Electronic Communications of the EASST
Volume 85 Year 2025

deRSE25 - Selected Contributions of the 5th Conference for Research Software Engineering in Germany

Edited by: René Caspart, Florian Goth, Oliver Karras, Jan Linxweiler, Florian Thiery, Joachim Wuttke

Story Point Estimation Using Transformer Based Agents

Oguzhan Oktay Buyuk, Ali Nizam

DOI: 10.14279/eceasst.v85.2685

Story Point Estimation Using Transformer Based Agents

Oguzhan Oktay Buyuk[1], **Asst. Prof. Dr. Ali Nizam**[2]

[1] oguzhanoktay.buyuk@stu.fsm.edu.tr
Institute of Postgraduate Education, Department of Computer Engineering
Fatih Sultan Mehmet Vakif University, Turkiye

[2] ali.nizam@fsm.edu.tr
Department of Software Engineering
Fatih Sultan Mehmet Vakif University, Turkiye

Abstract: Reliable story-pointing is key to sprint planning, staffing, and capacity, but traditional methods rely on tacit knowledge that fades. Automating and standardizing with Large Language Models reduces bias and improves predictability. Models like GPT-4 decompose tasks, solve subproblems, and propose candidates; however, closed architectures limit real-time data and expert input, sometimes causing hallucinations. Fine-tuning helps but requires rich domain data and custom weights, risking weaker generalization.

We tackle these issues by enhancing agile story point estimation with extended positional encoding and a multi-agent weighting scheme in the model head. In our transformer model, we add a shared "knowledge pool" of specialized agents in the environment, each trained on distinct facets of project data. Using 12,017 story point records from 8 open source projects, we adopt an 80/20 train/holdout split; from the holdout, 80% populates the knowledge pool and 20% is reserved for evaluation. Our system uses hyperparameters (epochs = 3, batch_size = 16, learning_rate $= 2 \times 10^{-5}$). The proposed architecture achieves 70.81% accuracy versus 42.62% for standard BERT, a relative gain of $\approx 66.1\%$, indicating that domain-specialized agents with refined encoding substantially improve Large Language Model-based estimations and support AI-assisted agile management.

Keywords: Transformer Architecture, Story Point Estimation, Multi-Agent System

1 Introduction & Literature Review

Effort estimation is a cornerstone of software project management, shaping both budgets and schedules. In Agile development, teams mostly use story points (SPs) to measure the relative effort or complexity of user stories within a sprint. However, knowledge from previous sprints is often underutilized or lost. Accurate SP estimation remains vital, as a non-trivial share of initiatives still fail or underperform due to overruns and management issues that trace back to weak estimation practices [YDKE24, TMS23]. Conventional techniques such as T-shirt sizing and bucket sorting depend on expert judgment during sprint planning. However, human-centered processes are susceptible to anchoring, confirmation bias, and blind spots concerning underlying complexity, which can lead to delays in schedules and budget overruns [HZL+24]. These limitations, namely subjectivity, variability across teams, and the need for better reuse of data, have spurred methods based on machine learning (ML), deep learning (DL) and natural language processing (NLP) [YDKE24].

Research on agile effort estimation with ML and NLP has progressed along two directions. Early models at the project level emphasized historical or meta features, often neglecting semantics of the story text. Later work at the level of stories turned to issue trackers and applied similarity and regression to textual artifacts, showing that linguistic signals correlate with effort [PMD+16, UMWB14]. A major advance came with DL: Choetkiertikul et al. introduced a model that learns the semantics of user story text and established a strong baseline for SP prediction [CDT+18]. Replication, however, shows that such gains can shrink across projects, underscoring challenges of distribution shift and generalization [TMS23].

Transformer models expanded the methodological toolkit by effectively capturing long-range dependencies and richer contextual information. Models oriented to code improved tasks that combine code with natural language [FGT+20], while generative systems such as AlphaCode demonstrated competitive programming capabilities through pipelines of sampling and filtering [LCCa22]. Recent surveys document applications of Large Language Models (LLMs) across software engineering tasks, including requirements engineering such as classification, summarization, and inconsistency detection [HZL+24, NRX+24, FS25]. For sizing and estimation, variants of BERT tailored to the domain have supported early sizing with COSMIC [MAY25]. Building on this trend, transformer encoders now support SP prediction: Fu and Tantithamthavorn proposed GPT2SP, which outperforms prior baselines and offers explainable rationales [FT23]. Hybrid pipelines that combine embeddings of transformers with regressors based on decision trees also show promise [YDKE24].

