32]Sabljo, J., Coric, R., Dumic, M., Durasevic, M. (2025). *Evolving Priority Rules to Optimize Multi-Objective Criteria for the Resource-Constrained Project Scheduling Problem*. MIPRO 48th ICT and Electronics Convention, Opatija, Croatia. <https://doi.org/10.1109/MIPRO65660.2025.11132077>
- [33]Chand, S., Singh, H., Ray, T. (2019). Evolving rollout-justification based heuristics for resource constrained project scheduling problems. *Swarm and Evolutionary Computation*, 50. <https://doi.org/10.1016/j.swevo.2019.07.002>
- [34]Chao, Y., Zhuang, C., Guo, H., Liu, J. (2026). A genetic programming hyper-heuristic with whale optimization algorithm for the dynamic resource-constrained multi-project scheduling problems. *Expert Systems with Applications*, 295. <https://doi.org/10.1016/j.eswa.2025.128881>
- [35]Luo, J. Y., Vanhoucke, M., Coelho, J., Song, J. (2026). Automated design of heuristics for resource-constrained project scheduling problem via regression algorithms. *Annals of Operations Research*. <https://doi.org/10.1007/s10479-026-07196-9>
- [36]Zhang, H., Li, L., Demeulemeester, E., Bai, S. (2025). A two-stage adjustable robust optimization approach for multi-work package project scheduling problem under uncertain environments. *Journal of the Operational Research Society*, 76 (11), 2210-2226. <https://doi.org/10.1080/01605682.2025.2463556>
- [37]Adamu, P. I., Okagbue, H. I., Oguntunde, P. E. (2019). A New Priority Rule for Solving Project Scheduling Problems. *Wireless Personal Communications*, 106 (2), 681-699. <https://doi.org/10.1007/s11277-019-06185-5>

- [38]Chakraborty, R. K., Rahman, H. F., Ryan, M. J. (2020). Efficient priority rules for project scheduling under dynamic environments: A heuristic approach. *Computers & Industrial Engineering*, 140. <https://doi.org/10.1016/j.cie.2020.106287>
- [39]Mihály, K., Kulcsár, G., Kulcsárné-Forrai, M. (2025). An Advanced Reactive Approach to Solve Extended Resource-Constrained Project Scheduling Problems. *Infocommunications Journal*, 17, 2-6. <https://doi.org/10.36244/ICJ.2025.5.1>
- [40]Hu, X., Li, J., Li, F., Wang, J., Wang, Y. (2025). Priority rule-based heuristics for distributed multi-project scheduling considering global resource failures. *Journal of the Operational Research Society*, 76 (10), 2112-2131. <https://doi.org/10.1080/01605682.2025.2457655>
- [41]Klevanskiy, N. N., Tkachev, S. I., Voloshchouk, L. A. (2019). Multi-project scheduling: Multicriteria time-cost trade-off problem. *Procedia Computer Science*, 150, 237-243. <https://doi.org/10.1016/j.procs.2019.02.048>
- [42]ElFiky, H., Owida, A., Galal, N. M. (2020). *Resource constrained multi-project scheduling using priority rules: Application in the deep-water construction industry*. International Conference on Industrial Engineering and Operations Management Dubai, UAE. <http://www.icomsociety.org/icom2020/papers/398.pdf>
- [43]Shi, Y., Du, Z., Li, J. (2019). *Hierarchy Model for Distributed Resource Constrained Multi-project Scheduling Problem*. ACM 5th International Conference on Industrial and Business Engineering (ICIBE), Hong Kong. <https://doi.org/10.1145/3364335.3364363>
- [44]Liu, W., Zhang, J., Vanhoucke, M., Guo, W. (2025). Resource allocation models and heuristics for the multi-project scheduling with global resource transfers and local resource constraints. *Computers & Industrial Engineering*, 200. <https://doi.org/10.1016/j.cie.2024.110843>
- [45]Tian, Y., Mei, Y., Zhang, M. (2025). *Genetic Programming with Activity Group Selection for Dynamic Multi-mode Resource-Constrained Project Scheduling Problems*. IEEE Congress on Evolutionary Computation, (CEC), Hangzhou, China. <https://doi.org/10.1109/CEC65147.2025.11042979>
- [46]Tian, Y., Mei, Y., Zhang, M. (2026). Scalable Knee-Point Guided Activity Group Selection in Multi-tree Genetic Programming for Dynamic Multi-Mode Project Scheduling. *PRICAI 2025: Trends in Artificial Intelligence. Lecture Notes in Computer Science*, 16454. https://doi.org/10.1007/978-981-95-7081-2_40
- [47]Zhang, Y., Li, X., Teng, Y., Shen, G. Q., Bai, S. (2024). A heuristic rule adaptive selection approach for multi-work package project scheduling problem. *Expert Systems with Applications*, 238. <https://doi.org/10.1016/j.eswa.2023.122092>
