SYM-AM-24-080

Excerpt from the Proceedings of the Twenty-First Annual Acquisition Research Symposium

Acquisition Research: Creating Synergy for Informed Change

May 8–9, 2024

Published: April 30, 2024

Prepared for the Naval Postgraduate School, Monterey, CA 93943.

The research presented in this report was supported by the Acquisition Research Program at the Naval Postgraduate School.

To request defense acquisition research, to become a research sponsor, or to print additional copies of reports, please contact any of the staff listed on the Acquisition Research Program website (www.acquisitionresearch.net).

ACQUISITION RESEARCH PROGRAM
DEPARTMENT OF DEFENSE MANAGEMENT
NAVAL POSTGRADUATE SCHOOL

Systems Acquisition Cost Modeling Initiative for AI Assistance

Ray Madachy—is a research staff member at the Institute for Defense Analyses (IDA), with expertise in legal, regulatory, and financial matters relating to the DoD. Prior to joining IDA in 2005, he held a number of positions in the commercial sector. He holds a bachelor's degree in economics from Dickinson College; a Doctor of Jurisprudence degree from Georgetown University Law Center; and a Master of Business Administration degree from The University of Chicago Booth School of Business. [rjmadach@nps.edu]

Ryan Bell—is a research staff member at the Institute for Defense Analyses (IDA) specializing in contract matters relating to the DoD. She has a doctoral degree in economics from The Ohio State University. [ryan.bell@nps.edu]

Ryan Longshore—is a research staff member at the Institute for Defense Analyses (IDA) specializing in contract matters relating to the DoD. She has a doctoral degree in economics from The Ohio State University. [ryan.longshore@nps.edu]

Abstract

Artificial Intelligence (AI) based tools that assist in generating system artifacts are transforming systems and software engineering lifecycles. Drastic reductions in effort are possible using tools that use large language models (LLMs). This research addresses the new challenges in systems and software cost modeling with the introduction of cost factors and size measures to incorporate into existing parametric cost models.

Keywords. Systems Acquisition Cost Modeling, Parametric Cost Modeling, Artificial Intelligence, Generative Artificial Intelligence, Large Language Models (LLMs), Systems Engineering, Software Engineering, COSYSMO, COCOMO

Introduction and Background

A disruptive transformation is occurring on how the Navy and the rest of the DoD develops, delivers, and sustains systems due to AI assistance in the systems and software engineering processes. AI tools can support virtually all non-hardware production lifecycle aspects from concept, AoA, architecture, requirements, design, software, V&V, testing, etc. A research goal is to better understand and codify the advantages and pitfalls of integrating AI into systems and software processes. We consider the benefits, challenges, dangers of over-reliance and potential inefficiencies.

We have observed that drastic reductions in effort are possible using AI assistant tools that use large language models (LLMs). LLMs are a type of generative AI that utilize a deep learning algorithm to generate human-like text based on natural language prompts. One typically interfaces with a chatbot such as ChatGPT, Gemini, Claude, Copilot and many others. They are well suited for tasks such as language translation, text summarization, and question answering. Some LLMs are exceptionally good at generating code and text-based system models like SysML 2.

AI-based tools can assist in generating system artifacts across virtually all phases and activities. The research initiative examines the quantitative AI impacts by lifecycle phase and activity since their effects may vary greatly. Examples of potential cost benefits and risks for traditional technical processes in systems and software engineering are shown in Tables 1 and 2.

Table 1. Example Engineering Activity Cost Benefits

Activity	Benefits
Require-ments	AI tools can help clarify terminologies and concepts on-the-fly, reducing the need for prolonged meetings or external consultations. Developers or analysts can query AI tools for insights or comparative analysis, which might help in re-fining requirements.
Design	Engineers can quickly consult AI tools for recommended design patterns or ar-chitectural practices suitable for their problem. Preliminary design ideas can be discussed with AI tools for quick feedback.
Code	Developers can seek assistance on coding challenges, syntax, and algorithmic solutions. AI tools can assist in code reviews, highlighting potential pitfalls or anti-patterns.
Testing and Integration	AI tools can suggest potential edge cases or testing scenarios. For failing tests or integration issues, developers can discuss potential causes and solutions with AI tools.
Maintenance	AI tools can assist in understanding old codebases, suggesting potential refac-toring techniques, or identifying deprecated methods. For known errors or bugs, AI tools can suggest common solutions or workarounds.

