Acadlore Transactions on Applied Mathematics and Statistics
https://www.acadlore.com/journals/ATAMS

A Stochastic Formulation of Kanban Pipeline Dynamics Under Work-in-Progress Constraints

Igor Lazov*

School of Computer Science and Information Technology, University American College Skopje, 1000 Skopje, North Macedonia

* Correspondence: Igor Lazov (igor.lazov@uacs.edu.mk)

Received: 01-23-2026 **Revised:** 03-09-2026 **Accepted:** 03-18-2026

Citation: I. Lazov, "A stochastic formulation of Kanban pipeline dynamics under work-in-progress constraints," *Acadlore Trans. Appl Math. Stat.*, vol. 4, no. 1, pp. 36–45, 2026. https://doi.org/10.56578/atams040103.

Abstract: Kanban is a pull-based workflow management methodology designed to improve delivery efficiency by limiting the number of tasks that may be processed concurrently within an execution pipeline. Its defining operational mechanism is the enforcement of a work-in-progress limit, which constrains the number of active tasks, mitigates excessive task accumulation, and promotes continuous workflow. Despite its widespread industrial adoption, a rigorous stochastic characterization of Kanban pipeline dynamics under finite work-in-progress constraints has remained limited. In this study, the Kanban workflow was formulated as a finite-capacity birth-death process, where the pipeline capacity was determined by the work-in-progress limit and the state space was truncated by a closed boundary condition. System behavior is characterized by the traffic intensity, defined as the ratio of the task arrival rate to the task service rate, thereby providing a quantitative measure of resource utilization and pipeline congestion. To evaluate the structural uncertainty of workflow evolution, an information-theoretic framework was established by introducing the information associated with each pipeline state and the corresponding entropy. In addition, closed-form analytical expressions are derived for the expected number of tasks within the execution pipeline, the pipeline entropy, and the average system lead time experienced by successfully admitted tasks. The effects of both system utilization and the work-in-progress limit on these performance measures were analytically quantified, thereby revealing the fundamental trade-offs between throughput, congestion, operational uncertainty, and delivery responsiveness. The proposed formulation provides a unified probabilistic and information-theoretic perspective for analyzing constrained workflow systems, and establishes a theoretical foundation for quantitatively evaluating Kanban performance and optimizing work-in-progress policies across knowledge-intensive project environments.

Keywords: Kanban; Information and entropy; Utilization; Work-in-progress; Lead time

1 Introduction

Traditional operations management literature frequently models project execution using network-based, continuous-duration techniques, such as the program evaluation and review technique and the critical path method. While suitable for highly sequential engineering workflows, these models struggle to accurately represent continuous-delivery frameworks, such as Kanban. Kanban is highly relevant to modern project management, particularly in environments that demand flexibility, continuous improvement, and rapid response to change. While Kanban originated in manufacturing [1], it has evolved into a premier agile project management framework widely used in software development, marketing, operations, and human resources [2]. Kanban is not just a tool; it is a methodology built on a few fundamental practices designed to optimize workflow. Limiting work-in-progress is the defining characteristic of Kanban. By strictly limiting how many tasks can be in a specific column at one time, teams prevent bottlenecks, avoid multitasking, and ensure that tasks are actually completed before new ones are started. Data provided in the empirical study [3] shows that strict work-in-progresslimits reduce the hidden costs of "context switching" (i.e., the cognitive friction caused by jumping between multiple unfinished tasks). High work-in-progress acts as a "queuing system" that increases risk, slows down feedback loops, and destroys product development profitability [4].

Kanban encourages teams to look at their process regularly, analyze metrics (like how long a task takes from start to finish), and make incremental changes to improve efficiency. Traditional project management (like Waterfall)

relies on fixed, upfront planning. Kanban, however, is a pull-based system. Tasks are drawn into production only when capacity becomes available. If a project's priorities shift overnight, the team simply rearranges the backlog (the "to do" column). No time is wasted on changing rigid, long-term Gantt charts. In a Kanban environment, project progression is more accurately characterized as a stochastic counting process running through an open queuing network. The primary variable of interest is not the single-task duration, but rather the instantaneous volume of concurrent active work and its associated systemic predictability.

In traditional project management, work is often invisible, multitasking is rampant, and bottlenecks are hidden until deadlines are missed. Kanban fixes this by treating a project like a physical pipeline. In knowledge work (like software engineering, research, or academic writing), inventory building upcannot be physically seen. If a factory floor has 500 car engines sitting idle in a hallway, everyone notices the bottleneck. If a software team has 50 half-finished, unreviewed code branches, it is completely invisible on a text list. Furthermore, the physical card system has been successfully mapped onto software engineering and modern corporate project management [5]. Kanban maps the abstract lifecycle of a task into discrete, physical columns. This immediately reveals where work is piling up, transforming project tracking from an administrative guessing game into an objective, visual reality. Human beings are notoriously bad at multitasking; context switching incurs a heavy cognitive tax that actively destroys productivity. Kanban tackles this directly through work-in-progresslimits. By focusing on limiting work-in-progress, Kanban reduces the time it takes for a single item to go from the initial request to delivery (known as lead time). It forces teams to finish what they start rather than juggling ten half-done tasks simultaneously.

