LLM vs Manual Analysis - Comparative Study

Search and Indexing Subsystem Analysis

Subsystem Analyzed: Apache Roller Search and Indexing Subsystem
Scope: 15 core classes/interfaces + 6 optional UI layer classes (21 total)

1. Analysis Comparison Summary

Metric	Manual Analysis	LLM-Assisted Analysis
Time Spent	~4-5 hours	~15-20 minutes
Core Classes Identified	15 classes	15 classes
Optional UI Classes	6 classes	0 classes
Total Classes	21 classes	15 classes
Relationships Documented	60+ relationships	25+ relationships
PlantUML Lines	625 lines	299 lines
Design Patterns Found	4 patterns	6 patterns
Documentation Pages	PDF + Markdown reports	Markdown files

2. Completeness Analysis

2.1 Classes and Interfaces Identified

Core Search Classes (15 classes)

Component	Manual ✓/✗	LLM ✓/✗
`IndexManager`	✓	✓
`LuceneIndexManager`	✓	✓
`SearchResultList`	✓	✓
`SearchResultMap`	✓	✓
`IndexOperation`	✓	✓
`WriteToIndexOperation`	✓	✓
`ReadFromIndexOperation`	✓	✓
`AddEntryOperation`	✓	✓
`RemoveEntryOperation`	✓	✓

Component	Manual ✓/✗	LLM ✓/✗
`ReIndexEntryOperation`	✓	✓
`RebuildWebsiteIndexOperation`	✓	✓
`RemoveWebsiteIndexOperation`	✓	✓
`SearchOperation`	✓	✓
`FieldConstants`	✓	✓
`IndexUtil`	✓	✓

Optional UI Layer Classes (6 classes - uses IndexManager)

Component	Layer	Manual ✓/✗	LLM ✓/✗
`SearchResultsModel`	UI Rendering	✓	✗
`SearchResultsFeedModel`	UI Rendering	✓	✗
`SearchResultsPager`	UI Paging	✓	✗
`SearchResultsFeedPager`	UI Paging	✓	✗
`SearchServlet`	Servlet	✓	✗
`WeblogSearchRequest`	Request Util	✓	✗

Result:

- Core classes: Both approaches achieved 100% identification (15/15)
- Optional UI classes: Manual identified 6 additional classes that use IndexManager
- **Total**: Manual: 21 classes | LLM: 15 classes

2.2 External Dependencies Modeled

Dependency	Manual	LLM
Lucene `IndexWriter`	✓	✓
Lucene `IndexReader`	✓	✓
Lucene `IndexSearcher`	✓	✓
Lucene `Analyzer`	✓	✓
Lucene `Term`	✓	✓
Lucene `TopFieldDocs`	✓	✓
Lucene `Document`	✗	✓
`WeblogEntry`	✓	✓
`Weblog`	✓	✓

Dependency	Manual	LLM
`WeblogCategory`	✓	✗
`WeblogEntryComment`	✓	✓
`WeblogEntryWrapper`	✓	✗
`WeblogEntryManager`	✓	✗

Observation: Manual analysis included more external domain classes; LLM focused on core subsystem

2.3 Relationship Coverage

Relationship Type	Manual Count	LLM Count
Interface Implementation	6	1
Inheritance (extends)	10	7
Composition (*--)	4	2
Aggregation (o--)	6	1
Association (-->)	35+	12+
Dependency (..>)	5+	8+

Observation:

- Manual analysis now includes UI layer relationships showing how SearchServlet, Models, and Pagers connect to IndexManager
- Manual captures the complete search request flow from HTTP to index
- LLM focuses only on core subsystem relationships

---

3. Correctness Analysis

3.1 Relationship Accuracy

Aspect	Manual Analysis	LLM Analysis
Inheritance hierarchy	✓ Correct	✓ Correct
Interface implementation	✓ Correct	✓ Correct
Composition vs Aggregation	✓ Correct	✓ Correct
Field associations	✓ Correct	✓ Correct
Method-level dependencies	✓ Detailed	○ Simplified

3.2 UML Notation Correctness

Notation Element	Manual	LLM

Notation Element	Manual	LLM
Visibility modifiers (+,-,#)	✓	✓
Static members	✓	✓
Abstract classes	✓	✓ with stereotypes
Interface notation	✓	✓ with stereotypes
Notes/Annotations	✗ None	✓ Design pattern notes

3.3 Errors/Inaccuracies Found

Issue	Manual	LLM
Missing `Runnable` interface	Noted but not shown explicitly	Shows `implements Runnable`
Logger field visibility	Shown inconsistently	Consistently shown as static
Constructor visibility	Public shown	Protected shown correctly