Table 2. Example Engineering Activity Cost Risks

Activity	Risks
Require-ments	Relying too much on AI tools for domain-specific knowledge might lead to missed nuances that an expert in the field would be aware of.
Design	Over-relying on AI for design decisions without human review can lead to subop-timal choices.
Code	If developers use AI tool suggestions verbatim without understanding, it might introduce bugs or inefficient code. Waiting on AI tool responses for every small issue can become a crutch and delay development if developers stop trying to problem-solve on their own.
Testing and Integration	If AI tool suggestions are taken without thorough review, it might lead to unnec-essary tests or efforts spent on non-issues. If teams over-rely on AI tools for maintenance, they might overlook deeper architectural or design issues that re-quire human expertise.
Maintenance	AI tools can assist in understanding old codebases, suggesting potential refac-toring techniques or identifying deprecated methods. For known errors or bugs, AI tools can suggest common solutions or workarounds.

This research is using the comprehensive IS0 15288 lifecycle standard for systems and software engineering (International Organization for Standardization [ISO], 2015) as a framework for data collection and analysis of detailed activities. The activities in Tables 1 and 2 contain a subset of the technical processes in ISO 15288. While this initiative is first addressing software development cost with formal definitions and data collection, we are performing allied research in systems engineering AI assistance and cost impacts.

This research is quantifying the impact against standard systems and software engineering tasks, generating and analyzing artifacts with the assistance of AI. Benchmarks are first necessary for well-defined engineering tasks for quantification and reproducibility.

SysEngBench is being designed to evaluate LLMs in the context of systems engineering concepts and applications to support benchmarking (Bell et al., in press). It will encompass a comprehensive set of tasks derived from core systems engineering processes, including requirements analysis, system architecture design, risk management, and stakeholder communication. By leveraging a diverse array of real-world and synthetically generated scenarios, *SysEngBench* aims to provide an assessment of LLMs' ability to interpret complex engineering problems and generate innovative solutions.

Our research also explores the ability of current LLMs to generate, modify, and query Systems Modeling Language (SysML) v2 models (Longshore et al., in press). Techniques such as Retrieval-Augmented Generation (RAG) are utilized to add domain-specific knowledge to an LLM and improve model accuracy. Preliminary case studies indicate that the number of prompts to generate models can be minimized.

Method

This research addresses the new challenges in systems and software cost modeling with model definitions, a lifecycle standard, and data analysis process to calibrate the model. This includes new cost factors and size measures to incorporate into existing parametric cost models. Empirical data collection and analysis for model calibration is also underway.

Already, there is very strong convincing data that substantial labor can be saved in steady-state AI tool usage by individuals and teams. To address the cost impacts, we have developed a road map for advancing the cost models by leveraging existing modeling and measurement frameworks. We are using the Constructive Cost Model (COCOMO) framework and calibration procedures (Boehm, 1981; Boehm et al., 2000).

The cost modeling and measurement framework incorporates a new factor for "AI Assistance Usage" with a defined rating scale and data analysis process to calibrate it. An online data collection and Delphi survey to improve the model with expert judgment has been developed for the community. A new measure, "query points," is being refined to quantify the size and complexity of the AI generated solutions.

From systems and software engineering case studies, we are gathering empirical data on generated solution sizes, actual effort, and effort estimates without AI assistance. Subsequent case studies will address larger scale team and enterprise processes assisted with AI.

Parametric Modeling

The general effort formula used in the parametric systems and software cost models is:

$$Effort = A * Size^{B} * \prod_{i=1}^{N} EM_i$$

where

- *Effort* is in Person-Months (PM)
- *A* is a constant derived from historical project data
- *Size* is a measure of the work product

- *B* is an exponent for the diseconomy of scale
- EM_i is an effort multiplier for the i^{th} cost driver. The geometric product of *N* multipliers is an overall Effort Adjustment Factor (EAF) to the nominal effort.

Size is interpreted within the context of the specific work being estimated in the units of the work products. The effort multipliers cover factors for product, platform, personnel and project attributes that affect cost. The resulting top-level effort can be decomposed for each phase, activity or increment.

