Shadow Technical Debt: The Dark Matter of Engineered Systems

Abstract. Shadow technical debt is not defined by who or what created it. It is defined by the fact that it exists before an organization has acknowledged, measured, or budgeted for it. Although the phenomenon is not new, recent discussion has highlighted the role of distributed and agentic development environments in accelerating the creation of hidden liabilities within complex engineered systems.

Over the past two decades, project management for systems has emphasized risk management and resource-constrained decision making. It stresses focusing limited resources where they can most effectively reduce uncertainty and system risk. This paper frames shadow technical debt as a long-standing condition within engineered systems and gives it socioeconomic context, enabling management within existing frameworks. Building on established risk and resource allocation methodologies, including cost-managed discovery and prioritization approaches, the paper outlines practical techniques for identifying shadow technical debt within tightly constrained budgets. As contemporary agentic and automated engineering processes increase the rate at which systems are created and modified, the volume of hidden technical debt is growing by orders of magnitude and may exhibit a superlinear relationship. Efficient methods for discovering and classifying these liabilities are therefore essential for converting shadow debt into actionable technical debt and bringing it onto the engineering balance sheet.

Keywords: Technical Debt · Risk-Based Decision Making · Resource Allocation · Cost-Benefit Analysis · Decision Making under Uncertainty.

0 Current Events

Prior to the recent popular emergence of the term shadow technical debt, the author informally described this phenomenon using the metaphor dark technical debt, drawing analogy to dark matter in physics: liabilities that cannot be directly observed but whose effects influence system behavior. The term shadow technical debt is used in this paper because it has recently entered industry discourse. Originally, this paper was intended to be an internal artifact, to guide our Accurate Project Management[65] in accordance with the Optimizing principle of CMMI level 5[11].

0.1 JetBrains Publicly Names "Shadow Technical Debt"

In March 2026, public discussion surrounding AI coding agents began describing the hidden liabilities left behind by rapid AI-assisted development as a distinct category of technical debt[83]. One widely circulated example came from JetBrains[1], which characterized the artifacts produced by agentic development workflows as shadow tech debt. The significance of this moment lies less in the appearance of a new engineering condition than in the emergence of shared terminology. Once a widely used development platform publicly recognizes a category of hidden liabilities, the concept becomes visible to a broader audience of managers, architects, and decision-makers. Whether JetBrains coined the phrase, popularized it, or attached it to an already existing pattern[72], the effect is similar: a previously diffuse engineering concern becomes recognizable within mainstream technical discussion.

0.2 An Observed Instance of Shadow Technical Debt

Events in early 2026 illustrate the same pattern from another direction. On 8 January 2026 an AI-generated commit introduced hidden technical debt into an internal system[10], which required subsequent cleanup several days later [64]. The example is not unusual. Code generated rapidly by AI-assisted tooling can embed undocumented assumptions, duplicated logic, fragile interfaces, unreviewed dependencies, or architectural inconsistencies. The provenance of the code may differ, but the resulting liabilities resemble patterns long familiar to engineering teams. The episode reinforced the need for clearer terminology and management practices around hidden technical liabilities in engineered systems.

1 Introduction

Hidden technical liabilities have long existed within engineered systems, but they often remain unrecognized until they are explicitly named and examined.

1.1 Shadow Technical Debt Predates AI

This paper argues that shadow technical debt has always existed, even when engineering organizations lacked a stable name for it. The current wave of AI-assisted and agentic development has not created a fundamentally new kind of debt. Instead, it has exposed an old pattern in a new medium: work created under pressure, outside full visibility, with deferred understanding and delayed accountability. The real contribution of the present moment is therefore linguistic and managerial as much as technical. Once the condition is named, it can move from a vague sense of unease to something that can be discussed, analyzed, and incorporated into engineering planning.

[1] [83] discusses approaches for preventing the creation of shadow technical debt through additional up-front investments.

1.2 Purpose and Contributions of This Paper

The central claim of this paper is straightforward: shadow technical debt is the class of technical liability that exists before it is formally acknowledged, documented, owned, or budgeted. Hidden debt can take many forms, including code, configuration, workflow dependencies, prompt chains, manual procedures, undocumented patches, integration logic, etc. Its defining feature is not the technology that produced the debt, but the fact the debt remains outside formal visibility.

This paper introduces a definition of shadow technical debt and examines its relationship to conventional technical debt. It proposes a severity spectrum for engineering liabilities and outlines cost-managed approaches for discovering and prioritizing hidden liabilities using established risk- and return-based resource allocation frameworks. By giving the phenomenon a name and a structure, shadow technical debt can move from an unknown condition to a visible liability that can be incorporated into engineering and management decision processes.

2 Background

Technical debt is a familiar concept in software engineering, but familiar concepts are not always complete concepts. The standard literature describes debt as the future cost of present expediency. That formulation is useful, yet it often assumes that the debt is at least visible enough to be discussed. In practice, much of the most consequential debt lives in the period before formal recognition. It behaves like debt before anyone writes it down as debt.

This paper focuses on that pre-recognition interval. The question is not whether technical debt exists; that is well established. The question is what to call the liabilities that are already shaping cost, fragility, and risk while still existing outside the organization's explicit debt model. Those liabilities are what this paper calls shadow technical debt.

2.1 Motivation and Context

In January 2026[10], our OpenAI Codex AI agent created hidden technical debt. The root cause and impact were the same as the hidden technical debt historically introduced by human developers. Codex did nothing fundamentally new. What changed was the rate at which the problem could occur.

I have been dealing with technical debt in enterprise systems for many years, including work on Navy ERP Business Systems data warehouse transformation at Naval Sea Systems Command (2007) and more recently supply chain risk management (SCRM) efforts in Department of Defense[2] and Defense Industrial Base systems[60].

[2] The United States Department of Defense is also known as Department of War, and is used interchangeably.

My perspective on this problem comes from work in requirements management, architecture analysis, cost modeling, and prioritization in environments where resources were constrained and systems were complex. In those contexts, the problem was never simply to find everything that might be wrong. The real problem was deciding what could be known, what could be tested, and what could be acted upon within explicit budget and staffing limits.

This work included the development and application of prioritization and governance methods such as Accurate Project Management (APM) and Criticality Analysis and Risk Assessment (CARA), along with large scale requirements testing and architecture cost modeling across Department of Defense and federal systems.

That experience matters here because shadow technical debt is not primarily a philosophical problem. It is a resource allocation problem. If organizations had unlimited time and unlimited money, hidden liabilities would still be inconvenient, but they would not be strategically important. Shadow debt becomes important because real organizations must decide what fraction of scarce resources should be spent converting unknown liabilities into known ones, and then deciding which known liabilities justify action.

2.2 Selected Technical Debt in the Literature and Practice

The history of technical debt begins with Cunningham's original metaphor[16] of expedient code that accelerates delivery in the present while increasing cost in the future. Over time, that metaphor broadened beyond code to include architecture, documentation, configuration, testing, operations, and other dimensions of software systems. Yet one practical issue remained persistent throughout that evolution: not all debt is visible when it is incurred.

McConnell[51, p. 6] made this visibility problem explicit by observing that technical debt is often much less visible than other engineering obligations and by recommending debt lists and backlogs to make it transparent. Fowler[22] sharpened the point by distinguishing deliberate from inadvertent debt, noting that teams can take on debt without fully realizing what they are doing and only later understand the liability they created. The familiar "just get it done" response sits across both cases. Sometimes it reflects conscious borrowing against future effort. Sometimes it produces liabilities that are only recognized after the fact. In that sense, technical debt has always had an epistemic dimension.

Brown[8] moved visibility closer to the center of the concept by identifying it as a core property of technical debt and warning that serious problems arise when debt is not visible enough to those who will ultimately pay for it. This is an important predecessor to what is now described as shadow technical debt. The liability may already exist in the system, but it is not yet legible to the organization that will inherit its cost.

Allman[1] added a temporal dimension by observing that technical debt can be almost invisible early in its life because the interest payments have not yet begun to come due. Kruchten[39] then cautioned against equating technical debt only with what tools can detect, arguing that significant categories, especially

architectural debt, include invisible elements that resist simple measurement. Taken together, these works show that invisibility was not a late addition to technical debt theory. It was present in the concept before recent vocabulary emerged around hidden, unknown, or shadow debt.

In practice, technical debt has long appeared in at least two broad forms: debt arising through system evolution and deferred maintenance, and debt incurred more explicitly through expedient borrowing against future effort. Both forms can produce shadow debt because the liability may be real long before it is visible to management, tooling, or budgeting.

This older literature also aligns with practice. Long before AI coding assistants, organizations accumulated undocumented scripts, brittle spreadsheets, one-off data pipelines, local patches, manual restart procedures, unsupported dependencies, and quietly tolerated shortcuts. These artifacts often existed because "just get it done" was a rational response to immediate delivery pressure. Over time, however, expedient work became habitual, habitual work became dependency, and dependency remained effectively invisible until a failure, audit, security review, or modernization effort forced it into view. At that point, the organization often behaved as though a new problem had appeared, when in fact an old problem had merely become visible.

Later case studies reinforced this pattern, Yli-Huumo[87] examined sources of technical debt and approaches to managing it in practice, while Guo[29] explored the costs of technical debt management and the economic tradeoffs involved in addressing it. These studies help connect the early conceptual literature to the organizational reality that technical debt is not only a design or implementation issue, but also a budgeting and prioritization issue.

More recent practice sources use newer vocabulary more directly. Bernhardsson[6] describes a team accumulating shadow tech debt in the form of critical monster SQL queries built outside the formal data function. Windward[86] links unmanaged app sprawl and shadow IT to growing tech debt and frames guardrails and centralized governance as the response. Russo[69] likewise warns that lightly governed internal tools built to bypass delivery gaps can create fragile solutions and shadow tech debt when review, ownership, and standards are weak. These sources do not define the category, but they show that practitioners increasingly recognize and name the phenomenon.

