American Impact Review

Theoretical Article

Estimation as Laundering: How Machine Learning Converts Software Commitments into Apparent Predictions, and What an Estimation Hygiene Framework Can Recover

Yauheni Kanavalik[1], Daria Firsova[2], Andrei Dzeikalo[3], Vadim Goncharov[4]

[1] Solutions Architect, EPAM Systems, San Francisco, California, USA

[2] Independent Senior QA Engineer & AI Testing Specialist

[3] Independent Researcher

[4] Jrnys Wellness Inc; Bachelor, Moscow Open Institute

* Corresponding author

ORCID Yauheni Kanavalik: 0009-0004-2354-4614

ORCID Andrei Dzeikalo: 0009-0009-4968-5323

ORCID Vadim Goncharov: 0009-0007-2613-0662

OPEN ACCESS

Citation:
Yauheni Kanavalik et al. (2026). Estimation as Laundering: How Machine Learning Converts Software Commitments into Apparent Predictions, and What an Estimation Hygiene Framework Can Recover. *Am. Impact Rev.*
10.66308/air.e2026041

Received: April 22, 2026
Accepted: May 4, 2026
Published: May 4, 2026

DOI:
10.66308/air.e2026041

ISSN: 3071-124X

Abstract

Software effort estimation has often been treated as if it were primarily a prediction problem. The framing was inherited from parametric cost models in the 1970s and has been carried forward, with substantial methodological refinement, into machine-learning and large-language-model estimators. The empirical record across this lineage is mixed: replication studies and honest-baseline comparisons consistently report smaller and less stable gains than headline numbers suggest. This paper develops two related claims. The diagnostic claim is that machine-learning estimators can launder what is structurally a human or team commitment into the appearance of an objective technical output, with consequences for contestability, authorship, and accountability that accuracy metrics do not capture. The constructive claim is that an estimation hygiene discipline, five practices and a single operational decision rule, retains the genuine value of machine learning as decision support without permitting it to displace human commitment. The study is a conceptual, theory-building paper. It draws on the Austin and Searle speech-act apparatus, the performativity-of-models tradition, the methodological-critique literature on machine learning in software engineering and adjacent fields, and the political economy of measured work. The paper should be read as a conceptual synthesis rather than a systematic review. Recurring failure modes in machine-learning effort estimation, including fragile baselines, leakage, target ambiguity, and the cost of estimation itself, are read here as symptoms of a deeper category error rather than as independent technical problems. Reframing the estimate as a commissive speech act, contingent on explicit felicity conditions, clarifies why methodologically careful models can still under-deliver once deployed inside real planning practices. Three structural features distinguish the software case from the canonical financial-model performativity case: the absence of arbitrage closure, the intra-firm setting, and the combinatorial novelty of software work. The paper offers a sustained conceptual critique of machine-learning software effort estimation that targets the ontology of the target rather than only the validity of the models; names and analyses a specific sociotechnical mechanism, ML-as-laundering, by which automated estimators obscure commitment under the appearance of measurement; and proposes an estimation hygiene framework, including a disagreement-flag rule, that defines a defensible role for machine learning as decision support alongside human commitment.

Keywords: software effort estimation, story points, machine learning, performativity, speech acts, algorithmic management, decision support, agile software development

1. Introduction

Software effort estimation has a longer track record of mixed results than almost any other measurement practice in software engineering. Substantial schedule and effort overruns have remained a recurring feature of the empirical literature for decades, persisting through the transitions from Function Points (Albrecht, 1979) and COCOMO (Boehm, 1981) to story-point estimation in agile teams (Cohn, 2005), through classical machine-learning baselines (Wen et al., 2012; Idri, Hosni, & Abran, 2016), the deep-learning Deep-SE family (Choetkiertikul et al., 2019), transformer estimators such as GPT2SP (Fu & Tantithamthavorn, 2023), and more recently retrieval-augmented and few-shot estimators built on large language models (Tawosi, Sarro, & Petrozziello, 2023; Radliński & Swacha, 2025). Each successive wave has often been presented as a methodological improvement; in aggregate, however, the underlying error rate has not improved as dramatically as headline reports suggest. A 2023 replication study found that the canonical Deep-SE model significantly outperformed a trivial median baseline in only a small fraction of project comparisons (Tawosi, Moussa, & Sarro, 2023), and the same pattern of fragile baselines has been documented in machine-learning-for-science more broadly (Kapoor & Narayanan, 2023; Liu, Gao, Xia, Lo, Grundy, & Yang, 2021).

We argue that this persistence is not only a methodological problem awaiting a better algorithm; it is, in important part, a problem of how the object of estimation is conceived. An estimate, as it is uttered inside a real planning practice, is rarely just a prediction. Once accepted into a sprint commitment, a delivery promise, or a capacity allocation, it functions as a *performative speech act*, a commitment that organizes the very labor it appears to forecast (Austin, 1962; Searle, 1969). Treating estimates as if they were primarily predictive, and training machine-learning systems to predict them better, can *launder* the commitment as a measurement and obscure the work the utterance is doing in a planning ceremony, a sprint review, or an executive dashboard. Once that laundering effect is named, the persistence of estimation error across four methodological generations becomes less surprising: it is what one expects when a commissive utterance is consistently treated as a constative one.

The argument of this paper proceeds in three movements. The first reconstructs how the prediction frame was assembled across the 1970s and 1980s and shows how it survives inside contemporary ML estimators through dataset design, evaluation protocol, and benchmark culture. The second develops the alternative: estimation as a performative speech act, with explicit felicity conditions, distinguished from MacKenzie's (2006) account of financial-model performativity by three structural features that make the software case qualitatively different, the absence of arbitrage closure, the intra-firm rather than market setting, and the combinatorial novelty of software work. The third asks what role remains for ML once estimation is no longer treated as a prediction problem alone, and proposes a discipline of practice, what we will call *estimation hygiene*, to keep that role from collapsing back into the frame it leaves behind. The corresponding guiding questions are:

- **GQ1.** What structural confusions are reproduced when software effort estimation is framed primarily as a prediction problem, and how do they manifest in machine-learning estimators?
- **GQ2.** What is gained by reframing estimation, in its planning-relevant sense, as a performative speech act rather than as a measurement-and-prediction problem alone?
- **GQ3.** What role remains for machine learning under the reframed view, and what discipline of practice can prevent ML-based estimators from re-collapsing into the prediction frame?

Contribution relative to prior work. Prior critiques of machine-learning software effort estimation, including the strong methodological tradition exemplified by Tantithamthavorn et al. (2017, 2020) and the replication

work of Tawosi, Moussa, and Sarro (2023), have largely targeted *model evaluation*: dataset construction, baselines, leakage, and statistical comparisons. The present paper is complementary to that tradition rather than competitive with it, and differs from it in three ways. First, it engages the *ontology of the target* (what is being estimated, and what kind of utterance an estimate is) rather than the validity of the predictor alone. Second, it distinguishes the software-estimation case from the canonical financial-model performativity literature (MacKenzie, 2006) on three structural grounds: the absence of arbitrage closure, the intra-firm setting, and combinatorial novelty. Third, the paper's constructive contribution is a deployment discipline (estimation hygiene) rather than a new estimator, identifying conditions under which machine learning can usefully support, rather than displace, human commitment.

The remainder of the paper proceeds as follows. Section 2 describes the conceptual methodology and the literature base. Section 3 reconstructs the genealogy of estimation from Function Points through the LLM era and locates three structural confusions in the founding frame. Section 4 reviews the methodological critique of ML-based estimation (replication, leakage, target ambiguity, and the cost of estimation itself) and reads each failure mode as a symptom of the deeper framing rather than its cause. Section 5 develops the speech-act reframing and the MacKenzie differentiation. Section 6 names the laundering mechanism and locates the political economy of measured work in which it operates. Section 7 specifies the estimation hygiene framework. Section 8 engages three counter-arguments and identifies boundary conditions. Section 9 concludes.

