Autonomous Design Optimization for Nuclear Energy (AutoDONE): An Agentic Framework for Nuclear Power Plant Design

Sonali Sinha Roy
[a] Idaho National Laboratory, Idaho Falls, USA, sonali.sinharoy@inl.gov

Abstract: Power generation systems encompass complex designs with numerous interdependent subsystems, posing significant engineering, logistical, and management challenges. Current practices in system design and procurement often focus on component-level specifications and upfront expenditure, overlooking holistic system performance and long-term operational costs. Limited tools for design exploration and time-consuming roadblocks to iterative design can lead to suboptimal systems and operational inefficiencies. This paper introduces the Autonomous Design Optimization for Nuclear Energy (AutoDONE) application, which leverages agentic artificial intelligence (AI) and digital engineering tools to automate and optimize the selection of commercial-grade equipment for nuclear power plants. The application accepts stakeholder requirements as input, identifies components that fulfill those requirements, and creates hierarchical system models for comprehensive lifecycle analysis and optimization. The capabilities of AutoDONE are demonstrated in a test case where an optimized set of components for a nuclear power plant cooling water loop are autonomously identified and the long-term operation of the selected system is simulated. Through AI-driven automation of data processing and verification, model generation, and human-in-the-loop decision-making, AutoDONE aims to shorten the initial design timeline by about 80% and design modification time by 50%. AutoDONE shifts the application of AI from hypothetical designs to verifiable components for real-world systems, which can expedite design workflows and enable engineering flexibility in nuclear power plants. The successful implementation of this application could significantly accelerate the process of developing new nuclear facilities and upgrading existing ones, thus helping scale the deployment of power generation capabilities to meet the rising global energy demand.

1. INTRODUCTION

Energy generation systems have complex designs involving several interdependent subsystems. In addition to multidisciplinary engineering considerations, there are multiple logistical and management-centric challenges. The design process for such systems must take into account engineering requirements and specifications, budgetary constraints, operational safety, as well as overall performance. Under current practices, most of this work is manually executed, and the system-level design is often not optimized across the plant's lifecycle. This is especially evident in the equipment selection and procurement processes, which can take several months to over a year [1, 2], especially if specialized equipment or customizations are needed. Automating the design workflow can accelerate the process and a rigorous analysis of the system before design finalization can enable informed decision-making. The emerging domain of agentic artificial intelligence has the potential to facilitate both process automation and operation-centric design. The following sections elaborate on the challenges involved in the current process, and the potential benefits that can be achieved by adopting an agentic semi-autonomous design process and a dedicated tool that implements it.

1.1. Current State of Practice

The lifecycle of energy facilities spans several phases (e.g., design, construction, procurement, commissioning, deployment, operations, decommissioning), each of which involves unique challenges. The plant design phase is driven by system complexities since, in addition to the primary subsystem (i.e., the reactor for nuclear power plants), several supporting subsystems also perform critical functions like cooling, power conversion, radiation monitoring, and fire protection. Each subsystem consists of a series of components like tanks, pumps, valves, sensors, and actuators. The process of designing the

system involves identifying the required equipment, specifying the requirements (for each piece of equipment, for the subsystems, and for the overall system), and designing the connections and interfaces for the subsystems to work cohesively. The goal of the procurement process is to acquire all the necessary components while adhering to the system specifications. It is often necessary to make procurement decisions (e.g., to make, buy, customize, or reuse) for many components. For commercial off-the-shelf (COTS) purchases or customizable equipment, next steps include conducting market surveys, contacting vendors, deciding the specific components to buy, managing the assigned budget, and monitoring delivery logistics. Most aspects of the current practice are handled manually, resulting in a slow process.