For that reason, the recent focus on AI-created shadow debt should be interpreted carefully. AI agents did not create the category, and agentic tech debt is not the only source of shadow tech debt. They intensified a preexisting mechanism. The debt created by an agent is not fundamentally different from the debt created by an exhausted team working under deadline pressure. In both cases, the organization accepted output without fully paying the visibility, validation, or governance cost up front.

2.3 Cross-Disciplinary Precedents on Naming

The underlying idea is not new. Across religion, strategy, economics, psychology, and organization theory, naming changes what can be perceived, communicated,

and acted upon. A condition without a name may still be felt, but it is harder to classify, discuss, and resource. Once named, it can become an object of attention and action. That is the role of shadow technical debt in this paper. The term does not create a new liability. It gives an existing liability a name that makes it easier to analyze and manage.

The Ancient Power of Naming Across human history, unnamed things have carried disproportionate power. A phenomenon without a name is difficult to communicate, classify, or confront. In mythic, religious, and folk traditions, naming is often associated with control, recognition, or diminished fear. Whether interpreted literally or metaphorically, the principle is instructive: ambiguity preserves power, while naming reduces it [9, pp. 47-55] [53, p. 96] [52, pp. 21, 80] [7, p. 303] and ultimately controls it.

Engineering organizations follow the same pattern. Teams often live with recurring liabilities long before they can describe them precisely: a fragile subsystem, a workflow dependent on a single person, or a shortcut that has quietly become permanent. Until a problem has a name, it cannot easily become shared knowledge. It is not easily discussed, not easily understood by leadership, and therefore unlikely to receive resources for correction.

Military Strategy Military strategy has long recognized that classification enables action. Commanders do not allocate resources against ambiguity; they allocate against identified threats, defined terrain, known constraints, and prioritized objectives. Moving from uncertainty to action requires reducing ambiguity enough to support planning. A threat that is not categorized is difficult to brief, defend against, or resupply against.

The same principle applies in engineering management. Shadow technical debt may be widely felt without being formally defined. Once named, however, it becomes possible to discuss severity, scope, cost, and priority. Naming does not solve the problem, but it converts an atmospheric concern into something that can enter planning.

Sun Tzu [13, pp. 24-25] captured this principle succinctly: "If you know the enemy and know yourself, you need not fear the result of a hundred battles". Mao [14, pp. 22-23] likewise emphasized the decisive role of information: "Intelligence is the decisive factor... Because of superior information, guerrillas always engage under conditions of their own choosing".

Strategic doctrine also recognizes the inverse principle[45, p. 158], that advantage often lies in concealing one's own plans: "No policy is better than that which remains hidden from the enemy until you have executed it ... To know in war how to recognize an opportunity and seize it is better than anything else".

Economics and Psychology Economics and psychology point to the same practical conclusion. Cialdini[12] shows how disclosure, commitment, and personal knowledge can increase influence and compliance pressure. Klein[37] shows

how exposure and dependency can create leverage once one party becomes committed to a specific position or asset. Schelling[71] extends the point into strategy. In mixed-motive settings, outcomes depend not only on incentives, but on signals, recognizable limits, and what he called "the signaling of intentions and the meeting of minds". Ocasio[54] provides the organizational parallel. Firms act according to how attention is distributed, which issues become salient, and how rules, resources, and relationships channel decision-makers into particular communications and procedures. Taken together, these perspectives explain why naming matters. Once the concept of otherwise unknown liabilities has a name, organizations can focus attention on that class of problems, discuss it in common terms, and make it part of organizational decision-making.

Cognitive and Neurological Basis At the cognitive level, ambiguity often produces avoidance rather than analysis. Leaders who do not understand a problem often defer it, minimize it, or treat it as too vague to fund. Once a condition is named, it becomes easier to ask concrete questions about it. How much exists? What does it threaten? What would it cost to discover? What would it cost to ignore?

This is consistent with the broader literature on labeling, ambiguity, and regulation. Siegel's phrase "name it to tame it" captures[79] the basic idea that verbal identification helps reduce diffuse reactivity and supports more deliberate response. Related work points in the same direction. The amygdala is particularly sensitive to ambiguous or uncertain threat signals[85,31], while studies of affect labeling and cognitive regulation show that putting feelings or stimuli into words recruits more deliberate cortical processing and can reduce limbic reactivity[55,58,42]. Bartel[2] points more broadly to the role of language in organizing meaning. The implication for enterprise systems is practical. When liabilities remain vague, they are more likely to evoke the fear and avoidance associated with uncertain threat than the analysis associated with a defined problem. Once the category is named, the problem becomes easier to count, model, prioritize, and govern. In that sense, naming shadow technical debt is the first step in taming it.

Partial Observability, Rational Inattention, and Distributed Decision-Making Several adjacent research areas examine decision-making when relevant facts are hidden, distributed, or costly to obtain. These literatures do not describe shadow technical debt directly, but they provide useful conceptual analogs, especially when it comes to later analysis.

A substantial body of work in multi-agent reinforcement learning studies agents operating under partial observability. Omidshafiei et al.[57], for example, examine deep decentralized multi-agent reinforcement learning in settings where agents must act using local observations, limited communication, and incomplete knowledge of the true system state. The central problem is that the agent cannot directly observe reality and must instead infer hidden state from history and interaction. This is analogous to one aspect of shadow technical debt: important

facts are missing, and action must proceed despite incomplete visibility. However, these models typically presume that the tasks, objectives, or reward structures are already defined, even if the state is only partially observed.

Related distributed Q-learning research makes a similar assumption. Kar et al.[36] study collaborative learning in which agents observe only local costs, exchange information over a network, and converge toward a joint policy through distributed coordination. These models capture the important idea that no single actor sees the entire system and that coordination emerges through communication. That is relevant to the distributed organizational setting considered later in this paper, including the role of CARA in enterprise-level prioritization. Even so, the task itself remains formally defined. The literature addresses fragmented observation, not the more organizational problem in which the liability has not yet identified as a task at all.

Further related literature concerns partially observable decision processes and belief-state reasoning. In Partially Observable Markov Decision Process style models, agents maintain probability distributions over hidden states and update those beliefs as new observations arrive[38,44,77]. More generally, this structure can be understood as observations $\rightarrow$ belief state $\rightarrow$ policy. That sequence aligns with the present paper's concern for hidden-fact probability, bounded discovery, and the allocation of limited resources under uncertainty. The key difference, again, is that these models ordinarily treat hidden state as the problem, not hidden liability. The state may be uncertain, but the governing task is still admitted and specified.

An especially relevant parallel comes from economics rather than reinforcement learning. Sims[80] frames decision-making under rational inattention, where actors face real costs in acquiring, processing, and using information. This perspective is particularly useful for the present paper because shadow technical debt is often not merely unseen, but uneconomical to fully inspect at all times. Limited attention and limited budget force selective observation. In that sense, shadow technical debt is not only a visibility problem; it is also a resource-allocation problem.

A final nearby, and much earlier, literature arises in multi-robot coordination and task allocation. Research in this area studies how distributed agents allocate work under constraints of information, communication, and resource cost[24]. This is the closest formal analogue to the economic aspect of the present paper. Even here, however, the framing typically assumes that the tasks already exist and that the problem is deciding who should perform them. Shadow technical debt introduces a prior problem: the organization may be experiencing the effects of a liability before that liability has been formally recognized as a task, an objective, or a budget item.

Taken together, these literatures help clarify what shadow technical debt is and is not from an analytical point of view. It resembles partial observability, distributed coordination, and rational inattention in that relevant facts may be hidden, fragmented, or costly to acquire. It differs in that the central difficulty is

not merely hidden state, but unrecognized liability. Shadow technical debt exists before it becomes an admitted object of management.

3 Defining Shadow Technical Debt

The most important practical fact about shadow technical debt is that the unnamed problem is typically the unfunded problem. In organizational settings, leaders do not usually allocate resources to diffuse anxieties; they allocate resources to recognized categories. When engineering cannot state clearly what the problem is, management often responds with avoidance, discomfort, or performative optimism. The liability may be sensed, but it remains too vague, too broad, or too socially costly to address directly.

For that reason, naming is not a cosmetic act. It is the first managerial intervention. Once the phenomenon has a name, engineers can make bounded and concrete requests: funding for automated code scanning, time for peer review, dependency analysis, structural differencing, or a limited discovery effort. These are no longer requests to "look into something scary." They are requests to execute defined work with defined outputs.

In this sense, naming performs two functions simultaneously. First, it reduces ambiguity enough to make rational planning possible. Second, it lowers the social cost of acknowledging the problem. A named problem may be funded because management genuinely understands it or because management prefers not to appear uninformed; from an engineering standpoint, that distinction is secondary. What matters is that the liability becomes discussable, budgetable, and therefore actionable.

3.1 Formal Definition

In this paper, shadow technical debt is defined as technical liability that exists operationally while remaining outside formal engineering visibility, ownership, or governance. It may be introduced by individual developers, teams, contractors, low-code or no-code builders, or AI agents. Its essential property is not who created it, but its status: it has not yet been converted into recognized technical debt within the organization's formal system of record.

At the point of recognition, the severity of a shadow liability may be unknown. That uncertainty does not exclude it from the category. What matters is that the liability already affects the system while remaining untracked, unowned, or outside explicit governance.

This definition is important because it separates the hiddenness of the liability from the mechanism of its creation. A hidden dependency introduced by a developer under deadline pressure and a hidden dependency generated by an AI coding agent are both instances of shadow technical debt if they remain relied upon while untracked. Shadow technical debt is therefore not a special AI-only subtype of technical debt. It is a category defined by visibility and governance.

3.2 Distinction from Conventional Technical Debt

The conventional literature on technical debt generally assumes that the liability is at least visible enough to be discussed. Cunningham[16] originally framed technical debt as expedient code that may satisfy immediate customer needs while imposing future cost. Kruchten et al.[39] later expanded the idea by emphasizing that debt can also arise through the evolution of technology and context over time, even when the original engineering decision was reasonable. They further acknowledged forms of debt that are not immediately visible to all, although their discussion largely concerns debt visible to software developers even when it may not be visible to the enterprise as a whole.

