LLM-Based Design Pattern Detection

Christian Schindler and Andreas Rausch
Institute for Software and Systems Engineering
Clausthal University of Technology
Clausthal-Zellerfeld, Germany
e-mail: {christian.schindler | andreas.rausch}@tu-clausthal.de

The Seventeenth International Conference on Pervasive Patterns and Applications
PATTERNS 2025
April 06, 2025 to April 10, 2025 - Valencia, Spain

The presenter – Christian Schindler

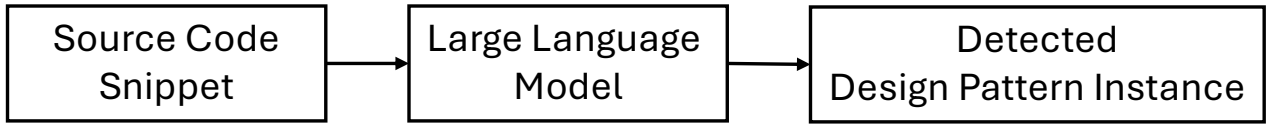
- PhD Candidate at ISSE TU Clausthal
- Holds B.Sc. and M.Sc. (2014 and 2016)

- Research interests
  - Software Engineering and AI
  - Software Architecture and Design Patterns Detection

Motivation

- Design patterns
  - Proven solutions addressing recurring software design problems
  - Define roles and responsibilities within a software structure
- Design Pattern Instance
  - Concrete implementation where roles are realized by specific code artifacts (classes, interfaces, packages)
- Importance of Identifying Pattern Instances
  - Enhances code quality and maintainability
  - Simplifies code comprehension and communication among developers
- Challenges in Identification
  - Patterns are typically implicit, with no direct language-level indicators (e.g. key words)
  - Rarely explicitly annotated, making automated identification challenging

Research Aim



- RQ1: How can LLMs be leveraged to automatically detect and annotate design pattern instances in software codebases?
- RQ2: How good can LLMs detect and annotate design pattern instances in software codebases?
- RQ3: What are challenges/limitations faced by LLMs in identifying design pattern instances in code bases, and how can these be addressed

Data and Experimental Runs

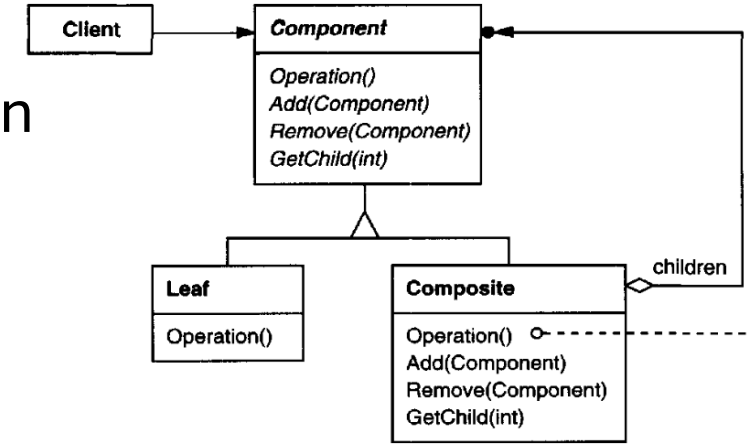
- P-MARt [1]
  - Peer-reviewed Design Pattern instances annotations
  - from different open software systems

```
<designPattern name="Composite">
    <microArchitectures>
        <microArchitecture number="4">
            <roles>
                <clients>
                    <client roleKind="AbstractClass"><entity>uml.ui.ToolPalette</entity></client>
                </clients>
                <components>
                    <component roleKind="AbstractClass"><entity>diagram.tool.Tool</entity></component>
                </components>
                <composites>
                    <composite roleKind="Class"><entity>diagram.tool.CompositeTool</entity></composite>
                </composites>
                <leaves>
                    <!-- <leaf roleKind="Class"><entity>diagram.tool.CompositeTool</entity></leaf> -->
                    <!-- <leaf roleKind="Class"><entity>diagram.tool.AbstractTool</entity></leaf> -->
                    <leaf roleKind="Class"><entity>uml.ui.CardinalityTool</entity></leaf>
                    <leaf roleKind="Class"><entity>diagram.tool.ClipboardTool</entity></leaf>
```