Instead of allowing a team to start ten different tasks simultaneously and finish none of them by a deadline, Kanban forces a strict policy: a new task cannot enter the "in-progress" lane until an existing one moves to "done". As for the traditional approach, it is characterized by high utilization and low completion. Everyone looks busy, but deliverables linger in limbo. Traditional management uses a push system—managers assign schedules and push tasks onto team members based on deadlines, regardless of whether the team has the current bandwidth to handle them. This causes stress, corner-cutting, and process errors. However, the Kanban approach is characterized by low multitasking and high throughput. The team concentrates energy to push items across the finish line faster. Kanban is a pull system. The team capacity is fixed by the work-in-progress limit. When an engineer finishes a task, they pull the next highest-priority item from the backlog. This guarantees that work moves through the pipeline at the exact maximum sustainable velocity of the team, matching demand to capacity in real time.

The adaptation of the Kanban framework outside of the technology sector has also been explored. For example, visual project management improves cross-functional communication and collaboration in modern hybrid work environments [6]. The following metrics are particularly important for project managers:

- Cycle time: Drastically reduces the time it takes for a single task to go from "started" to "shipped". For the project manager, cycle time means fast, reliable feedback loops and quicker time-to-market.
- Flexibility: No locked-in sprint commitments. The backlog can be reprioritized at any second. For the project manager, flexibility is perfect for volatile environments where client demands change weekly.
- Predictability: Stabilizes process flow, making delivery dates mathematically calculable. For the project manager, predictability allows accurate forecasting for stakeholders based on historical velocity rather than estimates.

The intersection of Kanban systems and queuing theory treats work-in-progress limits and card circulation loops as structural constraints within stochastic networks. Rather than focusing on deterministic schedules, this body of literature models manufacturing and software workflows using stochastic queuing networks to evaluate trade-offs between throughput, lead times, and capacity utilization under high variability. Key research directions in this field include:

- Synchronization and closed networks: Kanban card loops are structurally analyzed using join queuing networks, where the processing of a task or part is strictly dependent on the simultaneous availability of a free Kanban card [7, 8]. Because the total card count is fixed, these are frequently solved as closed or mixed Markovian networks.
- Optimal card allocation: A significant subset of the literature addresses the mathematical optimization of card counts across multi-stage, serial production lines. Matrix-geometric methods are utilizedto identify optimal card distribution to prevent upstream blocking and downstream starvation [9, 10].
- Comparative analysis of pull protocols: Queuing abstractions provide the baseline to mathematically compare classical Kanban against alternative pull control policies, such as constant work-in-process and hybrid extended Kanban control systems [11–13].

In this work, depending on the basic concepts of information i and entropy $S = E(i)$ for a Kanban process pipeline, a stochastic formulation of Kanban pipeline dynamics is developed. The pipeline is characterized by the current number of active tasks, denoted as N, and the maximum number of simultaneously active tasks, denoted as M.

The system is modeled as a birth-death process of finite size M + 1 with states n, where $n = 0, 1, \ldots, M$.

The information i owned by a system (i.e., carried by the system's random variable N) is a key feature of these discrete states of the birth-death process and, in essence, the system's entropy $S = E(i)$ reflects the uncertainty inherent in the system. This study relies on Gibbs' entropy formula [14], utilizing Shannon's adaptation for random variables [15] (confining the analysis to discrete variables). The approach introduced here can be used for any project type, and, therefore, any project is a suitable candidate for this form of analysis.

Thus, this study introduces an entropy-based characterization under work-in-progress truncation. Since the entropy of a system with an assumed infinite queue is a strictly increasing function, if a finite queue is considered (which is effectively the work-in-progress limit), the entropy acts as both an increasing and decreasing function, exhibiting a single maximum. As for the managerial insights, by framing the work-in-progress limit through the lens of system information and system entropy, the model helps managers pinpoint the exact threshold where a queue transitions from under-utilized (low entropy and throughput) to optimally balanced (maximum entropy and throughput) right before it becomes overly restricted and blocked (falling entropy and throughput due to the rigid boundary of the finite queue). In short, it helps managers identify the specific "sweet spot" where the system achieves maximum productivity before the negative effects of finite queue constraints (such as blocking or starvation) cause performance to degrade.

The remainder of this study is structured as follows. Section 2 gives a basic review of the project management frameworks. Section 3 introduces the performance measures in the methodology. In Section 4, the methodology is illustrated on a software feature pipeline scenario. Section 5 provides the conclusion of the study.

2 Project Management Frameworks

The project management frameworks are generally split into two distinct categories: linear/traditional (Waterfall) and agile (Scrum and Kanban). In a nutshell, Kanban's main rivals represent completely different structural answers to managing a project's timeline and constraints. When comparing frameworks, Scrum is the most direct day-to-day competitor [16], Waterfall represents the traditional structural opposite [17], and Scrumban serves as the modern hybrid [18]. Empirical evidence is provided on how combining elements of Scrum and Kanban (Scrumban) helps mitigate the rigid deadlines of Scrum, while keeping teams organized [19].

As for Kanban vs. Scrum (the core agile rivalry), while both fall under the agile umbrella, they handle execution very differently:

- Workflow: Kanban is a continuous flow (no start or end dates); Scrum is broken into strict, time-boxed intervals called sprints (usually 2 weeks).
- Core metric: Kanban tracks cycle time (how fast one item moves from start to finish); Scrum tracks velocity (how much story-point volume a team can clear in one sprint loop).
- Roles: Kanban has no prescribed roles; Scrum introduces strict structural identities like the Scrum Master and Product Owner.
- Changes: Kanban allows backlog changes in real time as long as capacity permits; Scrum locks the scope the moment a sprint begins.