1.2. Pitfalls of Current Practices

Designing energy systems is an iterative process that starts with a conceptual design and incorporates additional details over time. Equipment procurement is often tedious and time-consuming because it requires safety and cost assessments, vendor vetting, and delivery timelines. Design adjustments or modifications involve manual steps like recalculation, redrawing, and document revision, making the iterative design expensive and time-consuming. Due to phase-wise budget limits, projects usually prioritize minimizing capital expenditures while ensuring the design specifications and safety requirements are met. However, long-term factors like operating expenditures and overall system performance are often oversimplified or ignored, leading to a lack of end-to-end lifecycle analysis. Design and procurement processes focus on component-level requirements, costs, and risks, rather than the entire equipment portfolio, sometimes resulting in suboptimal operation across the system's lifecycle due to component incompatibility, high operational costs, and high failure rates at interfaces.

To understand some of the impacts of the above-mentioned pitfalls, consider a section of a hypothetical heat extraction system consisting of a chiller unit, a pump, and a thermally insulated storage tank. Water is cooled as it passes through the chiller, after which it is pumped to the tank. Let us assume that the chiller and tank need customization. Since these are critical components, a large part of the available budget is spent on selecting highly reliable models and then customizing them. However, to stay within budget, a relatively cheap and moderately reliable pump is selected. During the operational phase, the pump fails more often than the chiller and tank. Since it is a connecting piece, the overall reliability of the entire chiller-pump-tank section is heavily affected by the reliability of the pump. Therefore, despite buying two expensive and reliable components, the cost of downtime and maintenance required for the third component negatively affects the overall system's performance and operating expenditure. Lifecycle assessment of the entire system in the design phase could have prevented this outcome.

1.3. Mitigation of Pitfalls through Agentic Artificial Intelligence

Artificial intelligence [3] is a collective term for technology that enables machines to simulate complex human behaviors. It encompasses machine learning (ML) [4], natural language processing (NLP) [5], and generative AI (GenAI) [6], including large language models (LLMs) and multimodal foundation models [7]. A rapidly emerging area is agentic AI [8], where AI "agents" can interface with external tools and data. This allows for the creation of custom solutions that integrate AI with domain-specific software or databases to systematically tackle complex, multi-step problems. In engineering domains, agentic AI applications can include physics solvers, modeling and simulation software, custom codes for analytics, and so on. Digital engineering [9] and its subset, model-based systems engineering (MBSE) [10], utilize computer models and digital databases to support system lifecycle management. With agentic AI, there is an opportunity to harvest the combined benefits of digital engineering and AI.

The adoption of both digital engineering and AI has had a positive impact on several industries, including aerospace, automotive, and manufacturing. MBSE users have observed benefits [11] such as increased efficiency, improved communication, reduced costs, better analysis capability, and early error detection. Using digital engineering, Boeing achieved an 80% reduction [12] in assembly hours for the T-7A, an advanced trainer for the U.S. Air Force. Digital transformation of the Siemens Electronics Works Amberg factory in Germany resulted in a 14× improvement [13] in productivity. A survey led

by Standford University [14] found that the use of GenAI can result in efficiency gains of up to 60% [15]. In a Microsoft-commissioned study [16], IDC surveyed 2,109 companies worldwide and found an average return of $3.5 per dollar of AI investment. Companies like Tesla Motors and GE [17] have heavily invested in AI-powered digital twins for analytics, diagnostics, optimization, and predictive maintenance. The relatively nascent area of agentic AI has already initiated a paradigm shift in software development [18] and shows significant promise for engineering design and analysis [19].

Several use cases in the energy industry are particularly well-suited for agentic AI implementation, including power generation megaprojects, research and development for high-impact experiments, long-term technology development, and government-funded capital asset acquisition. A particularly lucrative use case is the design of complex energy systems such as nuclear power plants. The traditional process, as described earlier in this paper, involves several time-consuming manual steps. Plant-level design often takes years. Depending on complexity, system and sub-system design timelines range from a couple of weeks to about six months. Conversations with design engineers for nuclear facilities revealed that it takes three to six months to mature a conceptual (~20%) design into the final (~90%) design for a complex subsystem. For existing systems, design modifications or upgrades can take over two weeks.