Conventional technical debt is therefore debt that has already crossed a threshold of recognition. A team may disagree about urgency, cost, or remediation strategy, but the debt is known to exist. It can be logged, estimated, prioritized, deferred, or consciously accepted. It is debt inside the system of record.

Shadow technical debt, by contrast, is debt before admission. It affects behavior before it appears in planning. It may exist in source code, infrastructure, deployment procedures, undocumented manual steps, brittle interfaces, or dependencies that nobody remembers selecting. It may not yet exist as shared organizational knowledge, but it may still be experienced through symptoms such as fragility, recurring workarounds, unexplained delay, or dependence on a single individual.

The distinction can be stated simply: conventional technical debt is technical debt that is already recognized, whether actively managed or intentionally deferred. Shadow technical debt is technical debt that remains outside management because it has not yet been recognized as an object of management.

4 Severity Spectrum of Technical Debt

Not every shortcut constitutes technical debt. Some shortcuts are rational, temporary, and net-positive. The problem is not expediency itself, but unmanaged liability. For practical decision-making, technical debt, and especially shadow technical debt, should therefore be treated as a spectrum rather than as a binary condition. Organizations need a way to distinguish a tolerable shortcut from a continuing resource burden, and a continuing burden from a latent failure mode.

This paper adopts a deliberately coarse three-level spectrum: non-debt conditions, resource liability, and latent risk ("time bomb"). The purpose of this model is prioritization rather than metaphysical precision. It is intended to support triage, resource allocation, and governance. Hiddenness and severity are analytically distinct: a shadow liability may be benign, burdensome, or catastrophic.

4.1 Non-Debt Conditions

At the lowest end of the spectrum, a shortcut may not be debt at all. A tactical simplification can function as an asset when it enables delivery without

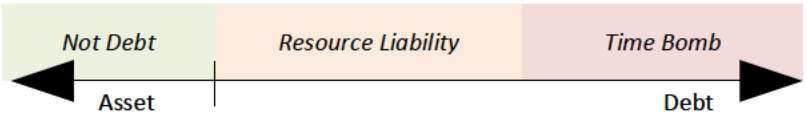

Fig. 1. Severity spectrum of technical debt, from non-debt conditions through resource liability to latent risk

materially increasing future cost, fragility, or risk. Some simplifications remain temporary, retain clean rollback paths, and impose no meaningful interest payment[51, p. 5]. In such cases, describing the artifact as debt would be analytically misleading.

This category is important because it prevents the present argument from collapsing into a general critique of speed. Rapid decisions are not inherently problematic. Technical debt begins when a shortcut creates persistent future liability rather than bounded present value.

4.2 Resource Liability

The next level is resource liability. Here the shortcut does not necessarily threaten immediate failure, but it imposes continuing cost in time, money, friction, or reduced agility. The system becomes slower to change, harder to review, harder to test, or more expensive to operate. The organization is not yet facing catastrophe, but it is paying interest.

Most day-to-day technical debt resides in this category. Resource liability consumes future engineering capacity gradually rather than dramatically, and for that reason it is often tolerated. It may never produce a major incident, but it steadily diverts effort that could otherwise be applied to new capability, assurance, or resilience.

4.3 Latent Risk ("Time Bomb")

At the highest end of the spectrum is latent risk: liability that functions as a concealed failure mode. The system may depend on an unsupported component, a vulnerable library, an undocumented recovery path, a single maintainer, a brittle hidden integration, or a body of generated code that no one fully understands. The costs associated with this form of debt may remain dormant for long periods and then materialize all at once.

This is the form of technical debt that makes shadow technical debt strategically dangerous. It is also the category most likely to remain underfunded, precisely because it appears inexpensive until failure occurs.

5 Making Shadow Technical Debt Visible

This paper does not attempt to provide a general solution to all technical debt. Such an ambition would be too broad to be practically useful. Its narrower concern is what organizations should do once they recognize that shadow technical debt exists. The approach proposed here is intentionally limited to two principles: first, name the phenomenon; second, address it through cost-managed resource allocation.

That limitation is deliberate. Organizations facing shadow technical debt are rarely operating with surplus time, staffing, or budget[87,41]. The hidden liability exists precisely because the original work was performed under constraint. Remediation models that assume abundant cleanup capacity are therefore poorly matched to the conditions that produced the debt in the first place. Cunningham's original metaphor makes the same point from the opposite direction: debt can accelerate delivery only when it is repaid promptly; otherwise the accumulated "interest" can bring an engineering organization to a standstill [16]. More recent practitioner guidance makes a similar argument in managerial terms, warning that technology used as a short-term fix can quickly add to technical debt and that firms must understand the financial exposure if critical systems fail[32].

5.1 Naming the Phenomenon

The first intervention is taxonomic rather than technical. An organization must first admit that there exists a class of liability that precedes formal recognition, and that this class deserves a name. Without that step, engineers remain confined to intuition, management remains confined to ambiguity, and the issue remains trapped in informal conversation.

Naming changes the resource conversation. Once shadow technical debt is accepted as a legitimate category, organizations can request bounded discovery work rather than open-ended remediation commitments. A request to "fix all the hidden problems" is easy to refuse because it is undefined in scope, cost, and outcome. By contrast, a request for a limited discovery effort, combining scanning, targeted review, and logging, defines both the work and the expected output. Naming does not resolve the liability. It makes the liability discussable, budgetable, and therefore admissible into planning.

5.2 Mechanisms of Emergence

Shadow technical debt usually emerges through pressure rather than malice. Its canonical cause is the imperative to "just get it done": deadlines approach, staffing is thin, features must ship, incidents must be resolved, and demonstrations must succeed. Under such conditions, teams build the path available to them, not necessarily the path they would have designed under less constrained circumstances. Empirical studies consistently describe technical debt in these terms: a short-term optimization driven by time-to-market pressure, customer

commitments, and competitive deadlines, later repaid as extra work, quality degradation, and economic drag[87,29,17,34]. Yli-Huumo et al. describe technical debt as arising from practical development pressures, where short-term decisions made to meet deadlines or delivery goals later manifest as technical debt[87].

The same pattern appears in the agentic era under a different slogan: AI is cheap, so throw AI at the problem. Low marginal generation cost encourages rapid production, but the liabilities created downstream are not free merely because the tokens were inexpensive. If generated artifacts are weakly reviewed, poorly documented, architecturally inconsistent, or weakly owned, they enter the same shadow zone as older human shortcuts. In this respect, the difference between human-created and agent-created debt is principally one of provenance and rate of accumulation, not one of operational effect.

Shadow technical debt also emerges through organizational bottlenecks, ownership gaps, tribal knowledge, workaround culture, shadow systems, local optimization, and deferred governance. The technologies change over time; the pattern does not. Bernhardsson explicitly describes workaround artifacts produced around data-team bottlenecks as "shadow tech debt"[6]. ServiceNow similarly uses the term to characterize liabilities created by rapid application development without sufficient guardrails[75]. Gosselin describes a related phenomenon in which users bypass prescribed systems, "McGyver[sic]" local workarounds, and create diffuse "shadow technical debt"[26]. Fowler's distinction between deliberate and inadvertent debt is also relevant here: teams can incur debt "without even realizing" what they are doing[22].

The consequences are not merely theoretical. Primary incident investigations show how hidden, weakly governed liabilities become operational failures. GitLab's 2017 postmortem linked prolonged outage and data loss to manual, undocumented recovery work and to backup testing that had no clear ownership [19]. The Equifax breach investigation emphasized the security burden created by complex legacy systems and governance failures that delayed corrective action[84]. In the Knight Capital case, regulators found that the firm responded narrowly to immediate defects without addressing deeper control weaknesses, leaving it exposed to catastrophic failure[73]. These cases illustrate not simply technical debt, but technical debt that remained insufficiently visible, insufficiently prioritized, or insufficiently governed until failure forced recognition.

A related literature connects technical debt directly to security exposure. Izurieta et al. argue that addressing technical debt associated with security weaknesses can improve earlier vulnerability detection in the lifecycle[33]. Siavvas et al. report a statistically significant positive correlation between technical debt and vulnerability density, suggesting that debt can function as an indicator of software security risk[78]. Shadow technical debt is therefore not merely a productivity issue. It is also a resilience and risk issue.

Taken together, these sources support a common conclusion: the mechanisms that produced pre-agentic "just get it done" debt are operationally indistinguishable from those now producing AI-assisted hidden debt. In both cases, speed-

over-quality tradeoffs generate artifacts that may be under-tested, poorly documented, weakly owned, and operationally relied upon. The difference is not the impact surface, maintainability, resilience, security, and long-term cost, but the provenance of the artifacts and the rate at which they are being produced.

5.3 Discovery and Knowing There Are Unknowns

A useful conceptual model comes from distributed learning under partial observability. Earlier distributed reinforcement-learning work treated the difficulty as one of coordination under incomplete local observation and as a hidden task model[3] (HTM) problem[28]. The author's earlier notes extended this line of thinking by recasting the problem as one of hidden knowledge: the task space is defined, yet the relevant facts available to any individual agent, or even to the agent population as a whole, remain incomplete[61].

Let F denote the universe[4] of relevant facts, let Σ denote observable signals, and let

$$\Sigma \Subseteq F \tag{2}$$

denote the subset[5] of facts that may appear as observable or transmissible signals. Each agent i maintains a history set

$$H_i(n) \Subseteq F \tag{3},$$

updated over time from direct observation, communication, and self-reflection.

The agent does not act directly on raw history, but on an attention-controlled projection

$$\pi_i(n) = \mathcal{P}_{\alpha_i(n)}(H_i(-\infty : n)) \tag{4},$$

where $\mathcal{P}_{\alpha_i(n)}$ denotes a bounded attention-and-compression operator. This projection induces an effective decision structure $Q_i(n)$, from which actions are selected. In this framework, "superstition" arises when distinct causal histories collapse into the same operational representation, so that action is guided by a compressed internal model rather than[6] by the full underlying truth.

Practically, we can define the attention based projection operation on historical memory as:

$$\mathcal{P}_{\alpha_i}(H) = \Pi_i\Big(\mathcal{T}_{\alpha_i}(H)\Big) \tag{5},$$

[3] The HTM is in which task identity is not directly observable and must be inferred from incomplete observations.