As for Kanban vs. Waterfall (the structural opposite), Waterfall is a rigid, linear sequence (requirements $\rightarrow$ design $\rightarrow$ build $\rightarrow$ test). The next stage cannot be moved to until the previous state is 100% complete. Kanban handles all stages concurrently across different tasks. A tester can test Task A, while a developer writes code for Task B, and a designer sketches Task C.

The comparison between Kanban and Scrumban (the hybrid successor) is described below. As project environments grow more complex, many teams find themselves stuck between the rigidity of Scrum and the loose structure of pure Kanban. This gave rise to Scrumban, which pulls structural properties from both worlds. It uses the visual workflow boards and strict work-in-progress limits of Kanban, but keeps the planning rituals, retrospective alignment, and occasional review cadences of Scrum to prevent long-term process drift.

3 System Model

3.1 System Entropy

It is assumed that a Kanban system consists of an unconstrained backlog, an active execution pipeline, and a deployment sink. The state of the system is uniquely identified by a random variable N, representing the bounded set of active tasks currently inside the pipeline, where $N \in \{0, 1, 2, \ldots, M\}$. The upper limit M represents the strict, policy-enforced work-in-progress limit. It is assumed that the ingestion of tasks from the backlog follows a Poisson process with arrival intensity λ, and task execution/clearance follows an independent exponential distribution with service intensity μ. Then, the pipeline utilization (or, traffic intensity) $\rho = \lambda/\mu$ is defined as a ratio of arrival rate to service rate, representing how busy the system or team is. Kanban uses a continuous flow regulation via ergodic queues. Kanban does not use periodic time boxes. Instead, it applies a strict upper bound on the number N of active tasks allowed simultaneously. This is the work-in-progress limit M.

The Kanban system can be represented as a birth-death process of size M + 1, where the state n , with $n = 0, 1, \ldots, M$, denotes the exact number of active tasks N currently inside the system pipeline (tasks being developed, reviewed, or tested). This study considers three metrics: i) arrival rate λ: the rate at which new tasks are pulled from the unconstrained backlog into the active pipeline, ii) service rate μ: the rate at which the team successfully completes and deploys tasks, clearing them from the active pipeline, iii) work-in-progress limit M: the strict upper bound on the number of active tasks allowed in the system simultaneously. Because of the work-in-progress limit, any arrival when the system is at capacity ($n = M$) is blocked or deflected back to the backlog. The state-dependent arrival rate λ_n and service rate μ_{n+1}, with $n = 0, 1, \ldots, M$, are defined as:

$$\lambda_n = \begin{cases} \lambda, & 0 \le n < M \\ 0, & n \ge M' \end{cases} \quad \mu_{n+1} = \begin{cases} \mu, & 0 \le n < M \\ 0, & n \ge M' \end{cases} \tag{1}$$

Therefore, this study can model the Kanban system as an open queueing system in a steady state. If tasks arrive into the backlog at rate λ, and are deployed at rate μ, then $\rho = \lambda/\mu$ represents the traffic intensity, or utilization factor, of the project team.

The measure N, which determines the distinct system's states, represents a random variable obeying a probability distribution $p_n = \mathrm{Prob}\{N = n\}$, where $n = 0, 1, 2, \ldots, M$. The distribution is managed by the probability continuity law (using the global balance equations) and the probability conservation law (using the normalization factor) [20–22]. As a result, the probability distribution p_n, where $n = 0, 1, 2, \ldots, M$, is specified by the expressions:

$$\frac{p_n(\rho)}{p_0(\rho)} = \prod_{j=0}^{n-1} \frac{\lambda_j}{\mu_{j+1}} = \prod_{j=0}^{n-1} \frac{\lambda}{\mu} = \rho^n, \quad n = 0, 1, \ldots, M,$$

$$p_0(\rho) = \frac{1}{\mathrm{geom}_M(\rho)}, \mathrm{geom}_M(\rho) = \sum_{n=0}^{M} \rho^n, \rho > 0.$$

Thus, the probability of having N items in progress follows a geometric distribution restricted by the limit M, and for the probability mass function for the steady-state of a Kanban pipeline, the following equation can be obtained:

$$p_n(\rho) = \frac{\rho^n}{\mathrm{geom}_M(\rho)} = \frac{\rho^n}{\sum_{j=0}^{M} \rho^j}, \quad n = 0, 1, \ldots, M; \quad \rho > 0.$$

i.e.,

$$p_n(\rho) = \begin{cases} \frac{1-\rho}{1-\rho^{M+1}} \cdot \rho^n, & \rho \ne 1. \\ \frac{1}{M+1}, & \rho = 1. \end{cases}, n = 0, 1, \ldots, M. \tag{2}$$

Therefore, for ρ = 1, the geometrical distribution collapses into a uniform distribution across all allowed states n, where $n = 0, 1, \ldots, M$. As a critical theoretical edge, it can be noted that unlike the open queues with infinite size (where $\rho < 1$ is strictly required to prevent infinite queue explosion), a Kanban system remains mathematically stable even if $\rho \ge 1$ (i.e., when work arrives faster than the team can finish it). The work-in-progress limit structurally forces stability by truncating the state space. Therefore, p_0 is the probability that the system is entirely idle, and p_M is the probability that the Kanban board reaches its work-in-progress limit and rejects incoming tasks.

