jinchenh@usc.edu **Jincheng He** 213-246-9074

Education

- Ph.D. in Computer Science, University of Southern California 2020-
- M.S. in Computer Science, University of Southern California 2016-2018
- B.S. Software Engineering, Huazhong University of Science and Technology 2012-2016

Relevant Core Ph.D. Courses:

- Advanced Program Analysis and Verification
- Advanced Natural Language Processing
- Advanced Analysis of Algorithms

Project and Research Experience

Software Quality as a Service (Development & Research) 2017.9-2020.1

CSCI-577a Project, instructed by Dr. Pooyan Behnamghader, Dr. Barry Boehm, CSSE, USC

- The project focuses on monitoring software evolution by evaluating code with static analysis tools.
- **My Role**: Developer (Backend and Front-end), QFP (Quality Focal Point), Data Analyst, Software Researcher.
- **Skill & Tool Set**: Node.js, React.js, MySQL, SonarQube, PMD, FindBugs (SpotBugs)

Center for Systems and Software Engineering Research Portal (Development) 2020.1-2022.12

Development project, instructed by Dr. Barry Boehm, Coordinating with Dr. Pooyan Behnamghader, CSSE

- The project focuses on maintaining and improving the website deployed on AWS that serve as a portal for student directed research project recruitment.
- **My Role**: AWS account management, server maintenance and student/intern recruitment for DR projects.
- **Skill & Tool Set**: Node.js, React.js, MySQL, AWS server management and maintenance.

Software Metadata Analysis (Ph.D. Research) 2020.1-2023.3

Research project, instructed by Dr. Barry Boehm, CSSE, USC

- The project focuses on analyzing GitHub project metadata, tracing software quality with static analysis tools, and classify commits (with software maintenance task taxonomy) to understand their intentions.
- **Skill & Tool Set**: Python, Java, Bash, Machine Learning and NLP techniques (Word of Bags, Decision Tree, etc.), SonarQube, PMD, FindBugs (SpotBugs), Pytorch
- **Background and Contributions:** Software repositories allow multiple developers to iteratively contribute commits, with the intention of improving the system. However, commits can negatively impact software quality, or even cause the software to become uncompilable. Recent studies show that uncompilable commits exist even in high-profile open-source software. Identifying broken code, a potential symptom of careless development, and analyzing how software changes when it becomes uncompilable can shed light on how software quality evolves when developers do not follow best practices. Since comprehensive software quality analysis tools are incapable of analyzing uncompilable commits, there is little insight as to what happens and how quality changes when a commit breaks the compilability. In this study, starting from an analysis of the software quality metric changes that happen when the project become uncompilable, we explore the purposes of commits and the relations between commit type, size and compilability, analyzed across 68 open-source Java repositories.

Tracing Sustainability in Architectures (Ph.D. Research) 2023.3-

Research project, instructed by Dr. Nenad Medvidović, USC

- The project focuses on analyzing software projects to understand the architectural design decisions, architectural changes, and architectural decays over time to preserve the sustainability of software.
- **Skill & Tool Set**: Python, Java, Bash, Generative LLM (e.g., GPT), RAG (Retrieval Augmented Generation)
- **Background and Contributions:** Software architecture sustainability has become increasingly critical with the prevalence of long-term service and support in modern systems. However, sustainability concerns remain insufficiently understood and poorly tracked in software development, making it challenging to design and maintain sustainable software architectures. This study presents TASR, a framework that helps developers track and understand sustainability concerns and their implications on software architecture. Our framework adapted an LLM classifier using prompt engineering to identify the five most common sustainability concerns in software issues that are associated with architectural

changes. These concerns were filtered from previously established taxonomy for technical aspects of software sustainability. To evaluate TASR, we extracted >7,700 issues from six high-profile GitHub projects and classified their sustainability concerns, achieving a micro precision of >83% and micro recall of >90%. Moreover, TASR tracks the correlation between sustainability concerns and architectural implications. Our findings suggest that sustainability concerns significantly correlate with previously reported architectural smells. Our analysis further revealed that sustainability concerns cause unintended introduction or removal of architectural smells. TASR offers a unique pathway for developers to gain a deeper awareness of the sustainability concerns and their impacts on software architecture.

ARCADE, Software Architecture Evaluation Workbench 2023.11-

Development project for software researcher and architects, instructed by Dr. Nenad Medvidović, USC

- The workbench provides architectural recovery (extracting architecture from existing projects), evaluation (with architectural metrics and decay detection), design decisions tracing, and visualization for software architecture researchers and architects to understand the software architecture and how it evolves over time.
- **My Role**: Maintaining the existing features, providing technical support for users, writing manuals, develop new features (especially by integrating Generative LLMs).
- **Skill & Tool Set**: Java, Python, NLP (Topic Modelling and LLM), Understand (SciTools)

First Author Publications

- *The Characteristics and Impact of Uncompilable Code Changes on Software Quality Evolution*, The 20th IEEE International Conference on Software Qualify, Reliability, and Security (QRS)
- *TSAR: Tracing Sustainability in Architectures*, first author (submitted)
- *LLM-enhanced Generalizable Software Architectural Smell Detection and Tracing* (On-going)

Teaching Experience

Teaching a course requires fully understanding the course content, designing questions for exams, and providing support for professors and students.

- *CSCI-577A Software Engineering* (M.S. course) with Prof. Barry Boehm (1 semester)
- *CSCI-510 Software Economics* (M.S. course) with Prof. Barry Boehm (1 semester)
- *CSCI-585 Database Systems* (M.S. course) (5 semesters)
- *CSCI-561 Foundations of Artificial Intelligence* (M.S. course) (9 semesters)

Skills and Interests

- Programming language: Python, Java, C/C++, Linux bash, PHP, Assembly, ReactJS, NodeJS.
- Language Proficiency: Chinese (Mandarin), English, Japanese (JPLT N1, Rank 98.3%)
- Software and Systems: Microsoft Visual Studio Code, IntelliJ IDEA, GitHub, GitLab, Google suite and Microsoft Office suite, Ubuntu, Windows and WSL2.
- Research Interests: Software Architecture, Software Engineering and Quality Analysis, Statistics, Neural Networks and Natural Language Processing.