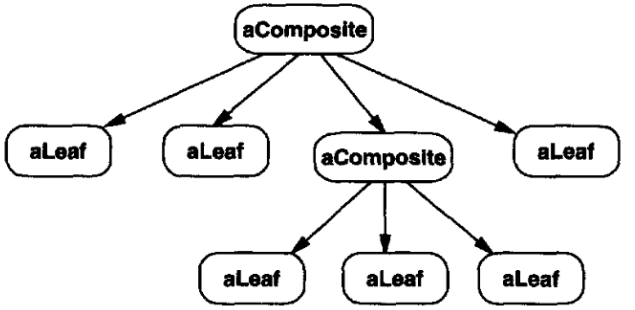
[1] Y.-G. Guéhéneuc, “P-mart: Pattern-like micro architecturerepository,” Proceedings of the 1st EuroPLoP Focus Group on pattern repositories, pp. 1–3, 2007.

Design Pattern Details

- Composite Pattern



[1]

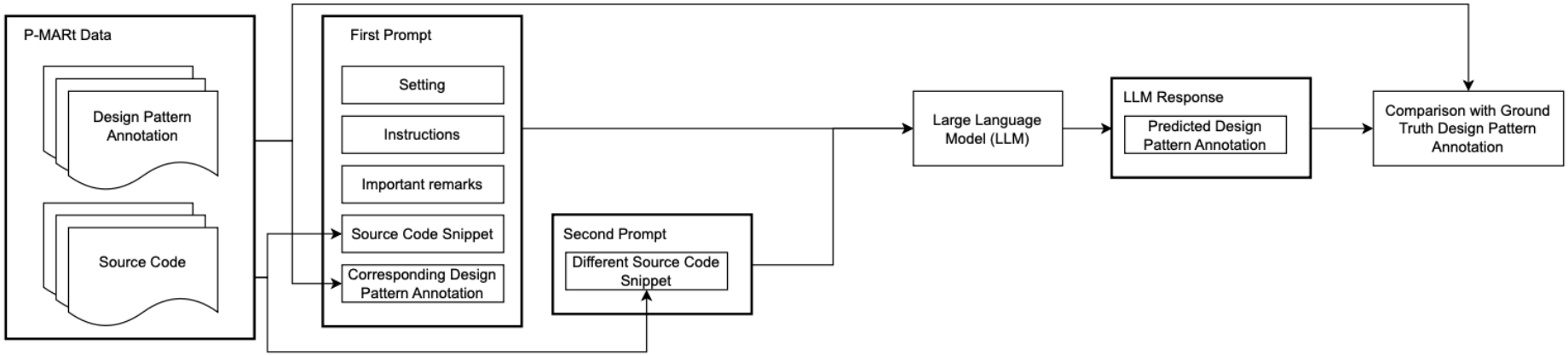


[1]

Instance	*Project*	*Common Root Package*	*# Classes Participating in Pattern instance*	*# Classes in Root Package*
4	QuickUML 2001		17	217
65	JUnit v3.7	junit	39	94
75	JHotDraw v5.1	CH.ifa.draw	35	155
98	MapperXML v1.9.7	com.taursys	29	234
129	PMD v1.8	net.sourceforge.pmd.ast	3	108
143	Software architecture design patterns in Java	src.COMPOSITE	5	5

[1] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: elements of reusable object-oriented software*.

Experimental Design & Methodology



- One-shot learning
- LLMs ChatGPT 3.5 and GPT 4
- 14 runs with example pairs

https://github.com/schindlerc/LLM-Based-Design-Pattern-Detection

Results – Confusion Matrix

TABLE II. CONFUSION MATRIX CHATGPT 3.5.

							With Hallucination		Without Hallucination	
	Hallucinated Class	Client	Component	Composite	Leaf	No Role	Classification Overall	Precision	Classification Overall	Precision
Hallucinated Role	0	0	1	0	0	3	4	-	-	-
Client	38	5	1	0	2	24	70	.071	32	.156
Component	0	0	8	1	0	1	10	.800	10	.800
Composite	1	0	0	4	12	10	27	.148	26	.154
Leaf	13	0	0	4	17	32	66	.258	53	.321
No Role	0	23	5	16	185	1064	1293	.823	1293	.823
Truth overall	52	28	15	25	216	1134	1470			
Recall	-	.179	.533	.160	.008	.938				
Truth overall	-	28	14	25	216	1131			1414	
Recall	-	.179	.571	.160	.008	.941				

- ChatGPT 3.5 hallucinated
  - Classes that do not exist in the Software System
  - Roles that do not exist in the Design Pattern
- Results differ from role to role

Results – Confusion Matrix

TABLE III. CONFUSION MATRIX GPT4.