2. Method

This study is a conceptual, theory-building paper (Jaakkola, 2020; MacInnis, 2011) that develops a theoretical reframing of software effort estimation rather than reporting new empirical results. The paper makes two claims. The *theoretical* claim is that software effort estimation, in its planning-relevant sense, has often been treated as if it were primarily a prediction problem. The *constructive* claim is that an alternative framing yields a defensible role for machine learning as decision support. Neither claim is reducible to a single quantitative test. The paper therefore does not present new empirical data; instead it draws on the published empirical record (replication studies, systematic reviews, and large-scale benchmarks) to substantiate its descriptive premises about the current state of the field. It should be read as a conceptual synthesis rather than as a systematic review or meta-analysis, and it does not claim exhaustive coverage of the machine-learning software effort estimation literature.

Literature was identified through three complementary procedures intended to assemble conceptual anchors, empirical premises, and methodological critiques rather than to produce an exhaustive corpus. Searches were conducted in IEEE Xplore, ACM Digital Library, Scopus, and Google Scholar using term combinations that paired *(software effort estimation OR story point estimation OR agile estimation)* with *(machine learning OR deep learning OR transformer OR large language model)* and *(replication OR calibration OR baseline OR performativity OR critique)*. Searches prioritized peer-reviewed articles, conference proceedings, and indexed monographs. Citations were then traced backward from canonical works in software effort estimation (Boehm, 1981; Cohn, 2005; Choetkiertikul et al., 2019; Tawosi et al., 2023) and from canonical works in the philosophy and sociology of measurement (Austin, 1962; Searle, 1969, 1995; MacKenzie, 2006; Bowker & Star, 1999). The literature on the political economy of measured work (Braverman, 1974; Espeland & Sauder, 2007; Moore, 2017) was incorporated to ground the analysis of Section 6.

The retained sources were organized analytically along three dimensions: the *level of analysis* (technical method, organizational practice, sociotechnical theory, normative claim); the *kind of evidence offered* (empirical, methodological-critical, conceptual, normative); and the *position taken on the prediction frame* (committed to it, agnostic, critical of it). This organization allowed the argument to be assembled around

explicit oppositions across literatures rather than treating disparate traditions as commensurable. No formal inter-rater coding, PRISMA protocol, or quantitative synthesis was undertaken; the procedure should be read as the analytical scaffolding of a conceptual paper rather than as a systematic review.

3. The Genealogy of the Prediction Frame

The contemporary framing of software effort estimation as cardinal prediction begins with Albrecht's (1979) Function Point Analysis. Albrecht proposed that the *function* delivered by a software system (measured by external inputs, outputs, inquiries, and interfaces) could serve as an input-language-independent proxy for effort. Total Function Points were computed as a weighted sum,

$$\text{FP} = \text{UFC} \cdot \Big(0.65 + 0.01 \cdot \sum_{i=1}^{14} F_i\Big),$$

where UFC is the unadjusted function count and F_i are fourteen Technical Complexity Factors rated on a 0 - 5 scale. The model produced a single cardinal number from a structured elicitation. This was a methodological achievement, but it embedded a frame: that the future cost of a software project is a *measurable quantity* that can in principle be estimated from observable features of the specification. Boehm's (1981) Constructive Cost Model formalized the same commitment in a more elaborate functional form, expressing effort in person-months as

$$\text{Effort} = a \cdot (\text{KLOC})^b \cdot \prod_{j=1}^{n} \text{EM}_j,$$

with a and b project-class constants, KLOC thousands of lines of code, and EM_j effort multipliers reflecting product, platform, personnel, and process attributes. COCOMO II (Boehm et al., 2000) refined the parameterization but preserved the core epistemic commitment: effort is a function, in the mathematical sense, of project attributes plus residual error.

The introduction of Story Points in early Extreme Programming and Scrum communities (Beck, 2000; Cohn, 2005) appeared to mark a break from this tradition. Story points were promoted as *ordinal*, *team-relative*, and *non-comparable across teams*. Yet in practice the ordinal scale was rapidly re-cardinalized. Velocity was computed as the arithmetic mean of completed story points per sprint, and sprint commitments were derived from velocity by linear projection. The ordinal-cardinal slippage is not incidental but structural: as long as a team must answer the question *"how many story points will we deliver next sprint?"*, the ordinal scale is forced through a cardinal procedure. The performative dimension of the utterance, which we develop in Section 5, is suppressed at exactly this moment.

The transition into the modern deep-learning era is marked by Deep-SE (Choetkiertikul et al., 2019), which trained a long short-term memory network with a recurrent highway network on issue title and description text and produced a real-valued story-point estimate. The architecture was canonized as a benchmark, and subsequent work introduced GPT-2-based estimators (Fu & Tantithamthavorn, 2023), domain-pretrained BERT variants (Amasaki, 2023), pruned transformers (Cheemaa et al., 2025), retrieval-augmented LLM estimators, and search-based few-shot prompting (Tawosi, Sarro, & Petrozziello, 2023). The dominant benchmark, TAWOS (Tawosi et al., 2022b), now contains over 500,000 issues from 44 open-source Jira projects. Yet the empirical performance of this lineage is sobering: Tawosi, Moussa, and Sarro (2023) re-ran Deep-SE under controlled conditions and found that it failed to significantly outperform a trivial median baseline in the majority of project comparisons. Subsequent work has documented similar baseline

fragility in transformer estimators, a pattern consistent with what Kapoor and Narayanan (2023) describe as a reproducibility crisis in machine-learning-for-science across disciplines.

Three structural confusions thread through this genealogy and are carried forward, intact, into each successive methodological generation. The first is a *cardinal-ordinal confusion*: effort is treated as a cardinal quantity even when its operational scale is ordinal (story points) or socially negotiated (deadline commitments). Once a model emits a real number, planning practice forces that number through arithmetic operations (averaging, summing, projecting) that the underlying quantity does not support. The second is a *historical-fit versus deployed-calibration confusion*: model accuracy on held-out historical data is conflated with calibration in the deployed organizational context, where assignees, dependencies, and team composition shift continuously (Tantithamthavorn et al., 2017). The third is a *single-output versus multi-actor confusion*: estimation is modeled as a function from issue features to a single output, but in practice an estimate is the outcome of a negotiation among reporter, assignee, manager, and customer. Reducing this negotiation to a single-output regression discards precisely what makes the estimate operative. These three confusions motivate the methodological critique that follows.

4. The Methodological Critique of ML-Based Estimation

The methodological literature on machine-learning software effort estimation contains a damaging finding: under controlled comparison with trivial baselines, the apparent gains of contemporary deep-learning estimators are smaller, less stable, and more dataset-dependent than the headline numbers in the original publications suggest. The replication study by Tawosi, Moussa, and Sarro (2023) re-ran Deep-SE (Choetkiertikul et al., 2019) on the original benchmark of sixteen open-source projects under hardened evaluation conditions and found that the model significantly outperformed a trivial per-component median in only 8 of 42 within-project comparisons and 9 of 32 cross-project comparisons. Hyperparameter tuning, data augmentation, and pretraining variants did not materially improve the picture. Subsequent transformer-based estimators (Fu & Tantithamthavorn, 2023; Cheemaa et al., 2025) and recent LLM-based estimators (Tawosi, Sarro, & Petrozziello, 2023; Radliński & Swacha, 2025) inherit a similar fragility: gains exist on specific projects under specific data-construction choices, but the gains are heterogeneous, and absolute error remains large relative to the operational meaning of a story-point unit. The pattern is consistent with what Kapoor and Narayanan (2023) describe across many machine-learning-for-science applications: a reproducibility crisis in which results that appear to constitute progress dissolve under honest baseline comparison. The lesson for ML-SEE is not that the methods are dishonest but that the field has often measured the wrong thing: model accuracy on held-out historical data is not the same quantity as deployed calibration in an organizational planning context (Tantithamthavorn et al., 2017).