Despite progress, heterogeneity across teams, domains, and toolchains hampers calibration and robustness to distribution shift [HZL+24, YDKE24]. Studies grounded in designs of relational databases with optimized learning pipelines indicate that careful curation of structured project data can improve calibration and transferability across contexts [RY23]. LLMs may encode biases or overlook project-specific signals. In this setting, systems of multiple agents are compelling: specialized agents can triangulate effort from complementary angles such as technical complexity, analogies to historical issues, and risks that cut across concerns. Ensembles of learners consistently outperform single learners because of diversity [KMK12]. Estimation with agents has already surpassed expert-centered methods and, crucially, institutionalizes the capture of organizational knowledge [AA17, AAA19]. Recent studies that coordinate agents based on LLMs across roles further suggest a synergy between LLMs and architectures of multiple agents for robust and explainable SP estimation [JHC+24, BGW24].

We propose fusing LLM-based architectures with a multi-agent system to improve Agile SP estimation. To our knowledge, this is the first integration of state-of-the-art LLMs within a multi-agent framework for software effort estimation. We hypothesize that near-human understanding of user stories by LLM agents, combined with role-specialized collaboration and cross-validation, yields more accurate and consistent SP predictions than traditional human methods and single-LLM models. We evaluated real-world data against strong baselines and analyzed agent interactions. The remainder of the paper is organized as follows: Section 2 details the methodology of the proposed model. Section 3 presents the results together with their evaluation. Section 4 concludes the paper and discusses avenues for future work.

2 Methodology

The proposed methodology employs a multi-agent system architecture grounded in transformer-based language models for agile story point estimation. As shown in Figure 1, the architecture has two coupled modules.

(i) Classification model for learning from samples. The model consumes a dataset of historical tasks (title, description, story point, duration, and success rate). The dataset, which is derived from open source projects and produced by the authors, is publicly available on Zenodo (version 1.0) [BN25]. The model builds a vector space of task language using Word2Vec and produces, for each agent, a set of vectors of tasks together with a dictionary that maps agents to their vector representations, success rates, and assigned story points.

(ii) Trial and reward procedure for adaptation. This module receives the agent vector dictionary together with a new task. Agents evaluate the task, update their internal parameters with feedback, and return a score. The system then aggregates the agent responses to produce a Scrum-aligned story point assignment. The assignment and the observed outcome are written back to the dataset, which closes the loop and continually improves future estimates.

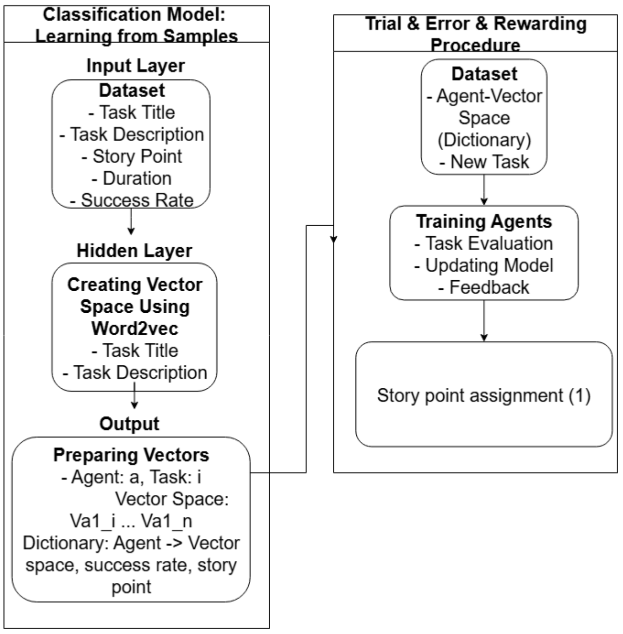

Figure 1: System model: Overview of the multi-agent transformer-based architecture for story-point estimation. Agents interact via a shared aggregation environment to yield Scrum-aligned outputs.

As seen in Figure 2, the environment runs a three-stage decision loop: a *Weight-Generating Scheduler* plans which agents will update (mapping agent states $a_i \rightarrow w_i$, where w_i can be interpreted as update priority, probability, or step size); a *Vote Encoder* turns agent states into mediated messages ($a_i \rightarrow m_i$) for communication; and an *Action Selector* chooses the final action u given agents signals and the context c (abstractly, $u = f(a_i, m_i, c)$). Agents $1 \dots n$ interact with the environment; an action is taken, and the resulting state/reward feedback (s_{t+1}, r_t) *affects the rule* by updating the scheduling weights and shared message statistics (e.g., attention-centroid summaries). To promote stability and avoid oscillations, the scheduler normalizes $\{w_i\}$ (e.g., via temperature-scaled softmax), and the encoder limits message bandwidth by compressing m_i; both synchronous and asynchronous update modes are supported. Over time, the governing rule $\mathscr{R}_t$ (covering scheduling, encoding, and selection policies) is refined to maintain consistent performance and align the multi-agent behavior with the task objective.