[4] $\{\, H \subseteq F : |H| < \aleph_0, |F| = \aleph_0 \,\}, \quad \forall n \in \mathbb{N},\ \exists H \subseteq F$ such that $|H| = n$

[5] The operator $\Subseteq$ is used to denote a subset relationship that is formally identical to $\subseteq$, but where the left-hand set is materially small relative to the right-hand set. It is intended to convey practical scale disparity rather than introduce a new mathematical construct, for example selecting a bounded working set from a combinatorially large or effectively unbounded universe.

[6] This has both desired beneficial and undesired detrimental effects, and through reinforcement the good will outweigh the bad.

which requires the attention based search, $\mathcal{T}_{\alpha_i}$, of historical memory followed by the transformation, Π_i, or synthesis of the gathered knowledge in to an actionable (Q-value) decision matrix. This structure allows the snap decisions based on $\Pi_i(n)$ mapping, which is how we humans work[62].

At the population level, knowledge at time n may be written as

$$K(n) = \bigcup_i H_i(n) \tag{6},$$

with the basic constraint

$$K(n) \subseteq F \tag{7}.$$

In principle, equality would require that the entire agent population collectively possess (fill Figure 2 completely) every relevant fact. In practice, that condition is unrealistic. Facts must first be observable, then transmitted, then retained, then survive attention and projection. Storage limits, communication limits, filtering, bias, and compression all make complete knowledge unattainable. Operational knowledge is smaller still, because only a subset of stored history survives projection into the representations that actually guide action.

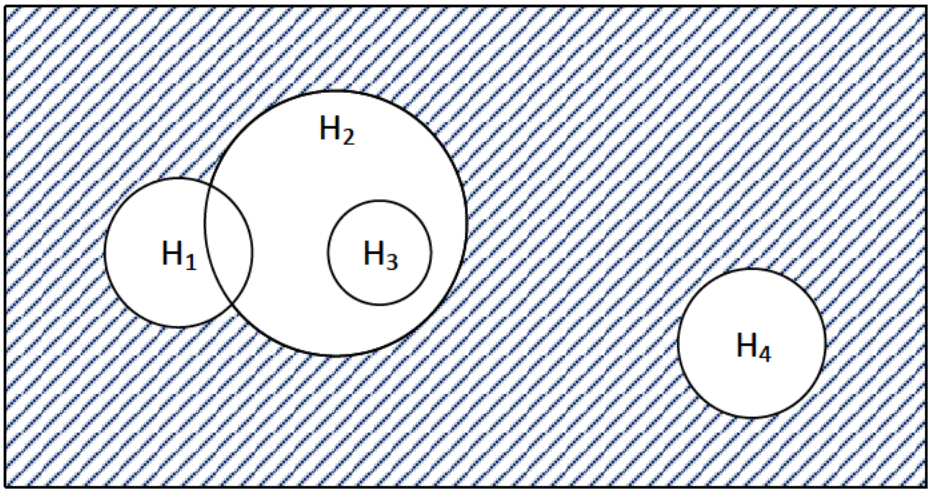

Fig. 2. Universe of facts with sets of agent knowledge (not to scale), see Eq. ((3),)

This distinction between facts, stored history, and operational belief is important. A system may stabilize around internally consistent projections even while large portions of F remain outside the knowledge frontier representable under bounded memory and attention of its agents. In other words, the system can function coherently while remaining materially ignorant. The relevant unknowns,

$$unknowns = F - K(n) \tag{8},$$

are not an edge case; they are a structural consequence of bounded observation, bounded memory, and bounded attention.

The implication for shadow technical debt is direct. The present paper does not require proof that the unknown liabilities are vast in magnitude, only that they necessarily exist. If organizational knowledge is always a strict subset of the relevant fact space, then some technical liabilities will remain outside formal recognition. Shadow technical debt (see: Figure 3) is therefore not an anomaly but an expected consequence of bounded knowledge.

That conclusion motivates a second one: because exhaustive discovery is infeasible, discovery itself must be managed under explicit cost constraints. The goal is not to spend whatever is required to find every hidden liability. The goal is to set a budget, however small, and use that budget to convert as much shadow technical debt as possible into known technical debt. Once the liability is known, it is no longer in the shadows; it can be logged, sized, risk-ranked, deferred, accepted, or remediated.

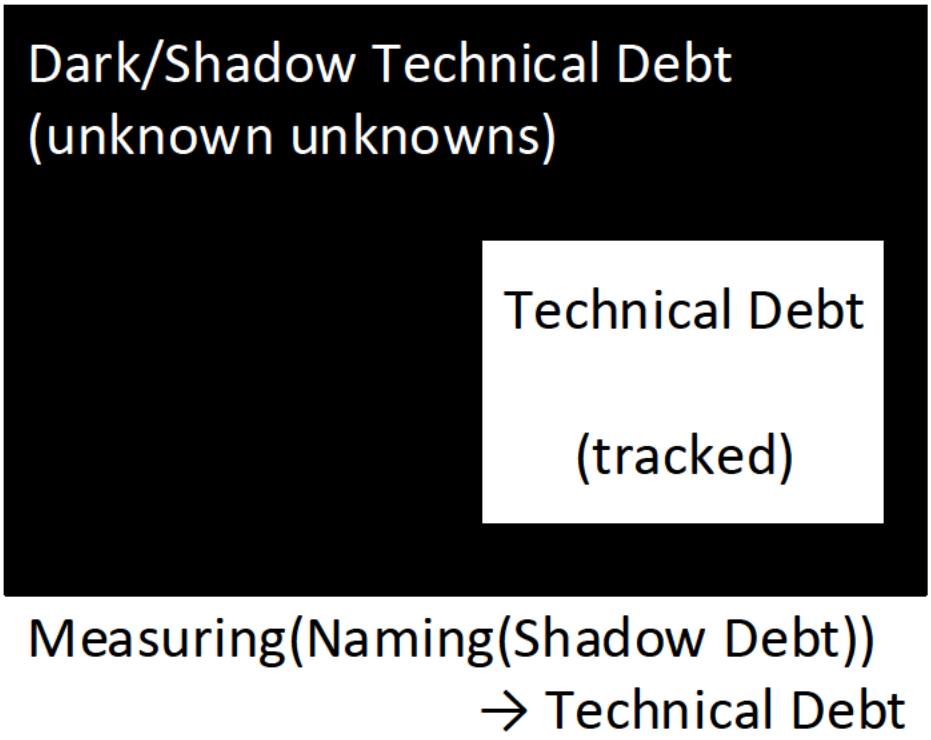

Fig. 3. The Debt Visibility Model

This is where criticality analysis and risk assessment (cost managed) frameworks become essential. Viewed from one perspective, the problem is one of risk management. Viewed from another, it is one of return on investment. In practice, both views serve the same purpose: scarce resources should be allocated where they reduce the greatest uncertainty or the greatest potential loss per unit of cost. Search therefore becomes a problem of contrast and prioritization, finding which artifacts, dependencies, interfaces, workflows, or generated outputs most strongly deviate from the expected operational model and therefore deserve bounded investigation.

A useful way to frame this principle is proportionality. If an organization spent a large amount generating or acquiring software quickly, it need not spend a comparable amount to determine whether that work introduced hidden liabilities. A first-pass discovery budget may be one or even two orders of magnitude smaller than the original development budget. The exact ratio is context dependent; the important point is that discovery itself can be budgeted, bounded, and managed.

For example, if a team spent $4,000 in AI tokens or equivalent resources producing software, an initial discovery effort might reasonably be budgeted at $4, $40, or $400 depending on system criticality, risk exposure, and mission impact. That discovery effort is not intended to repair everything. Its purpose is to answer a cheaper and more urgent question: what was just created that now needs to be understood? Once the resulting liabilities are identified and logged, they have moved from unknown unknowns to known debt.

This prioritization approach reflects a long-recognized strategic principle: actors with superior information and situational awareness choose when and where to engage. Earlier military doctrine described this advantage as engaging only when the conditions of success are favorable[14, pp. 22-23]. The same principle applies here. Discovery is not free, and organizations cannot afford to search everywhere at once. They must decide where limited discovery effort is most likely to reveal consequential hidden liabilities. This transition is the practical core of the paper. The first task is not exhaustive cleanup. The first task is the low-cost conversion of hidden liabilities into visible liabilities under explicit resource constraints.

6 Cost-Managed Discovery and Prioritization

The discovery and prioritization problem addressed in this paper is not captured by partial observability alone. Most formal models treat uncertainty as something to be inferred within an already formed task–action problem, rather than as something that determines whether known tasks become salient at all. In Gu and Maddox's distributed reinforcement learning framework, for example, the hidden layer is a Hidden Task Model that maps incomplete perception into latent tasks before action[28]. Recent partially observable multi-agent reinforcement learning remains in that same analytical lane[44]: it studies POSGs, information sharing, and tractable learning under hidden state, but still assumes

that the decision problem itself has already been specified. The present paper shifts the unit of uncertainty from "which task or state am I in?" to "which facts are missing, and therefore which known tasks fail to activate?" That is precisely the layer these models leave exogenous. They also generally do not model the further possibility that a task may be recognized and still be deferred because the cost of analysis is too high.

Adjacent literatures approach parts of this problem, but only in fragments. Rational inattention formalizes action under capacity-limited information processing[80,50]. Metareasoning formalizes the tradeoff between additional deliberation and improved decision quality[68], and Conitzer shows that even restricted metareasoning variants are computationally hard to solve optimally[15]. More recent work extends this logic by treating computational effort itself as a choice variable[70,43]. Observation-cost and active-perception research comes closest on the control side by making measurement itself costly and by allowing agents to learn when additional observation is worth acquiring[4,5,3,77]. Rationally inattentive inverse reinforcement learning adds attention costs to behavioral modeling[30], but still does so within a pre-specified decision problem rather than a setting in which hidden facts determine whether tasks are activated, deferred, or ignored.