A second class of problems concerns *leakage*. Kapoor and Narayanan (2023) catalogue eight distinct leakage modes; at least four of them apply directly to issue-tracker pipelines. Random train-test splits violate the temporal order in which issues were actually estimated and resolved, and because story points are often revised after an issue has been partially worked on, TAWOS (Tawosi et al., 2022b) preserves change-history for this reason, training on the *final* story-point label predicts a quantity unavailable at the moment estimation is performed. Issue text often contains identifying tokens such as project codenames, reporter aliases, and environment paths that allow the model to infer organizational facts the deployed system cannot use without retraining for each new team. Cross-validation that randomly mixes components from a single project allows memorization of per-component effort distributions that simply do not exist on a new project. And late issue comments routinely contain effort signals, estimated hours, blockers, dependencies, added *after* an early estimate was committed; including comment text in features predicts effort partly from outcomes. These leakage modes are not symptomatic of bad practice in any single paper. They reflect a structural fact about

issue-tracker data, which is generated *by* a planning process that is the very thing the model claims to predict.

A third confusion concerns the dependent variable itself. Across the ML-SEE literature, that variable is variously the original story-point estimate, the final story-point label after revision, completed calendar duration, completed person-hours, or a binary *"fits in sprint / does not."* These are different quantities, generated by different organizational processes, and they are not monotonically related. The follow-up study by Tawosi, Moussa, and Sarro (2022c) on the relationship between story points and development effort in agile open-source software demonstrates this empirically: the Spearman rank correlation between story points and actual effort is, on most projects, weak. A model that achieves low mean absolute error on story points may have negligible practical value for sprint planning if story points themselves are poorly correlated with the calendar quantities planners actually want to know.

A final consideration is the cost of the estimator itself. Recent LLM-based pipelines consume non-trivial compute per estimate, but the published literature rarely reports the inference cost of a story-point prediction in dollars or kilowatt-hours, nor compares accuracy *per unit of compute* across methods. A reasonable order-of-magnitude cost model for an LLM estimator is

$$C_{\text{LLM}} \;=\; N \cdot \left(c_{\text{prompt}} \cdot t_{\text{in}} + c_{\text{completion}} \cdot t_{\text{out}}\right) \;+\; N \cdot e_{\text{infra}},$$

where N is the number of estimates per quarter, t_{in} and t_{out} are mean input and output token counts per estimate, c_{prompt} and $c_{\text{completion}}$ are the provider's per-token prices, and e_{infra} is the amortized infrastructure cost per call. For a team estimating two hundred issues per sprint and ten sprints per year, with $t_{\text{in}} \approx 400$ and $t_{\text{out}} \approx 50$ at current frontier-model pricing, C_{LLM} falls in the range of hundreds to low thousands of dollars per team-year, small in absolute terms, but non-trivial when measured against the marginal accuracy gain documented above.

Taken together, the four problems converge on one diagnosis: machine-learning software effort estimation pipelines are well-engineered solutions to a question that has been incompletely specified. No amount of careful cross-validation, trivial-baseline comparison, or per-token cost accounting will produce a calibrated *prediction* of an effort quantity that is itself constituted, in part, by the act of producing the prediction. The next section develops the argument that follows from this observation.

5. Estimation as Speech Act

Austin's (1962) lectures introduced the distinction that will do the heaviest work in what follows. A *constative* utterance describes a state of affairs and is evaluated as true or false, *the cup is on the table*; *the train arrives at noon*. A *performative* utterance, in contrast, does not describe a pre-existing state but, by being uttered under appropriate conditions, brings the state into being, *I promise to be there*; *I name this ship Queen Elizabeth*; *I declare the meeting closed*. The criterion of evaluation is correspondingly different: a performative cannot be true or false, but it can be *felicitous* or *infelicitous* depending on whether the conditions of its issuance are satisfied. Searle (1969, 1995) refined Austin's distinction into a typology of five illocutionary acts: assertives, directives, *commissives*, expressives, and *declaratives*. He argued that institutional facts (a deadline, a marriage, a debt) are created by declarative acts issued by authorized speakers under appropriate conditions. The framework has been applied widely outside its native linguistic domain, most prominently in the performativity-of-economics tradition where MacKenzie (2006) shows that financial models do not merely describe markets but participate in producing them. What follows applies the same apparatus to a different and, we will argue, structurally distinct case.

An important clarification is required at the outset. The claim is not that every numerical assessment is

automatically a commitment. A story-point value written on an isolated issue, for example, by an analyst tagging a backlog before any planning conversation has taken place, may function purely as an **assertive** or classificatory judgment: "I assess this issue at three points." It becomes **commissive** only when it is taken up into a planning practice that allocates capacity, fixes a delivery promise, sets a sprint scope, or assigns accountability. The locus of the speech act is the *uptake*, not the utterance considered in isolation.

Consider the canonical case. An engineer, in a sprint-planning ceremony, says of an issue *"this will take three story points"*. In isolation, such an utterance may function as an assessment of effort. In sprint planning, however, once accepted into a capacity decision or delivery commitment, it takes on a commissive role: the engineer (or, after team agreement, the team) is now publicly committed to performing the work within an effort envelope. After the team's acceptance, the same utterance also operates as a **declarative**: the sprint commitment exists as an institutional fact, with consequences for stakeholder communication, scope management, and the team's accounting of its capacity. Only secondarily does the utterance function as a **constative**, a description of how much effort the work will in fact consume. The peer-reviewed software-engineering literature has, with notable exceptions, treated estimates almost exclusively in the constative register: as candidate forecasts to be scored, after the fact, against an eventual completion quantity (Jørgensen, 2007; Choetkiertikul et al., 2019; Tawosi et al., 2023). The category error consists in collapsing the planning-relevant role of the utterance into the constative role: by *measuring* estimates only against outcomes, the field tends to define estimates implicitly as the kind of utterance that can only be measured against outcomes.

The felicity conditions of a commissive estimation utterance are explicit in Searle's account, and it is worth running through them in this domain to make the contrast precise. *Propositional content.* The utterance must concern a future act of the speaker, in our case, the engineer's own work, not anyone else's. *Preparatory condition.* The speaker must have reason to believe that the act is possible, which in software work amounts to a non-trivial epistemic condition: the engineer must have inspected the issue, considered dependencies, and judged the task within the team's competence and the sprint's remaining capacity. *Sincerity condition.* The speaker must intend to perform the act, must, in the social register, *mean* the estimate. An estimate offered under coercion, or with the private intention to renegotiate it later, is a defective commissive in Austin's sense. *Essential condition.* The utterance must count, in the relevant institutional context, as a commitment to perform. Sprint planning ceremonies are designed precisely to make the utterance count: by ratifying the estimate the team converts it from a private opinion into a public obligation. When any of the four conditions fails, the utterance ceases to be an estimate in the speech-act sense and becomes something else, an opinion, an aspiration, a contractual feint, a piece of automated output without a speaker. The felicity-condition lens already does diagnostic work: many of the practitioner pathologies catalogued in the estimation literature (Halkjelsvik & Jørgensen, 2012; Lovallo & Kahneman, 2003; Mahnič & Hovelja, 2012) can be redescribed as failures of one or another condition.

The irreducibility argument can now be stated more precisely. *Planning-relevant* software effort, the quantity that a sprint commitment is supposed to control, is not merely a hidden physical quantity to be uncovered by skillful measurement. Actual elapsed time and person-hours can, of course, be measured after the fact. But the quantity that the planning utterance ranges over is partly *constituted* by the speech act that names it: the act of saying "three days" reorganizes the engineer's week, triggers calendar holds, sets stakeholder expectations, structures the conversation about scope at the next stand-up, and, when the work begins to overflow the commitment, sets in motion the scope-cutting and re-prioritization that ultimately determine what the team books as completion. The eventual completion duration is therefore not an independent target awaiting prediction; it is co-produced by the work itself and the prior commitment. A predictor that stands wholly outside the system it claims to predict, and most contemporary ML estimators do, cannot be calibrated, in the relevant sense, on a quantity that its own deployment would alter.

This is a stronger claim than the classical observation that measurement perturbs its object. A thermometer

perturbs the fluid it samples weakly and bidirectionally; the perturbation is treatable as noise around a stable signal. An estimate, by contrast, *constitutes* the deadline it appears to describe: there is no underlying signal independent of the act. The asymmetry is what distinguishes performative from constative utterances, and it is what the prediction frame, by treating the estimate as a regression target, must, by design, suppress. Once the asymmetry is named, the field's persistent failures (Section 4) cease to look like methodological artifacts and begin to look like the predictable behavior of a research programme that has misclassified its object.

