SDU

# Generative AI and Software Variability

An earlier research vision, today's reality, and a possible future

**Sandra Greiner**

Artificial Intelligence, Cybersecurity, Programming Languages Section (ACP)
Institute of Mathematics and Conputer Science (IMADA)
University of Southern Denmark (SDU)

**September, 29th 2026** **GAIV '26 in Limmasol, Cyprus & AoE**

# Generative AI and Variability
# A Research Vision

**Sandra Greiner** **Klaus Schmid[2] Thorsten Berger[3] Sebastian Krieter[4] Kristof Meixner[5]**

Software Engineering Group
University of Bern, Switzerland

**Februray, 9th 2024** **VaMoS '24 in Bern**

[2]University of Hildesheim [3]Ruhr University Bochum [4]University of Ulm [5]TU Vienna

# What’s the agenda for today?

I) Research Vision from 2024

II) Today’s state

III) Possible future directions and open questions

# What's the agenda for today?

**I) Research Vision from 2024**

II) Today's state

III) Possible future directions and open questions

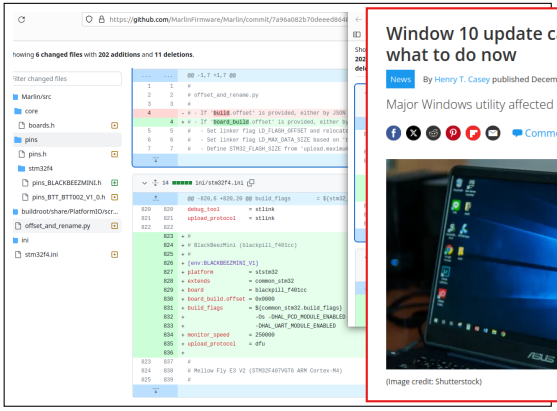
# (Current*) State of

# Managing Variable-Intensive Software

**current? (as of Jan 2024)*

# Complex, configurable software, which evolves

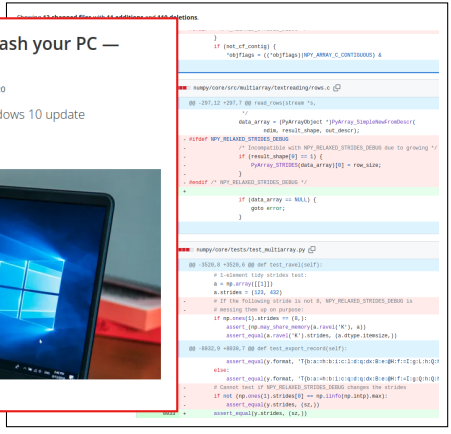
SDU




Marlin (3D printer software)

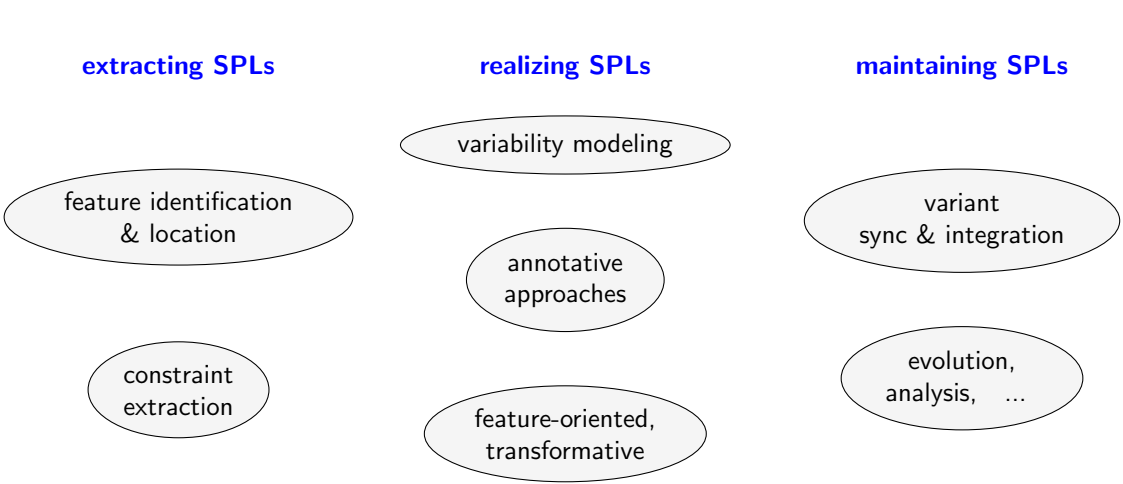



(Image credit: Shutterstock)




*numpy*-lib commit: markdown, tests, C-file, ...