4. Effort Comparison

4.1 Time Breakdown

Task	Manual Time	LLM Time	Savings
Reading source files	~60 min	~3 min	95%
Identifying classes	~30 min	~1 min	97%
Understanding relationships	~45 min	~2 min	96%
Creating PlantUML diagram	~60 min	~5 min	92%
Writing documentation	~45 min	~5 min	89%
Review and refinement	~30 min	~4 min	87%
Total	**~4.5 hours**	**~20 min**	**~93%**

4.2 Cognitive Load

Aspect	Manual	LLM
Understanding code structure	High (reader fatigue)	Low (automated parsing)
Tracking relationships	High (mental mapping)	Low (pattern matching)
PlantUML syntax	Medium (lookup required)	Low (generated)
Consistency checking	High (manual review)	Medium (may need verification)

5. Quality Comparison

5.1 Documentation Quality

Aspect	Manual	LLM	Winner
Technical accuracy	High	High	Tie
Depth of explanation	Medium	High	LLM
Code examples	None	Included	LLM
Design pattern identification	4 patterns	6 patterns	LLM
Assumptions documented	Limited	Comprehensive	LLM

5.2 Diagram Quality

Aspect	Manual	LLM	Winner
Visual organization	Good	Good	Tie
Relationship clarity	More detailed	More focused	Manual
UI Layer coverage	Complete (6 classes)	None	Manual
End-to-end flow	Full request flow	Core only	Manual
Annotations/Notes	Detailed comments	Design pattern notes	Tie
External dependencies	More complete	Core focused	Manual
File size	20 KB (625 lines)	9.4 KB (299 lines)	Manual (comprehensive)

---

6. Strengths and Weaknesses

6.1 Manual Analysis

Strengths:

- Deep understanding of code nuances
- Captures subtle relationships (e.g., method-level dependencies)
- Better external dependency coverage
- Developer gains intimate knowledge of codebase

Weaknesses:

- Time-intensive (4+ hours for 15 classes)
- Prone to fatigue-related omissions
- Inconsistent notation
- No automated verification

6.2 LLM-Assisted Analysis

Strengths:

- Extremely fast (~20 minutes)
- Consistent notation and formatting
- Identifies design patterns automatically
- Generates comprehensive documentation
- Produces well-structured output

Weaknesses:

- May miss subtle domain-specific relationships
- Requires verification for accuracy
- Can oversimplify complex interactions
- May not understand business context

---

7. Recommendations

When to Use Manual Analysis

- Critical/safety-sensitive systems
- When deep code understanding is required
- Small, focused components (< 5 classes)
- Security-sensitive code review

When to Use LLM-Assisted Analysis

- Large codebases (50+ classes)
- Initial exploration/onboarding
- Documentation generation
- Rapid prototyping of diagrams
- Time-constrained analysis

Optimal Approach: Hybrid

1. **Start with LLM** - Generate initial analysis (90% complete, 10% of time)
2. **Manual Review** - Verify key relationships and fix errors
3. **Human Enhancement** - Add domain knowledge and business context
4. **LLM Refinement** - Polish documentation and diagrams

---

8. Conclusion

Criterion	Winner
Speed	LLM (93% faster)
Completeness	Tie (both found all 15 classes)
Correctness	LLM (slightly better notation)
Documentation	LLM (more comprehensive)
Relationship Detail	Manual (more granular)

Criterion	Winner
Design Insight	LLM (found 6 vs 4 patterns)
Overall	**LLM** (for efficiency-to-quality ratio)

Final Assessment: LLM-assisted analysis provides comparable or better results in approximately 7% of the time required for manual analysis. For the Search and Indexing Subsystem, the LLM approach successfully identified all classes, produced accurate UML diagrams, and generated more comprehensive documentation including design pattern annotations.

Appendix: Files Compared

File Type	Manual File	LLM File
PlantUML Diagram	`search_manual.puml` (625 lines)	`llm_Search_subsystem.puml` (299 lines)
PDF Report	`Manual_Search.pdf`	N/A
Documentation	`Readme.md` (comprehensive)	`llm_Search_llm_Subsystem_Documentation.md`
Observations	N/A	`llm_Observations_and_Assumptions.md`

Classes Coverage Comparison

Category	Manual Analysis	LLM Analysis
Core Search (15)	IndexManager, LuceneIndexManager, SearchResultList, SearchResultMap, IndexOperation, WriteToIndexOperation, etc.	Same 15 classes
UI Rendering (2)	SearchResultsModel, SearchResultsFeedModel	Not included
UI Paging (2)	SearchResultsPager, SearchResultsFeedPager	Not included
Servlet Layer (1)	SearchServlet	Not included
Request Util (1)	WeblogSearchRequest	Not included
Total	**21 classes**	**15 classes**