TILBURG UNIVERSITY

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

JADS Jheronimus Academy of Data Science

Large Language Models for Service-Oriented Computing (**LLM4SOC**): Review and Research Directions

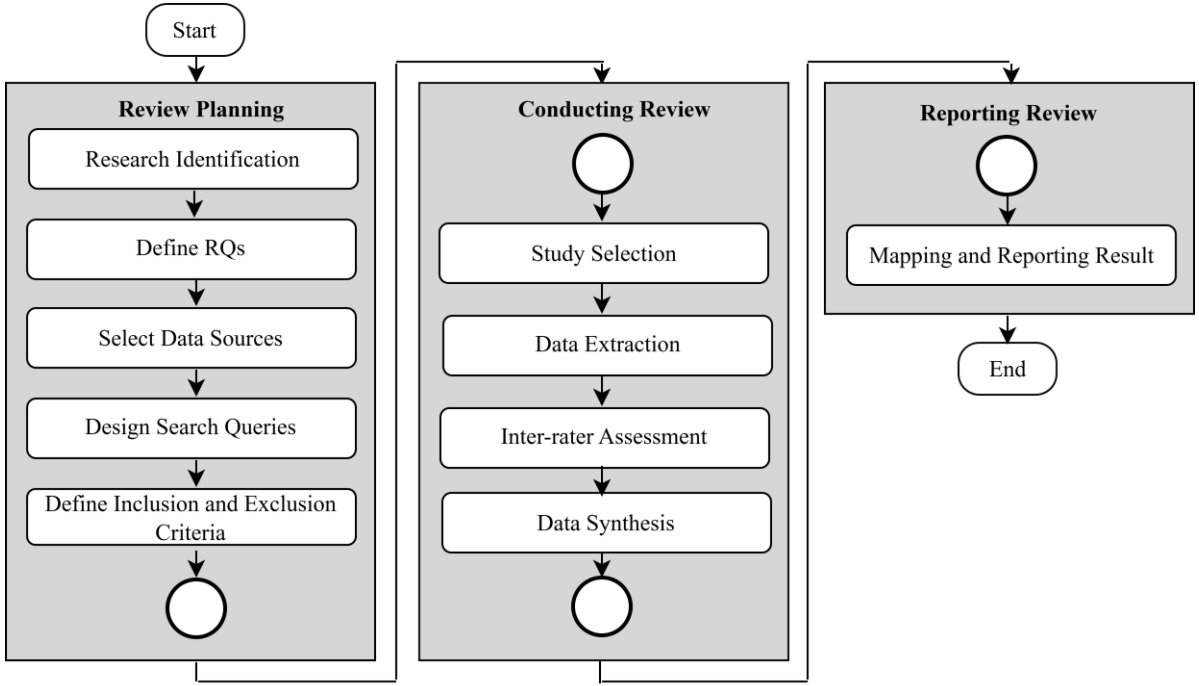
Indika Kumara, Hasan Kaplan, Joshua Owotogbe, Damian Andrew Tamburri, and Willem-Jan van den Heuvel

The Problem and Research Goals

- Problem
  - There is a considerable amount of academic and gray literature on LLMs for SOC.
  - However, there is no systematic literature review on the topic.
- Research Goals
  - Review the existing work on LLM4SOC
  - Develop a research agenda for LLM4SOC

[2]