Example Cost Driver and Effort Multipliers

An example cost driver, *Applications Experience*, from COCOMO II is visualized in Figure 1 with its effort multipliers. The effort multipliers for each rating represent the relative effort to Nominal. For example, the EM for a Very Low rating of Applications Experience is 1.22, indicating a 22% increase in effort from Nominal. The Nominal rating is always 1.0 by definition for a typical project. The High rating EM is 0.88, or a 12% decrease in effort from Nominal. The overall Effort Multiplier Ratio (EMR) for Applications Experience is the ratio of the highest to lowest multipliers, or 1.22/.81 = 1.5.

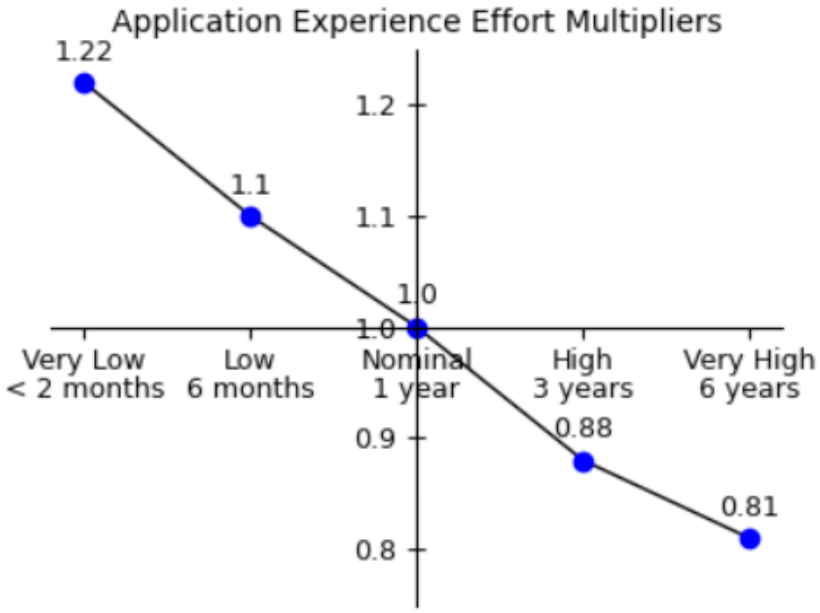


Figure 1. Application Experience Effort Multiplier Example

The initial rating scale for the proposed cost driver "AI Assistance Usage" has been defined using the COCOMO framework. It consists of five ratings from Very Low to Very High, corresponding to the degree of AI usage on a project per Table 3. The default setting is Nominal, corresponding to a typical project. The data collection will be used to calibrate the effort multipliers for each rating level.

Table 3. AI Assistance Usage Initial Rating Scale

Very low	Low	Nominal	High	Very High
Minimal to no AI assistance. Development relies primarily on traditional methods and tools. AI tools may be present but are rarely, if ever, consulted.	Occasional AI consultation, typically for clarification or basic information retrieval. AI tools are not deeply integrated into the development workflow.	Regular use of AI tools for various tasks like code help, design insights, or testing assistance. AI tools are a recognized part of the toolkit but aren't central to development	Frequent and strategic use of AI assistance. AI tools play a central role in multiple phases of development, from design to code review.	AI tools are deeply ingrained in most development phases. They are crucial for decision making, problem solving, and automating specific tasks. The development process is designed around maximizing AI tool benefits.

We have also identified other affected cost factors and parameters for using generative AI. For example, the relative cost of achieving reliability may change, and AI may help reduce impacts of experience and capability. Overall cost model coefficients will change. Usage of AI will shortly become an assumed skillset of engineers. Subsequent data collection will help us assess these impacts as well.

The initial definition is oriented to software. The factor definition and its data collection are setting the stage for further exploration into systems engineering process impacts. We are defining an analogous usage factor for systems engineering to incorporate in the Constructive System Engineering (COSYSMO) model (Valerdi, 2005). In our research, we are generating SysML 2 artifacts and capturing data on effort, solution accuracy, size and complexity for activities covered in COSYSMO. In additional case studies, we will collect similar data from large team projects.

We are also investigating phase sensitive effort multipliers to account for different AI tool impacts across the lifecycle. We are codifying the practices by phase and activity, and empirical data collection is being aligned with those in the ISO/IEC/IEEE 15288 lifecycle.