For a given system with limit M, every value ρ' of the parameter ρ specifies one (equilibrium) system's macrostate. The parameter ρ, being a frequency ratio, is a kinematic metric. Staying in any of its particular states n in amacrostate, the system owns (i.e., the variable N carries) a quantity of information $i_N(\rho)$, with feasible values $i_n(\rho)$, with $n = 0, 1, \ldots, M$, and its expected value, i.e., entropy $S(\rho)$, specified as:

$$i_n(\rho) \stackrel{\text{def}}{=} -\ln p_n(\rho), \quad n = 0, 1, \ldots, M, \quad \rho > 0, \tag{3}$$

$$S(\rho) = E(i(\rho)) \stackrel{\text{def}}{=} \sum_{n=0}^{M} i_n(\rho) \cdot p_n(\rho), \quad \rho > 0. \tag{4}$$

The entropy of this steady-state Kanban pipeline is static over time. Because $S(\rho)$ does not scale with the project duration T, Kanban mathematically eliminates the risk of an entropy explosion over long timelines. The mean value of the quantity N, representing the expected value of items in the pipeline (i.e., the average number of tasks inside the pipeline), i.e., the function $\overline{N}(\rho)$, is given by:

$$\overline{N}(\rho) \stackrel{\text{def}}{=} \sum_{n=0}^{M} n \cdot p_n(\rho) = \rho \cdot \frac{[\mathrm{geom}_M(\rho)]'_\rho}{\mathrm{geom}_M(\rho)} = \frac{\sum_{j=0}^{M} n \cdot \rho^j}{\sum_{j=0}^{M} \rho^j}, \quad \rho \in (0, +\infty). \tag{5}$$

To resolve this equation completely as a pure function of system input parameters ρ and M, the following can be derived:

$$\overline{N}(\rho) = \begin{cases} \frac{\rho}{1-\rho} - \frac{(M+1)\cdot\rho^{M+1}}{1-\rho^{M+1}}, & \rho \neq 1 \\ M/2, & \rho = 1 \end{cases}. \tag{6}$$

This is the expected value of work-in-progress. The team canhave an average of $\overline{N}$ tasks active on the board at any given time. For any finite size $M \geq 1$, the mean $\overline{N}$ is a consistently increasing function with respect to the parameter ρ, with values in the interval (0, M). It is worth noting that, if $M \to \infty$, then $\overline{N}$ is a strictly limitless function with respect to the parameter ρ (then, $\rho < 1$). Eq. (6) is split into two distinct parts. The first term is the open queue baseline, which is the standard formula for an infinite-capacity single-server M/M/1 queueing system (i.e., when $M \to \infty$), where the first M denotes memoryless arrivals (meaning the time between arrivals follows an exponential distribution), and the second M indicates memoryless service (meaning the service time follows an exponential distribution). It represents how many tasks would be in the system if there were no work-in-progress limit. As utilization ρ approaches 1, this term explodes toward infinity. The second term is the capacity correction factor, which is the "stabilizer". Because the system has a strict work-in-progress limit M, the queue cannot grow infinitely. This term calculates the exact number of tasks that cannot enter the system due to blocking, shaving them off the infinite baseline to keep $\overline{N}$ strictly bounded between 0 and M.

The operational chaos or uncertainty associated with the Kanban pipeline is represented by the entropy of the probability distribution. The next equation can be derived through Eqs. (3) and (4):

$$S(\rho) = \ln\left(\mathrm{geom}_M(\rho)\right) - (\ln\rho)\cdot\overline{N}(\rho), \rho > 0. \tag{7}$$

Eq. (7) is well known, and also presents the relationship between the entropy $S(\rho)$ of the system and its expected number $\overline{N}(\rho)$ of tasks inside the pipeline. Combining Eqs. (6) and (7) yields the final closed-form expression for the Kanban process entropy:

$$S(\rho) = \begin{cases} \ln\left(\frac{1-\rho^{M+1}}{1-\rho}\right) - (\ln\rho)\cdot\left(\frac{\rho}{1-\rho} - \frac{(M+1)\cdot\rho^{M+1}}{1-\rho^{M+1}}\right), & \rho \neq 1 \\ \ln(M+1), & \rho = 1 \end{cases}. \tag{8}$$

The function $S(\rho) \to 0$, as $\rho \to 0^+$ or $\rho \to \infty$. Moreover, the significant maximum uncertainty point $\rho_{M,\max}$, at which system entropy $S(\rho)$ reaches its maximum value, is the solution of the equation obtained as $S'(\rho) = 0$,

$$\sum_{n=0}^{M}[n - \overline{N}(\rho)]\cdot(\rho^n)\cdot\ln(\rho^n) = 0 \tag{9}$$

It is obvious that $\rho_{M,\max}$ = 1. Therefore, in a perfectly balanced system, when the arrival rate perfectly matches service capacity (i.e., $\lambda = \mu$), the resulting entropy collapses to its absolute upper bound. It is important to note that, if $M \to \infty$, then entropy $S(\rho)$ is a strictly limitless function with respect to the parameter ρ (then, $\rho < 1$). For a rigorous analytical description of how the number of active tasks N and carried information i_N change mutually, please refer to the study [23].