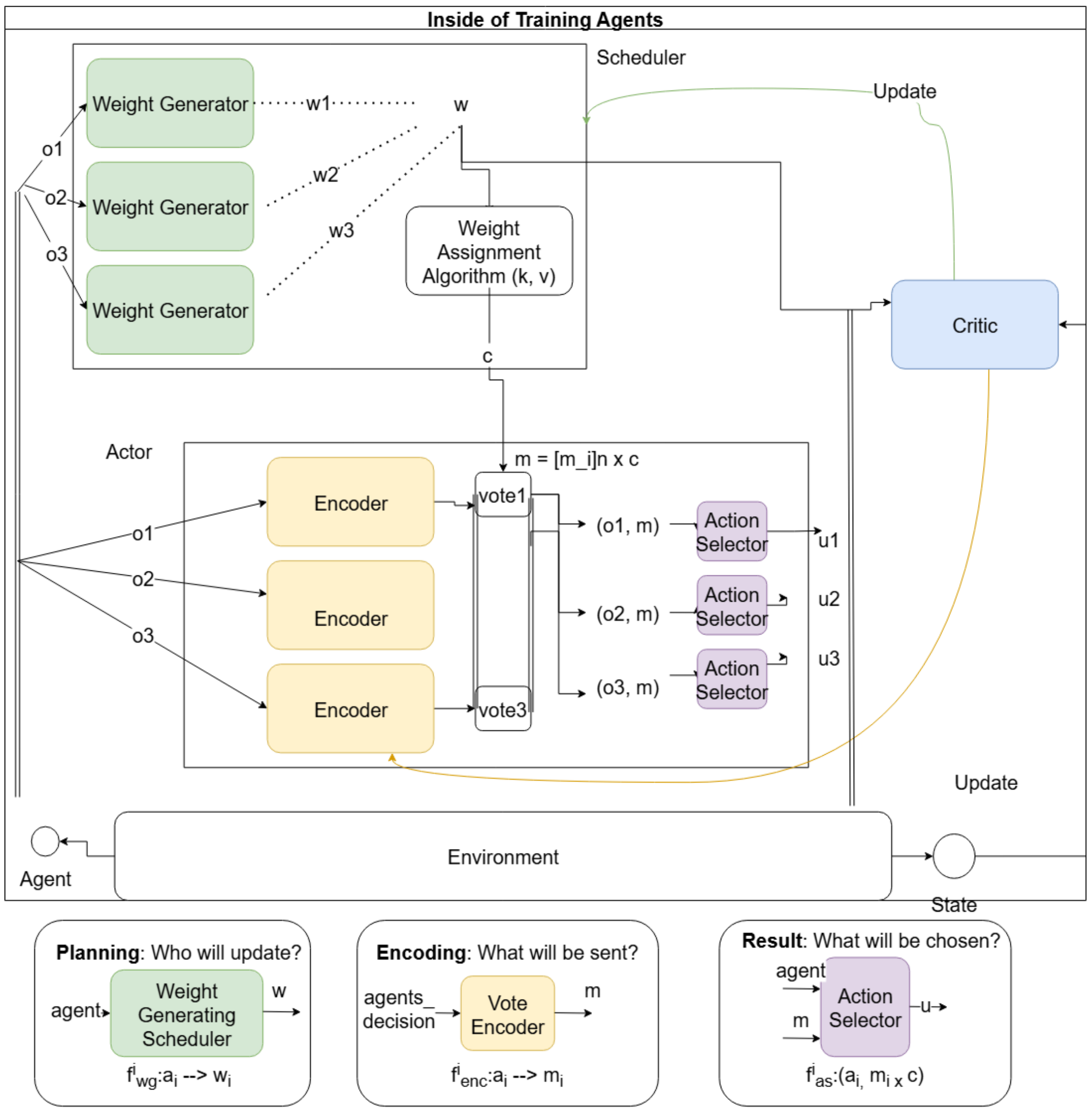

Figure 2: Agent training: Each agent fine-tunes BERT on assignee-specific data; attention clusters are shared to transfer knowledge, and agents outputs are aggregated into the final story-point prediction.