Data Collection and Analysis

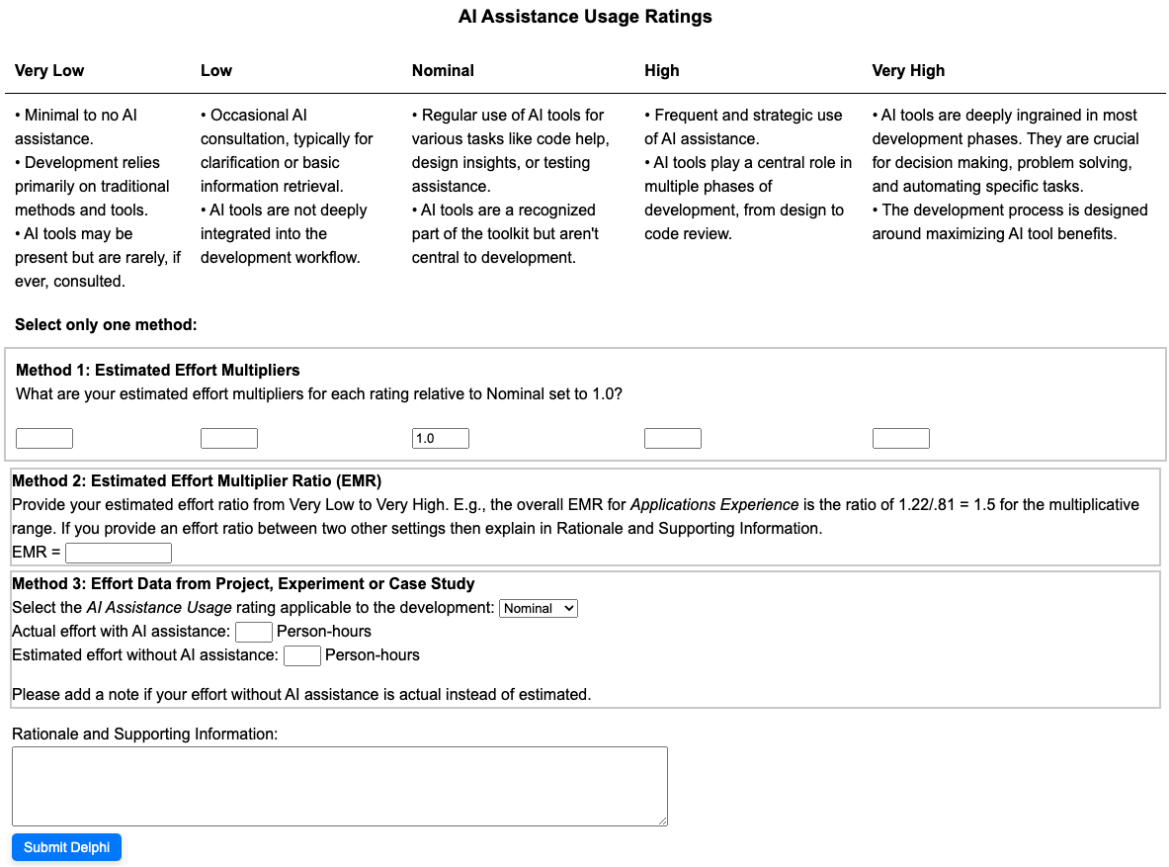
Different empirical methods are used for parametric cost model analysis and calibration. These include:

- multi-project data collection in conjunction with other cost factors (e.g., COCOMO II; Boehm et al., 2000)
- controlled group experiments (e.g., Github Copilot; GitHub, 2022)
- Delphi surveys for expert judgment
- Bayesian approaches combining empirical project data and Delphi results (e.g., COCOMO II; Boehm et al., 2000; Chulani et al., 1999)
- small-scale empirical case studies and expert judgment

A variety of data sources are being drawn from to support model calibration and provide insights. Multi-project data collection in conjunction with other cost factors is going forward for the COCOMO III model. Small-scale empirical case studies and controlled group experiments are being performed. We are also collecting classroom data.

We have developed an online Delphi survey form to capture both expert judgment and actual data to help calibrate the model. A portion is shown in Figure 2. It is available at

http://softwarecost.org/data/ai. The data collection is being supported by the Boehm Center for Systems and Software Engineering.