# Key activities in our domain...



# Key activities in our domain...

**extracting SPLs** | **realizing SPLs** | **maintaining SPLs**

variability modeling

feature identification & location

annotative approaches

variant sync & integration

language-specific, often not scalable, hard to maintain consistently ...

**still insufficient genericity and automation**

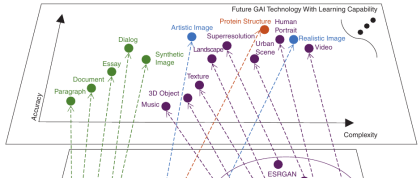
SDU

Business / Tech

# A year after ChatGPT's release, the AI rev is just beginning

By Catherine Thorbecke, CNN

8 minute read · Updated 10:32 AM EST, Thu November 30, 2023




[jovanovic22]

On Programming Variability with Large Language Model-based Assistant

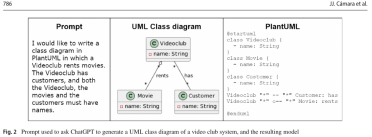



Fig. 2 Prompt used to ask ChatGPT to generate a UML class diagram of a video club system, and the resulting model

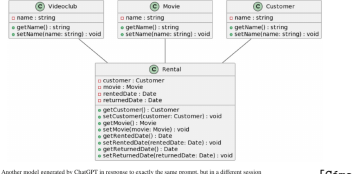



Fig. 3 Another model generated by ChatGPT in response to exactly the same prompt, but in a different session

[Cámara23]

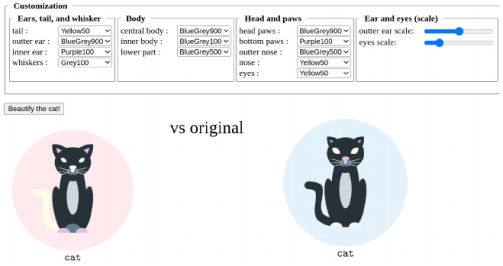


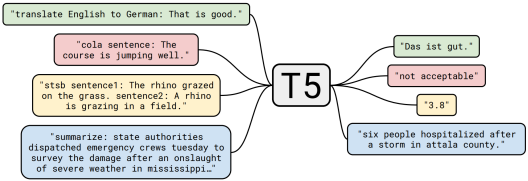

Figure 1: End-user customization of Cat TikZ code

[Acher23]

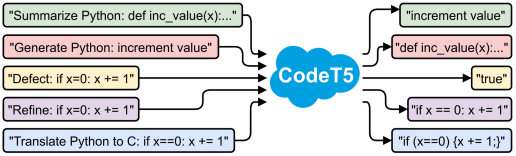
What does it need from research
to support maintaining variability-intensive software systems
with Generative AI?

# Background

(as of the beginning of 2024)




[T5]




[CodeT5]

> The basic idea is to use a large language corpus to train a neural network to learn the language, by hiding part of the text and asking the network to guess the missing parts.

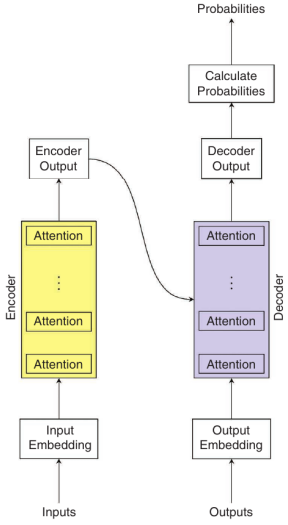
[Ebert23]

large pre-trained transformer models: convert *input* text into *output* text (encoder-decoder)

Generative AI for SE: *pre-trained* on several programming languages

- defect & clone detection
- code summarization
- code translation,
- ...

# Transformer architecture (beginning of 2024)




[Ebert23]

transformer architecture enabled building first LLMs

context window defines size of input and output (in tokens)

typical sizes: 8k-32k

fine-tuning through separate train-validate-test runs on custom task while modifying different hyperparameters (learning rate, ...)

