Domain Applications

Week 11: Code, Finance, and Healthcare Agents

PhD Course in Agentic Artificial Intelligence

12-Week Research-Level Course

Learning Objectives

Bloom's Taxonomy Levels Covered

- **Remember**: Define SWE-bench, code agent, FinAgent, clinical decision support
- **Understand**: Explain domain-specific requirements for agent deployment
- **Apply**: Implement a code agent using flow engineering (structured pipelines)
- **Analyze**: Compare agent architectures across different domains
- **Evaluate**: Assess regulatory and safety requirements for each domain
- **Create**: Design a domain-specific agent with appropriate safeguards

By end of lecture, you will understand how agents adapt to real-world domains.

Domain Maturity Landscape

High Maturity: Software Development

- Clear success criteria (tests pass, code works)
- Sandboxed execution environments
- Active deployment: GitHub Copilot, Cursor, Devin

Medium Maturity: Finance

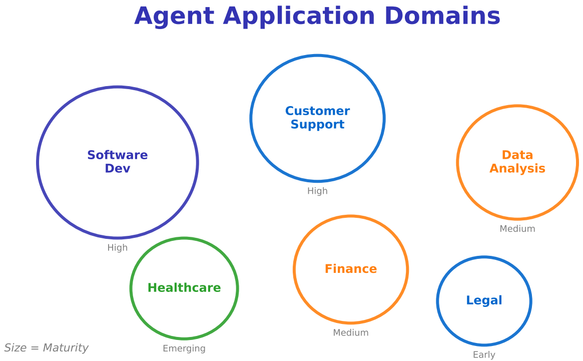
- Well-defined tasks (analysis, research, reporting)
- Heavy regulatory constraints (compliance)
- Emerging: Trading assistants, document analysis

Emerging: Healthcare

- High stakes, requires human oversight
- Regulatory approval (FDA, HIPAA)
- Focus: Decision support, not autonomous action

Maturity correlates with ability to verify outputs and contain errors.

Application Domain Maturity



Software development leads in maturity; healthcare is emerging.

Code Agents: The Leading Domain

Why Code is Ideal for Agents

- Clear success criteria: Tests pass or fail
- Safe sandbox: Run code in containers
- Immediate feedback: Execution reveals errors
- Rich context: Codebase provides grounding

Key Capabilities

- Bug fixing and debugging
- Feature implementation from specifications
- Code review and refactoring
- Documentation generation

Current State

- SWE-bench: Best agents solve ∼50% of real GitHub issues
- Production systems: Copilot, Cursor, Devin, Claude Code

Code agents now outperform average developers on specific benchmarks.

SWE-bench and AlphaCodium

SWE-bench (Jimenez et al., 2024)

- 2,294 real GitHub issues from 12 Python repositories
- Task: Generate code patch to resolve issue
- Verification: Patch must pass repository tests

AlphaCodium: Flow Engineering (Ridnik et al., 2024)

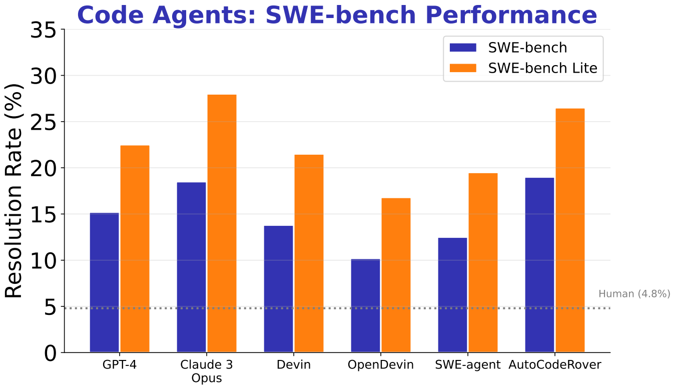
- Structured multi-stage pipeline (not single-shot)
- Stages: Problem reflection, public tests, AI tests, code iteration
- Key insight: Test against multiple cases before submitting

Flow Engineering Principles

- Break complex tasks into simpler stages
- Generate and run tests iteratively
- Use structured output at each stage

Flow engineering = structured pipelines for complex coding tasks.

Code Agents: SWE-bench Performance



Code agents now outperform average human developers on SWE-bench.