- [48]Peng, W., Wang, Z., Xie, F., Li, H. (2025). A heuristic approach for the critical chain project scheduling problem based on resource flows. *Computers & Operations Research*, 180. <https://doi.org/10.1016/j.cor.2025.107054>
- [49]Hu, X., Wang, J., Leng, K. (2019). The Interaction Between Critical Chain Sequencing, Buffer Sizing, and Reactive Actions in a CC/BM Framework. *Asia-Pacific Journal of Operational Research*, 36 (3). <https://doi.org/10.1142/S0217595919500106>
- [50]Vermeire, G., Vanhoucke, M. (2025). A priority rule heuristic for the multi-skilled resource-constrained project scheduling problem. *Applied Soft Computing*, 171. <https://doi.org/10.1016/j.asoc.2025.112776>
- [51]You, W., Xu, Z., Yu, Y., Zhao, S. (2024). A two-layer approach for solving robust decentralized multiproject scheduling problem with multi-skilled staff. *International Transactions in Operational Research*, 31 (3), 1631-1670. <https://doi.org/10.1111/itor.13392>
- [52]Rezaei, F., Najafi, A. A., Ramezani, R., Demeulemeester, E. (2021). Simulation-based priority rules for the stochastic resource-constrained net present value and risk problem. *Computers & Industrial Engineering*, 160. <https://doi.org/10.1016/j.cie.2021.107607>
- [53]Chandrasekaran, T., Mohammed, N. A., Saranya, R., Indumathi, G. S., Srinivasan, C., Sakthi Kanmani, K. (2025). *A Structured Approach to Feature Prioritization for Dynamic Roadmap Optimization*. 11th International Conference on Communication and Signal Processing (ICCSP), Melmaruvathur, India. <https://doi.org/10.1109/ICCSP64183.2025.11089247>
- [54]Wieggers, K. (2024). *First things first: Prioritizing requirements*. <https://medium.com/analysts-corner/first-things-first-prioritizing-requirements-or-anything-else-d43aa7bad4b1>

- [55]Lencastre, M., Silva, D., Pimentel, J. H. C., Castro, J. B. (2023). PRIUS: Applying Gamification to User Stories Prioritization. *ACM SIGAPP Applied Computing Review*, 23 (4), 27-44.
<https://doi.org/10.1145/3642964.3642967>
- [56]Pattyn, F., Kazmi, R., Goetz, P., Aldhahri, A., Aqsa (2025). RECTV: Enhancing Agile Decision-Making in Startups Through Value-Driven Requirements Prioritization. *IEEE Access*, 13, 142546-142560.
<https://doi.org/10.1109/ACCESS.2025.3589942>
- [57]Chakravarty, K., Singh, J. (2023). *A Deep Dive into Product Backlog Prioritization Technique in Scrum*. 21st International Conference on Information Technology, Proceedings (OCIT), Raipur, India.
<https://doi.org/10.1109/OCIT59427.2023.10430672>
- [58]Asad, K., Muqem, M. (2023). Enhancing Requirements Change Request Categorization and Prioritization in Agile Software Development Using Analytic Hierarchy Process (AHP). *International Journal on Recent and Innovation Trends in Computing and Communication*, 11 (5), 148-159.
<https://doi.org/10.17762/ijritcc.v11i5.6589>
- [59]Muhammad, A., Siddique, A., Mubasher, M., Aldweesh, A., Naveed, Q. N. (2023). Prioritizing Non-Functional Requirements in Agile Process Using Multi Criteria Decision Making Analysis. *IEEE Access*, 11, 24631-24654. <https://doi.org/10.1109/ACCESS.2023.3253771>
- [60]Yaseen, M., Mustapha, A., Shah, M. A., Ibrahim, N. (2023). A hybrid technique using minimal spanning tree and analytic hierarchical process to prioritize functional requirements for parallel software development. *Requirements Engineering*, 28 (3), 347-376. <https://doi.org/10.1007/s00766-023-00397-9>
- [61]Abdelazim, K., Moawad, R., Elfakharany, E. (2020). A Framework for Requirements Prioritization Process in Agile Software Development. *Journal of Physics: Conference Series*, 1454, 1-11.
<https://doi.org/10.1088/1742-6596/1454/1/012001>
- [62]Roy, M., Deb, N., Cortesi, A., Chaki, R., Chaki, N. (2021). NFR-aware prioritization of software requirements. *Systems Engineering*, 24 (3), 158-176. <https://doi.org/10.1002/sys.21572>
- [63]Mougouei, D., Ghose, A., Dam, H., Fahmideh, M., Powers, D. (2021). *A Fuzzy-Based Requirement Selection Method for Considering Value Dependencies in Software Release Planning*. IEEE International Conference on Fuzzy Systems, Luxembourg, Luxembourg.