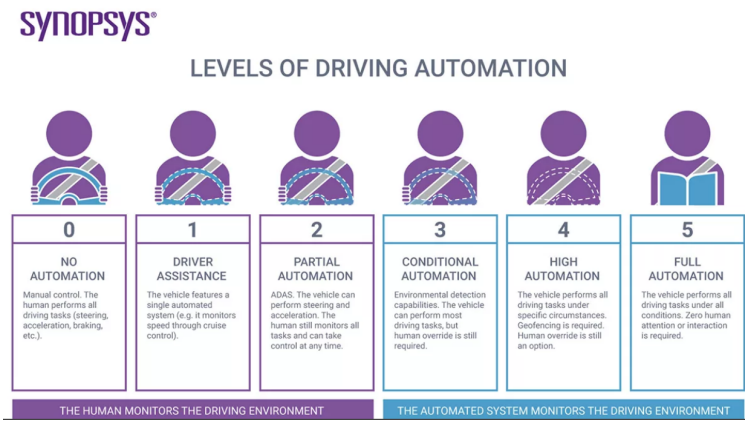
What does it need from research
to support maintaining variability-intensive software systems
with Generative AI?



[idea]

Our initial idea
and approach

How can we measure success of Gen-AI-enabled development of variability-intensive software?




[synopsys]

How can we classify and assess progress
in using Generative AI for variability-aware SE?

| | |
|---|---|
| **Scale** | information amount processed in one step |
| **Task Type** | analytical vs. adaptive, to a productive end* |
| **Quality** | level of reliability: random, junior developer, senior developer, even better |
| **Automation** | level of manual intervention |

* ⚠ not trimmed to a **specific** variability-intensive task

# Automation levels



| automation | scale | quality |
|---|---|---|
| **L4** Full Generative AI for Variability | unlimited | expert |
| **L3** Advanced Generative AI for Variability | multiple (project**s**) | high |
| **L2** Basic Generative AI for Variability | small (project) | low |
| **L1** No Generative AI for Variability | small (files) | low |
| **L0** No Generative AI | | |

# How can we achieve a level-up?

**Increase Scale**

- generic solutions (e.g., larger context windows)
- SE solutions (e.g., abstractions)
- SPLE solutions (e.g., focus on features)

**Increase Task**

towards reaching productive usage

- when and how to integrate existing tools?
- ensure traceability
- self-optimizations, multiple agents

**Increase Quality (and reliability)**

- decrease error probability
  → pre-training, and fine-tuning
- support existing (hidden) knowledge
  → prompting, model-driven techniques
- assessment strategies
  → novel datasets and metrics (real-world projects are too large, accuracy not sufficient)
- increase reliability
  → reasoning for results, certainty values

# What's the agenda for today?

I) Research Vision from 2024

II) **Today's state**

III) Possible future directions and open questions

# Updated Background

GenAI as model

$\Rightarrow$

GenAI as a holistically engineered system

# Technology changes rapidly...



[waterfall]

context windows

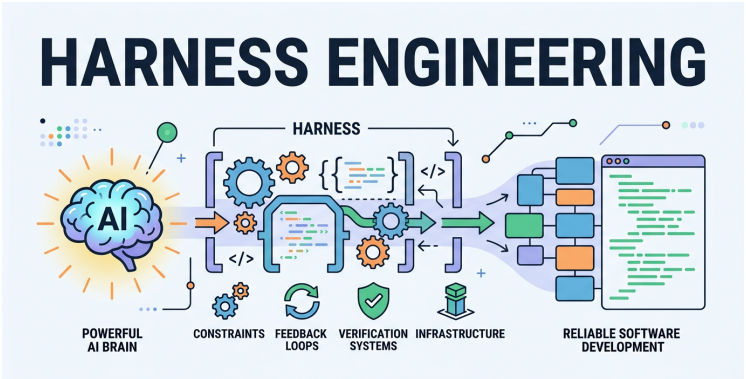
- increased: up to 1Mio tokens
- use and store only necessary information (e.g., when doing code analyses)
- build knowledge bases (atm: typically RAG)

shift from fine-tuning to *context or even harness engineering*

use of agents instead of single prompts, few-shot examples, etc

harness to manage the agents at runtime

- ...




How can one and the same LLM be used to accomplish completely different tasks?