Organizational theories address a related issue from a different direction. Ocasio's attention-based view of the firm explains how organizations channel limited attention across competing issues[54]. Rational inattention to discrete choice[49], problemistic search under multiple feedback signals[81], and broader behavioral theories of the firm[23] likewise help explain why some problems are noticed, escalated, deferred, or ignored. Taken together, these literatures provide strong models of hidden state, costly information, bounded attention, and constrained deliberation, but still no standard model in which hidden facts determine whether known tasks are never recognized, recognized but deferred, or actively pursued under explicit enterprise cost constraints. That is the gap addressed by the shadow-technical-debt framing developed here.

Once an organization accepts that some liabilities will remain outside formal visibility, the practical question is no longer abstract inference but bounded discovery: which hidden liabilities should be investigated first, and how much should be spent to make them visible? The remainder of this chapter treats that question as a resource-allocation problem. Discovery is not free, and in resource-constrained organizations it cannot be exhaustive. It must therefore be prioritized under explicit budgets using methods that weigh informational value, risk reduction, and expected loss against the cost of inspection.

6.1 Why Discovery Must Be Budgeted

The premise of finite resources is foundational. Robbins' classic formulation defines economics as the study of choices made under scarcity, where ends compete for scarce means with alternative uses[67, p. 16]. Shadow technical debt does not suspend that condition; it intensifies it. The organization must decide not only what to fix, but what it can afford to discover.

This logic is reflected in both public- and private-sector governance. U.S. government policy requires agencies to establish safeguards "commensurate with the risk and magnitude of the harm"[56], a formulation structurally analogous to the private sector's need to allocate limited resources in proportion to expected loss. NIST continuous-monitoring guidance makes the same point explicitly: timely and relevant information is especially important when resources are limited and agencies must prioritize their efforts[18]. In both settings, the operative principle is not maximal effort, but measured effort under constraint.

A second reason discovery must be budgeted is temporal instability. What is valuable, urgent, or dangerous at one moment may not remain so at the next. Robbins' wartime example is instructive: assets that were productive immediately before the Armistice became waste immediately after it, although their physical substance had not changed[67, pp. 47-48]. The same principle applies to discovery. The value of inspecting a hidden liability depends on timing, because mission priorities, architectural dependencies, adversarial conditions, and business context all evolve. Discovery is therefore not simply a search problem; it is a search problem under changing value conditions.

A third reason is organizational asymmetry. Information is rarely located where budget authority resides. Szukits and Móricz[82] show that information asymmetry creates a structural separation between where data exists and where decisions are made. Under such conditions, management actions necessarily incorporate subjective inputs, not merely as preference, but as a consequence of incomplete and unevenly distributed information. In the present context, this means that whether a hidden liability is recognized, acted upon, or funded depends on who can observe it, how it is interpreted, and how effectively it can be communicated upward.

Distance reinforces this effect. Labro, Lang, and Omartian[40] show that predictive analytics can convert dispersed local data into more objective and quantitative inputs for higher-level decision making, while also altering the centralization of authority. The analogy here is direct: the organizational distance between headquarters and plant resembles the distance between the budget-setting PMO and the engineer closest to the system. In both cases, local knowledge must be translated into a form that can travel upward, survive compression, and compete for scarce attention. Discovery therefore depends not only on what is known locally, but also on how cheaply and credibly that knowledge can be converted into decision-relevant evidence for distant budget holders.

Uncertainty further complicates the problem. Sengul et al.[74, pp. 27-30] note that under environmental uncertainty and complexity, comparison and selection rely more heavily on subjective and qualitative judgments shaped by beliefs and threat perceptions. Drawing on Bower and Gilbert, they further emphasize that the resource-based view treats heterogeneous, often non-financial assets and capabilities as sources of value creation. Maritan and Lee[47] reinforce the broader point that resource allocation is shaped not only by formal optimization, but also by how organizations interpret uncertainty, capability, and strategic importance. Taken together, these perspectives imply that organizations must convert dis-

similar opportunities and risks into a common decision scale. In practice, this is a utility mapping: a way to compare unlike items, such as technical risk, mission impact, and cost, in order to support prioritization under constraint. Discovery decisions therefore operate in a domain where incomplete information, bounded analysis, and organizational interpretation jointly determine what gets funded.

By analogy, just as the portfolio with the highest expected return is not necessarily the one with the lowest variance[48], the discovery posture that maximizes visibility is not necessarily the one that minimizes cost or loss under constraint. Discovery therefore requires budgets, not maximalism.

Taken together, timing, information control, organizational distance, and asymmetric information all point to the same conclusion. The value of discovery is time-sensitive because conditions change; what is important to know now may matter less later, while new liabilities may emerge in the meantime. Information and authority are rarely co-located: the engineer, system owner, and executive budget holder usually see different slices of reality and operate under different incentives. Distance, whether geographic, organizational, or technical, introduces delay, compression, and loss of context. Asymmetry determines who can observe a problem, who can explain it, and who can fund its investigation. These conditions do not eliminate discovery; they make discovery an allocation problem. The next question, therefore, is how to frame discovery not as an open-ended search, but as an investment decision evaluated through return and risk.

ROI and Risk Framing

> The resource allocation process (RAP) articulated by Bower, Burgelman and their colleagues tries to understand how organizations decide to commit their resources, the forces that influence such decisions, and the strategic consequences of the resource commitments made. Since its inception 43 years ago, this stream of research has significantly evolved in terms of its conceptualization of RAP...[46]

Manukonda's review is useful here because it frames resource allocation as a structured organizational process rather than as an isolated budgeting exercise. That perspective aligns closely with the present argument. Discovery is not merely a technical activity; it is a resource-allocation process in which organizations decide which uncertainties are worth paying to reduce, under what constraints, and with what expected strategic consequences.

Pringle and Van Orden[59] provide a practical bridge by showing how return-on-investment reasoning can be applied even to non-revenue-generating activities. More broadly, corporate finance supplies a formal vocabulary for a behavior organizations already display in practice: they trade off cost, uncertainty, and expected loss when allocating limited resources, whether or not those decisions are expressed through explicit models such as net present value or CAPM. In that sense, ROI logic is not foreign to discovery; it is simply one way of making the prioritization rule explicit.

Sharpe[76] provides the standard linkage between expected return and systematic risk, and thereby the basis for discount-rate reasoning in present-value

analysis. Markowitz[48], by contrast, provides the more general intuition needed here: organizations can accept greater uncertainty in exchange for lower cost, or spend more in order to reduce uncertainty. The point is not that discovery literally behaves like a securities portfolio, but that bounded discovery requires the same kind of constrained tradeoff. A search posture that maximizes visibility is not necessarily the one that minimizes expected loss or total cost under real-world constraints.

This same logic appears outside corporate finance. NIST SP 800-30 frames risk assessments as providing leaders with information needed to choose appropriate courses of action, explicitly including cost-benefit analysis among its topical emphases[35]. Gordon and Loeb[25] start from information-security economics and show that even within justifiable investments, optimal spending is bounded relative to expected loss. ENISA's ROSI guidance[20] and the FAIR model[21] make the same move operationally by expressing security decisions in financial terms and enabling comparison across budget lines. These approaches differ in notation and emphasis, but they converge on a common principle: risk and return are two views of the same allocation problem.

Graham and Harvey[27] show that executives rely heavily on practical, informal rules when making capital-allocation decisions. In the language of this paper, such rules can be interpreted as compressed projections of accumulated organizational experience rather than outputs of exhaustive optimization. This is not a deviation from rationality under constraint; it is how bounded decision systems commonly operate. Informal rules, however, leave the allocation logic implicit. Once discovery is treated as an investment decision, the question becomes operational: how should bounded resources be distributed across candidate investigations so as to maximize expected uncertainty reduction, risk reduction, or coverage gain per unit cost? That is the role of Criticality Analysis and Risk Assessment (CARA). CARA does not assume exhaustive inspection. It makes the allocation rule explicit by linking traceability, likelihood, impact, and budget constraints into a prioritization method for deciding what to inspect first.

6.2 CARA: Criticality Analysis and Risk Assessment (Cost Managed)

Criticality Analysis and Risk Assessment (CARA) is the mechanism used in this paper to convert bounded discovery into a formal selection problem. Its purpose is not to eliminate all uncertainty, but to decide which investigations should be funded first when inspection time, staffing, and execution resources are constrained.

The method grew out of earlier work on capturing and communicating known unknowns across distributed actors[61,62], and was later integrated with requirements traceability, cost allocation, and constrained project-planning methods [63,66]. In its present form, CARA extends Accurate Project Management[65] and related cost-modeling approaches by making the discovery problem explicit: different candidate investigations have different expected value, different cost,

and different traceability reach, and these differences must be compared under budget.

CARA operates in two levels. At the requirement level, it assigns a priority score to each requirement using a small number of operationally meaningful factors. At the test level, it projects those requirement priorities through the requirements traceability matrix and selects a feasible investigation plan under resource constraints.

Let

$$r \in \mathcal{R} \tag{9}$$

denote a requirement, and let

$$v \in \mathcal{V} \tag{10}$$

denote a verification or investigative test case[7]. The set $\mathcal{R}$ contains all requirements relevant to the system under analysis, and $\mathcal{V}$ contains all candidate test cases available for inspection or execution.

Requirement-Level Priority CARA begins by assigning each requirement a base priority. Let $K(r)$ denote the criticality of requirement r, $\lambda_r(r)$ the requirement-level likelihood that the requirement will be violated or implicated in failure, $m(r)$ the impact associated with such failure, and $D(r)$ a documentation-quality penalty, with larger values reflecting poorer supporting documentation and therefore greater uncertainty.

The resulting requirement-level CARA priority is

$$P_r(r) = K(r)\,\lambda_r(r)\,m(r)\,D(r) \tag{11}.$$

This score is intentionally simple. It is not intended to model all aspects of risk exhaustively. It provides a practical first-pass priority that reflects how important the requirement is, how likely it is to be implicated in failure, how costly such failure would be, and how much uncertainty is introduced by weak documentation.

Traceability Mapping CARA then projects requirement priority into the space of executable investigations through the requirements traceability matrix.