The most serious objection to the speech-act framing is that it is borrowed wholesale from MacKenzie's (2006) treatment of the Black-Scholes option-pricing formula and the broader performativity-of-economics tradition (Callon, 1998; MacKenzie, 2006; Espeland & Sauder, 2007). MacKenzie's case is now a canonical example: the formula does not merely describe option prices but, through the practice of traders, produces the very statistical regularities the formula assumes. If the present argument is no more than the software analogue of *An Engine, Not a Camera*, the contribution reduces to a footnote in an established literature. Three structural differences, set out in turn, justify treating the software case as a separate object of theoretical attention.

The absence of arbitrage closure. In financial markets, deviations between model-predicted prices and observed prices are damped by arbitrage. Traders who detect a mispricing have an incentive to trade against it, and their aggregated trades restore approximate consistency between model and market. The mechanism is mathematically tractable in MacKenzie's account: arbitrage is what closes the loop between the model's prescriptions and the market's behavior, producing the Barnesian performativity in which the model becomes self-validating. Software estimation has no equivalent closure mechanism. An overoptimistic estimate is not corrected by a counterparty who takes the opposite side and profits when the work overruns; the only "correction" is the overrun itself, by which point the political consequences, missed deadline, contracted scope, dropped feature, blamed engineer, have already accrued and are irreversible. Performativity in software is therefore *open-loop* in a way that performativity in finance is not, and this asymmetry has substantive consequences: the open loop is what allows estimation pathologies (planning fallacy, anchoring, deadline pressure) to persist over decades without being damped.

The intra-firm rather than market setting. MacKenzie's models perform inside a market populated by formally equal traders, in which prices are produced by anonymous interactions between counterparties. Estimation performs inside a hierarchy. The engineer who issues the commissive and the manager who receives it have asymmetric power, asymmetric information, and asymmetric exposure to the consequences of an inaccurate estimate. The estimate is therefore embedded in a labor relation rather than a market relation, and the appropriate sociology is the political economy of measured work (Braverman, 1974; Espeland & Sauder, 2007; Moore, 2017; Zuboff, 2019) rather than the sociology of markets. This shift is not cosmetic. It changes what counts as the relevant counterperformative force. In finance, counterperformativity arises when the model itself begins to undermine the conditions of its own success; in software, counterperformativity is more often a labor-relations phenomenon, workers learn to game the metric, to estimate strategically, to convert an evaluative instrument into a defensive one.

The combinatorial novelty of software work. The third difference is epistemic. Classical maintenance tasks aside, a substantial portion of professional software development is novel in the strong sense: the engineer has not previously executed a task identical, or even closely similar, to this one. The base rates that ground prediction in stable domains (actuarial tables for insurance, return distributions for finance, weather climatologies for meteorology) are largely absent. Brooks's (1987) "no silver bullet" observation, read with hindsight, was a claim about precisely this absence: not a pessimistic productivity forecast but a statement about the unavailability of stable base rates in a field whose work is, by its institutional logic, perpetually new. Where base rates are absent, the very preconditions of cardinal prediction collapse, and the speech-act dimension of the estimate (the work it does to commit, coordinate, and align expectations) is left as the only

operative dimension.

Taken together, the three differences justify a software-specific treatment of performativity and, more importantly, motivate the diagnosis we develop next: that machine-learning estimators do not merely re-describe the prediction frame but actively *launder* the underlying speech act, with consequences for the political economy of software work that the financial-model literature has no occasion to address.

6. The Laundering Diagnosis

If estimation is, in its planning-relevant sense, structurally a commissive speech act, why should ML-based estimation be a worse problem than the expert-judgment estimation that preceded it? Expert judgment is itself subject to well-documented pathologies: anchoring (Tversky & Kahneman, 1974), the planning fallacy (Buehler, Griffin, & Ross, 1994), the summation fallacy when subtask estimates are aggregated (Halkjelsvik & Jørgensen, 2022), strategic misrepresentation under deadline pressure (Lovallo & Kahneman, 2003; Flyvbjerg, 2006), and the social anchoring observed in planning-poker ceremonies (Mahnič & Hovelja, 2012). A reasonable position would be that ML estimators, by averaging over many issues and exposing the estimator to a richer feature space, should at worst inherit and at best damp some of these biases. The empirical record (Section 4) does not yet support that hope: replication studies and honest-baseline comparisons leave the headline gains of ML-SEE fragile and project-dependent (Tawosi, Moussa, & Sarro, 2023; Liu, Gao, Xia, Lo, Grundy, & Yang, 2021). The deeper concern, however, is not that the estimators fail to outperform humans on accuracy; it is that, once embedded in planning practice, they perform a different kind of speech act, one that disguises itself as the same kind. We name this disguise *laundering*: a sociotechnical translation in which a commitment that has socially identifiable authors and accountability is converted into a technical output whose contestability, authorship, and responsibility become obscured.

By laundering, we do not mean deliberate deception. We mean the structural effect of replacing a commissive utterance, issued by a person or team and held to felicity conditions of the kind set out in §5, with an automated output presented as objective. The replacement need not be intended; it can emerge from the institutional incentives that govern how planning systems are introduced and from the conventional reading practices that surround model outputs. What follows analyses the three mechanisms by which the laundering effect operates.

The first laundering mechanism is *opacity of derivation*. A deep-learning estimator emits a number whose justification cannot, in practice, be inspected by a human in the seconds available during sprint planning. Saliency maps, attention heatmaps, and SHAP values address the technical-explainability question but not the planning-meeting question: a senior engineer who wishes to contest the estimate cannot, in a fifteen-minute ceremony, audit the model's reasoning the way she could audit a colleague's reasoning. The opacity is not a bug to be patched; it is constitutive of how trained models work. Its political consequence is that the estimate becomes asymmetrically *contestable*: the speaker (the model) cannot be cross-examined, while the listener (the team) can.

The second laundering mechanism is *the rhetoric of objectivity*. ML outputs are wrapped in a vocabulary that signals computational rigour, confidence intervals, mean absolute error, model versions, validation accuracy, calibration plots, even when the underlying validity is poor. The rhetorical effect is independent of the technical content. A team confronted with a number labelled "estimated story-point complexity = 5 (95% CI: 3.8 - 6.2)" treats the output differently from a number labelled "Marko's gut feeling: about a 5", even when the two numbers come from systems of comparable reliability. The laundering operates through what Espeland and Sauder (2007) call the reactivity of measurement: the apparent precision of the output reorganizes the discussion around it, rather than being interrogated by the discussion.

The third laundering mechanism is *the displacement of accountability*. When the estimate is produced by a

model, the engineer who would otherwise own the commitment can defer to the system ("the model said five"), and the manager who would otherwise own the planning consequence can defer to the engineer who deferred to the system ("we used the recommended estimate"). The chain that runs from utterance to commitment to consequence, the chain that makes a speech act felicitous in Searle's sense, is interrupted at every link by an opportunity to disclaim. What remains is a number that no one issued, no one stood behind, and no one need answer for when it proves wrong. This is the deepest sense in which ML-based estimation is not merely an inaccurate predictor but a structurally different speech act: it is a *defective commissive*, a commitment with no committed party.

Consider a stylized sprint-planning case to make the mechanism concrete. (The case is illustrative rather than empirical; it composes plausible features of the setting in order to display the laundering effect, and it is not drawn from any specific deployment.) A team is sizing a backlog issue concerning a payment-flow change. The engineer assigned to the work estimates the issue at eight story points, reasoning that an upstream dependency on a third-party vendor's API is uncertain and may require a workaround. The team's ML estimator, trained on the team's historical issues, suggests three points: the issue text is short, the file paths are familiar, and the model has not seen comparable dependency uncertainty in the recent training window. The product manager, reviewing the planning board, treats the model's output as the more "objective" of the two estimates and gently presses the engineer to accept it. The engineer, lacking the time and the social standing to litigate the difference in a fifteen-minute ceremony, accepts three. Two sprints later, the dependency materialises: the work overruns, the release slips, and the post-mortem is convened. The engineer's record shows a "missed estimate"; the manager records that the team "followed the data"; the model continues to be used in subsequent sprints with no recorded fault. The number that drove the commitment had no committed author, and when the commitment failed, the responsibility for the failure was distributed in a way that left no party fully answerable for it. The example is small, but the structural pattern it instantiates is the laundering effect this section names.

