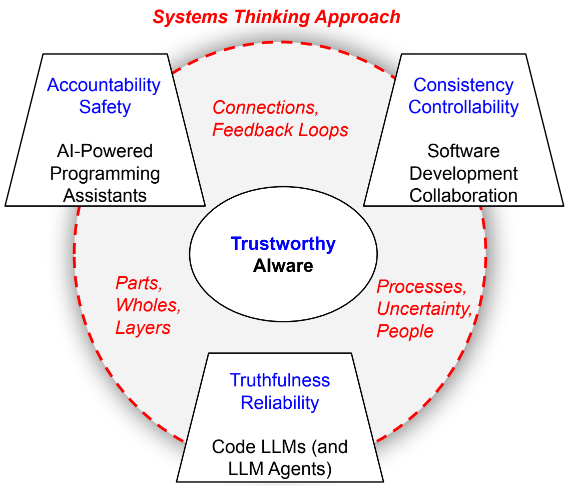
Jie "JW" Wu — Research Statement

I wouldn't be here as an academic researcher without my childhood experience learning Euclid geometry. The simple, powerful yet mysterious axioms that explain complex phenomena deeply impressed me and motivated me to pursue research dreams of using science to simplify complex phenomena, and pulled me back from industry to a postdoc at the University of British Columbia. Having worked as a software engineer in the industry for a decade at Snapchat, Microsoft, and a startup, I envision that the industries are transitioning from software to *AIware* [1], i.e., AI-powered software, that will greatly democratize software creation and ease the jobs of 27 million professional software engineers in the world. **How to build reliable and safe software is critical for society, especially in the current era of *Large Language Models (LLMs)*.**

The goal of my research is to build trustworthy AIware. To achieve this, I'm pioneering a systems thinking approach that emphasizes connections, feedback loops, layers, and new processes to address the trustworthiness issues in AIware. I often start with the question: "What the AI model cannot do that the software engineer can do?", and then find events that do not work well for AI models. I don't stop there and further analyze the patterns and systemic structures in the systems thinking iceberg. For example, I studied the truthfulness of Code LLMs by creating a new benchmark for them, called *HumanEvalComm* [2], in their ability to ask clarifying questions when needed. I further invented a new LLM-based agent framework, *Okanagan* [2], and a new instruction fine-tuning method, *ClarifyCoder* [3], to improve Code LLMs' ability to ask clarifying questions. In the future, I will continue to use a systems thinking approach to enhance more dimensions of trustworthiness such as controllability, safety, and robustness for AIware components. **Ultimately, I believe, systems thinking will provide a *new perspective* that complements the statistical and reductionist view, a *new language* for communicating AIware components and relationships, and *new tools* that form fundamental, practical building blocks in the evolution of trustworthy AIware.**

Background

As we are entering the new era of AIware, I argue that systems thinking is more important given the stochastic and black-box nature of large language models (LLMs). Systems thinking is not only thinking systematically but, more importantly, considering the whole system and how its parts are interconnected, and developing a deep understanding of the underlying structure to make reliable inferences. Systems thinking typically involves moving back and forth between different levels of abstraction to develop a more complete picture [4]. This approach goes beyond single *Events*, into *Patterns* (Emergence), *Systemic Structures* (Relationship Among Components, Feedback Loop) and *Mental Models* (Culture, Values, Paradigms). Hence, this approach helps in identifying patterns, predicting outcomes, and finding leverage points for effective intervention. My research vision is to enhance trustworthiness in the growing autonomy of AI-powered software engineering through a systems thinking lens.


Systems Thinking Approach
Accountability
Safety
AI-Powered Programming Assistants
Connections, Feedback Loops
Consistency
Controllability
Software Development Collaboration
Trustworthy
AIware
Parts, Wholes, Layers
Processes, Uncertainty, People
Truthfulness
Reliability
Code LLMs (and LLM Agents)

Current Research

1) Truthfulness in Code LLMs. Here I illustrate the use of systems thinking with an example. Let us take a step back and view the AI model as a holistic system. If the AI model is given unlimited resources and time for a software engineering task, the model has to get the necessary information for it to complete the task, and the best way to get this information is by asking questions. In my recent work [2], based on the observation that **top-level software engineers often ask clarifying questions to reduce ambiguity in both requirements and coding solutions, I argue that the same should be applied to LLMs for code generation tasks** [5]. I created a new benchmark and empirical study to evaluate the communication skills of large language models (LLMs) in code generation. My findings indicate that modifying the problem description significantly reduced both the Test Pass Rates and Pass@1 metrics. In terms of communication skills, over 60% of responses from code LLMs continue to generate code rather than ask clarifying questions when the problem descriptions are manually modified.