Let

$$\mathcal{V}(r) \subseteq \mathcal{V} \tag{12}$$

be the set of test cases that validate requirement r, and let

$$\mathcal{R}(v) \subseteq \mathcal{R} \tag{13}$$

be the inverse mapping, that is, the set of requirements exercised by test case v.

[7] In [66], $t \in \mathcal{T}$ was used. In this paper, this has been replaced with $v \in \mathcal{V}$ to avoid confusion with $\mathcal{T}_{\alpha_i}$ in Eq. ((5),).

This inverse mapping is important because a test designed for one requirement often exercises more than that requirement alone. Formally,

$$\{r\} \subseteq \bigcup_{v_r \in \mathcal{V}(r)} \mathcal{R}(v_r) \tag{14},$$

and in practical systems the containment is often strict. This is the traceability fan-out effect: tests provide broader coverage than their immediate target suggests.

That property is useful under constraint. If a single test case covers multiple important requirements, then its effective value is higher than a one-to-one traceability interpretation would imply.

Test-Level Impact, Likelihood, and Risk Given the traceability mappings, CARA defines the impact of a test case in terms of the requirements it exercises. One admissible impact function is

$$I(v) = \prod_{r \in \mathcal{R}(v)} P_r(r) \tag{15}.$$

This multiplicative form is not meant to express probabilistic independence. Its purpose is to amplify test cases that simultaneously touch multiple high-priority requirements. In that sense, it is a scoring function for compound significance rather than a probabilistic model of joint failure.

Let $L(v)$ denote the likelihood that test case v will expose a meaningful failure, hidden liability, or nonconformance. This likelihood may be estimated from historical defect patterns, implementation novelty, change intensity, architectural fragility, prior test results, expert judgment, or other APM-derived project evidence.

Test-level risk is then defined in the usual way, product of likelihood and impact, as

$$\rho(v) = L(v) \cdot I(v) \tag{16}.$$

A corresponding test priority follows:

$$P_v(v) \propto \rho(v) \tag{17}.$$

Higher-risk tests are therefore assigned higher priority, not because they are more likely to fail alone, but because they combine higher failure likelihood with greater consequence if failure is exposed.

Universal Candidate Set The universal candidate set of test cases is the union of all test cases associated with all requirements:

$$\mathcal{V} = \bigcup_{r \in \mathcal{R}} \mathcal{V}(r) \tag{18}.$$

This is the full space from which a feasible discovery plan may be selected. In principle, the organization could attempt to evaluate or execute all such candidates. In practice, this is exactly what resource limits prevent.

Cost Function and Budget Constraint Let $C(v)$ denote the cost of executing test case v. This cost may incorporate analyst time, environment setup, tool execution, review effort, test data generation, coordination overhead, or other discovery-related expenses.

Let B denote the bounded discovery budget available for the current planning period.

A feasible test plan is then a subset

$$\mathcal{V}_P \subseteq \mathcal{V} \tag{19}$$

such that

$$\sum_{v \in \mathcal{V}_P} C(v) \leq B \tag{20}.$$

This is analogous to "The return... is a weighted sum... where the investor can choose the weights." $E = \sum_{i=1}^{N} X_i \mu_i$ such that X_i be the relative amount invested[48]. The test plan is therefore not a wish list or a complete inventory. It is the subset of candidate investigations that can be justified and funded under explicit resource constraints.

Budget-Constrained Selection The ideal CARA objective is to maximize covered risk subject to the budget constraint:

$$\mathcal{V}_P^* = \arg\max_{\mathcal{S} \subseteq \mathcal{V}} \sum_{v \in \mathcal{S}} \rho(v) \quad \text{subject to} \quad \sum_{v \in \mathcal{S}} C(v) \leq B \tag{22}.$$

This is a knapsack-style optimization over risk-weighted investigations. Stated informally, the organization seeks the collection of tests that yields the greatest expected discovery value without exceeding the budget.

The cardinality $|\mathcal{R}(v)|$ captures the structural fan-out of a test case, that is, the number of requirements exercised by that test. However, because requirements differ in importance, this measure is extended to a priority-weighted form. Let

$$W(v) = \sum_{r \in \mathcal{R}(v)} P_r(r) \tag{23},$$

which represents the aggregate priority of the requirements covered by v. This weighted formulation captures the fact that some tests provide disproportionately higher value by exercising multiple high-priority requirements.

The efficiency of a test case may then be expressed as

$$\text{efficiency}(v) = \frac{W(v)}{C(v)} \tag{24},$$

allowing comparison of candidate investigations in terms of expected value per unit cost under bounded resources.

The point is not to discover everything. The point is to spend scarce discovery resources where they are expected to reveal the most consequential hidden liabilities.

Budget-Constrained CARA Selection with Investigation Limit Let $c_{\text{eval}}(v)$ denote the cost of evaluating or analyzing a candidate test case v prior to execution. This cost may be zero in environments where evaluation is negligible, or non-zero where analysis, setup, or coordination effort is required.

The parameter $c_{\text{eval}}(v)$ acts as a tunable control on discovery depth, allowing organizations to model environments ranging from near-zero marginal evaluation cost to scenarios where analysis itself is a dominant resource constraint.

Input: Candidate set $\mathcal{V}$, execution budget B, investigation budget B_I
Initialize:
$\mathcal{V}_P \leftarrow \emptyset$
$C_{\text{total}} \leftarrow 0$
$C_{\text{invest}} \leftarrow 0$
Sort $\mathcal{V}$ in descending order of $\frac{W(v)}{C(v)}$
for each $v \in \mathcal{V}$ **do**
    $C_{\text{invest}} \leftarrow C_{\text{invest}} + c_{\text{eval}}(v)$
    if $C_{\text{invest}} > B_I$ **then**
        break
    end if
    if $C_{\text{total}} + C(v) \leq B$ **then**
        $\mathcal{V}_P \leftarrow \mathcal{V}_P \cup \{v\}$
        $C_{\text{total}} \leftarrow C_{\text{total}} + C(v)$
    end if
end for
return $\mathcal{V}_P$

Practical Interpretation In practice, CARA provides a disciplined way to answer a question that organizations usually handle implicitly: which hidden liabilities are worth paying to make visible first? Requirement-level priority captures mission and governance significance. Traceability propagates that significance into the test space. Likelihood and impact convert that propagated significance into a risk-based ranking. The budget constraint then forces the organization to confront what can actually be inspected within the available resources.

That structure is what makes CARA appropriate for shadow technical debt. Shadow liabilities are not discovered by exhaustive search; they are surfaced through bounded, prioritized, traceability-aware investigation. CARA does not deny uncertainty. It prices and allocates around it.

The next section applies this logic directly to the discovery of shadow technical debt by treating bounded inspections as the mechanism by which hidden liabilities are converted into logged technical debt.

6.3 Applying CARA to Discover Shadow Technical Debt

CARA may be applied directly to changes rather than only to verification tests. For present purposes, let $c \in \mathcal{C}$ denote a change introduced into the system, its configuration, or its governing artifacts. Under this interpretation, every change

is treated as requirement-connected in principle: it either corrects implementation relative to an existing requirement, or it corrects the requirement set itself, which in practice appears as an enhancement. This makes changes useful probes for shadow technical debt because they identify where understanding, implementation, and planning have diverged.

Let

$$\mathcal{C}(r) \subseteq \mathcal{C} \tag{25}$$

denote the set of changes associated with requirement r, and let

$$\mathcal{R}(c) \subseteq \mathcal{R} \tag{26}$$

denote the inverse mapping, that is, the set of requirements implicated by change c. As with test traceability, the mapping is frequently one-to-many: a single change may fan out across multiple requirements. This inverse mapping also exposes an important exception. Academically, all changes should trace to requirements. In practice, some do not. When $\mathcal{R}(c) = \varnothing$, the change should not be treated as a harmless zero-impact case. The absence of documented requirement traceability is itself evidence of technical debt and should be red-flagged as an exception requiring review. This prevents undocumented changes from disappearing from the prioritization model merely because no requirement linkage was recorded.

The resulting change and requirement-level CARA priority is

$$P_r(r, c) = K(r)\, \lambda_c(c)\, m(r)\, D(c) \tag{27}.$$

The change-level impact function is then defined over the requirement fan-out of the change. One admissible form is

$$I(c) = \prod_{r \in \mathcal{R}(c)} P_r(r, c) \tag{28}$$

where $K(r)$ remains the criticality of requirement r, $m(r)$ remains the impact associated with failure of that requirement, $\lambda_c(c)$ is the likelihood that change c carries or introduces hidden liability, and $D(c)$ is a change-management penalty associated with the way the change moved through the schedule and governance process.

The interpretation of $D(c)$ is important. In this application, $D(c)$ is not merely haste in the colloquial sense, but a measure of scheduling turbulence or rate-of-change-of-schedule. It is minimized when the change proceeds substantially as forecast through the ordinary backlog, review, and execution cadence. It increases when work is accelerated into an earlier work period, compressed far below its expected duration, expanded unexpectedly, or inserted and released outside the normal planning and grooming process. In the limiting case, where the effective displacement in schedule is discontinuous relative to the ordinary management cadence, $D(c)$ approaches its maximum. The practical intuition is that abrupt schedule distortion is often a proxy for the kinds of shortcuts, omissions, and weakly governed decisions that create shadow technical debt.

A second change-level modifier is temporal recency. Let $\delta_c(c)$ denote a monotone measure of the newness of change c, with larger values assigned to newer changes and smaller values assigned to older changes. No particular decay law is required here; only the ordering matters. The newest changes are given greater emphasis because they are more likely to contain newly introduced shadow technical debt that has not yet been discovered, while older changes are less likely to represent newly emerging hidden liability in the present planning period.

A corresponding change-level risk score may therefore be written as

$$\rho(c) = \delta_c(c) \cdot I(c) \tag{29}$$

with priority assigned in the usual way:

$$P_c(c) \propto \rho(c) \tag{30}.$$

This formulation gives highest priority to recent changes that touch important requirements and that were introduced under conditions suggesting elevated hidden-liability risk.