The three laundering mechanisms do not float free of organizational structure; they intersect with the political economy of measured work in a determinate way. Braverman's (1974) classic analysis describes how industrial labor is reorganized through the separation of *conception* from *execution*: the planning of work is concentrated in a managerial layer while the worker is reduced to an executor of pre-specified tasks. The mechanism Braverman identified for manual labor has been extended to knowledge work under the rubric of algorithmic management (Mateescu & Nguyen, 2019; Moore, 2017), in which planning, monitoring, and evaluation are increasingly performed by software systems whose outputs structure the worker's day. ML-based effort estimation is a specific instance of this pattern, located at the moment in which the planning conception is generated.

The dynamic is straightforward when stated plainly. The conception of how long work *should* take is removed from the worker who will perform the work and relocated to a model owned and operated by the firm. The model's outputs then re-enter the worker's planning environment as authoritative inputs against which the worker must justify any deviation. An estimate that the engineer would once have issued as a commissive, *I will do this in three days*, is now issued by the system as a quasi-directive, against which the engineer's role is reduced to acceptance, contestation, or strategic accommodation. Espeland and Sauder's (2007) analysis of how measurement systems reshape the entities they measure applies here with particular force: the worker who must justify deviations from a model's estimate is in a different work relation from the worker who issues her own commitments and is held to them.

This reframing also clarifies what is at stake when ML-SEE outputs are integrated with performance management. Goodhart's (1984) law, that a measure ceases to be a good measure once it becomes a target, is in evidence whenever individual estimation accuracy is tracked across sprints. But the deeper concern is structural rather than statistical. The use of estimation accuracy as a performance metric converts the

engineer's commitment from a sincere speech act into a strategic move under surveillance, and over time produces the well-documented failure mode of metric gaming: estimates inflate to protect against overrun penalties, conservative buffering becomes a defensive practice, and the commissive register that originally made estimation useful as a coordination device is hollowed out (Espeland & Sauder, 2007). What appears as a productivity tool is, viewed through the political-economy lens, an instrument of intra-firm bargaining that systematically advantages the party that controls the model.

We do not claim that this outcome is the conscious intent of the engineers and product managers who deploy ML-SEE tools; in our experience it is rarely so. The point is rather that the institutional incentives that govern how planning systems are introduced, the search for "objectivity," the wish to standardize across teams, the procurement preference for measurable artifacts, produce the laundering effect as a likely default unless explicit countermeasures are put in place. Naming the mechanism is the first step toward the countermeasures developed in Section 7.

The estimation act, on the present account, sits at the intersection of three irreducible dimensions: an *epistemic* dimension concerning what is knowable in advance about future effort, a *social* dimension concerning the negotiation through which a team commits, and a *political* dimension concerning the asymmetries of power over the commitment and its consequences. A given estimation methodology can be located by where it places its center of gravity across these three dimensions; characteristic pathologies arise when one dimension is treated as if it had absorbed the others.

The schema can be represented as a triangle whose vertices are the three dimensions. A given methodology can be located by the vertex toward which its center of gravity is pulled, and characteristic pathologies arise when the two other dimensions are treated as if they had been absorbed into the dominant one. Table 1 summarizes the four regions of this conceptual space. The first three regions correspond to the three vertices, each populated by methodologies that have collapsed onto that vertex; the fourth, interior region is the location proposed by the estimation hygiene framework of Section 7.

Table 1. Conceptual location of estimation methods across the epistemic, social, and political dimensions.

Region	Position in the triangle	Methodologies	Failure mode
A	Near the epistemic vertex	COCOMO, Deep-SE, GPT2SP, LLM estimators	Treats estimation as hidden-variable inference. Machine learning deepens the collapse by adding apparent objectivity without adding stakes.
B	Near the social vertex	Planning Poker, expert consensus, story-point negotiation	Honest about negotiation, but tends to suppress political asymmetry: the senior engineer who speaks first anchors the rest.
C	Near the political vertex	Top-down deadlines, contractual fixed-bid estimates	Honest about power, but presumes epistemic content it does not have.
D	Interior (proposed)	Estimation hygiene	Holds the three dimensions in tension: machine learning used as epistemic sanity-check only; commitments named and bounded as commissives; revision rights treated as a political counterweight.

The schema's argumentative work is twofold. Machine-learning interventions are not a new vertex; they intensify a pre-existing center of gravity at the epistemic corner. The constructive proposal of Section 7 is not a new vertex either; it is a deliberate refusal to collapse onto any single one.

7. Toward an Estimation Hygiene Framework

If estimation, in its planning-relevant sense, is structurally a commissive speech act and machine-learning systems do not, by themselves, satisfy the felicity conditions of commissives, the constructive question is not *how to make machine learning predict better* but *how to deploy machine learning in a way that does not displace the speech act it cannot replace*. *Estimation hygiene*, as we use the term, is a deployment discipline rather than a model architecture. It leaves the choice of estimator open and constrains the way the estimator's outputs are integrated into planning practice. The discipline can be specified through a small number of standing commitments and a single operational rule that wires those commitments into a sprint-planning workflow.

The first commitment is *baseline transparency*. Reviewers, internal tooling teams, and procurement processes should require that any deployment of a machine-learning effort estimator report performance against a per-component or per-reporter median baseline. The lesson of Tawosi, Moussa, and Sarro (2023), that Deep-SE significantly outperforms a trivial median in a minority of project comparisons, should be the field's default expectation rather than its surprise. The second commitment is *uncertainty-aware reporting*. Sprint planning needs prediction intervals, not point estimates: the conformal-prediction and uncertainty-quantification literatures (Liu et al., 2025) supply methods for converting any point predictor into an interval with formal coverage guarantees under specified assumptions, and a deployment output should take the form *"with 80% confidence this issue is between L and U story points"* rather than *"this issue is $\hat{y}$ story points."* The validity of such intervals, however, is conditional. It depends on exchangeability between the calibration distribution and the deployment distribution, and on stable team and process conditions; neither can be taken for granted in production, and reported coverage must be monitored against realized outcomes and recalibrated when monitoring reveals drift.

The third commitment is *calibration accountability*. Calibration is not stable across team composition changes, tool migrations, or process shifts. A hygienic deployment monitors the calibration of the deployed model over a rolling window of W sprints and triggers an explicit halt-and-recalibrate cycle when miscalibration exceeds a threshold δ. The fourth commitment, the most consequential of the set, is *human authorship of the commitment*. The model's output is a candidate input to the planning conversation; the commitment remains the engineer's commissive utterance, accepted by the team. This protects the felicity conditions of the speech act (§5) and the chain of accountability (§6) at once. The fifth and final commitment is the *separation of estimation from individual performance management*. Estimation accuracy must not be tracked as an individual performance metric. Doing so converts the engineer's commitment from a sincere utterance into a strategic move under surveillance, and over time produces the well-documented failure mode of metric gaming (Goodhart, 1984; Espeland & Sauder, 2007). Of all the normative recommendations in this paper, this is the one whose violation reliably destroys the commissive register that hygiene is intended to protect.