The work presented in this paper is a preliminary proof-of-concept for AutoDONE (Autonomous Design Optimization for Nuclear Energy), an agentic AI application for design workflow automation. It utilizes AI agents for specialized tasks and leverages digital engineering methods like behavioral modeling and lifecycle analysis to ensure that systems are designed for the most desirable operational metrics (e.g., cost, risk, performance). AutoDONE can significantly accelerate the current design timelines. For example, once fully deployed and validated, it is projected to reduce the conceptual-to-final design timeframe from three months to two weeks (including iterative optimization), thus resulting in a nearly 84% gain in efficiency. Similarly, the time required for design changes is estimated to drop by at least 50% from the status quo of two weeks to about one week (with human-in-the-loop verification). Additionally, AutoDONE facilitates systematic data cataloging, statistical analysis of the design, and multi-objective optimization, thereby providing a basis for informed decision-making backed by traceable, quantitative data.

2. AGENTIC AI OVERVIEW

Agentic artificial intelligence (agentic AI) refers to a class of AI systems in which foundation models are embedded within a control architecture that can interpret objectives, plan intermediate steps, invoke external tools or functions, access databases, observe results, and iteratively update its behavior until a task is completed. In contrast to a conventional single-turn LLM interaction (e.g., prompting to an AI chatbot), an agentic system is organized as a closed-loop process. The model receives a goal and context, decides what information or action is needed, executes one or more operations through software interfaces, incorporates the resulting observations, and continues until it reaches a stopping condition. This loop may be simple, such as querying a database to produce a response, or complex, such as coordinating multiple specialized agents over several task phases.

The central distinction between a generic LLM application such as an AI chatbot and an agentic AI application is therefore the execution architecture surrounding the underlying model. A basic LLM application maps an input prompt to an output. An agentic application is often created to specialize in certain tasks, and includes capabilities such as external tool use, planning, task-level orchestration, error recovery, state and memory management, and so on. Foundational work on reasoning-and-acting methods, such as ReAct, showed that language models can interleave reasoning traces with task-specific actions, enabling the model to gather information from external environments and revise its trajectory based on observations [20]. Similarly, tool-use research demonstrated that LLMs can be trained or prompted to decide when to call an external API (application programming interface), what arguments to pass, and how to incorporate tool outputs into subsequent reasoning [21]. These ideas form the technical basis for modern agentic systems, where the model is treated less as a passive text generator

and more as a policy component within a larger decision-making framework. **Table 1** summarizes the key concepts relevant to agentic AI.