As a managerial implication, operating at a perfect ratio of demand to capability, without an excess buffer capacity, results in maximum pipeline uncertainty. At any random evaluation threshold (i.e., check-in), the system is equally likely to be completely blocked, completely idle, or anywhere in between, maximizing the cognitive tracking load of the project manager. If demand drastically outstrips team capability ($\lambda \gg \mu \Rightarrow \rho \to \infty$), then a team is continuously overwhelmed, system entropy drops to zero, and the system becomes entirely deterministic—it is locked perpetually at the maximum capacity M. While the predictability is high, this signifies a severely bottlenecked process where lead times skyrocket. Therefore, in an overloaded Kanban framework, this represents a permanent gridlock, and the pipeline is continuously maxed out at its work-in-progress ceiling, i.e., the expected number $\overline{N}(\rho)$ of tasks inside the pipeline approaches the capacity M.

The structural behavior of the Kanban process entropy S (through Eq. (8)), and its expected number of active tasks $\overline{N}$ (through Eq. (6)), with respect to the system utilization parameter ρ (i.e., mapped across changing operational macrostates) and system size M, are presented in Figure 1.

3.2 System Lead Time

To connect information-based analysis directly to delivery schedule performance, this study bridges the proposed entropy framework with Little's Law [24]. The effective arrival rate (i.e., throughput) of the pipeline, accounting for tasks turned away due to work-in-progress saturation, is defined as:

$$\lambda_{\text{eff}} = \lambda\cdot(1 - p_M) = \mu\cdot(1 - p_0). \tag{10}$$

where,

$$p_M(\rho) = \frac{\rho^M}{\sum_{j=0}^{M} \rho^j}, \quad p_0(\rho) = \frac{1}{\sum_{j=0}^{M} \rho^j}.$$

This represents the actual rate at which tasks are admitted into the workflow, i.e., λ_{eff} is the rate of the tasks that actually "makes it through the gate".

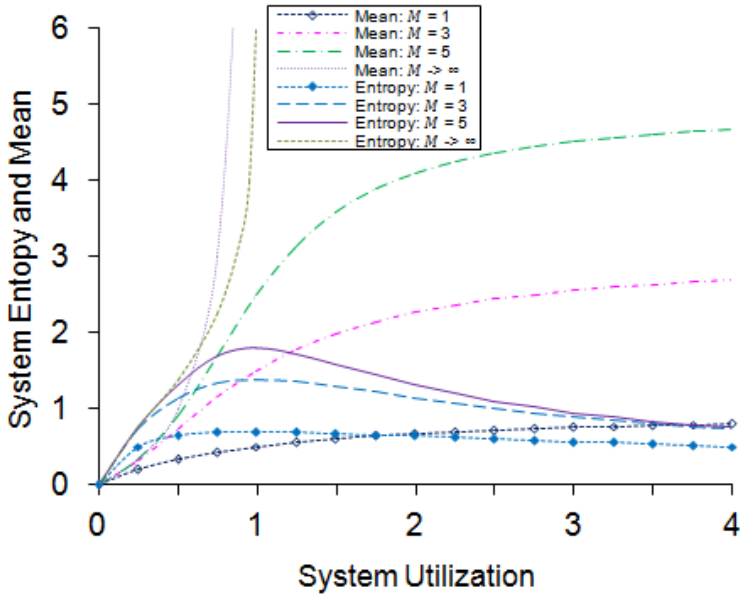

Figure 1. Entropy S and mean $\overline{N}$ vs. utilization ρ for a Kanban pipeline

Note: For any $M \geq 1 \Longrightarrow \rho_{M,\max} = 1$; $M \to \infty \Longrightarrow S$ and $\overline{N}$ are unlimited; for any $M \geq 1, \rho \to \infty \Rightarrow \overline{N} \to M$.

The average time a successfully admitted task spends in the active Kanban pipeline from start to finish (i.e., system lead time) is given by:

$$T_{\text{lead}}(\rho) = \frac{\overline{N}(\rho)}{\lambda \cdot (1 - p_M(\rho))}. \tag{11}$$

Isolating the expected inventory parameter $\overline{N}$ from the core entropy Eq. (7) allows us to map out a unified equation linking delivery lead time directly to system entropy as follows:

$$T_{\text{lead}}(\rho) = \frac{i_0(\rho) - S(\rho)}{\lambda \cdot (1 - p_M(\rho)) \cdot \ln \rho} = \frac{\ln\left(\text{geom}_M(\rho)\right) - S(\rho)}{\lambda \cdot \left(\frac{\text{geom}_{M-1}(\rho)}{\text{geom}_M(\rho)}\right) \cdot \ln \rho},$$

i.e.,

$$T_{\text{lead}}(\rho) = \begin{cases} \frac{\ln\left(\frac{1-\rho^{M+1}}{1-\rho}\right) - S(\rho)}{\lambda \cdot \left(\frac{1-\rho^M}{1-\rho^{M+1}}\right) \cdot \ln \rho}, & \rho \neq 1. \\ \frac{M+1}{2 \cdot \lambda}, & \rho = 1. \end{cases}, \quad n = 0, 1, \ldots, M. \tag{12}$$

It proves that a project manager cannot optimize or stabilize delivery lead time T_{lead} without modifying the underlying workflow boundary conditions M to compress and bound the process entropy of the delivery pipeline. Considering an 8-hour working day basis to complete a task, then if an accepted task takes 0.5 times units to complete, $T_{\text{lead}} = 0.5 \cdot 8\text{ h} = 4\text{ h}$ can be obtained for the average lifecycle duration of a task in the system.

3.3 Managerial Insights

In an actual Kanban environment, regarding managerial insights, evaluating the work-in-progress limit using system information and entropy allows decision-makers to identify the exact transition point (i.e., the significant maximum uncertainty point $\rho_{M,\max}$) where a Kanban pipeline reaches its maximumoperational state diversity before capacity constraints limit performance.