The operational question is how to wire these commitments into a sprint-planning workflow without re-introducing the laundering effect. The simplest such wiring is what we call the *disagreement-flag rule*: the machine-learning estimator is used not to produce estimates but to flag, for human review, those issues on which the team's estimate and the model's estimate diverge by more than a calibrated threshold. Formally,

$$\text{Commit}(i) = \begin{cases} \hat{e}_H(i), & \text{if } d\big(\hat{e}_H(i),\, \hat{e}_M(i)\big) \le \tau, \\ \text{review}(i), & \text{if } d\big(\hat{e}_H(i),\, \hat{e}_M(i)\big) > \tau, \end{cases}$$

where $\hat{e}_H(i)$ is the human team's estimate for issue i, $\hat{e}_M(i)$ is the model's output for the same issue, $d(\cdot,\cdot)$ is a domain-appropriate distance, and τ is a per-team disagreement threshold. For story-point estimates on a Fibonacci-like ordinal scale, the natural choice for d is rank distance, the absolute difference between scale

positions, which respects the ordinal nature of the underlying unit; for calendar-time estimates, log-ratio distance is a sensible default. The threshold τ is not a universal constant but a calibration parameter best derived from the team's own history. A defensible starting calibration is

$$\tau \;=\; \underset{i \in \mathcal{H}}{\text{median}}\, d\big(\hat{e}_H(i),\, \hat{e}_M(i)\big) \;+\; k \cdot \underset{i \in \mathcal{H}}{\text{MAD}}\, d\big(\hat{e}_H(i),\, \hat{e}_M(i)\big),$$

where $\mathcal{H}$ is the team's set of recently estimated and resolved issues, MAD is the median absolute deviation of the human-model disagreement on that history, and k is a tunable factor that controls the share of issues flagged for review. A starting value of $k = 2$ produces a flag rate corresponding to the upper tail of disagreement under a near-normal disagreement distribution; teams whose review capacity is constrained will set k higher, and teams that wish to surface more disagreements for discussion will set it lower.

The defining property of the rule is that the model never produces the commitment. The committed estimate is always $\hat{e}_H(i)$, the team's own number, and the model's role is exclusively to direct attention. When the model and the team agree within τ, the team's estimate stands without modification; when they disagree, the disagreement triggers a brief human conversation about why, after which the team revises or does not revise its estimate. The model output is never read out as the answer, and it is never recorded as the team's commitment in the issue tracker. This wiring preserves the felicity conditions of the commissive speech act analyzed in Section 5: the propositional content remains the engineer's act, the preparatory and sincerity conditions remain the engineer's beliefs and intentions, and the essential condition, that the utterance counts as a commitment, remains anchored in human authorship rather than in algorithmic output.

Three failure modes deserve explicit attention before the rule is adopted. *Threshold drift* arises when a team's composition or process changes and its disagreement distribution shifts: a τ calibrated on a stale window will under- or over-flag, and the remedy is rolling-window recalibration on a moving history together with a halt-and-recalibrate trigger that binds the rule's behavior to the team's current state rather than its past. *Anchoring on disagreement* arises in the human conversation that follows a flag: the team may unconsciously regress its estimate toward the model's number, defeating the intent that the model only direct attention. The remedy is procedural rather than technical, a brief norm in the team's planning ceremony that the model's number is not introduced into the discussion until *after* the team has articulated, in qualitative terms, why it might have under- or over-estimated the issue. *Quiet co-option for performance management* arises when a manager observes the flag log over time and begins to use it as a record of which engineers "disagree with the model" most often. This is precisely the failure pattern that the separation of estimation and performance management is meant to prevent, and a team that adopts the disagreement-flag rule must adopt the prohibition with it. Without the prohibition, the rule becomes a surveillance device wearing the costume of a coordination device.

The rule is deliberately a minimum specification rather than a complete protocol. Real teams will need to decide who participates in the review when a flag is raised, how the review is logged or whether it is logged at all, and how disagreement statistics are reported up the management chain or whether they are reported at all. These choices are properly local: they should be made by teams in light of their own labor relations, trust climate, and history with measurement. What the rule provides is the structural commitment that holds across local choices, the model is an instrument that organizes human attention, never an instrument that authors commitments. Estimation hygiene, in this sense, is not a replacement for machine-learning methodology. It does not propose a new architecture, dataset, or loss function. It does not eliminate planning failure either: under hygiene, teams will still mis-estimate, deadlines will still slip, and senior engineers will still anchor sprint planning. What it removes is the specific contemporary failure mode in which a model's output, taken as objective, displaces a commitment that the system is structurally incapable of making.

8. Discussion

Three objections to the argument deserve direct response. The first is empirical: contemporary transformer- and LLM-based estimators report mean absolute error improvements over expert judgment on held-out benchmarks (Fu & Tantithamthavorn, 2023; Tawosi, Sarro, & Petrozziello, 2023), and if the prediction frame were truly incoherent, the objection runs, no such gains would be possible. The response is to draw a distinction the field has not yet drawn cleanly. The benchmark gains live within an *established commitment regime*: a stable team, a fixed scope, deadlines and dependencies that the experimental design holds constant. They reflect interpolation under such conditions, not prediction of effort as a free variable. The replication evidence (Tawosi, Moussa, & Sarro, 2023) shows that even these interpolation gains are smaller and more dataset-dependent than headline numbers suggest; and even where the gains are real, they are produced precisely by holding the speech-act variables fixed, which is the configuration in which the negotiation has already happened and in which the model's predictions are least useful in deployed planning.

A second, deeper objection is methodological. The speech-act framing, it might be said, is unfalsifiable: Austin-Searle vocabulary can re-describe any estimation outcome and so generates no testable predictions. We concede part of the objection. The contribution of the framing is not that it generates novel quantitative predictions; it is that it generates better explanations for anomalies the prediction frame cannot accommodate, the persistent accuracy plateau across four methodological generations (Jørgensen & Shepperd, 2007), the asymmetric political consequences of over- versus under-estimation (Lovallo & Kahneman, 2003), and the failure of debiasing interventions to converge on a stable improvement (Halkjelsvik & Jørgensen, 2012). The paper's reframing is best read as inference to the best explanation: a saving of appearances that, unlike the prediction frame, opens normative recommendations the prediction frame cannot motivate.

The most serious objection is theoretical. Performativity of models is by now a mature literature (Callon, 1998; MacKenzie, 2006), and a software analogue might appear to be no more than a corollary. We have argued in Section 5 that three structural differences distinguish the software case: the absence of arbitrage closure, the intra-firm rather than market setting, and the combinatorial novelty of software work. Without these differentiations the present paper would indeed be a footnote in the performativity literature; with them, it is a distinct contribution about the political economy of intra-firm measurement, in which the open-loop dynamics of estimation, the asymmetry of authority over the commitment, and the absence of stable base rates jointly produce a phenomenon not reducible to MacKenzie's account of financial-model performativity.

The argument has limits, and they deserve to be named directly. The paper presents no new empirical data and does not include a production deployment test of the estimation hygiene framework. Its descriptive premises, the replication failure of Deep-SE, the leakage modes documented in machine-learning-for-science, the calibration fragility of LLM estimators, are inherited from the cited literature, and the analysis stands or falls with the reliability of those secondary sources. The speech-act reframing is explanatory and theoretical: it is defended through conceptual analysis and analogical differentiation rather than through a single empirical test, and it is not directly falsified by any benchmark result on any single dataset. Because the paper is conceptual, the estimation hygiene framework should be read as a design proposal rather than as a validated intervention. The argument is also strongest for novel, feature-development work in internal product teams under conditions where the estimators have stakes in the outcome. It is weaker, by the paper's own logic, for highly repetitive maintenance work where base rates exist and the prediction frame may be locally valid; for fixed-bid contracting between firms, where the estimate is an instrument in a different speech act, namely an offer or tender; and for very early-stage estimation, where the propositional content of the commitment is itself unstable.

The implications, with these limits in view, are of three kinds. For practitioners, the paper offers the deployment commitments and the disagreement-flag rule of Section 7 as concrete starting points; teams adopting machine-

learning estimation should treat the trivial-baseline comparison and the calibration-monitoring loop as defaults rather than as enrichments. For peer reviewers and editors of software-engineering venues, the standard for ML-SEE manuscripts should require explicit comparison against a trivial median baseline, disclosure of leakage controls in the eight modes catalogued by Kapoor and Narayanan (2023), and explicit treatment of the deployment context, not only the held-out test set. For organizational leaders, the most consequential implication is the separation of *estimation-as-planning* from *estimation-as-performance-management*: the use of estimation accuracy as an individual performance metric should be explicitly excluded from any engineering policy that introduces machine-learning estimation, on pain of producing the metric-gaming pathology the framework is designed to prevent.