Table 1: Key concepts associated with agentic AI

Concept	Explanation
Agent	A software entity that uses a model, instructions, state, and tools to pursue a user- or system-defined objective under specified constraints. An agent acts in a controlled capacity to select actions in response to context and instructions. Its action space is constrained by available tools, permissions, guardrails, human approval points, logging, and domain-specific policies. It is important to note that most high-value agentic applications are not fully autonomous systems; they are semi-autonomous workflows in which an AI component performs information gathering, classification, planning, drafting, or task execution while humans retain oversight for consequential decisions.
Tool	An executable capability exposed to an agent, such as a database query, file retrieval operation, simulation routine, web request, calendar action, or domain-specific API. A function call is a structured invocation of such a tool, typically with validated arguments.
Memory	Information retained across turns or task steps. Short-term memory may include the current conversation and intermediate observations, whereas long-term memory may include stored user preferences, retrieved documents, embeddings, prior task outcomes, or persistent application state.
Agent actions	An agent may execute one or more categories of specialized tasks. *Planning* denotes the decomposition of a high-level objective into intermediate actions or subtasks. *Perception* is the agent's ability to "see" or read its environment. It collects data from files, live user inputs, or connected software. *Reasoning* is the "thinking" core of an AI agent. It uses logic to process information, draw conclusions, and solve problems. *Reflection* or *self-evaluation* is a type of reasoning process in which the agent critiques an intermediate output, detects errors, or revises a plan before continuing.
Orchestration	Process of coordinating, managing, and governing multiple specialized AI agents. An orchestrator could be a set of hard-coded rules or a fluid AI agent that can route tasks, share context, and manage tool usage across specialized task agents to achieve complex outcomes.
Framework and harness	These generally refer to the software layer that surrounds the model and implements the agent loop. A framework usually provides reusable abstractions for agents, tools, memory, tracing, evaluation, and multi-agent workflows. A harness is the runtime configuration that supplies the model with the right context at the right time and governs how the model interacts with tools. In this sense, an agent can be viewed as a model plus a harness, where the model supplies probabilistic reasoning and language understanding, while the harness supplies prompts, context retrieval, tool schemas, middleware, safety checks, state management, and termination logic [22]. Frameworks such as LangChain/LangGraph, AutoGen, Semantic Kernel, Claude Agent SDK, and OpenAI Agents SDK operationalize these concepts by managing tool loops, handoffs, tracing, guardrails, or multi-agent coordination [23].
Model Context Protocol (MCP)	MCP [24] is an open protocol for connecting LLM applications to external context, tools, and data sources. It treats the LLM application as the host and the connector within that application as the client. The server exposes resources, prompts, or tools through standardized protocol messages. Conceptually, MCP addresses the "agent-to-tool" integration problem. Rather than building custom connectors for every combination of model application and external system, MCP provides a common interface through which tools and data sources can be exposed to LLM-powered applications.
Agent2Agent (A2A)	A2A [25] addresses a different problem: communication among independent agentic systems. The A2A specification defines an open standard for interoperability between agents that may be implemented by different vendors, frameworks, or organizations. Its goals include agent discovery, capability description, task delegation, negotiation of interaction modalities, and secure exchange of information without requiring direct access to another agent's private tools, memory, or internal state.

2.1. Agentic AI Architecture

Agentic AI applications can use various architectures, agent configurations, and orchestration patterns. A typical agentic AI system can be represented as a layered architecture. At the top is the **user**, who provides a goal, question, or task. The request is passed to an **orchestration layer**, which processes and decomposes the request, performs high-level planning, routes work to specialized agents, and enforces stopping conditions. The orchestrator also manages the application's **state, memory, and context**, which may be accessed by the **AI agents** invoked by the orchestrator. The agents utilize models such as general LLMs, reasoning models, domain-specialized models, embedding models, and classifiers. Model Context Protocol (MCP) allows them to interact with and invoke **tools**, which could be databases, internal APIs, external services, custom codes, simulation engines, and so on. For production-grade systems, it is also important to surround these functional layers with governance services like authentication and authorization, policy enforcement, human-in-the-loop approval, monitoring, tracing, evaluation, and audit logging.