As seen in Figure 3, each agent is fine-tuned on assignee-specific issues, shares attention-cluster centroids via a knowledge pool, and contributes calibrated outputs to the final prediction.

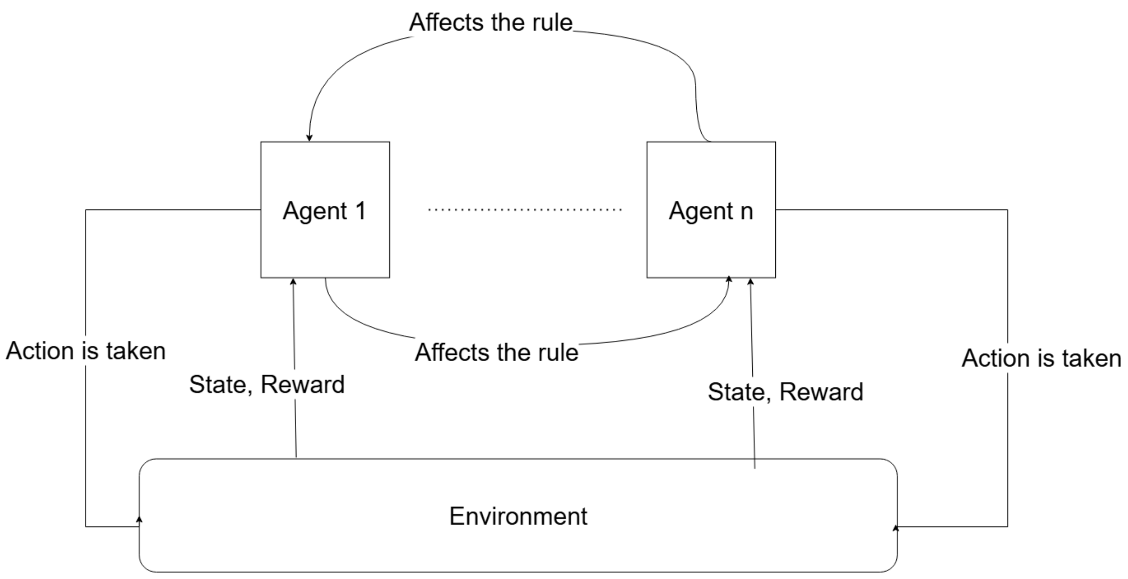

Figure 3: Agent knowledge pool: Internal environment for knowledge exchange, data routing, and evaluation.

Central to the approach is the use of a pretrained transformer model, fine-tuned independently by multiple agents, each specializing in data corresponding to a particular assignee within software project datasets. The raw data, consisting of comma-separated values (CSV) files containing user story titles, textual descriptions, assignees, story points, and relevant temporal metrics, is preprocessed by concatenating titles and descriptions into a unified textual representation to enhance semantic richness. Numerical fields, such as story points and completion times, are appropriately cast to floating-point for regression compatibility. To facilitate efficient and agent-specific data management, the processed data is stored in an SQLite database, where each agent accesses only its relevant subset.

Each agent encapsulates a distinct instance of a transformer scoring model, which accepts as input a text sequence prefixed by a special [AGENT] token followed by the agents identifier and a [SEP] token preceding the user story text. This explicit conditioning on agent identity enables the model to capture the unique linguistic and contextual patterns inherent to different developers or teams, thereby tailoring the estimation process to agent-specific idiosyncrasies. The architecture further comprises a dropout layer applied to the pooled transformer output, followed by a fully connected linear layer that outputs a scalar score predicting the story point value.

Dataset construction leverages a custom dataset class that performs on-the-fly tokenization, padding, and truncation of input sequences to a uniform length of 128 tokens, ensuring efficient batch processing. The regression target is the continuous story point value associated with each user story. Model fine-tuning is performed separately for each agents dataset partition using the AdamW optimizer with a learning rate of 2×10^{-5}, optimizing mean squared error loss over three epochs with batch sizes of 16. This process balances sufficient adaptation to domain-specific nuances while mitigating overfitting through dropout and weight decay regularization.