Finance Agents

High-Value Applications

- **Research**: Earnings analysis, market research synthesis
- **Trading**: Strategy backtesting, execution assistance
- **Compliance**: Regulatory document analysis, audit trails
- **Operations**: Report generation, data reconciliation

Unique Challenges

- **Regulatory**: SEC, FINRA, MiFID II compliance requirements
- **Explainability**: Must justify recommendations
- **Latency**: Markets move in milliseconds
- **Risk**: Errors have direct financial consequences

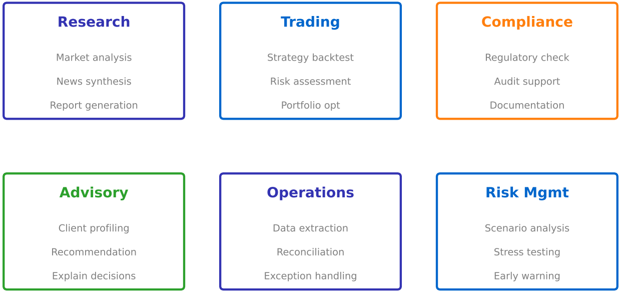
Current Deployments

- FinAgent: Multimodal trading agent (research)
- Bloomberg Terminal AI: Document analysis, Q&A

Finance requires compliance (regulatory) awareness at every step.

Finance Agent Applications

Finance Agent Applications



Research

Market analysis

News synthesis

Report generation

Trading

Strategy backtest

Risk assessment

Portfolio opt

Compliance

Regulatory check

Audit support

Documentation

Advisory

Client profiling

Recommendation

Explain decisions

Operations

Data extraction

Reconciliation

Exception handling

Risk Mgmt

Scenario analysis

Stress testing

Early warning

Finance agents span research, trading, compliance, and operations.

FinAgent: Multimodal Finance Agent

Architecture (Li et al., 2024)

- Multimodal: Text (news, filings), numeric (prices, fundamentals), charts
- Dual memory: Short-term (recent trades), long-term (market patterns)
- Tool use: Market data APIs, technical indicators, portfolio analytics

Key Components

- **Market Perception**: Process multi-modal market signals
- **Agent Memory**: Store and retrieve trading experience
- **Decision Module**: ReAct-style reasoning for trade decisions

Results

- Outperforms baselines on paper trading benchmarks
- Caveat: Simulated environment, not live trading

Multimodal perception is critical for financial markets.

Healthcare Agents

Potential Applications

- Clinical decision support (diagnosis assistance)
- Drug interaction checking
- Patient triage and routing
- Medical literature synthesis

Critical Constraints

- **Regulation**: FDA approval for clinical use
- **Privacy**: HIPAA compliance required
- **Liability**: Who is responsible for errors?
- **Verification**: Medical claims must be evidence-based

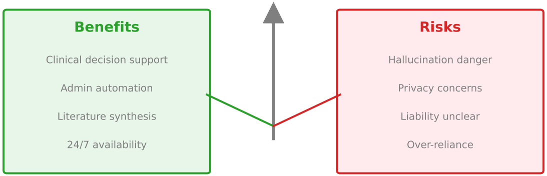
Current Approach

- Human-in-the-loop: Agents suggest, clinicians decide
- Focus on augmentation, not automation

Healthcare agents must support clinicians, not replace them.

Healthcare Considerations

Healthcare Agent Considerations



Benefits

- Clinical decision support
- Admin automation
- Literature synthesis
- 24/7 availability

Risks

- Hallucination danger
- Privacy concerns
- Liability unclear
- Over-reliance

Human oversight essential | Narrow use cases first

Healthcare requires careful balance of benefits vs risks.

Cross-Domain Design Patterns

Verification Strategy by Domain

- **Code**: Run tests, syntax checking, type checking
- **Finance**: Backtesting, compliance rules, risk limits
- **Healthcare**: Evidence linking, confidence thresholds, human review

Human-in-the-Loop Intensity

- **Code**: Low (automated tests catch most errors)
- **Finance**: Medium (compliance review for significant actions)
- **Healthcare**: High (clinician approval for all recommendations)

Common Success Factors

- Domain-specific tools and knowledge bases
- Clear escalation paths for uncertainty
- Audit trails for accountability

Adapt verification intensity to domain risk level.

Required Readings

This Week

- Jimenez et al. (2024). “SWE-bench: Can Language Models Resolve Real-World GitHub Issues?” arXiv:2310.06770
- Ridnik et al. (2024). “AlphaCodium: Code Generation with Flow Engineering.” arXiv:2401.08500
- Li et al. (2024). “FinAgent: A Multimodal Foundation Agent for Financial Trading.” arXiv:2402.18485

Supplementary

- Yang et al. (2024). “SWE-agent: Agent-Computer Interfaces Enable Software Engineering.” arXiv:2405.15793
- Singhal et al. (2023). “Large Language Models Encode Clinical Knowledge.” Nature

Focus on SWE-bench for understanding code agent evaluation.

Summary and Key Takeaways

Domain Insights

- **Code**: Most mature; clear success criteria, safe sandboxing
- **Finance**: High value but requires compliance awareness
- **Healthcare**: Highest stakes; human oversight essential

Design Principles

- Match verification intensity to domain risk
- Build domain-specific tools and knowledge
- Design clear human escalation paths

Next Week

- Research Frontiers and Final Projects

Domain expertise + agent capabilities = real-world impact.