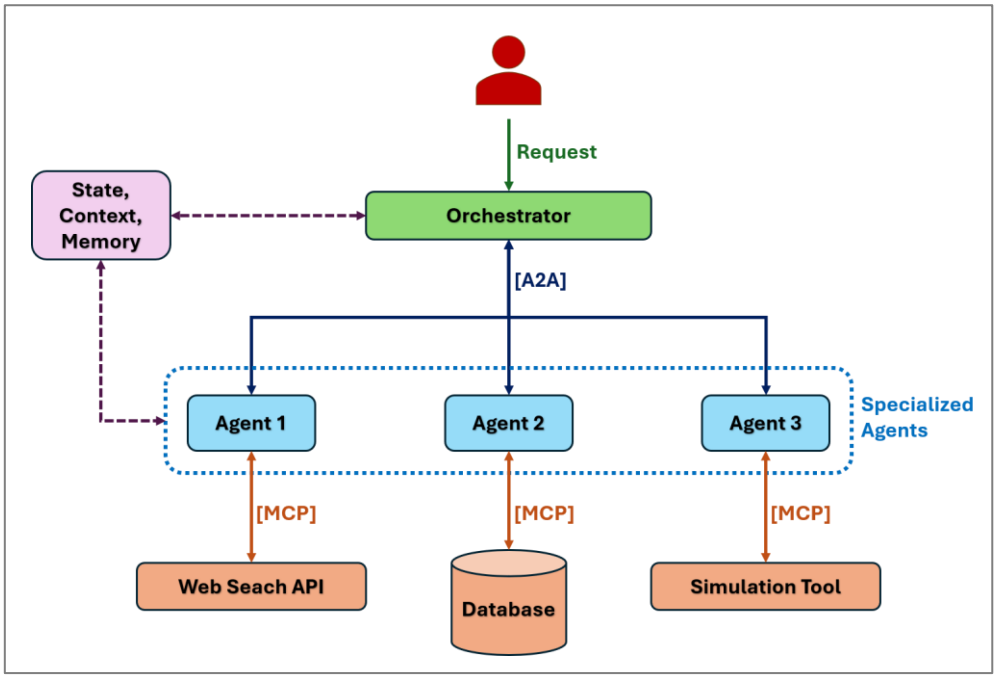

Figure 1: Notional agentic AI application with three task agents and an orchestrator. The agent-to-agent (A2A) protocol enables communication among agents, while MCP allows agents to call tools.

Figure 1 shows a notional agentic AI application, which has three specialized task agents and an orchestrator. When a user submits a request to this agentic application, the orchestrator initializes the appropriate application-level state, retrieves relevant context, decomposes the user's request into agent-level tasks, and sends structured prompts to the corresponding agents using the agent-to-agent (A2A) protocol. If invoked, the Agents 1, 2, and 3 leverage MCP to execute a web search, query a database, and run a simulation, respectively. Depending on the specific task and orchestration pattern, the agents may be invoked parallelly, sequentially, or in a loop. The application's operation terminates when the orchestrator determines that the user's request has been completed, confidence is sufficient, or a policy or resource limit has been reached.

Agentic AI is best understood as an engineering pattern for placing LLMs inside controlled, observable, tool-using workflows. The agent is the decision-making unit; the harness or framework is the runtime structure that supplies context and governs action; MCP standardizes access to tools and external context; and A2A standardizes communication among independently operating agents. The conceptual overview provided in this section acts as the foundation for understanding the agentic application developed in this study, which is presented in the next section.

3. THE AUTODONE FRAMEWORK

AutoDONE (Autonomous Design Optimization for Nuclear Energy) is an agentic AI application developed to accelerate the early-stage design of nuclear power plant systems. Given a set of system-level process requirements or specifications (e.g., required flow rate, pressure rise), the application autonomously identifies compatible off-the-shelf components (e.g., pumps, control valves, air handlers), ranks candidate combinations by user-defined optimization objectives, and builds quantitative operational models used to estimate the operational statistics (e.g., cost, reliability) of the resulting system over its service life. The goal is to compress a tedious design task that traditionally requires a manual catalog search, spreadsheet costing, and a separate reliability analysis into a single automated, reproducible workflow, thereby significantly decreasing design time, increasing flexibility, and enabling informed decision-making.

3.1. Application Overview

AutoDONE is a multi-agent application built on the LangGraph [26] framework. The intended end-users are systems engineers or design engineers in the nuclear industry. As inputs to the tool, they will provide the conceptual system architecture and the system-level specifications. Any data sources, such as vendor and supplier catalogs, inventory, and manufacturer-provided product details can also be provided. AutoDONE uses this data to identify and simulate optimal design options, generating operational insights that can facilitate better decisions in the design phase.