That is, this model helps managers to identify the critical threshold where a Kanban pipeline transitions from under-utilized (low entropy, low throughput, and low lead time) to optimally balanced (maximum entropy, maximum throughput, and medium lead time), right before it becomes over-loaded and blocked (falling entropy and throughput, due to the rigid boundary of the finite work-in-progress limit, but increasing lead time).

4 A Software Feature Pipeline Scenario

This study imagines a specialized software development team operating a strict Kanban pipeline for implementing user stories. The parameters are:

- Arrival rate (λ): New feature requests arrive at an average rate of λ tasks per day. It is assumed that arrivals follow a Poisson process.
- Service rate (μ): The team can process and complete tasks at an average rate of μ tasks per day. Service times are exponentially distributed.
- Work-in-progress limit (M): The Kanban board has a strict total work-in-progress limit of M tasks in progress at any one time. If the pipeline is full ($n = M$), any incoming request is rejected (or, blocked from entering) until a task is completed.

Because this is a birth-death process, the system can only transition from state n to n + 1 (birth), or to $n - 1$ (death). Therefore, no more tasks can enter when the work-in-progress limit is reached ($n = M$), and the team cannot finish a task if the pipeline is empty ($n = 0$). Furthermore, this study defines the traffic intensity (utilization factor) as $\rho = \lambda/\mu$. Using Eq. (2), the stationary probability for every state in the Kanban pipeline can be computed. Using these steady-state probabilities, this studycan extract vital operational metrics for this specific Kanban pipeline (i.e., key performance metrics for the project manager).

Thus, this study calculates the average number of cards on the board at any given time through Eq. (6), the entropy of the pipeline through Eq. (8), the actual rate at which tasks enter the pipeline through Eq. (10), and the average lead time from the moment a task successfully enters the Kanban board until it is completed through Eq. (12), and converts this to hours (assuming an 8-hour workday). Considering M = 1, 3, 5, and taking ρ = 1/2; 3/4; 1; 4/3; 2, the results are presented in Tables 1–3. In the scenarios where the average cycle time for a single Kanban card exceeds 1 time unit, then one full 8-hour workday is not enough for completing the task. As it could be seen from Tables 1–3, this study analyzes the traffic intensity (i.e., utilization factor) $\rho = \lambda/\mu$, ranging from smaller to higher values, investigating the output metrics behavior. These input values can serve as a good theoretical demonstration, but they also represent some software project scenarios when there is an increasing demand for software developers, who subsequently process tasks at a reduced rate. In this direction, it can be also observed that as software developer demand rate λ rises (at given processing rate μ) the pipeline throughput is increasing, and as software developer processing rate μ falls (at given demand rate λ) the pipeline throughput is decreasing (but, in both cases, the traffic intensity $\rho = \lambda/\mu$ grows). Note that pairs of arrival rate λ and service rate μ with inverse values exhibit the same throughput. At ρ = 1 (i.e., $\lambda = \mu$), after Eqs. (2) and (10), the throughput equals $\lambda_{\text{eff}} = \lambda \cdot M/(M + 1)$. As M increases, this value approaches λ, but it never reaches it for any finite M.

Table 1. Output Kanban pipeline parameters with the work-in-progress limit M = 1 for different values of λ and μ

λ	μ	ρ	p_0	p_1	$\overline{N}$	S	λ_{eff}	T_{lead}	$T_{\text{8-hour}}$
2	4	0.5	0.67	0.33	0.33	0.64	1.33	0.25	2 h
3	4	0.75	0.57	0.43	0.43	0.68	1.71	0.25	2 h
3	3	1	0.5	0.5	0.5	0.69	1.5	0.33	2.67 h
4	3	1.33	0.43	0.57	0.57	0.68	1.71	0.33	2.67 h
4	2	2	0.33	0.67	0.67	0.64	1.33	0.5	4 h

Note: λ: arrival rate; μ: service rate; ρ: utilization factor; p_0: idle probability; p_1: blocking probability for M = 1; $\overline{N}$: mean number of active tasks; S: system entropy; λ_{eff}: effective arrival rate; T_{lead}: average lead time; $T_{\text{8-hour}}$: lead time in hours based on an 8-hour workday.

Table 2. Output Kanban pipeline parameters with the work-in-progress limit M = 3 for different values of λ and μ

λ	μ	ρ	p_0	p_3	$\overline{N}$	S	λ_{eff}	T_{lead}	$T_{\text{8-hour}}$
2	4	0.5	0.53	0.07	0.73	1.14	1.87	0.39	3.14 h
3	4	0.75	0.37	0.15	1.15	1.34	2.54	0.45	3.62 h
3	3	1	0.25	0.25	1.5	1.39	2.25	0.67	5.33 h
4	3	1.33	0.15	0.37	1.85	1.34	2.54	0.73	5.84 h
4	2	2	0.07	0.53	2.27	1.14	1.87	1.21	9.71 h

Note: λ: arrival rate; μ: service rate; ρ: utilization factor; p_0: idle probability; p_3: blocking probability for M = 3; $\overline{N}$: mean number of active tasks; S: system entropy; λ_{eff}: effective arrival rate; T_{lead}: average lead time; $T_{\text{8-hour}}$: lead time in hours based on an 8-hour workday.