AI Assistance Usage Ratings

Very Low	Low	Nominal	High	Very High
• Minimal to no AI assistance. • Development relies primarily on traditional methods and tools. • AI tools may be present but are rarely, if ever, consulted.	• Occasional AI consultation, typically for clarification or basic information retrieval. • AI tools are not deeply integrated into the development workflow.	• Regular use of AI tools for various tasks like code help, design insights, or testing assistance. • AI tools are a recognized part of the toolkit but aren't central to development.	• Frequent and strategic use of AI assistance. • AI tools play a central role in multiple phases of development, from design to code review.	• AI tools are deeply ingrained in most development phases. They are crucial for decision making, problem solving, and automating specific tasks. • The development process is designed around maximizing AI tool benefits.

Figure 2. Data Collection Form Portion

Ideal Effort Multiplier

A method is needed to isolate the effect of AI assistance among the contaminating effects of other parametric cost factors when analyzing multiple project datapoints. The Ideal Effort Multiplier (IEM) method will be used to determine calibrated multipliers for each project and perform regression across the rating scale to attain global effort multipliers for the model. The IEM quantifies the contribution of AI Assistance Usage eliminating other cost factor sources per:

$$IEM(P, \text{Cost Factor}) = PM(P, \text{actual}) \,/\, PM(P, \text{Cost Factor})$$

where

- *IEM(P*, Cost Factor) is the ideal effort multiplier for project *P*
- *PM* is Person-Months of effort
- *PM(P*, actual) is the actual development effort of project *P*
- *PM(P*, Cost Factor) is the cost model estimate excluding the Cost Factor

The IEM method is visualized in Figure 3 with representative values from COCOMO II. The values at each setting represent the best fit against all the project data points.

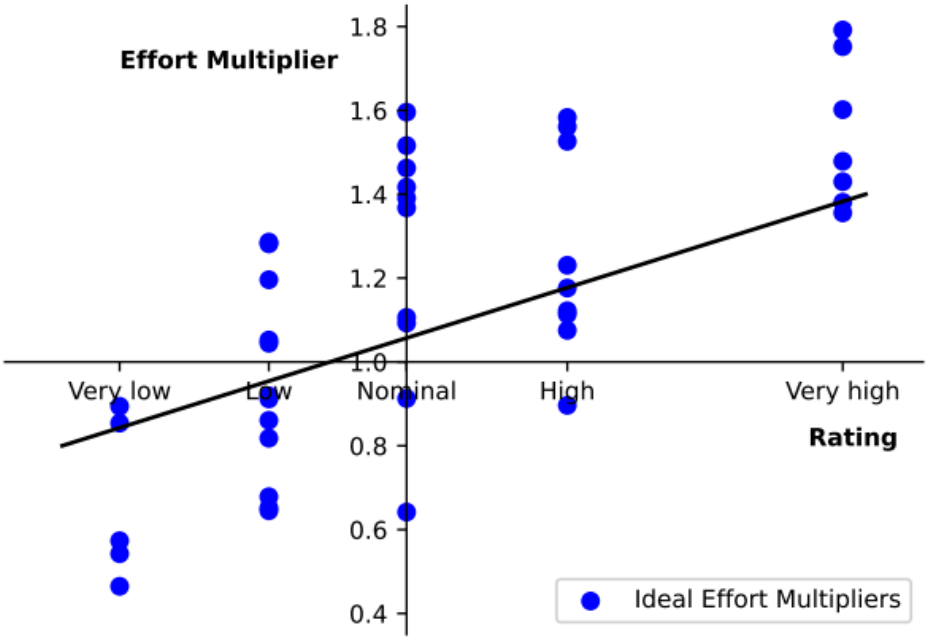


Figure 3. Ideal Effort Multiplier Normalization Method

Detailed Phase Impacts

Phase-sensitive effort multipliers account for different impacts across the lifecycle versus the project level effects in the previous multipliers. This is necessary because some cost drivers vary more across phases than others. An example is shown in Figure 4 from the Detailed COCOMO model for the factor *Use of Software Tools* (Boehm, 1981). It is a highly relevant example because software tools now include AI assistance.