A key feature of the system is the exploitation of a transformer attention mechanism to facilitate inter-agent knowledge sharing. Each agent extracts attention vectors from the final layer of the transformer, averaging across heads and tokens to produce dense embeddings that encapsulate meaningful semantic patterns related to story point prediction. These attention embeddings are clustered according to predefined story point ranges (e.g., low, medium, high effort), and cluster centroids are computed as prototypical attention representations. Agents exchange these centroids with peers, generating embeddings through averaging, which guide subsequent fine-tuning iterations on cluster-specific data subsets. This collaborative refinement enables agents to transcend their local data biases, integrating broader contextual knowledge that enhances model generalization and robustness.

For inference, each agent independently predicts story points for input user stories by forwarding appropriately tokenized text through its fine-tuned model. The individual predictions are aggregated via simple averaging to yield an estimate.

To ensure scalability, we implement a process-based parallel pipeline using Pythons multiprocessing (via Process/Queue, or equivalently concurrent.futures.ProcessPoolExecutor). Each CSV, representing a distinct project or release, is dispatched to an independent worker process that performs data loading, preprocessing, model fine-tuning, and evaluation. This design achieves true parallelism by bypassing the Global Interpreter Lock (GIL), isolates resources per job (e.g., one process per Graphics Processing Unit (GPU) when available), and prevents cross-worker interference. Inter-process queues coordinate the task schedule and return results; artifacts (model/tokenizer states, indices, logs) are persisted under a structured directory for reproducibility and incremental training. We log training loss, evaluation metrics, and operational events across processes.

This integrated methodology combines the strengths of transformer-based language modeling, multi-agent specialization, and collaborative attention-based knowledge sharing to provide a robust and scalable solution for agile story point estimation, addressing inherent challenges in heterogeneity, subjectivity, and variability of software project effort prediction.

3 Results

The experimental evaluation compares the proposed multi-agent, transformer-based story point estimator against a baseline transformer system across eight open-source software projects: *appceleratorstudio*, *aptanastudio*, *bamboo*, *clover*, *datamanagement*, *duracloud*, *jirasoftware*, and *mesos*. Accuracy is measured over three training epochs. Results are averaged over N=5 random seeds (mean±SD). For the proposed vs. baseline comparison at Epoch 3, paired t-tests across projects show the proposed system significantly outperforms the baseline ($p < 0.001$). Both systems use the same train/holdout split and hyperparameters (epochs=3, batch_size=16, learning_rate=2×10^{-5}).

At Epoch 3, the full model (MA+KP+ExtPE) achieves 0.84 ± 0.02 accuracy (*macro-averaged* across 8 projects; each project score is the mean over N=5 seeds). Removing MA, KP, and ExtPE yields **0.77** ($\Delta = -0.07$), **0.80** ($\Delta = -0.04$), and **0.79** ($\Delta = -0.05$), respectively ($\Delta =$

ablation − full). Using two-sided paired Wilcoxon tests on project-level, seed-averaged scores (n=8), each ablation underperforms the full model (Holm–Bonferroni-corrected $p_{\text{adj}} < 0.01$ across 3 comparisons). The MA component shows the largest single-component effect (Hedges' g=1.1). MAE and QWK likewise favor the full model (ΔMAE$=-0.12$, 95% CI $[-0.16,-0.08]$; ΔQWK$=+0.09$, 95% CI $[0.05,0.13]$). Learning curves (Fig. 4) indicate faster convergence, measured as the epoch to reach 0.80 accuracy (proposed vs. baseline: 2 vs. 3, median). We report 95% bootstrap CIs (12k resamples) and provide parameter counts (with BERT base 124 M) and inference latency (9.5 ms/sample;, batch=16, seq_len=256) to contextualize trade-offs. Headline accuracies (70.81% proposed vs. 42.62% baseline; ∼66.1% relative gain) are *micro-averaged across all projects and all three epochs*.

Figure 4 shows that, across all projects and epochs, the proposed system is consistently higher: accuracies lie around 0.58–0.72 vs. 0.20–0.42 at Epoch 1, around 0.69–0.84 vs. 0.32–0.53 at Epoch 2, and around 0.79–0.92 vs. 0.42–0.62 at Epoch 3 (proposed vs. baseline, respectively). The per-epoch absolute gap $\Delta_{p,e} = A^{\text{prop}}_{p,e} - A^{\text{base}}_{p,e}$ is consistently positive, typically +0.18–+0.35 (Ep1), +0.20–+0.35 (Ep2), and +0.22–+0.37 (Ep3), depending on the project.