The framework also opens a tractable empirical agenda. Whether teams that adopt machine-learning estimation under hygiene constraints exhibit smaller calibration drift over rolling windows than teams that adopt the same models without those constraints is a question that comparative field studies could answer. The conjecture that explicitly attributing commitments to named engineers, thus preserving the commissive speech act, increases the team's perceived ability to contest estimates relative to settings in which model outputs are treated as authoritative inputs is similarly testable through survey instruments combined with observational data on planning ceremonies. The hypothesis that the use of estimation outputs for individual performance management produces faster onset of metric-gaming than adoption that excludes such use can be examined longitudinally in firms that institute one or the other regime. At the firm level, the political-economic dimension developed in Section 6 implies that the laundering effect varies with employment relations: firms with stronger collective worker representation should display greater resistance to it than firms without, a question that quantitative organizational studies are well placed to address. And the distinction we drew in this section between interpolation within an established commitment regime and prediction of effort as a free variable invites direct study: production-deployed estimators should show their accuracy gains concentrated in the former regime and attenuating to zero, within a small number of sprints, when teams change composition, tools, or process. None of these conjectures requires the speech-act framing to be true in order to be tested; the framing makes them salient as questions worth asking, and the prediction frame, by contrast, has no occasion to ask them.

9. Conclusions

Software effort estimation has often been treated as if it were primarily a prediction problem, and successive methodological generations (parametric, classical-ML, transformer, and LLM-based) have inherited much of that framing. The persistence of estimation difficulties suggests, on the present account, that the question is not only one of model quality but also one of how the object of estimation is conceived. Reframing the planning-relevant estimate as a performative speech act, a commitment that participates in producing the future it claims to describe, clarifies what ML-based estimators can and cannot do: they cannot, by themselves, make commitments, but they can usefully support the humans who do. An *estimation hygiene* discipline retains machine learning as decision support while protecting the speech act in which estimation, in its planning-relevant sense, actually consists.

Author Contributions

- **Yauheni Kanavalik (lead, corresponding author):** Conceptualization, Methodology, Writing, original draft, Writing, review & editing, Project administration, Supervision.
- **Daria Firsova:** Conceptualization, Investigation, Writing, original draft, Writing, review & editing, Validation.

- **Andrei Dzeikalo:** Investigation, Writing, original draft, Writing, review & editing, Resources.
- **Vadim Goncharov:** Investigation, Writing, review & editing, Resources.

All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Data Availability Statement

No new data were generated or analyzed in this study.

Conflicts of Interest

The authors declare no conflicts of interest.

References

1. Albrecht, A. J. (1979). Measuring application development productivity. In *Proceedings of the IBM Applications Development Symposium* (pp. 83 - 92). IBM.
2. Albrecht, A. J., & Gaffney, J. E. (1983). Software function, source lines of code, and development effort prediction: A software science validation. *IEEE Transactions on Software Engineering*, *SE-9*(6), 639 - 648. https://doi.org/10.1109/TSE.1983.235271
3. Amasaki, S. (2023). On effectiveness of further pre-training on BERT models for story point estimation. In *Proceedings of the 19th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE '23)*. ACM. https://doi.org/10.1145/3617555.3617877
4. Austin, J. L. (1962). *How to do things with words*. Harvard University Press.
5. Beck, K. (2000). *Extreme programming explained: Embrace change*. Addison-Wesley.
6. Boehm, B. W. (1981). *Software engineering economics*. Prentice-Hall.
7. Boehm, B. W., Abts, C., Brown, A. W., Chulani, S., Clark, B. K., Horowitz, E., Madachy, R., Reifer, D. J., & Steece, B. (2000). *Software cost estimation with COCOMO II*. Prentice Hall.
8. Bowker, G. C., & Star, S. L. (1999). *Sorting things out: Classification and its consequences*. MIT Press.
9. Braverman, H. (1974). *Labor and monopoly capital: The degradation of work in the twentieth century*. Monthly Review Press.
10. Brooks, F. P. (1987). No silver bullet: Essence and accidents of software engineering. *Computer*, *20*(4), 10 - 19. https://doi.org/10.1109/MC.1987.1663532

11. Buehler, R., Griffin, D., & Ross, M. (1994). Exploring the "planning fallacy": Why people underestimate their task completion times. *Journal of Personality and Social Psychology*, *67*(3), 366 - 381. https://doi.org/10.1037/0022-3514.67.3.366

12. Callon, M. (1998). Introduction: The embeddedness of economic markets in economics. *The Sociological Review*, *46*(S1), 1 - 57. https://doi.org/10.1111/j.1467-954X.1998.tb03468.x

13. Cheemaa, A. S., Azhar, M., Arif, F., ul Haq, Q. M., Sohail, M., & Iqbal, A. (2025). EGPT-SPE: Story point effort estimation using improved GPT-2 by removing inefficient attention heads. *Applied Intelligence*. https://doi.org/10.1007/s10489-025-06824-4

14. Choetkiertikul, M., Dam, H. K., Tran, T., Pham, T., Ghose, A., & Menzies, T. (2019). A deep learning model for estimating story points. *IEEE Transactions on Software Engineering*, *45*(7), 637 - 656. https://doi.org/10.1109/TSE.2018.2792473

15. Cockburn, A. (2006). *Agile software development: The cooperative game* (2nd ed.). Addison-Wesley.

16. Cohn, M. (2005). *Agile estimating and planning*. Prentice Hall.

17. De Bortoli Fávero, E. M., & Casanova, D. (2022). SE3M: A model for software effort estimation using pre-trained embedding models. *Information and Software Technology*, *147*, 106886. https://doi.org/10.1016/j.infsof.2022.106886

18. Espeland, W. N., & Sauder, M. (2007). Rankings and reactivity: How public measures recreate social worlds. *American Journal of Sociology*, *113*(1), 1 - 40. https://doi.org/10.1086/517897

19. Flyvbjerg, B. (2006). From Nobel Prize to project management: Getting risks right. *Project Management Journal*, *37*(3), 5 - 15. https://doi.org/10.1177/875697280603700302

20. Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate: The science of lean software and DevOps*. IT Revolution Press.

21. Forsgren, N., Storey, M.-A., Maddila, C., Zimmermann, T., Houck, B., & Butler, J. (2021). The SPACE of developer productivity. *ACM Queue*, *19*(1). https://doi.org/10.1145/3454122.3454124

22. Fu, M., & Tantithamthavorn, C. (2023). GPT2SP: A transformer-based agile story point estimation approach. *IEEE Transactions on Software Engineering*, *49*(2), 611 - 625. https://doi.org/10.1109/TSE.2022.3158252

23. Ghotra, B., McIntosh, S., & Hassan, A. E. (2015). Revisiting the impact of classification techniques on the performance of defect prediction models. In *Proceedings of the 37th International Conference on Software Engineering (ICSE 2015)*. IEEE. https://doi.org/10.1109/ICSE.2015.91

24. Goodhart, C. A. E. (1984). Problems of monetary management: The UK experience. In *Monetary theory and practice* (pp. 91 - 121). Macmillan. https://doi.org/10.1007/978-1-349-17295-5_4

25. Halkjelsvik, T., & Jørgensen, M. (2012). From origami to software development: A review of studies on judgment-based predictions of performance time. *Psychological Bulletin*, *138*(2), 238 - 271. https://doi.org/10.1037/a0025996

26. Halkjelsvik, T., & Jørgensen, M. (2018). *Time predictions: Understanding and avoiding unrealism in project planning and everyday life*. SpringerBriefs on Computing. Springer. https://doi.org/10.1007/978-3-319-74953-2

27. Halkjelsvik, T., & Jørgensen, M. (2022). When 2 + 2 should be 5: The summation fallacy in time prediction. *Journal of Behavioral Decision Making*, *35*(2), e2265. https://doi.org/10.1002/bdm.2265

28. Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., & Wang, H. (2024). Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology*. https://doi.org/10.1145/3695988

29. Idri, A., Hosni, M., & Abran, A. (2016). Systematic literature review of ensemble effort estimation. *Journal of Systems and Software*, *118*, 151 - 175. https://doi.org/10.1016/j.jss.2016.05.016

30. Jaakkola, E. (2020). Designing conceptual articles: Four approaches. *AMS Review*, *10*, 18 - 26. https://doi.org/10.1007/s13162-020-00161-0

31. Jørgensen, M. (2004). A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, *70*(1 - 2), 37 - 60. https://doi.org/10.1016/S0164-1212(02)00156-5

32. Jørgensen, M. (2007). Forecasting of software development work effort: Evidence on expert judgement and formal models. *International Journal of Forecasting*, *23*(3), 449 - 462. https://doi.org/10.1016/j.ijforecast.2007.05.008

33. Jørgensen, M. (2014). What we do and don't know about software development effort estimation. *IEEE Software*, *31*(2), 37 - 40.