AutoDONE uses A2A for inter-agent communication, transporting requests between the orchestrator and task agents over standard web protocols. A common agent interface defines a single request-and-response contract that every agent must satisfy. The agent roster is registry-driven, i.e., new agents that correspond to additional component categories or analysis steps can be added to the architecture without completely changing the orchestration logic. The underlying LLM provider and model are fully configurable in the setup, allowing users to customize the application and use their own API keys. To accomplish the end-to-end design workflow, AutoDONE includes four agent types:

- **Component search agents:** One per component category; each searches a domain-appropriate data source (e.g., a local catalog of product information sheets, a structured database, or the open web), extracts candidate component specifications, and returns a ranked list with the supporting rationale for each candidate. These agents interact with MCP servers that expose the database and web search tools.
- **Optimization agent:** An engineering/economics calculator that checks every combination of candidate components against physical feasibility constraints (e.g., a valve must be able to pass the required flow without excessive pressure loss), computes the user-defined objective function (e.g., operating cost) of feasible combinations from first-principles physics equations, and returns the top-ranked combinations. A data-quality penalty is applied on combinations that rely on LLM-estimated rather than manufacturer-confirmed specifications. Once the ranked list is produced, the user selects the design to carry forward to simulation, keeping an engineer in the loop at the key step that commits the design.
- **Behavior modeling agents:** One per component category; each converts a selected component's specifications into a finite-state operational using Idaho National Laboratory's EMRALD software [27], wrapped in a versatile MCP server. The behavior model for each component includes a small set of discrete states, along with the rates or probabilities of each state transition. State structure is fixed by engineering design and the key states for various types of components are derived from United States Nuclear Regulatory Commission's (NRC) Industry Average Parameter Estimates [28]. The LLM's primary role in this type of agent is to extract numeric parameters for the EMRALD model from free-text specifications, as well as data passed by the search and optimization agents.
- **System simulation agent:** Combines the component-level state models into one coupled system model and performs a stochastic (Monte Carlo) simulation forward in time, including cross-component dependencies (e.g., a stuck-closed valve stressing the pump) and common-

cause failure events that can disable multiple components simultaneously (e.g., loss of electrical power). Outputs include system availability, downtime, mean time between failures, and life-cycle cost, each with an associated statistical distribution.

Two front ends expose the same underlying pipeline: (i) an interactive browser interface, in which the user enters requirements through a form and reviews ranked designs, state diagrams, and reliability results as they are produced; and (ii) a command-line interface intended for scripted or batch use. **Figure 2** shows a high-level architecture of the AutoDONE application.