Table 3. Output Kanban pipeline parameters with the work-in-progress limit M = 5 for different values of λ and μ

λ	μ	ρ	p_0	p_5	$\overline{N}$	S	λ_{eff}	T_{lead}	$T_{\text{8-hour}}$
2	4	0.5	0.51	0.02	0.90	1.30	1.97	0.46	3.68 h
3	4	0.75	0.30	0.07	1.70	1.68	2.78	0.61	4.89 h
3	3	1	0.17	0.17	2.5	1.79	2.5	1	8 h
4	3	1.33	0.07	0.30	3.30	1.68	2.78	1.18	9.48 h
4	2	2	0.02	0.51	4.09	1.30	1.97	2.08	16.65 h

Note: λ: arrival rate; μ: service rate; ρ: utilization factor; p_0: idle probability; p_5: blocking probability for M = 5; $\overline{N}$: mean number of active tasks; S: system entropy; λ_{eff}: effective arrival rate; T_{lead}: average lead time; $T_{\text{8-hour}}$: lead time in hours based on an 8-hour workday.

4.1 Key Observations and Symmetries

By modeling the Kanban system this way as an operational insight, a project manager can mathematically justify the work-in-progress limit. For instance, if the manager notices that the blocking probability p_M is too high (meaning client requests are rejected or delayed in the backlog too often), they can use this birth-death framework to simulate how increasing the work-in-progress limit, or adding capacity to increase service rate μ, would alter both the team's idle probability p_0 and the average lead time T_{lead} .

The perfect equilibrium (uniform state probabilities) is as follows:

i) When ρ = 1, every single state in the birth–death process has an identical probability of occurring: $p_n = 1/(M+1)$, with $n = 0, 1, \ldots, M$. At any random point in the day, the board is just as likely to be completely empty, as it is to be fully maxed out, or partially occupied.

ii) The risk of developer idleness p_0, is perfectly balanced against the risk of blocking customer demand p_M.

iii) The average number of cards on the board sits precisely at the midpoint of the available pipeline capacity ($\overline{N} = M/2$).

As for symmetrical balances, given two points related by inversion symmetry of the pipeline utilization parameter ρ measured from the point $\rho_{M,\max}$ = 1 (that is, $\rho' \cdot \rho'' = 1$), the following can be derived (Tables 1–3):

$$\overline{N}(\rho') + \overline{N}(\rho'') = M, \quad S(\rho') = S(\rho''),$$

$$p_0(\rho') = p_M(\rho''), \quad p_M(\rho') = p_0(\rho''), \quad \lambda_{\text{eff}}(\rho') = \lambda_{\text{eff}}(\rho'').$$

Remark 1. The analytical framework presented assumes Poisson arrivals and exponential service times, i.e., this study models the Kanban pipeline as a finite-capacity single-server M/M/1/M queueing system, where the current number of active tasks N obeys a truncated geometrical distribution. In a real Kanban environment, if the situations where the quantity N follows any other discrete distribution (i.e., represents any other queueing system type) are considered, this framework could also be adapted. Then, different formulas can be obtained for the entropy and the mean.

Remark 2. Using a given sample with size m for the number of active tasks N, when the estimation of the pipeline utilization parameter ρ is of interest, then the entropy-based estimation (using the average carried information) coincides with the maximum likelihood estimation (using the average task count) through Eq. (7).

Remark 3. Several studies have integrated uncertainty into the performance optimization of various systems. For instance, a stochastic analyses accounting for system uncertainty have been applied tomanufacturing environments comprised of automated machinery and conveyor belt [25]. Moreover, besides the uncertainty (i.e., risk) captured by an outside observer,the risk captured by arriving and departing customersin a stochastic service system could also be investigated and contrasted. It can be noted that this current study examines solely the risk captured by an outside observer's viewpoint.

5 Conclusion

Kanban is a visual agile framework focused on continuous delivery and managing work-in-progress. It does not use fixed-time sprints like Scrum; instead, work flows continuously. Teams use a visual board (a Kanban board) with columns representing stages of work (e.g., to do, in progress, review, and done). Tasks are represented by sticky notes or digital cards. The team sets "work-in-progress limits" to ensure no single stage becomes a bottleneck. It is best for teams that need flexibility to change priorities at any moment and want to optimize efficiency (e.g., support teams, content creation, or operational work). To construct a rigorous mathematical representation, this study maps the project pipeline to abirth-death process with finite size, operating within a closed state-space. By applying system information principles, this study treats the operational variation within this pipeline as entropy associated with the pipeline. This allows us to formalize "process chaos" not as a subjective cultural attribute, but as an explicit, measurable property of system topology.

When a team tries to manage too many active tasks at once, they split their attention, deal with more tracking overhead, and suffer from high operational friction. This effectively lowers the actual service rate μ, causing T_{lead} to spike exponentially rather than linearly. This is exactly why Kanban systems enforce strict work-in-progress limits to actively keep $\overline{N}$ low, making T_{lead} remain fast and predictable and enabling system uncertainty, represented by the system entropy, to be a bounded measure (that is, introducing the maximum uncertainty point). The policy-driven work-in-progress limit acts as a structural boundary condition that truncates the state space, guaranteeing a steady-state solution even under continuous overload conditions. Therefore, because Kanban does not require chaining the team structures or job titles immediately, it is an excellent gateway framework for traditional organizations transitioning to agile.

Data Availability

The data supporting our research results are included within the article.

Conflicts of Interest

The author declares no conflicts of interest.