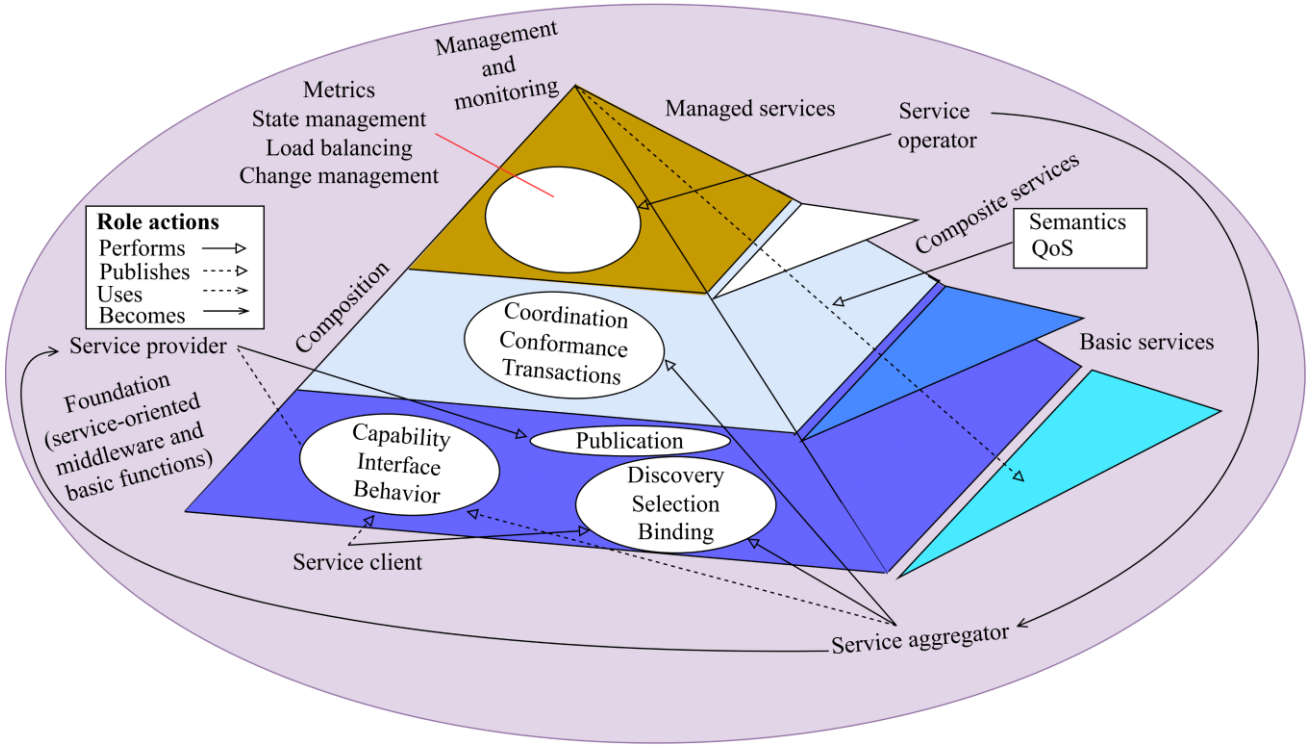
Research Methodology



Of 703 papers (search results), 64 were selected.

[3]

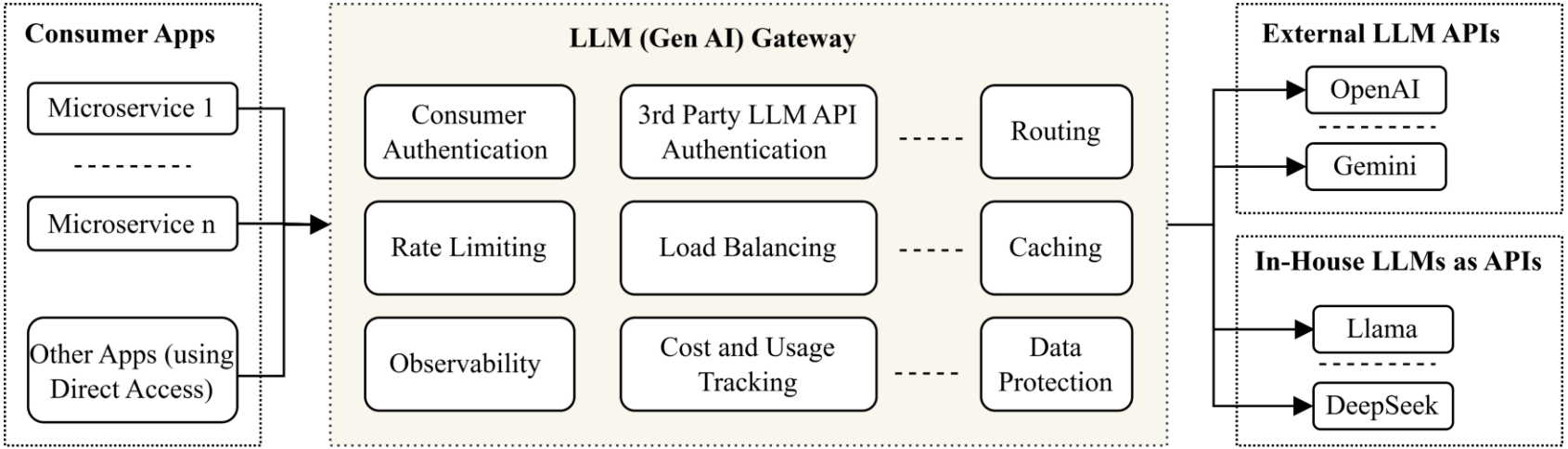
Organizing Results: Extended SOA Model



[4]

Service Foundation

- Enterprise service bus (ESB) was the SOA middleware for classical Web services.
- Microservices Architecture: ESB is replaced by an API gateway, service mesh, and event bus.
- LLM-based microservices require an additional middleware: an **LLM (Gen AI) gateway**.



[5]

Service Foundation

- **State of the Art**
  - A few works on LLM-based microservices and SOA middleware capabilities, such as *service recommendation* and *deployment optimization.*
- **Research Opportunities and Challenges**
  - LLM Inference Resource Management
    - Resource management for *LLM workloads*.
    - Optimize resource usage and energy consumption for *mixed workloads* in LLM-integrated microservice architectures.
  - LLM Gateway Capabilities
    - Request routing, rate control, caching, load balancing, LLM API usage and cost monitoring, etc.

[6]

Service Foundation

- **Research Opportunities and Challenges**
  - LLMs for Service Capabilities
    - Different types of business service logic (e.g., data and API access, data transformation, and extracting information from documents)
  - LLMs for Middleware Capabilities
    - Service annotation, publication, discovery, etc.
    - Conversational assistants to help developers correctly use and configure middleware platforms and tools. e.g., K8sGPT
  - Trustworthy and Secure LLMs in Microservice Architectures
    - Detecting and mitigating hallucinations and bias in LLM inferences consumed by microservices
    - Detecting and mitigating potential threats in the LLM-integrated microservice architecture (LLM Gateway)
      - prompt injection, model theft, model denial of service, insecure output handling, over-reliance, and information disclosure.