<https://doi.org/10.1109/FUZZ45933.2021.9494422>
- [64]Mougouei, D., Mougouei, E., Powers, D. M. W. (2021). Partial Selection of Software Requirements: A Fuzzy Method. *International Journal of Fuzzy Systems*, 23 (7), 2067-2079. <https://doi.org/10.1007/s40815-021-01093-y>
- [65]Amelia, T., Mohamed, R. B. (2021). *A Proposed Requirements Prioritization Model Based on Cost-Value Approach with Collaboration Perspective*. International Conference on Software Engineering and Computer Systems and 4th International Conference on Computational Science and Information Management, (ICSECS-ICOCSIM), Pekan, Malaysia. <https://doi.org/10.1109/ICSECS52883.2021.00009>
- [66]Gerogiannis, V. C., Kitsis, N. K., Tzimos, D., Son, L. H. (2021). *A Novel Requirements Prioritization Approach based on 360 Degree Feedback and Group Recommendation*. ACM 5th International Conference on Algorithms, Computing and Systems (ICACS), Xi'an, China. <https://doi.org/10.1145/3490700.3490709>
- [67]Rottoli, G. D., Casanova, C. (2021). Multi-criteria group requirement prioritization in software engineering using fuzzy linguistic labels. Workshops at the Fourth International Conference on Applied Informatics (ICAIW), Buenos Aires, Argentina. https://ceur-ws.org/Vol-2992/icaiw_waai_2.pdf
- [68]Bugayenko, Y., Farina, M., Kruglov, A., Pedrycz, W., Plaksin, Y., Succi, G. (2023). Automatically Prioritizing Tasks in Software Development. *IEEE Access*, 11, 90322-90334.
<https://doi.org/10.1109/ACCESS.2023.3305249>
- [69]Talele, P., Phalnikar, R. (2021). *Classification and prioritisation of software requirements using machine learning - A systematic review*. 11th International Conference on Cloud Computing, Data Science and Engineering (Confluence), Noida, India. <https://doi.org/10.1109/Confluence51648.2021.9377190>
- [70]Silva, D., Lencastre, M., Pimentel, J., Castro, J., Lira, L. (2023). *Applying Gamification to Prioritize Requirements in Agile Projects*. 38th ACM/SIGAPP Symposium on Applied Computing, Tallinn, Estonia.
<https://doi.org/10.1145/3555776.3577708>
- [71]Kaur, K., Singh, P., & Kaur, P. (2021). A review of artificial intelligence techniques for requirement engineering. En J. Kacprzyk (Ed.) *Advances in Intelligent Systems and Computing* (pp. 259-278). Springer.
https://doi.org/10.1007/978-981-15-7907-3_20

- [72]Amelia, T., Mohamed, R. (2023). A Review: Requirements Prioritization Criteria Within Collaboration Perspective. *Pertanika Journal of Science and Technology*, 31 (1), 161-185.
<https://doi.org/10.47836/pjst.31.1.11>
- [73]Sufian, M., Khan, Z., Rehman, S., Haider Butt, W. (2018). *A systematic literature review: Software requirements prioritization techniques*. International Conference on Frontiers of Information Technology (FIT), Islamabad, Pakistan. <https://doi.org/10.1109/FIT.2018.00014>
- [74]Gull, S., Umer, R., Uddin, N., Khan, S. (2022). Current Stance on Requirement Prioritization Techniques: Challenges, Opportunities, and Future Directions. *VFAST Transactions on Software Engineering*, 10 (2), 198-210. <https://doi.org/10.21015/vtse.v10i2.1082>
- [75]Bugayenko, Y., Bakare, A., Cheverda, A., Farina, M., Kruglov, A., Plaksin, Y., Pedrycz, W., Succi, G. (2023). Prioritizing tasks in software development: A systematic literature review. *PLoS ONE*, 18 (4), 1-31. <https://doi.org/10.1371/journal.pone.0283838>
- [76]Arshad, H., Shaheen, S., Khan, J. A., Anwar, M. S., Aurangzeb, K., Alhussein, M. (2023). A novel hybrid requirement's prioritization approach based on critical software project factors. *Cognition Technology & Work*, 25 (2), 305-324. <https://doi.org/10.1007/s10111-023-00729-3>
- [77]Breyter, M. (2022). *Agile Product and Project Management: A Step-by-Step Guide to Building the Right Products Right*. Apress Media LLC. <https://doi.org/10.1007/978-1-4842-8200-7>
- [78]Záček, M., Hamplová, A., Tyrychtr, J., Vrana, I. (2025). Improvements for the Planning Process in the Scrum Method. *Applied Sciences*, 15 (1), 1-20. <https://doi.org/10.3390/app15010202>
- [79]Kuengjai, Y., Ramingwong, L. (2021). *A Pilot Study of Requirement Prioritization Techniques in Agile Software Development*. ACM 4th International Conference on Computer Science and Software Engineering, Singapore. <https://doi.org/10.1145/3494885.3494912>
- [80]Ohri, S., & Ohri, I. B. (2025). *Stakeholder Hierarchy and Conway's Law: Impact on Backlog Management Strategies in Agile Teams Post-COVID*. IEEE 5th International Conference on ICT in Business Industry and Government (ICTBIG), Indore, Madhya Pradesh, India.
<https://doi.org/10.1109/ICTBIG68706.2025.11323987>
- [81]Model, K., Herzwurm, G. (2022). *Software-Supported Product Backlog Prioritization in Scrum Software Development Projects*. International Conference on Software Business (ICSOB), Bolzano, Italy. <https://ceur-ws.org/Vol-3316/phd-paper2.pdf>