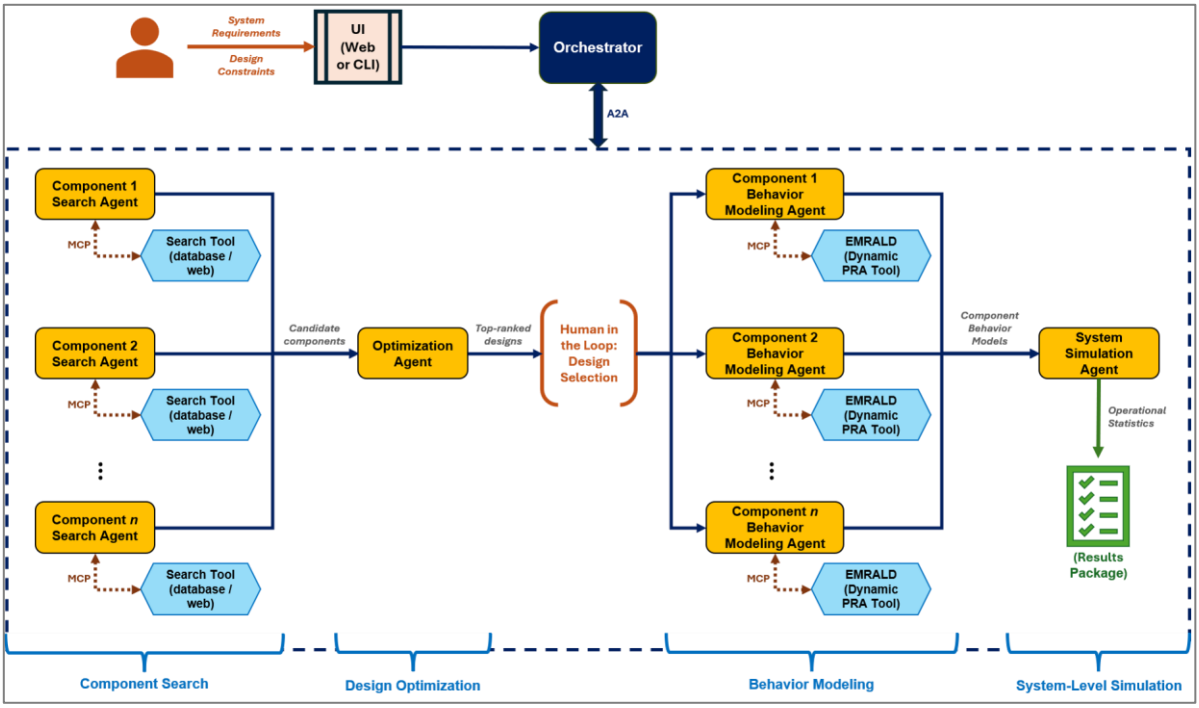

Figure 2: High-level architecture of the AutoDONE application. There are specialized agents for four key types of tasks: component search, design optimization, behavior modeling, and system-level simulation.

3.2. Demonstration: Simplified Component Cooling Water (CCW) Loop

The AutoDONE application is demonstrated on a simplified single-train cooling loop representative of a nuclear plant's secondary or auxiliary cooling system. It consists of a pump that circulates a working fluid through a motor-operated control/isolation valve and a shell-and-tube (or equivalent) heat exchanger, arranged in series. This three-component loop was chosen because it exercises all four agent roles (selection, optimization, behavior modeling, simulation) while remaining small enough to interpret by hand, providing a controlled test case ahead of application to more complex, multi-train plant configurations.

In this example, the workflow is as follows:

1. The user specifies the loop's process requirements at the start of the run.
2. The orchestrator processes the user's input to identify component-level specifications (e.g., calculating heat exchanger duty using the physical properties of the working fluid, targeted temperature difference, and flow rate) and passes the specifications to the search agents.
3. Using the orchestrator-supplied specifications, three search agents parallelly and independently retrieve candidate pumps, valves, and heat exchangers. The pump is identified from a local database, while the valve and heat exchanger are shortlisted through a web search.
4. The optimization agent evaluates every feasible pump–valve–heat-exchanger combination, discards combinations that violate a physical constraint (insufficient pump head, undersized

valve, insufficient heat-exchanger duty, or an inadequate pressure/temperature rating), and ranks the remainder by projected annual operating cost, adjusted for data quality.

5. The user is presented with the top three (lowest cost) options, out of which they select one.
6. The three behavior modeling agents parallelly generate a finite-state operational model for the selected pump, valve, and heat exchanger. The states correspond to normal operation, degraded operation, and various failure modes during startup and while in service, as well as maintenance. The state-transition rates are derived from the component's specifications and NRC's reliability data.
7. The system simulation agent combines the three component models into a coupled model of the full loop (including cross-component dependencies and common cause failure events) and runs a stochastic simulation forward in time under a selected simulation preset (trial count and mission duration).

Both the optimization and simulation agents assume 8,760 operating hours per year and an electricity rate of $0.10/kWh. The underlying foundation model for all the agents in the current example is Gemma 4 31B (Reasoning) hosted on high-performance computing clusters at the author's organization.

To demonstrate the workflow, the input values shown in **Figure 3** were provided to the AutoDONE application through the web interface. The top three combinations are summarized in **Table 2**.