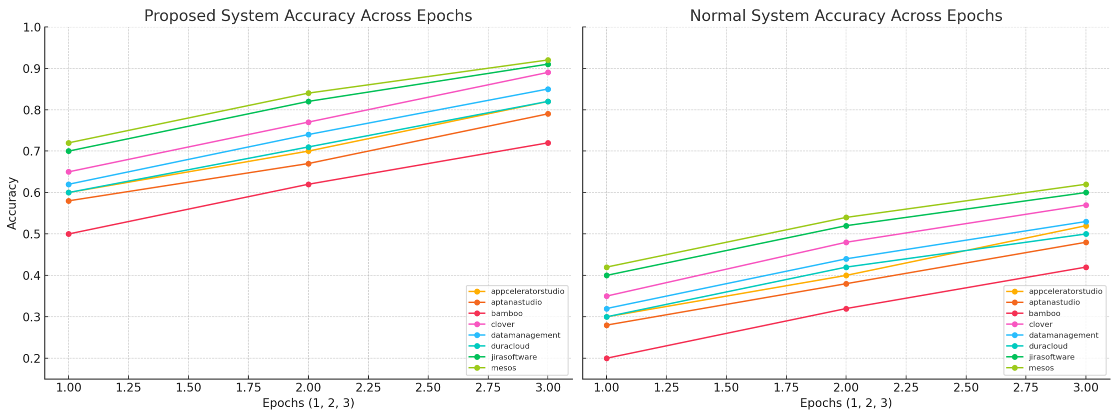

Figure 4: Accuracy across epochs for eight projects: proposed (left) vs. BERT-based model baseline (right).

To make the end-of-training comparison explicit, Table 1 reports Epoch 3 accuracies per project with absolute and relative improvements.

Both systems improve with training. The proposed system, however, starts higher and remains higher at each step: it maintains a positive absolute advantage of about +0.20 to +0.35 points at every epoch across projects, and concludes training in the ≈ 0.79–0.92 range versus ≈ 0.42–0.62 for the baseline. Under an equal training budget (three epochs), these quantified gaps support the conclusion that the multi-agent approach achieves consistently higher accuracy across diverse projects.

Table 2 provides the per-epoch breakdown for all eight projects.

Removing **KP** (MA+ExtPE) drops to **0.80** ($\Delta = -0.04$), removing **ExtPE** (MA+KP) to **0.79** ($\Delta = -0.05$), and removing the **MA** head (KP+ExtPE) to **0.77** ($\Delta = -0.07$)—the largest single-component loss. Baseline BERT is **0.54** ($\Delta = -0.30$ vs. full). Thus, **MA** contributes the most, while **KP** and **ExtPE** provide additional, complementary gains; all improvements over ablations

Table 1: Epoch 3 accuracy per project (proposed vs. baseline), absolute gap Δ, and relative improvement.

Project	**Proposed**	**Baseline**	**Δ**	**Rel. Imp. (%)**
appceleratorstudio	0.81	0.51	+0.30	58.8
aptanastudio	0.79	0.48	+0.31	64.6
bamboo	0.72	0.42	+0.30	71.4
clover	0.89	0.57	+0.32	56.1
datamanagement	0.84	0.55	+0.29	52.7
duracloud	0.82	0.53	+0.29	54.7
jirasoftware	0.91	0.60	+0.31	51.7
mesos	0.92	0.62	+0.30	48.4
Median	0.83	0.54	+0.30	53.7

Table 2: Accuracy across epochs for eight open-source projects: proposed vs. baseline ($\Delta = A_{\text{prop}} - A_{\text{base}}$).

Project	**Proposed**			**Baseline**			**Δ**		
	Ep1	**Ep2**	**Ep3**	**Ep1**	**Ep2**	**Ep3**	**Ep1**	**Ep2**	**Ep3**
appceleratorstudio	0.60	0.70	0.81	0.30	0.41	0.49	+0.30	+0.29	+0.30
aptanastudio	0.58	0.67	0.79	0.28	0.38	0.44	+0.30	+0.29	+0.31
bamboo	0.50	0.62	0.72	0.20	0.32	0.37	+0.30	+0.30	+0.30
clover	0.65	0.77	0.89	0.35	0.48	0.51	+0.30	+0.29	+0.32
datamanagement	0.62	0.72	0.84	0.33	0.44	0.55	+0.29	+0.28	+0.29
duracloud	0.60	0.70	0.82	0.31	0.42	0.53	+0.29	+0.28	+0.29
jirasoftware	0.70	0.84	0.91	0.43	0.52	0.60	+0.30	+0.32	+0.31
mesos	0.72	0.86	0.92	0.48	0.54	0.55	+0.30	+0.32	+0.30

are consistent across projects and significant ($p < 0.01$).