The priority-weighted coverage formulation remains structurally the same, with changes substituted for tests:

$$W(c) = \sum_{r \in \mathcal{R}(c)} P_r(r, c) \tag{31}.$$

This preserves the earlier idea that a single candidate may derive additional value by touching multiple important requirements. What changes in the present application is the cost structure. In discovery-oriented use, the direct cost of the change itself is often negligible for prioritization purposes, while the cost of interrogating the change for shadow technical debt is not[29]. Many of the relevant tools, techniques, and procedures, such as static code analysis, structural differencing, dependency inspection, or comparable review mechanisms, are effectively quantized at the level of the change: their operational runtime may vary somewhat, but in practice they are treated as fixed-price evaluation steps.

For that reason, the efficiency term should explicitly include evaluation cost:

$$\text{efficiency}(c) = \frac{W(c)}{C(c) + c_{\text{eval}}(c)} \tag{32}$$

where $C(c)$ is the direct cost attributable to change c, and $c_{\text{eval}}(c)$ is the cost of evaluating or interrogating that change for hidden liability. In many discovery settings, $C(c) \approx 0$, while $c_{\text{eval}}(c)$ is positive and approximately constant after quantization. This term is therefore no longer negligible, as it often was in the earlier testing context. It must appear explicitly in the denominator both to reflect the actual economics of discovery and to avoid degenerate zero-cost cases. Where $c_{\text{eval}}(c)$ is approximately constant across changes, the ranking collapses toward weighted coverage, but the constant still matters because it determines how many changes can be inspected within a very small discovery budget.

This point becomes especially important when discovery funding is intentionally small relative to the change budget. If only a very small fraction of overall change resources is allocated to finding shadow technical debt, then fixed evaluation costs dominate the economics of discovery. Under those conditions, the practical selection problem is not whether the organization can inspect everything, but which changes are most worth interrogating first. The budget-constrained CARA logic from the previous section therefore applies with $\mathcal{C}$ substituted for $\mathcal{V}$, and with the investigation term $c_{\text{eval}}(c)$ treated as a first-order cost driver rather than a negligible one.

The specific mechanisms used to surface hidden liabilities remain domain-dependent and are therefore not prescribed here. Static analysis, change-history review, dependency inspection, audit, telemetry, and other methods may all be appropriate depending on the system and operating context.

CARA operates one level above the underlying discovery tools. Its role is not to prescribe how candidate observations are generated, but to prioritize them once they exist. The mechanisms used to surface hidden liabilities are inherently domain-dependent, including static analysis, change-history review, dependency inspection, audit, and telemetry. These techniques produce candidate signals; CARA determines which of those signals should be investigated first under explicit resource constraints.

In this way, CARA converts a collection of potential observations into a ranked discovery plan, ensuring that a bounded effort yields the greatest possible conversion of shadow technical debt into visible, governed technical debt.

7 Conclusion

Once a hidden liability is surfaced through bounded discovery, it moves from an unobserved condition to a governed artifact. In practical terms, this means incorporation into formal tracking mechanisms such as defect logs, risk registers, or technical-debt backlogs. The important distinction is not the specific tooling used, but the state change itself: the liability becomes visible, accountable, and subject to prioritization within ordinary resource-allocation processes. CARA thus functions as the bridge between latent conditions and governed technical debt.

The central conclusion of this paper is that shadow technical debt persists whenever organizations neglect either of two core principles. If they fail to name the phenomenon, they remain trapped in ambiguity, avoidance, and underfunding. The liabilities remain real, but they do not become discussable enough to attract resources. If they fail to use cost-managed approaches, they may acknowledge the problem in principle while leaving it untouched in practice, because remediation appears open-ended and unaffordable.

Naming and cost management therefore operate together. Naming converts an unnamed threat into a recognizable class of liability and creates a basis for action. Cost-managed discovery converts that class from an abstract concern

into a bounded and predictable program of work. Together, these steps move an organization from shadow technical debt to known technical debt.

The practical importance of this conclusion is amplified by current economics and tooling. What could not previously be inspected at acceptable cost can increasingly be examined at lower cost points. This does not eliminate the underlying debt, but it does change the economics of discovering it. Cheaper discovery methods reduce the justification for leaving technical liabilities in shadow form. This is the opening modern engineering organizations should exploit.

References

1. Allman, E.: Managing technical debt. ACM Queue **10**(3) (2012).
2. Bartel, B.B., Kello, C.T., Lupyan, G.: The emergence of meaning through neural network models of natural language. Topics in Cognitive Science **12**(4) (2020).
3. Bellinger, C., Crowley, M., Tamblyn, I.: Dynamic observation policies in observation cost-sensitive reinforcement learning. In: Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@NeurIPS 2023) (2023), https://openreview.net/forum?id=3GL2GETaaL
4. Bellinger, C., Drozdyuk, A., Crowley, M., Tamblyn, I.: Scientific discovery and the cost of measurement – balancing information and cost in reinforcement learning (2022).
5. Bellinger, C., Drozdyuk, A., Crowley, M., Tamblyn, I.: Scientific discovery and the cost of measurement – balancing information and cost in reinforcement learning (2022), arXiv:2112.07535, extended version of [4]
6. Bernhardsson, E.: Building a data team at a mid-stage startup: A short story., https://erikbern.com/2021/07/07/the-data-team-a-short-story.html
7. Betz, H.D., editor: The Greek Magical Papyri in Translation. University of Chicago Press, Chicago, 2nd edn. (1992), ISBN 0-226-04444-0
8. Brown, N., Cai, Y., Guo, Y., Kazman, R., Kim, M., Kruchten, P., Lim, E., MacCormack, A., Nord, R., Ozkaya, I., Sangwan, R., Seaman, C., Sullivan, K., Zazworka, N.: Managing technical debt in software-reliant systems. In: Workshop on Future of Software Engineering Research, FoSER 2010, at the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering. Santa Fe, NM (November 2010).
9. Budge, E.A.W.: Legends of the Gods: The Egyptian Texts, Edited with Translations. Kegan Paul, Trench, Trübner & Co., Ltd., London (1912)
10. Chat GPT (Codex): Wire fm:cad soap lookups to db (git commit). In: Internal Git repository. vol. 9faa78d7f71e0349cc8f61949291fe71ee1b884d. [redacted]
11. Chrissis, M.B., Konrad, M., Shrum, S.: Capability maturity model integration for development (CMMI-DEV), version 1.2. Tech. Rep. CMU/SEI-2006-TR-008, Software Engineering Institute, Pittsburgh, PA (August 2006)
12. Cialdini, R.: Influence: The Psychology of Persuasion (1984)
13. 孫子 (Sun Tzu): 孫子兵法: Sun Tzŭ on the Art of War: The Oldest Military Treatise in the World. Luzac & Co., London (1910)
14. 毛泽东 (Mao, Tse-tung): Fleet Marine Force Reference Publication 12-18: Mao Tse-tung on Guerrilla Warfare. U.S. Marine Corps, Quantico, Virginia (1989)
15. Conitzer, V.: Metareasoning as a Formal Computational Problem. The MIT Press (2011).

16. Cunningham, W.: The WyCash portfolio management system. In: Addendum to the Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '92). Vancouver, British Columbia, Canada (October 1992).
17. De Almeida, R.R., Treude, C., Kulesza, U.: What's behind tight deadlines? business causes of technical debt. In: 2023 IEEE/ACM 16th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE). pp. 25–30 (2023).
18. Dempsey, K., Chawla, N.S., Johnson, A., Johnston, R., Jones, A.C., Orebaugh, A., Scholl, M., Stine, K.: NIST SP 800-137: Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations. Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD (Sep 2011).
19. Engineering, G.: Postmortem of database outage of january 31 (2017), https://about.gitlab.com/blog/postmortem-of-database-outage-of-january-31/
20. European Network and Information Security Agency: Introduction to Return on Security Investment: Helping CERTs assessing the cost of (lack of) security. Tech. rep., European Network and Information Security Agency, Heraklion, Greece (Dec 2012)
21. FAIR Institute: Factor Analysis of Information Risk (FAIR) Model, Version 3.0. Tech. rep., FAIR Institute (Jan 2025)
22. Fowler, M.: Technical debt quadrant (2009), https://martinfowler.com/bliki/TechnicalDebtQuadrant.html
23. Gavetti, G., Greve, H.R., Levinthal, D.A., Ocasio, W.: The behavioral theory of the firm: Assessment and prospects. Academy of Management Annals **6**(1), 1–40 (2012), https://doi.org/10.5465/19416520.2012.656841
24. Gerkey, B.P., Matarić, M.J.: A formal analysis and taxonomy of task allocation in multi-robot systems. Intl. J. of Robotics Research **23**(9) (2004).
25. Gordon, L.A., Loeb, M.P.: The economics of information security investment. ACM Trans. Inf. Syst. Secur. **5**(4), 438–457 (Nov 2002).
26. Gosselin, C.: What is technical debt and how to manage it (2021), https://www.andplus.com/blog/what-is-technical-debt-and-how-to-manage-it
27. Graham, J.R., Harvey, C.R.: The theory and practice of corporate finance: evidence from the field. Journal of Financial Economics **60**(2-3) (2001).
28. Gu, P., Maddox, A.B.: A framework for distributed reinforcement learning. In: Weiß, G., Sen, S. (eds.) Adaption and Learning in Multi-Agent Systems. pp. 97–112. Springer Berlin Heidelberg, Berlin, Heidelberg (1996).
29. Guo, Y., Spínola, R.O., Seaman, C.: Exploring the costs of technical debt management – a case study. Empir Software Eng **21** (2016).
30. Hoiles, W., Krishnamurthy, V., Pattanayak, K.: Rationally inattentive inverse reinforcement learning explains youtube commenting behavior. Journal of Machine Learning Research **21**(170) (2020), http://jmlr.org/papers/v21/19-872.html
31. Hsu, M., Bhatt, M., Adolphs, R., Tranel, D., Camerer, C.F.: Neural systems responding to degrees of uncertainty in human decision-making. Science **310**(5754) (2005).
32. Ivy-Rosser, L., Nguyen, A.: Key questions for tackling technical debt (2025), https://www.forrester.com/blogs/key-questions-for-tackling-technical-debt/
33. Izurieta, C., Rice, D., Kimball, K., Valentien, T.: A position study to investigate technical debt associated with security weaknesses. In: TechDebt'18 (International Workshop on Technical Debt). Gothenburg, Sweden (2018).
34. Jaspan, C., Green, C.: Defining, measuring, and managing technical debt. IEEE Software **40**(3) (2023).