[harness]

# Agent harness [harness-engineering]

**definition**

- infrastructure layer
- runtime environment
- operating system for LLM
- moat vs. commodity

**core elements**

- context engineering
- tool orchestration
- state and memory
- verification and safety
- human-in-the-loop
- lifecycle management

**failure modes**

- context rot
- tool explosion
- infinity retry loops
- state corruption

**implementation strategy**

- Anthropic: two-agent pattern
- OpenAI Codex: Machine-readable artifacts
- Vercel: Tool reduction
- Manus: architectural rewrite

# Latest research on software variability with GenAI

⚠ The following list does not result from an SLR and might miss relevant work...

(mainly proceedings of "variability" conferences since 2024)



# Publications @VARIABILITY 2026

*Fine-Tuning Small Language Models For Valid Universal Variability Language Generation* - Stümpfle et al.

*Do LLMs Know UVL? Quantifying Factual Knowledge and Generative Capabilities in Variability Engineering* - Horcas et al.

*Fluent in Variability? An Empirical Study of Large Language Models Across Variability-Related Tasks and Formats* - Zine et al.

*Investigating LLMs for Feature Presence Prediction in Software Products* - Nguyen et al.

*LLM-assisted Code Generation and Reuse: An Exploratory Study on Contextual Information* - Reinhartz-Berger et al.

*Semantic-Driven Feature Identification In Software Variants Using Transformers* - Stümpfle et al.

*Software Reuse in the Age of AI: A Survey of Practices, Satisfaction, and Barriers* - Iris Reinhartz-Berger, Zarima Fram

*Usability Analysis of Configurator User Interfaces with Multimodal Large Language Models* - Lubos et al.

# Further publications...

SDU

analytical

- *Configuration Bugs Classification using LLMs and Encoders* - Lopez-Duran et al. SPLC 2025

generative

- On Programming Variability with Large Language Model-based Assistant. - Acher et al. SPLC 2023
- *LLM Code Customization with Visual Results: A Benchmark on TikZ.* -Reux et al. EASE 2025
- *Prompting for Performance: Exploring LLMs for Configuring Software.* - Spieker et al. ICTAI 2025: 114-121
- *Large Language Models to generate meaningful feature model instances.* - Galindo et al. SPLC 2023
- *LLM-based Co-Evolution of Configurable Software Systems* - Zine et al

adaptive

- *Large Language Model assisted Transformation of Software Variants into a Software Product Line* - Stümpfle et al. ICSR 2025

# Further ...

*Bridging Feature Models and Users: LLM-Generated Low-Code Configurators for Software Product Lines* - Wegerer et al. MODELS 2026

*Towards Automated Test Adaptation in Fork Ecosystems via Large Language Models* - Mukelabai et al. FSE 2026 Companion

*Linux Kernel Configurations at Scale: A Dataset for Performance and Evolution Analysis.* - Borges et al. EASE 2025

*A Demonstration of End-User Code Customization Using Generative AI.* Acher. VaMoS 2024

But also the flip side

*Managing Variability in Industrial AI Agents for Manufacturing: Experiences at Hitachi* - Yoshimura et al. CAIN 2026

*Where Did the Variability Go? From Vibe Coding to Product Lines by Regeneration* - X. Tërnava

*Variability Management for Large Language Model Tasks: Practical Insights from an Industrial Application.* - Kentaro Yoshimura, Haruki Oishi. SPLC 2024



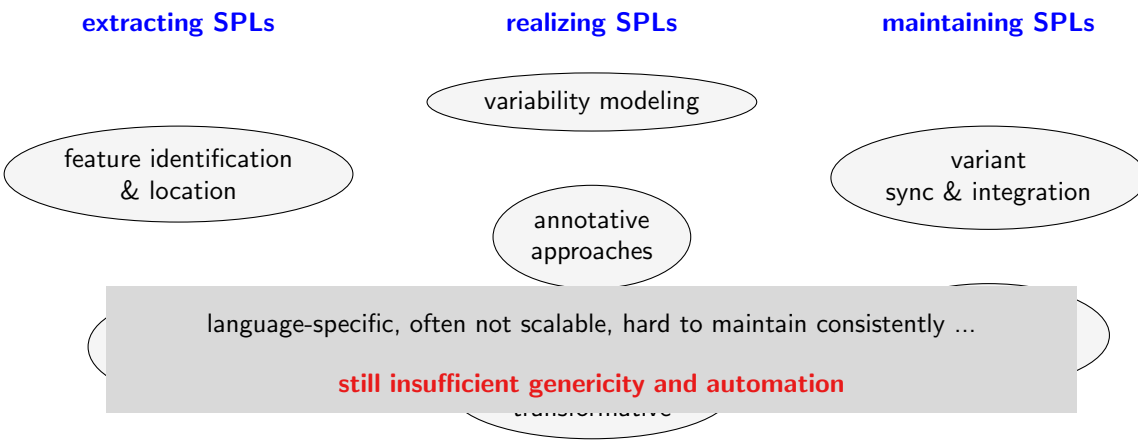
[turtle]



Have we solved our variability tasks?