Besides benchmarking the communication skills of Code LLMs, we further proposed an LLM-agent approach, Okanagan, to enhance the communication capability of the models, and thus lead to better code generation capability [2]. The contribution of Okanagan is a multi-round structure with customized prompts for asking clarifying questions when needed in code generation tasks. Furthermore, motivated by the observation that the instruction tuning methods in Code LLMs enforce the models to generate code [6], regardless of any issues in the problem description, my ongoing work further proposes a new instruction-tuning approach, called ClarifyCoder, to empower the model to have the ability to ask clarifying questions when needed for coding tasks [3].

2) Accountability in Programming Assistant. During my PhD, I leveraged the systems thinking mindset and focused on the issue in an industrial context, creating online A/B tests [7], evaluating online A/B testing [8, 9], and creating offline A/B tests (i.e. offline-policy evaluation) [10] are manual processes. I developed several approaches [7, 8, 9, 10] to automate these processes. For example, for evaluating online A/B testing, I identified the problem that a formal, generalized, systematic framework is required to assist the decision-makers in making launch decisions using A/B testing results. I proposed an MCDM-based framework to address this problem using A/B testing results [8]. My work indicates that the empirical decision-making process from analyzing big data could be formalized and automated.

Furthermore, for creating online A/B tests, I formulated the Variant Creation and Evaluation (VCE) problem in A/B testing and proposed MOVSW (Multi-Objective Variant Selection Wrapper) that utilizes a Multi-Objective Evolutionary Algorithm (MOEA) to automate the process of creating and selecting variants as launch candidates in A/B testing [7]. I found that MOEA/D is the most stable method with fast convergence compared with other MOEAs. In the era of AIware, I'm working to develop an LLM-based agent approach to reliably automate the creation of more sophisticated changes or features rather than only automate the adjustment of parameter values in the system. **I've started initial discussions with software companies from the industry, including Coupang and Snapchat, for potential collaboration and adoption.**

3) Consistency in Software Development Collaboration. My recent work [11] focused on the problem that traditional process models (such as waterfall, and agile model) are limited in interdisciplinary collaboration when building products with ML components. I developed a set of propositions based on interviews with practitioners in several software companies and proposed V-model, originated from systems thinking, to address this problem. My study revealed that despite requiring additional efforts, **the characteristics of V-Model align effectively with collaboration challenges encountered by practitioners when building ML-enabled systems.** I believe this work opens up the next steps to enhance consistency in interdisciplinary, human-bot collaborations of AIware that leverage the characteristics of V-Model such as the system decomposition, clear system boundary, and consistency of Validation & Verification (V&V) for building ML-enabled systems where bots co-evolve with humans with different roles. This work received the Distinguished Paper Award Candidate at CAIN 2024 and was selected to appear in IEEE Software Practioners' Digest column.

Future Direction: Systems Thinking for Trustworthy AIware

I envision the future to be increasing autonomy in AI-powered software engineering, where we are rapidly moving from no automation (manual coding) to partial automation, and ultimately to high or full automation, in a way similar to the autonomy in the self-driving field. However, I do see the high complexity in the system and the identified issues in this process, such as missing interaction, missing steps in the process, missed opportunities for higher trust, and better controllability.

Use Feedback Loop for Reliable Data-driven Automation. Existing automation tools in software engineering lack the ability to take the feedback loop to perform automation reliably. The most recent trend is using LLM-based agents to iterate the code generation process. With my previous tool-building experiences such as online sketch recognition [12, 13] and offline sketch parsing [14] for efficient flowchart creation, I aim to build practical tools to integrate the recent advances in LLM-based agent and feedback loop in a principled way for automated data-driven experiments in the AIware lifecycles. I will focus on vertical domains such as recommender systems where many data-driven insights are available for automation.