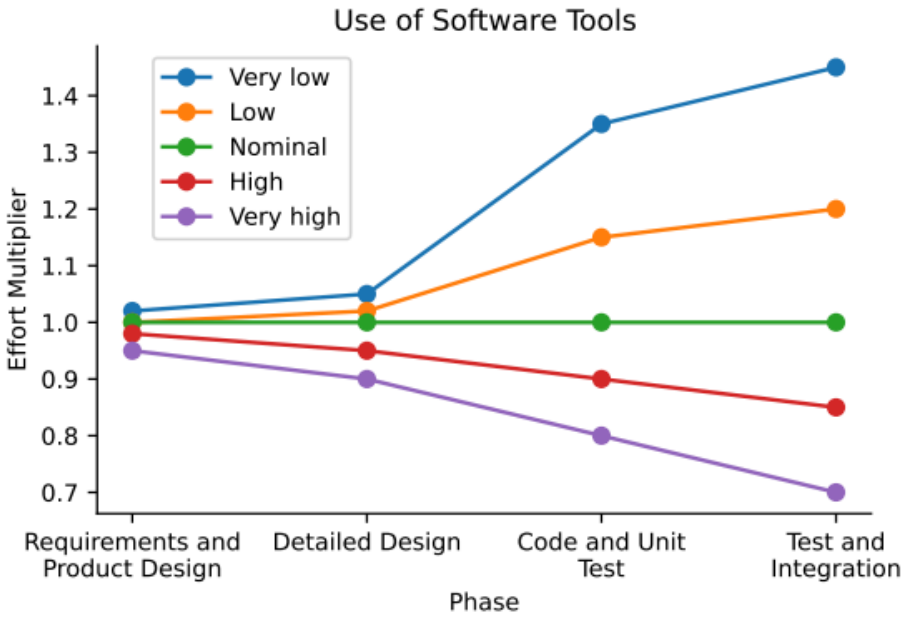


Figure 4. Phase Sensitive Effort Multiplier Example

Empirical data collection will be commensurate with phases so that AI impact will be evaluated by both phases and activities. This will include codifying the practices and ratings with detailed descriptions and examples.

Current and Future Work

We are early in the initiative, and the community is highly encouraged to provide feedback on the model definitions and submit data and feedback on the data collection. Community support is imperative to develop the new models. We are instituting the Delphi data collection and will continue iterative analysis with COCOMO III research to update the models. For this, we will provide open-source cost modeling tools with new factor(s) in the models for public usage.

A current research focus is on how query task complexity impacts AI correctness and effort impact. We are elaborating query points as a complexity measure for this to measure SysML 2 model artifacts. We will perform further analysis of AI tool impacts across lifecycle aligning artifacts and effort data with ISO/IEC/IEEE 15288 systems and software engineering phases and activities. This harmonization will also help address large scale team and enterprise processes assisted with AI.

References

Bell, R., Madachy, R., & Longshore, R. (in press). Introducing SysEngBench: A novel benchmark for assessing large language models in systems engineering. *Proceedings of the 2024 Acquisition Research Symposium*.

Boehm, B., Abts, C., Brown, W., Chulani, S., Clark, B., Horowitz, E. … Steece, B. (2000). *Software cost estimation with COCOMO II*. Prentice-Hall.

Chulani, S., Boehm, B., & Steece, B. (1999, December). Bayesian analysis of empirical software engineering cost models. In *IEEE Transactions on Software Engineering*, *25*(4).

GitHub. (2022). *Research: Quantifying GitHub Copilot's impact on developer productivity and happiness*.

https://github.blog/2022-09-07-researchquantifying-github-copilots-impact-on-developerproductivity-and-happiness.

International Organization for Standardization. (2015). *ISO/IEC 15288:2015 Systems engineering – System life cycle processes*. Tech. Rep. International Organization for Standardization.

Longshore, R., Madachy, R., & Bell, R. (in press). Leveraging generative AI to create, modify, and query MBSE models. *Proceedings of the 2024 Acquisition Research Symposium*.

Valerdi, R. (2005). *The constructive systems engineering cost model (COSYSMO)* [Doctoral dissertation, University of Southern California].

Acquisition Research Program
Department of Defense Management
Naval Postgraduate School
555 Dyer Road, Ingersoll Hall
Monterey, CA 93943

www.acquisitionresearch.net