[7]

Service Composition

- **State of the Art**
  - A few works on *generating executable service orchestration logic* (including process models)
  - An LLM-based multi-agent framework for *automated service composition* - agents that can execute services/APIs or use tools
  - A study on mining *microservice dependency graphs* from service configuration files
- **Research Opportunities and Challenges**
  - Composability Analysis
    - Composability analysis that incorporates the information from *multi-model data* such as service descriptions, policies, SLAs (service level agreements), service usage history, and user comments on APIs.
    - Composability analysis for *LLM-based microservices* that provide a generic service interface that takes an NL input.

[8]

Service Composition

- **Research Opportunities and Challenges**
  - Dynamic, Adaptive, and QoS-aware Compositions
    - Autonomously and on demand, decide on the service composition plans.
    - Interpreting runtime *service execution errors using the domain knowledge* and taking the most appropriate actions.
    - Translate QoS requirements (*NL*) into the optimization logic.
  - Multi-agents for Service Composition
    - There is no comprehensive evaluation of the scalability and robustness of the agentic approach for composing services.

Service Composition

- **Research Opportunities and Challenges**
  - Trustworthy and Secure LLMs in Service Composition
    - Fairness issues, e.g., the popularity bias in service selection can lead to the dominant use of popular services.
    - Robustness of LLMs for hallucinations, prompt perturbations, etc.
    - Security vulnerabilities in LLM's decisions, e.g., selecting an insecure API or a deprecated API.
      - Prompt injection can be used to trick LLMs into making such decisions.

Service Management and Monitoring

- **State of the Art**
  - Provide *human-understandable insights* about the health of microservices observability data
  - Anomaly detection and generation of service dependency graphs
  - Process mining, including detecting anomalies in complex service workflows
- **Research Opportunities and Challenges**
  - Observability in LLM-powered Microservice Architectures
    - Creating the *taxonomies of failures and errors* in LLM-powered architectures (including multi-agent based microservices).
    - Accurately *collecting data from diverse LLM technology stacks* at a low cost to support diagnosing such issues.
    - Identifying and evaluating the potential *mitigation or recovery strategies* for those issues.

Service Management and Monitoring

- **Research Opportunities and Challenges**
  - Self-* Management Capabilities
    - LLMs can enable building sophisticated self-* management systems.
      - Multi-domain adaptability (e.g., making management systems adaptable to diverse microservice environments, ranging from cloud to edge).
      - Natural language understanding (e.g., interpreting natural language policies and configuration instructions, and identification of anomalies and faults using the complex textual telemetry data from diverse system components).
      - Generation and agentic capabilities (e.g., automatically generating and sending alerts and carrying out system reconfigurations using the relevant APIs and tools).
  - Trustworthy and Secure LLMs in Service Monitoring and Management
    - Errors in LLM's interpretation of data and system configuration decisions
    - Attackers can potentially compromise self-* systems to create system misconfigurations intentionally

[12]

Service Design and Development

- **State of the Art**
  - Many works focus on applying LLMs in business process modeling.
  - A few works on generating and validating test cases for microservices.
  - A chatbot to answer queries from developers regarding web services in a service marketplace.
- **Research Opportunities and Challenges**
  - Service-based Application Design
    - Domain-driven design (DDD), pattern-based design, contract/code-first approaches to service design and implementation, etc.
  - Design Maintenance (including Evolution and Migration)
    - Unique design maintenance use cases:
      - Substituting or replacing services in the architecture
      - Changing activities and their control and data flows in process models
      - Migrating from a monolithic application architecture to a microservice architecture (and vice versa)
      - Service orchestration into choreography (and vice versa)
      - A containerized, resource-oriented microservice model to an event-driven serverless model (and vice versa).

[13]

Service Design and Development

- **Research Opportunities and Challenges**
  - Implementation Artifact Generation
    - Generation/repair of domain-specific artifacts such as IaC scripts, service policies, monitoring queries (observability logic), and microservice middleware configurations.
    - Consider typical business logic in microservices (e.g., API request and response validation, consuming and publishing events, database access, and API access).
    - Representative benchmark datasets for evaluating LLM-based approaches in SOC design and implementation.
  - Implementation Artifact Maintenance (including Evolution and Migration)
  - Design and Implementation Guidelines and Assistants
  - Trustworthy and Secure LLMs in Service Design and Development

Recap

- Large language models (LLMs) can significantly impact the service-oriented competing (SOC) landscape, and hence, warrant revisiting the traditional research challenges in SOC.
- This study systematically reviewed the literature on LLMs for SOC.
- It formulated a research roadmap (LLM4SOC research roadmap), providing research directions for the key pillars of SOC: service foundation, service composition, service monitoring and management, and service design and development.

Q/A

www.jads.nl