References

[1] T. Ohno, *Toyota Production System: Beyond Large-Scale Production*. Portland, USA: Productivity Press, 1988.

[2] C. A. Sathe and C. Panse, "An empirical study on impact of project management constraints in agile software development," *Braz. J. Oper. Prod. Manag.*, vol. 20, no. 3, p. 1796, 2023. https://doi.org/10.14488/BJOPM.1796.2023

[3] N. Damij and T. Damij, "An approach to optimizing Kanban board workflow and shortening the project management plan," *IEEE Trans. Eng. Manag.*, vol. 71, pp. 13 266–13 273, 2021. https://doi.org/10.1109/TEM.2021.3120984

[4] D. De Grandis, *Making Work Visible: Exposing Time Theft to Optimize Work & Flow (2nd ed)*. Portland, USA: IT Revolution Press, 2022.

[5] D. J. Anderson, *Discovering Kanban: The Evolutionary Path to Enterprise Agility*. Bilbao, Spain: Kanban University Press, 2023.

[6] L. Griffiths and A. Tyson, "Visualising success: The Kanban approach to improving collaboration and communication in the library," *Leg. Inf. Manag.*, vol. 23, no. 4, pp. 245–250, 2023. https://doi.org/10.1017/S1472669623000580

[7] S. Koukoumialos and G. Liberopoulos, "An analytical method for the performance evaluation of echelon Kanban control systems," *OR Spectrum*, vol. 27, pp. 339–368, 2005. https://doi.org/10.1007/s00291-004-0188-0

[8] L. Zhang, "Kanban-controlled exponential production lines: Analysis and design," *J. Manuf. Technol. Manag.*, vol. 24, no. 3, pp. 358–383, 2013. https://doi.org/10.1108/17410381311318873

[9] N. M. Dizbin and B. Tan, "Optimal control of production-inventory systems with correlated demand inter-arrival and processing times," *Int. J. Prod. Econ.*, vol. 228, p. 107692, 2020. https://doi.org/10.1016/j.ijpe.2020.107692

[10] B. Tan, O. Karabağ, and S. Khayyati, "Energy-efficient production control of a make-to-stock system with buffer- and time-based policies," *Int. J. Prod. Res.*, vol. 62, no. 16, pp. 5809–5827, 2023. https://doi.org/10.1080/00207543.2023.2298488

[11] M. Thürer, N. O. Fernandes, H. Lödding, and M. Stevenson, "Material flow control in make-to-stock production systems: An assessment of order generation, order release and production authorization by simulation," *Flex. Serv. Manuf. J.*, vol. 37, pp. 1–37, 2025. https://doi.org/10.1007/s10696-024-09532-2

[12] Z. G. Zhang, *Fundamentals of Stochastic Models (1st ed)*. Boca Raton, USA: CRC Press, 2023.

[13] A. S. Xanthopoulos and D. E. Koulouriotis, "A comparative study of different pull control strategies in multi-product manufacturing systems using discrete event simulation," *Adv. Prod. Eng. Manag.*, vol. 16, no. 4, pp. 473–484, 2021. https://dx.doi.org/10.14743/apem2021.4.414

[14] J. W. Gibbs, "On the equilibrium of heterogeneous substances," *Am. J. Sci.*, vol. s3-16, no. 96, pp. 441–458, 1878. https://doi.org/10.2475/ajs.s3-16.96.441

[15] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

[16] W. T. Lee and C. H. Chen, "Agile software development and reuse approach with Scrum and software product line engineering," *Electronics*, vol. 12, no. 15, p. 3291, 2023. https://doi.org/10.3390/electronics12153291

[17] T. Reilly, *The Staff Engineer's Path: A Guide for Individual Contributors Navigating Growth and Change*. Sebastopol, USA: O'Reilly Media, Inc., 2022.

[18] C. Ladas, *Scrumban: Essays on Kanban Systems for Lean Software Development*. Seattle, USA: Modus Cooperandi Press, 2009.

[19] M. Alqudah and R. Razali, "An empirical study of Scrumban formation based on the selection of Scrum and Kanban practices," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 8, no. 6, pp. 2315–2322, 2018. https://doi.org/10.18517/ijaseit.8.6.6566

[20] U. N. Bhat, *An Introduction to Queueing Theory: Modeling and Analysis in Applications*. Birkhäuser Boston, MA, USA: Springer, 2015.

[21] J. F. Shortle, J. M. Thompson, D. Gross, and C. M. Harris, *Fundamentals of Queueing Theory (3rd ed)*. New York, USA: John Wiley & Sons, 2018.

[22] D. Bertsimas and D. Gamarnik, *Queueing Theory: Classical and Modern Methods*. Charlestown, MA, USA: Dynamic Ideas LLC, 2022.

[23] I. Lazov and P. Lazov, "Geometrical interpretation of the population entropy maximum," *Stoch. Models*, vol. 40, no. 3, pp. 569–582, 2024. https://doi.org/10.1080/15326349.2023.2297959

[24] J. D. C. Little, "A proof for the queuing formula: $L = \lambda W$," *Oper. Res.*, vol. 9, no. 3, pp. 383–387, 1961. https://doi.org/10.1287/opre.9.3.383

[25] I. Lazov, "Risk-based analysis of manufacturing systems," *Int. J. Prod. Res.*, vol. 57, no. 22, pp. 7089–7103, 2019. https://doi.org/10.1080/00207543.2019.1577564