35. Joint Task Force Transformation Initiative: NIST SP 800-30 Rev. 1: Guide for Conducting Risk Assessments. Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD (Sep 2012).
36. Kar, S., Moura, J.M.F., Poor, H.V.: Qd-learning: A collaborative distributed strategy for multi-agent reinforcement learning through consensus+innovations. IEEE Transactions on Signal Processing **61**(7) (2013).
37. Klein, B., Crawford, R.G., Alchian, A.A.: Vertical integration, appropriable rents, and the competitive contracting process. Journal of Law and Economics **21**(2) (1978).
38. Kong, D., Indelman, V.: Simplified POMDP planning with an alternative observation space and formal performance guarantees. In: International Symposium of Robotics Research 2024. Long Beach, California (2024).
39. Kruchten, P., Nord, R.L., Ozkaya, I.: Technical debt: From metaphor to theory and practice. IEEE Software **29**(6), 18–21 (2012).
40. Labro, E., Lang, M., Omartian, J.D.: Predictive analytics and centralization of authority. Journal of Accounting and Economics **75**(1) (February 2023).
41. Lenarduzzi, V., Orava, T., Saarimäki, N., Systä, K., Taibi, D.: An empirical study on technical debt in a finnish sme. In: International Symposium on Empirical Software Engineering and Measurement. Brazil (2019).
42. Lieberman, M.D., Eisenberger, N.I., Crockett, M.J., Tom, S.M., Pfeifer, J.H., Way, B.M.: Putting feelings into words: Affect labeling disrupts amygdala. Psychological Science **18**(5) (May 2007).
43. Lieder, F., Plunkett, D., Hamrick, J.B., Russell, S.J., Hay, N.J., Griffiths, T.L.: Algorithm selection by rational metareasoning as a model of human strategy selection. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. (eds.) Advances in Neural Information Processing Systems. vol. 27. Curran Associates, Inc. (2014), https://proceedings.neurips.cc/paper_files/paper/2014/file/87b5e7e570f757a0b99f0a370ce2438c-Paper.pdf
44. Liu, X., Zhang, K.: Partially observable multiagent reinforcement learning with information sharing. SIAM Journal on Control and Optimization **64**(2), 673–697 (2026).
45. Machiavelli, N.: Art of War. University of Chicago Press, Chicago and London (2003)
46. Manukonda, S.: Resource Allocation Process: Contributions, Synthesis and Future Directions. Tech. Rep. Working Paper No. 415, Indian Institute of Management Bangalore (2013), https://www.iimb.ac.in/sites/default/files/2018-08/WPNo.415.pdf
47. Maritan, C.A., Lee, G.K.: Resource allocation and strategy. Journal of Management **43**(8) (November 2017).
48. Markowitz, H.: Portfolio selection. The Journal of Finance **7**(1), 77–91 (1952).
49. Matějka, F., McKay, A.: Rational inattention to discrete choices: A new foundation for the multinomial logit model. American Economic Review **105**(1), 272–98 (January 2015).
50. Maćkowiak, B., Matějka, F., Wiederholt, M.: Rational inattention: a review. Working Paper Series 2570, European Central Bank (Jun 2021), https://ideas.repec.org/p/ecb/ecbwps/20212570.html
51. McConnell, S.: Managing technical debt. Tech. rep. (June 2008)
52. Morgan, M.A., translator: Sepher Ha-Razim: The Book of the Mysteries. Scholars Press, Chico, CA (1983)

53. Nestle, E.: Novum Testamentum Graece cum apparatu critico ex editionibus et libris manu scriptis collecto. Stuttgart: Privilegierte Württembergische Bibelanstalt, secunda edn. (1899)
54. Ocasio, W.: Towards an attention-based view of the firm. Strategic Management Journal **18**(Summer 1997 Special Issue: Organizational and Competitive Interactions) (July 1997).
55. Ochsner, K.N., Bunge, S.A., Gross, J.J., Gabrieli, J.D.E.: Rethinking feelings: An fMRI study of the cognitive regulation of emotion. Cognitive Neuroscience **14**(8) (Nov 2002).
56. Office of Management and Budget: OMB Circular No. A-130: Managing Information as a Strategic Resource. Tech. rep., Office of Management and Budget, Washington, DC (Jul 2016), https://obamawhitehouse.archives.gov/sites/default/files/omb/assets/OMB/circulars/a130/a130revised.pdf
57. Omidshafiei, S., Pazis, J., Amato, C., How, J.P., Vian, J.: Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 2681–2690. PMLR, Sydney, Australia (06–11 Aug 2017).
58. Phelps, E.A., Delgado, M.R., Nearing, K.I., LeDoux, J.E.: Extinction learning in humans: Role of the amygdala and vmpfc. Neuron **43**(6) (2004).
59. Pringle, S., VanOrden, M.A.: Asset Pricing Model to DoD's Information Technology Investments. Master's thesis, Naval Postgraduate School, Monterey, CA (Mar 2009), aDA497225
60. [redacted]SCRM & reverse engineering of COTS: Finding undisclosed vulnerabilities and patching them (CUI). In: Fall 2025 Techex DC3 DIB Cybersecurity. Department of Defense Cyber Crime Center, Linthicum Heights, MD
61. [redacted]Research notes on multi-agent reinforcement learning architectures and inter-agent state communication (1998–2002)
62. [redacted]Dissimilar multi-agent q-learning adaptation with teacher-student knowledge sharing, unbounded state lookback, memory compaction, and n-1 dimension q value projection (2002)
63. [redacted]Requirements traceability CONOPS. Tech. rep., U.S. Department of Defense Office of the Chief Management Officer, Washington DC (2017)
64. [redacted]: Big clean up (git commit). In: Internal Git repository. vol. df3ff7cc744a3c7e1cde4ad13b2c88e491186a29. [redacted] (2026)
65. [redacted]: Accurate project management, version 3.1b. Tech. rep., [redacted] (August 7, 2019)
66. [redacted]: Department of veterans affairs enterprise testing service support services proposal. Tech. rep., [redacted] (October 18, 2019)
67. Robbins, L.: An essay on the nature and significance of economic science. Macmillan and Co., Limited, London, 2nd ed., revised and extended edn. (1945)
68. Russell, S.J.: Metareasoning. Tech. rep., Computer Science Division, University of California, Berkeley (Nov 1997)
69. Russo, D.: How to become a software engineer: Lessons from mid-career professionals (2025), https://www.bairesdev.com/blog/become-a-software-engineer/
70. Sabbata, C.N.D., Sumers, T.R., AlKhamissi, B., Bosselut, A., Griffiths, T.L.: Rational metareasoning for large language models (2025), https://arxiv.org/abs/2410.05563

71. Schelling, T.C.: The strategy of conflict prospectus for a reorientation of game theory. The Journal of Conflict Resolution **2**(3) (1958)
72. Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M.: Machine learning: The high interest credit card of technical debt. In: SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop) (2014)
73. Securities and Exchange Commission: In the matter of knight capital americas llc (Oct 2013), release No. 70694, Administrative Proceeding File No. 3-15570
74. Sengul, M., Costa, A.A., Gimeno, J.: The allocation of capital within firms. Academy of Management Annals **13**(1) (2019).
75. ServiceNow: Leadership trends report: Global app development: A roadmap from building and automating to celebrating. Tech. rep. (2023), https://resources.itsecuritywire.com/ebook/ServiceNow-CRWF-EN-3.pdf
76. Sharpe, W.F.: Capital asset prices: A theory of market equilibrium under conditions of risk. The Journal of Finance **19**(3) (1964)
77. Shi, C., Dorothy, M.R., Fu, J.: Integrated control and active perception in pomdps for temporal logic tasks and information acquisition (2025)
78. Siavvas, M., Tsoukalas, D., Jankovic, M., Kehagias, D., Chatzigeorgiou, A., Tzovaras, D., Anicic, N., Gelenbe, E.: An empirical evaluation of the relationship between technical debt and software security. In: International Conference on Information Systems and Technologies (ICIST 2019) (2019)
79. Siegel, D.J., Bryson, T.P.: The Whole-Brain Child: 12 Revolutionary Strategies to Nurture Your Child's Developing Mind. Bantam Books, New York, NY (2011)
80. Sims, C.A.: Implications of rational inattention **50**(3), 665–690 (2003).
81. Syrigos, E., Kostopoulos, K.C., Mammassis, C.S.: Multiple corporate and functional performance feedback and problemistic search. European Management Review **22**(1740-4754) (2023).
82. Szukits, Á., Móricz, P.: Towards data-driven decision making: the role of analytical culture and centralization efforts. Review of Managerial Science **18**(2849–2887) (2024).
83. Taft, D.K.: Jetbrains names the debt ai agents leave behind (2026), https://thenewstack.io/jetbrains-names-the-debt-ai-agents-leave-behind
84. U.S. House of Representatives, Committee on Oversight and Government Reform: The equifax data breach. majority staff report. Tech. rep., Washington, DC (December 2018)
85. Whalen, P.J.: Fear, vigilance, and ambiguity: Initial neuroimaging studies of the human amygdala. Current directions in psychological science : a journal of the American Psychological Society **7**(6) (1998).
86. Windward: Leadership trends report: Global app development. Tech. rep. (2024), https://windward.com/wp-content/uploads/2024/08/ServiceNow-Leadership-trends-report-global-app-development.pdf
87. Yli-Huumo, J., Maglyas, A., Smolander, K.: The sources and approaches to management of technical debt: A case study of two product lines in a middle-size finnish software company. In: Product-Focused Software Process Improvement. PROFES 2014. Lecture Notes in Computer Science. vol. 8892. Cham (2014)