[thinking]

# Recap: Key activities in our domain



| extracting SPLs | realizing SPLs | maintaining SPLs |
| --- | --- | --- |
| Semantic-Driven Feature Identification | Generation of Configurations | evolution support??? |
| LLMs for Migrating Variants into a SPL | Usability of configurators | |
| | Feature presence prediction (?) | |

language-independent, scalable ?, consistent maintenance?
**still insufficient automation**

# Revisiting the research needs

# How can we achieve increase? - Revisited

SDU

**Increase Scale**

- generic solutions (e.g., larger windows) ✔
- SE solutions (e.g., abstractions) ✔
- SPLE solutions (e.g., focus on features) ?

**Increase Task**

towards reaching productive usage

- when and how integrate existing tools ?
  (*partly, e.g., to generate variants)
- ensure traceability ✔
- self-optimizations, multiple agents ✔

**Increase Quality (and reliability)**

- decrease error probability → pre-training, and fine-tuning *use harness instead*
- support existing (hidden) knowledge → prompting, model-driven techniques ✔studies of what LLMs understand
- assessment strategies → novel datasets and metrics (real-world projects are too large, accuracy not sufficient) ✘
  * existing datasets can now be used
- increase reliability → reasoning for results, certainty values
  better focus shift ?

*We reach full GenAI support for variability if we have* **expert-developer** *quality at full scale.*

**Focus on observable/verifiable outcomes**

- Are the changes syntactically valid (compile)?
- Are intended constraints preserved?
- Are *relevant* configurations preserved?
- Are results reproducible?
- Are the results consistent (across variants), traceable?
- How robust is our AI-enable system?



[eyes]

I) Research Vision from 2024

II) Today's state

III) **Possible future directions and open questions**

# Recap: Automation Levels



| automation | scale | quality |
|---|---|---|
| **L4** Full Generative AI for Variability | unlimited | expert |
| **L3** Advanced Generative AI for Variability | multiple (project**s**) | high |
| **L2** Basic Generative AI for Variability | small (project) | low |
| **L1** No Generative AI for Variability | small (files) | low |
| **L0** No Generative AI | | |

**extracting SPLs**

Semantic-Driven Feature Identification

LLMs for Migrating Variants into a SPL

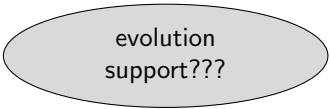
**realizing SPLs**

Generation of Configurations

Usability of configurators

Feature presence prediction (?)

**maintaining SPLs**




language-independent, scalable ?, consistent maintenance?
**still insufficient automation**

# Still open questions

Do we scale to massively configurable software systems?

If not, what is missing?

*How* to use agents for detecting (language-based) features and tracing them?

*How* to use agents for safely evolving feature implementations (update one feature without breaking another)?

How to ensure the quality of using Generative AI?

Do we (really) need novel benchmarks?
Can we reuse existing benchmarks or benchmark generating systems?

**Plot Twist**

*How and where can SPL techniques help to manage the variability induced by using Generative AI?*




**Let's** develop methods to ...

- **scale** highly-configurable systems into context windows and
- handle **state and memory**
- decide the optimal **existing tools** to be used for configurable software
- that decide when **humans** need to be in the loop

**Let's** use our methods to ...

- manage tons of vibe-coded source code
- manage different agents (that may handle different software features)
- ...

[next]

Thanks for the feedback and behind-the-scenes discussions to

**Alessio Pellegrino**



**Matteo Esposito**



and
Narongrit Unwerawattana, Fabrizio Montesi, Kristof Meixner, Klaus Schmid, Sebastian Krieter, and Thorsten Berger

SDU

# Thanks ☺!

## Feedback, Questions, ...?

greiner@imada.sdu.dk