# Conference Talk / Scientific Presentation Text

## Topic: Neurosymbolic Software Engineering: A New Paradigm for AI-Driven Automation

**Good afternoon, ladies and gentlemen,**

My presentation today focuses on an emerging paradigm in the field of software engineering known as **Neurosymbolic Software Engineering**, or **NSE**. The purpose of my talk is to present the motivation behind NSE, describe its core components, and explain why this hybrid approach may represent the next step in AI-driven software development. This presentation is based on the research study *"A Path Less Traveled: Reimagining Software Engineering Automation via a Neurosymbolic Paradigm"* by Mastropaolo and Poshyvanyk (2025).

**Firstly**, let me outline the problem addressed in the study.
Over the past decade, Large Language Models and Large Code Models have dramatically improved the automation of software engineering tasks such as code generation, bug fixing, documentation, and refactoring. However, these impressive results come at a cost. LLMs require enormous datasets, huge computational resources, and lack interpretability. Moreover, researchers predict the approach will soon face a "**singularity of automation**", a point at which we run out of human-generated data to train increasingly large models.

**Secondly**, the authors highlight that purely neural approaches are no longer sustainable. Neural models are powerful, but they remain black boxes, expensive to train, difficult to validate, and vulnerable to biases and adversarial examples. Therefore, the research community has begun exploring alternatives to unlimited scaling. One promising direction is **neurosymbolic AI**, which combines neural learning with symbolic reasoning.

**Thirdly**, the study proposes a generalized framework - **Neurosymbolic Software Engineering** - which expands earlier work on neurosymbolic program comprehension. This new paradigm unifies three components:

1. **Probabilistic (Neural) Methods**, based on LLMs and LCMs, which excel at pattern recognition and flexible generalization;

2. **Symbolic Methods**, such as formal rules, logic constraints, and verification systems, which provide correctness, interpretability, and structure;

3. **A Chaos-Driven Component**, which introduces controlled randomness to simulate real-world uncertainty in evolving software systems.

Let me now turn to these components in more detail.
The neural part provides semantic understanding and supports tasks like code completion and generation. The symbolic component ensures that automated solutions remain aligned with logical constraints and formal specifications. Finally, the chaos-driven module allows the

system to explore edge cases, handle unpredictable behaviours, and model non-determinism - something essential in complex software ecosystems.

**After that**, the authors discuss key opportunities offered by NSE.
Firstly, NSE improves **interpretability** because symbolic rules make system decisions more transparent.
Secondly, it enhances **trustworthiness**, allowing integration of formal verification and reducing model bias.
Thirdly, NSE promotes **energy efficiency**, because it mitigates the need to continuously scale neural models.
Finally, NSE supports **context-aware and adaptive automation**, enabling tools to adjust to changing requirements without retraining massive models.

The study also outlines several open challenges. Managing randomness carefully is difficult; encoding symbolic knowledge requires expertise; and deploying hybrid AI models demands new tools and infrastructures. Additionally, ensuring that NSE systems generalize across different software engineering tasks remains a nontrivial problem.

**In conclusion**, Neurosymbolic Software Engineering offers a promising alternative to the current trend of endlessly scaling large neural models. By combining neural learning, symbolic reasoning, and carefully controlled chaos, the NSE paradigm aims to create software engineering tools that are not only powerful and accurate but also interpretable, efficient, and sustainable. This hybrid approach may well become a key direction in the future of AI-driven software development.

**Thank you for your attention. I am ready to answer your questions.**