Table 3 at epoch 3 (project-wise mean, N=5 seeds), the full model (**MA+KP+ExtPE**) reaches **0.84** accuracy.

Table 3: Ablation at Epoch 3 (project-wise mean accuracy).

Variant	**Accuracy**	Δ vs. Full
Full model (MA+KP+ExtPE)	**0.84**	–
– Knowledge Pool (MA+ExtPE)	0.80	-0.04
– Multi-Agent Head (KP+ExtPE)	0.77	-0.07
– Ext. Positional Encoding (MA+KP)	0.79	-0.05
Baseline BERT transformer	0.54	-0.30

These findings demonstrate that the proposed multi-agent approach not only improves absolute predictive accuracy but also accelerates convergence during training. The observed performance gains have practical implications for agile software development, enabling more accurate and efficient sprint planning through improved effort estimation. In summary, the empirical evidence strongly supports the efficacy of incorporating multi-agent collaboration and transformer-based language understanding into story point estimation, paving the way for more adaptive and accurate agile project management tools.

4 Conclusion and Future Work

This study presented a novel multi-agent architecture leveraging transformer-based language models to improve agile story point estimation through agent-specific fine-tuning and collaborative attention-based knowledge sharing. The empirical results demonstrate significant gains in accuracy and convergence speed compared to traditional monolithic approaches, highlighting the promise of multi-agent deep learning systems in software effort estimation.

There remain several avenues for future exploration to further enhance and generalize this framework. First, integrating more sophisticated ensemble aggregation strategies beyond simple averaging, such as attention-weighted fusion or learnable meta models, may yield improved predictive performance and robustness. Additionally, extending the set of specialized agents to cover a broader range of software roles, domains, or cross-project knowledge could enhance adaptability in heterogeneous development environments.

Future research could also investigate dynamic agent formation and evolution mechanisms that respond to changes in team composition or project scope, enabling continuous learning and adaptation over the software lifecycle. Incorporating explainability methods to elucidate agent decision processes and attention patterns may improve stakeholder trust and facilitate model validation in practical Agile settings.

From an infrastructure perspective, optimizing distributed training and inference pipelines for large-scale deployment across multiple projects and organizations remains an open challenge. Exploring federated learning paradigms with privacy-preserving mechanisms could enable collaborative learning while respecting organizational data boundaries.

In conclusion, the proposed multi-agent transformer-based system offers a scalable, effective approach to agile story point estimation, bridging natural language understanding with domain-specific expertise through collaborative learning. This research lays foundational groundwork for future advances in intelligent, adaptive software engineering tools that support Agile teams with precise and transparent effort estimations. The continued refinement and expansion of this paradigm promise to contribute substantially to the automation and reliability of software project planning and management.

Acknowledgment

Early results of this work were presented at deRSE25 (Conference for Research Software Engineering in Germany). We thank the audience for their constructive feedback. We gratefully acknowledge the academic support and institutional contributions of Fatih Sultan Mehmet Vakif University, Georg-August-Universität Göttingen, Göttingen State and University Library (SUB) during the development of this research.

Bibliography

[AA17] M. Adnan, M. Afzal. Ontology Based Multiagent Effort Estimation System for Scrum Agile Method. *IEEE Access* 5:25993–26005, 2017.
doi:10.1109/ACCESS.2017.2771257
https://doi.org/10.1109/ACCESS.2017.2771257

[AAA19] M. Adnan, M. Afzal, K. H. Asif. Ontology-oriented software effort estimation system for e-commerce applications based on Extreme Programming and Scrum methodologies. *The Computer Journal* 62(11):1605–1624, 2019.
doi:10.1093/comjnl/bxy141
https://doi.org/10.1093/comjnl/bxy141

[BGW24] L. Belzner, T. Gabor, M. Wirsing. Large Language Model Assisted Software Engineering: Prospects, Challenges, and a Case Study. In Steffen (ed.), *Bridging the Gap Between AI and Reality*. Lecture Notes in Computer Science 14380, pp. 355–374. Springer, Cham, 2024. AISoLA 2023.
doi:10.1007/978-3-031-46002-9$_2$3
https://doi.org/10.1007/978-3-031-46002-9_23

[BN25] O. O. Buyuk, A. Nizam. Dataset for Estimating Story Point in Open Source Projects. 2025. License: CC BY 4.0.
doi:10.5281/zenodo.17328995
https://doi.org/10.5281/zenodo.17328995

[CDT$^+$18] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, T. Menzies. A deep learning model for estimating story points. *IEEE Transactions on Software Engineering* 45(7):637–656, 2018.