Use Holistic View for Enhanced Consistency Across AIware Lifecycles. Software artifacts play a crucial role in software development and maintenance because professional software engineering activities go beyond merely writing code. Some artifacts help describe the function, architecture, and design of software, such as design docs, specifications, class diagrams, meeting conversations, etc. Other artifacts are related to the development process, such as project plans and business cases. However, much less attention was received in developing holistic models that reason and perform at various software artifact levels, beyond the code level. In the era of AIware and LLM, I will investigate how the model of artifacts beyond code performs on attributes like predictability, ability to correct errors, and ability to refresh from new updates is insufficiently known. The objective is to systematically evaluate the models' ability to predict, correct, and refresh software artifacts to preserve the single source of truth.

Use Connections for Safe Controllability in Collaboration. The degree of controllability is still low in interdisciplinary and Human-Robot collaborations for AIware. It is important to have robust control of AIware during collaborations to ensure AI safety and to meet requirements in compliance and regulations. I plan to focus on 'connections' between components to develop control mechanisms to enhance robust controllability in both interdisciplinary and Human-Robot collaborations.

References

[1] Ahmed E Hassan, Dayi Lin, Gopi Krishnan Rajbahadur, Keheliya Gallaba, Filipe Roseiro Cogo, Boyuan Chen, Haoxiang Zhang, Kishanthan Thangarajah, Gustavo Oliva, Jiahuei Lin, et al. Rethinking software engineering in the era of foundation models: A curated catalogue of challenges in the development of trustworthy fmware. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, pages 294–305, 2024.

[2] Jie JW Wu and Fatemeh H. Fard. Benchmarking the communication competence in large language models for code generation. *ACM Transactions on Software Engineering and Methodology*, 2024. (Accept with Minor Revision).

[3] Jie JW Wu, Manav Chaudhary, and Fatemeh H. Fard. Clarifycoder: Instructing code large language models to ask clarifying questions. (In Preparation).

[4] Marc Steen. The problem with the trolley problem and the need for systems thinking. *Communications of the ACM*, 67(6):30–32, 2024.

[5] Jie JW Wu. Large language models should ask clarifying questions to increase confidence in generated code. *The 7th Annual Symposium on Machine Programming (MAPS '23 Workshop)*, 2023.

[6] Sathvik Joel, Jie JW Wu, and Fatemeh Hendijani Fard. Survey on code generation for low-resource and domain-specific programming languages. *ACM Computing Surveys (ACM CSUR)*, 2024. (Under Review).

[7] Jie JW Wu, Thomas A. Mazzuchi, and Shahram Sarkani. A multi-objective evolutionary approach towards automated online controlled experiments. *Journal of Systems and Software*, 2023.

[8] Jie JW Wu, Thomas A. Mazzuchi, and Shahram Sarkani. Comparison of multi-criteria decision-making methods for online controlled experiments in a launch decision-making framework. *Information and Software Technology*, 155:107–115, 2023.

[9] Jie JW Wu. *Towards Formalizing Data-Driven Decision-Making from Big Data: A Systematic Multi-Criteria Decision-Making Approach in Online Controlled Experiments*. PhD thesis, The George Washington University, 2023.

[10] Jie JW Wu. Autooffab: Toward automated offline a/b testing for data-driven requirement engineering. In *Foundations of Software Engineering (FSE), Ideas, Visions and Reflections Track*, 2024.

[11] Jie JW Wu. An exploratory study of v-model in building ml-enabled software: A systems engineering perspective. In *3rd International Conference on AI Engineering – Software Engineering for AI (CAIN 2024)*. Lisbon, Portugal.

[12] Jie Wu, Changhu Wang, Liqing Zhang, and Yong Rui. Sketch recognition with natural correction and editing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

[13] Jie Wu, Changhu Wang, Liqing Zhang, and Yong Rui. Smartvisio: Interactive sketch recognition with natural correction and editing. In *Proceedings of the 22nd ACM International Conference on Multimedia*, 2014. (demo).

[14] Jie Wu, Changhu Wang, Liqing Zhang, and Yong Rui. Offline sketch parsing via shapeness estimation. In *IJCAI*, volume 15, 2015.