34. Jørgensen, M., & Shepperd, M. (2007). A systematic review of software development cost estimation studies. *IEEE Transactions on Software Engineering*, *33*(1), 33 - 53. https://doi.org/10.1109/TSE.2007.256943

35. Kapoor, S., & Narayanan, A. (2023). Leakage and the reproducibility crisis in machine-learning-based science. *Patterns*, *4*(9), 100804. https://doi.org/10.1016/j.patter.2023.100804

36. Liu, C., Gao, C., Xia, X., Lo, D., Grundy, J., & Yang, X. (2021). On the reproducibility and replicability of deep learning in software engineering. *ACM Transactions on Software Engineering and Methodology*, *31*(1), 1 - 46. https://doi.org/10.1145/3477535

37. Liu, X., Chen, T., Da, L., Chen, C., Lin, Z., et al. (2025). Uncertainty quantification and confidence calibration in large language models: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM. https://doi.org/10.1145/3711896.3736569

38. Lovallo, D., & Kahneman, D. (2003). Delusions of success: How optimism undermines executives' decisions. *Harvard Business Review*, *81*(7), 56 - 63.

39. MacInnis, D. J. (2011). A framework for conceptual contributions in marketing. *Journal of Marketing*, *75*(4), 136 - 154. https://doi.org/10.1509/jmkg.75.4.136

40. MacKenzie, D. (2006). *An engine, not a camera: How financial models shape markets*. MIT Press.

41. Mahmood, Y., Kama, N., Azmi, A., Khan, A. S., & Ali, M. (2022). Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation. *Software: Practice and Experience*, *52*(1), 39 - 65. https://doi.org/10.1002/spe.3009

42. Mahnič, V., & Hovelja, T. (2012). On using planning poker for estimating user stories. *Journal of Systems and Software*, *85*(9), 2086 - 2095. https://doi.org/10.1016/j.jss.2012.04.005

43. Mateescu, A., & Nguyen, A. (2019). *Workplace monitoring & surveillance*. Data & Society Research Institute. https://datasociety.net/library/explainer-workplace-monitoring-surveillance/

44. McConnell, S. (2006). *Software estimation: Demystifying the black art*. Microsoft Press.

45. Moore, P. V. (2017). *The quantified self in precarity: Work, technology and what counts*. Routledge. https://doi.org/10.4324/9781315561523

46. Murphy, A. H., & Winkler, R. L. (1987). A general framework for forecast verification. *Monthly Weather Review*, *115*(7), 1330 - 1338. https://doi.org/10.1175/1520-0493(1987)115%3C1330:AGFFFV%3E2.0.CO;2

47. Pérez Castillo, R., et al. (2024). Sprint management in agile approach: Progress and velocity evaluation applying machine learning. *Information*, *15*(11), 726. https://doi.org/10.3390/info15110726

48. Phan, H., & Jannesari, A. (2022a). Heterogeneous graph neural networks for software effort estimation. In *Proceedings of the 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2022)*. ACM. https://doi.org/10.1145/3544902.3546248

49. Phan, H., & Jannesari, A. (2022b). Story point level classification by text level graph neural network. In *Proceedings of the 1st International Workshop on Natural Language-Based Software Engineering (NLBSE 2022)*. ACM. https://doi.org/10.1145/3528588.3528654

50. Pinch, T. J., & Bijker, W. E. (1984). The social construction of facts and artifacts: Or how the sociology of science and the sociology of technology might benefit each other. *Social Studies of Science*, *14*(3), 399 - 441. https://doi.org/10.1177/030631284014003004

51. Putnam, L. H. (1978). A general empirical solution to the macro software sizing and estimating problem. *IEEE Transactions on Software Engineering*, *SE-4*(4), 345 - 361. https://doi.org/10.1109/TSE.1978.231521

52. Radliński, Ł., & Swacha, J. (2025). Large language models for early-stage software project estimation: A systematic mapping study. *Applied Sciences*, *15*(24), 13099. https://doi.org/10.3390/app152413099

53. Sarim, M., Masood, F., Maheshwari, M., Faridi, A. R., & Shamsan, A. H. (2025). Generating reliable software project task flows using large language models through prompt engineering and robust evaluation. *Scientific Reports*, *15*. https://doi.org/10.1038/s41598-025-19170-9

54. Searle, J. R. (1969). *Speech acts: An essay in the philosophy of language*. Cambridge University Press. https://doi.org/10.1017/CBO9781139173438

55. Searle, J. R. (1995). *The construction of social reality*. Free Press.

56. Suchman, L. (2007). *Human-machine reconfigurations: Plans and situated actions* (2nd ed.). Cambridge University Press.

57. Tantithamthavorn, C., Hassan, A. E., & Matsumoto, K. (2020). The impact of class rebalancing techniques on the performance and interpretation of defect prediction models. *IEEE Transactions on Software Engineering*, *46*(11), 1200 - 1219. https://doi.org/10.1109/TSE.2018.2876537

58. Tantithamthavorn, C., McIntosh, S., Hassan, A. E., & Matsumoto, K. (2017). An empirical comparison of model validation techniques for defect prediction models. *IEEE Transactions on Software Engineering*, *43*(1), 1 - 18. https://doi.org/10.1109/TSE.2016.2584050

59. Tawosi, V., Al-Subaihin, A., & Sarro, F. (2022a). Investigating the effectiveness of clustering for story point estimation. In *Proceedings of the 29th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER 2022)*. https://doi.org/10.1109/saner53432.2022.00101

60. Tawosi, V., Al-Subaihin, A., Moussa, R., & Sarro, F. (2022b). A versatile dataset of agile open source software projects. In *Proceedings of the 19th International Conference on Mining Software Repositories (MSR '22)*. ACM. https://doi.org/10.1145/3524842.3528029

61. Tawosi, V., Moussa, R., & Sarro, F. (2022c). On the relationship between story points and development effort in agile open-source software. In *Proceedings of the 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2022)*. ACM. https://doi.org/10.1145/3544902.3546238

62. Tawosi, V., Moussa, R., & Sarro, F. (2023). Agile effort estimation: Have we solved the problem yet? Insights from a replication study. *IEEE Transactions on Software Engineering*, *49*(4), 2677 - 2697. https://doi.org/10.1109/TSE.2022.3228739

63. Tawosi, V., Sarro, F., & Petrozziello, A. (2023). Search-based optimisation of LLM learning shots for story point estimation. In *Proceedings of the 15th International Symposium on Search-Based Software Engineering (SSBSE 2023)* (pp. 123 - 138). Springer. https://doi.org/10.1007/978-3-031-48796-5_9

64. Tetlock, P. E. (2005). *Expert political judgment: How good is it? How can we know?* Princeton University Press.

65. Tetlock, P. E., & Gardner, D. (2015). *Superforecasting: The art and science of prediction*. Crown Publishers.

66. Tversky, A., & Kahneman, D. (1974). Judgment under uncertainty: Heuristics and biases. *Science*, *185*(4157), 1124 - 1131. https://doi.org/10.1126/science.185.4157.1124

67. Wen, J., Li, S., Lin, Z., Hu, Y., & Huang, C. (2012). Systematic literature review of machine learning based software development effort estimation models. *Information and Software Technology*, *54*(1), 41 - 59. https://doi.org/10.1016/j.infsof.2011.09.002

68. Yalçıner, B., & Ülker, E. (2024). Enhancing agile story point estimation: Integrating deep learning, machine learning, and ensemble learning approaches. *Applied Sciences*, *14*(16), 7305. https://doi.org/10.3390/app14167305

69. Zuboff, S. (2019). *The age of surveillance capitalism: The fight for a human future at the new frontier of power*. PublicAffairs.