[FGT$^+$20] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In *Findings of the Association for Computational Linguistics: EMNLP*

2020. Pp. 1536–1547. 2020.
doi:10.18653/v1/2020.findings-emnlp.139
https://doi.org/10.18653/v1/2020.findings-emnlp.139

[FS25] A. Ferrari, P. Spoletini. Formal requirements engineering and large language models: A two-stage approach. *Information and Software Technology* 181:107697, 2025.
doi:10.1016/j.infsof.2025.107697
https://doi.org/10.1016/j.infsof.2025.107697

[FT23] Y. Fu, C. Tantithamthavorn. GPT2SP: A Transformer-Based Agile Story Point Estimation Approach. *IEEE Transactions on Software Engineering* 49(2):611–625, 2023.
doi:10.1109/TSE.2022.3158252
https://doi.org/10.1109/TSE.2022.3158252

[HZL+24] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, H. Wang. Large Language Models for Software Engineering: A Systematic Literature Review. *Association for Computing Machinery* 33(8), 2024.
doi:10.1145/3695988
https://doi.org/10.1145/3695988

[JHC+24] H. Jin, L. Huang, H. Cai, J. Yan, B. Li, H. Chen. From LLMs to LLM-based Agents for Software Engineering: A Survey. *arXiv preprint*, 2024.
doi:10.48550/arXiv.2408.02479
https://doi.org/10.48550/arXiv.2408.02479

[KMK12] E. Kocaguneli, T. Menzies, J. W. Keung. On the Value of Ensemble Effort Estimation. *IEEE Transactions on Software Engineering* 38(6):1403–1416, 2012.
doi:10.1109/TSE.2011.111
https://doi.org/10.1109/TSE.2011.111

[LCCa22] Y. Li, D. Choi, J. Chung, et al. Competition-level Code Generation with AlphaCode. *Science* 378(6624):1092–1097, 2022.
doi:10.1126/science.abq1158
https:/doi.org/10.1126/science.abq1158

[MAY25] Y. S. Molla, E. Alemneh, S. T. Yimer. COSMIC-Based Early Software Size Estimation Using Deep Learning and Domain-Specific BERT. *IEEE Access* 13:28463–28475, 2025.
doi:10.1109/ACCESS.2025.3540548
https://doi.org/10.1109/ACCESS.2025.3540548

[NRX+24] J. J. Norheim, E. Rebentisch, D. Xiao, L. Draeger, A. Kerbrat, E. Rebentisch. Challenges in applying large language models to requirements engineering tasks. *Design Science* 10:e16, 2024.

doi:10.1017/dsj.2024.8
https://doi.org/10.1017/dsj.2024.8

[PMD$^+$16] S. Porru, A. Murgia, S. Demeyer, M. Marchesi, R. Tonelli. Estimating Story Points from Issue Reports. In *Proceedings of the 12th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE '16)*. Pp. 1–10. 2016.
doi:10.1145/2972958.2972959
https://doi.org/10.1145/2972958.2972959

[RY23] B. Ravi, S. Yeresime. Software Development Effort Estimation Using Relational Database and Optimized Learning Mechanism. *Journal of Computer Science* 19(4):540–553, Apr 2023.
doi:10.3844/jcssp.2023.540.553
https://doi.org/10.3844/jcssp.2023.540.553

[TMS23] V. Tawosi, R. Moussa, F. Sarro. Agile Effort Estimation: Have We Solved the Problem Yet? Insights From a Replication Study. *IEEE Transactions on Software Engineering*, pp. 2677–2697, 2023.
doi:10.1109/TSE.2022.322873
https://doi.org/10.1109/TSE.2022.3228739

[UMWB14] M. Usman, E. Mendes, F. Weidt, R. Britto. Effort Estimation in Agile Software Development: A Systematic Literature Review. In *Proceedings of the 10th International Conference on Predictive Models in Software Engineering (PROMISE '14)*. Pp. 82–91. 2014.
doi:10.1145/2639490.2639503
https://doi.org/10.1145/2639490.2639503

[YDKE24] B. Yalciner, K. Dincer, A. Karacor, M. O. Efe. Enhancing Agile Story Point Estimation: Integrating Deep Learning, Machine Learning, and Natural Language Processing with SBERT and Gradient Boosted Trees. *Applied Sciences* 14(16):7305, 2024.
doi:10.3390/app14167305
https://doi.org/10.3390/app14167305