	No Role	Client	Component	Composite	Leaf	Classification Overall	Precision
Client	13	9	1	3	8	34	.265
Component	1	0	10	0	0	11	.910
Composite	5	0	0	12	12	29	.414
Leaf	34	2	0	1	45	82	.549
No Role	1079	17	4	11	150	1261	.856
Truth overall	1132	28	15	27	215	1417	
Recall	.953	.333	.833	.522	.209		

- ChatGPT 4 did not hallucinate
- Results are all better than by ChatGPT 3.5

Results – ChatGPT 4

- Positive prediction class
  - all roles of the design pattern
- Negative prediction class
  - “No role”-class

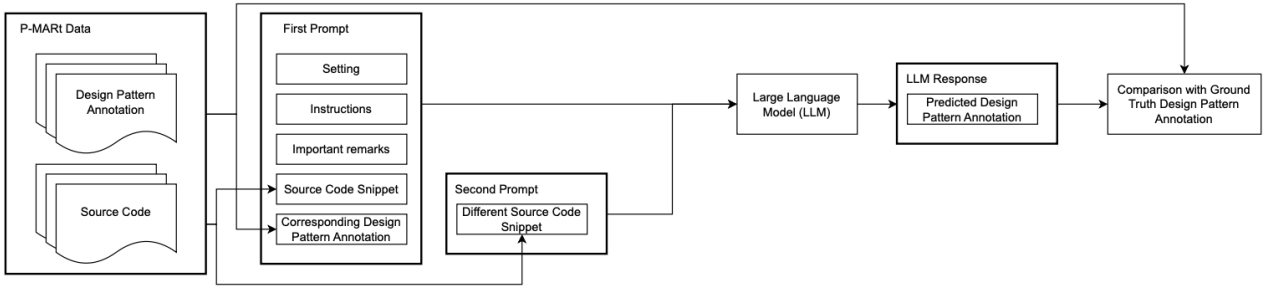
Run	TP	FP	FN	TN	Precision	Recall	F1	Accuracy
1	6	3	34	54	.667	.150	.245	.619
2	21	32	14	98	.396	.600	.477	.721
3	8	4	32	53	.667	.200	.308	.629
4	0	0	39	55	0	0	0	.585
5	0	0	17	200	0	0	0	.922
6	2	1	3	0	.667	.400	.500	.333
7	3	1	2	0	.750	.600	.667	.500
8	3	1	2	0	.750	.600	.667	.500
9	3	1	2	0	.750	.600	.667	.500
10	2	1	3	0	.667	.400	.500	.333
11	11	0	6	200	1.000	.647	.786	.972
12	0	0	3	105	0	0	0	.972
13	0	11	29	203	0	0	0	.835
14	17	15	18	111	.531	.486	.507	.795

Limitations and Future Work

- Limitations
  - Results obtained from analyzing one design pattern—other patterns may yield different outcomes
  - Evaluation included only two LLM models.

- Future Directions
  - The presented approach can be consistently applied to other design patterns
    -> validate the approach across multiple design patterns
  - Provide LLMs with multiple annotated examples to improve context and accuracy.
  - Investigate prompt refinement strategies for better detection accuracy.
  - Explore hybrid approaches combining:
    - Logical reasoning
    - Static analysis
    - Dynamic analysis

TLDL(isten);



- LLM-Based Design Pattern Detection
- Collected pairwise code samples annotated by other authors
  - Provided one labeled code example as context in an initial prompt
  - Applied the LLM to a second code sample to automatically detect and annotate the design pattern instance
- Minimal preprocessing required, streamlining the detection process
- Evaluated the accuracy by comparing the LLM-generated annotations to manually labeled ground truth data
- Observed improved detection performance with newer and larger LLM models