Bridging Natural Language and Reservoir Simulation:

A Configurable Architecture for Democratizing Subsurface Modeling

Abstract for Call for Papers

Reservoir simulation remains one of the most technically demanding disciplines in the energy sector. Despite decades of software evolution, the fundamental barrier persists: translating engineering intent into executable simulation input requires specialized expertise that takes years to develop. This paper presents the architecture of the Oxy Reservoir Simulation Agent (ORSA), a system designed to bridge the gap between natural language interaction and domain-specific simulation syntax.

The core technical challenge lies not in the simulation itself, but in the translation layer. When a reservoir engineer says "reduce injection in Well A-15 by 500 barrels per day starting March," the system must parse intent, validate entities against actual field assets, and generate syntactically correct input files that respect the complex interdependencies of reservoir simulation keywords. We propose a multi-stage pipeline that decomposes this translation into discrete, validated steps: speech recognition, intent classification, entity extraction, asset validation, syntax generation, and physics-based verification. Each stage includes rollback capabilities, enabling graceful degradation when confidence thresholds are not met.

From a software architecture perspective, ORSA addresses the heterogeneity problem inherent in global energy operations. International deployments face strict data sovereignty requirements, often mandating air-gapped installations with no external connectivity. Simultaneously, cloud-native deployments demand elastic scaling and managed service integration. We present a configuration-driven approach where every major component—from database backends to inference engines—can be substituted without architectural modification. The data layer implements a domain mesh pattern with purpose-built storage: columnar databases for time-series simulation results, document stores for flexible training datasets, relational systems for transactional user data, and graph databases for modeling well connectivity patterns.

The system architecture explicitly separates concerns between synchronous user interaction and asynchronous backend processing. Users submit requests through conventional API calls and receive immediate job identifiers; an orchestration layer manages workflow execution, resource allocation, and result delivery through event-driven messaging. This pattern accommodates simulation runs ranging from seconds to hours

without blocking user interfaces or requiring persistent connections.

A significant design consideration involves model flexibility. The translation pipeline requires language understanding capabilities, but the optimal model varies by deployment context. Air-gapped installations may run locally-hosted open-source models; connected environments might leverage external APIs; specialized deployments could use custom fine-tuned models trained on proprietary simulation datasets. The architecture abstracts model access behind a unified interface, enabling runtime selection based on availability, performance requirements, and regulatory constraints.

We also address the observability challenges inherent in distributed, multi-database systems. Request tracing spans service boundaries; centralized logging correlates events across asynchronous workflows; health monitoring tracks model performance and triggers automatic failover when degradation is detected. These capabilities prove essential for diagnosing issues in production deployments where traditional debugging approaches fall short.

The broader objective extends beyond technical convenience. Reservoir simulation expertise remains concentrated among a limited pool of specialists, creating organizational bottlenecks and knowledge preservation risks. By lowering the barrier to simulation interaction—while maintaining rigorous validation—the system enables broader participation in subsurface decision-making. Business teams can explore scenarios without waiting for specialist availability; junior engineers can learn by interacting with systems that explain their transformations; experienced practitioners can focus on interpretation rather than syntax.

This paper contributes: (1) a reference architecture for natural language interfaces to domain-specific technical systems, applicable beyond reservoir simulation; (2) patterns for configuration-driven deployment across heterogeneous infrastructure environments; (3) strategies for hybrid synchronous-asynchronous communication in long-running technical workflows; and (4) practical approaches to observability in polyglot persistence architectures. We discuss implementation considerations, risk mitigation strategies, and lessons learned from architectural decisions made under real-world operational constraints.

Keywords: reservoir simulation, natural language processing, microservices architecture, domain-specific languages, configurable systems, event-driven architecture, subsurface modeling, enterprise software architecture

Word count: approximately 650 words