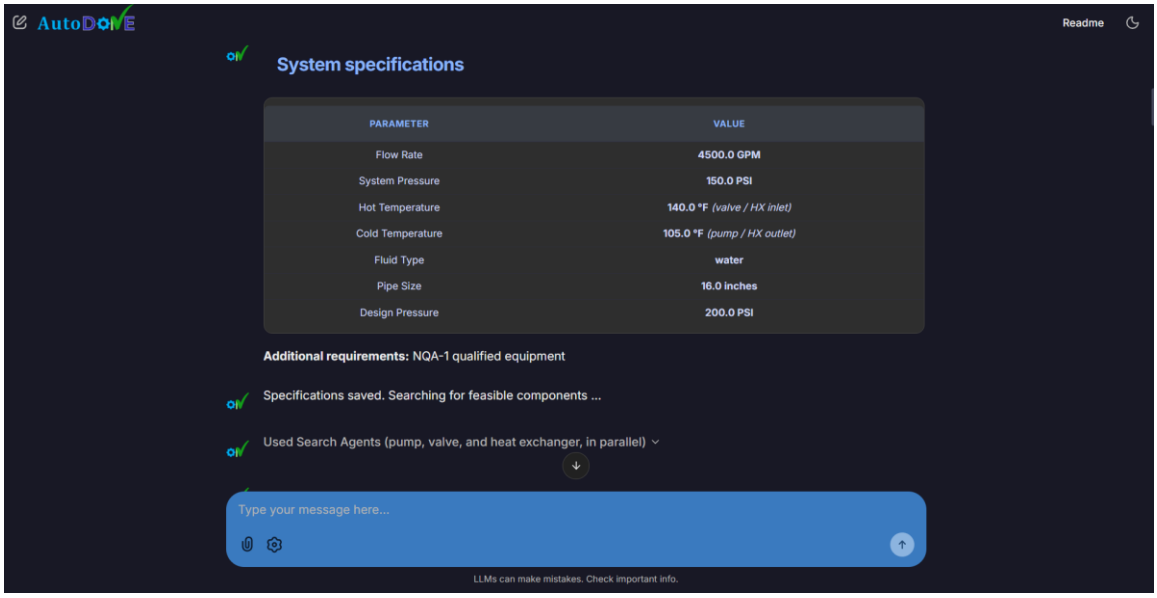

Figure 3: The AutoDONE web interface showing the input parameters (i.e., system-level specifications) for the simplified cooling loop.

Table 2: Top three system designs (component combinations) based on the user-defined system-level specifications.

Rank	Pump	Valve	Heat Exchanger	Estimated Annual Operating Cost	Selected?
1	Flowserve DMX/DMXD	Emerson Vanessa-MOV-16	Alfa Laval Shell-and-Tube ST-Nuclear	$43,512	Yes
2	Flowserve DVSS	Emerson Vanessa-MOV-16	Alfa Laval Shell-and-Tube ST-Nuclear	$43,512	No
3	KSB RUV	Emerson Vanessa-MOV-16	Alfa Laval Shell-and-Tube ST-Nuclear	$43,512	No

A few observations regarding these options:

- All three combinations were flagged for low data quality, meaning the complete set of specifications was not found, and the agents had to make some assumptions. Therefore, the operating cost is a broad estimate due to the lack of concrete data.
- The three options each have the same valve and heat exchanger, indicating that these components are the cost drivers. Since the pump's operating cost is primarily dependent on its operating speed and the user input specifies a target flow rate, the cost is not impacted heavily by the pump options for the same operating conditions.
- Since the application is sourcing the pump from a small, restricted, local catalog, it tries to select one that is the closest match, even if all the specifications are not appropriately mapped. For example, the minimum head of the Flowserve DMX/DMXD pump (300+ feet) is more than twice that of the requirement (~150 feet), whereas the other two pumps are the incorrect type for the application.
- The valve and heat exchanger correspond to more technically appropriate products, but the displayed name involves some hallucination, since they include additional characters (-MOV-16 and ST-Nuclear) that do not match the actual product IDs.

The first combination was selected, and state diagrams for the individual components were autonomously generated by the behavior modeling agents. The diagrams are shown in **Figure 4**.

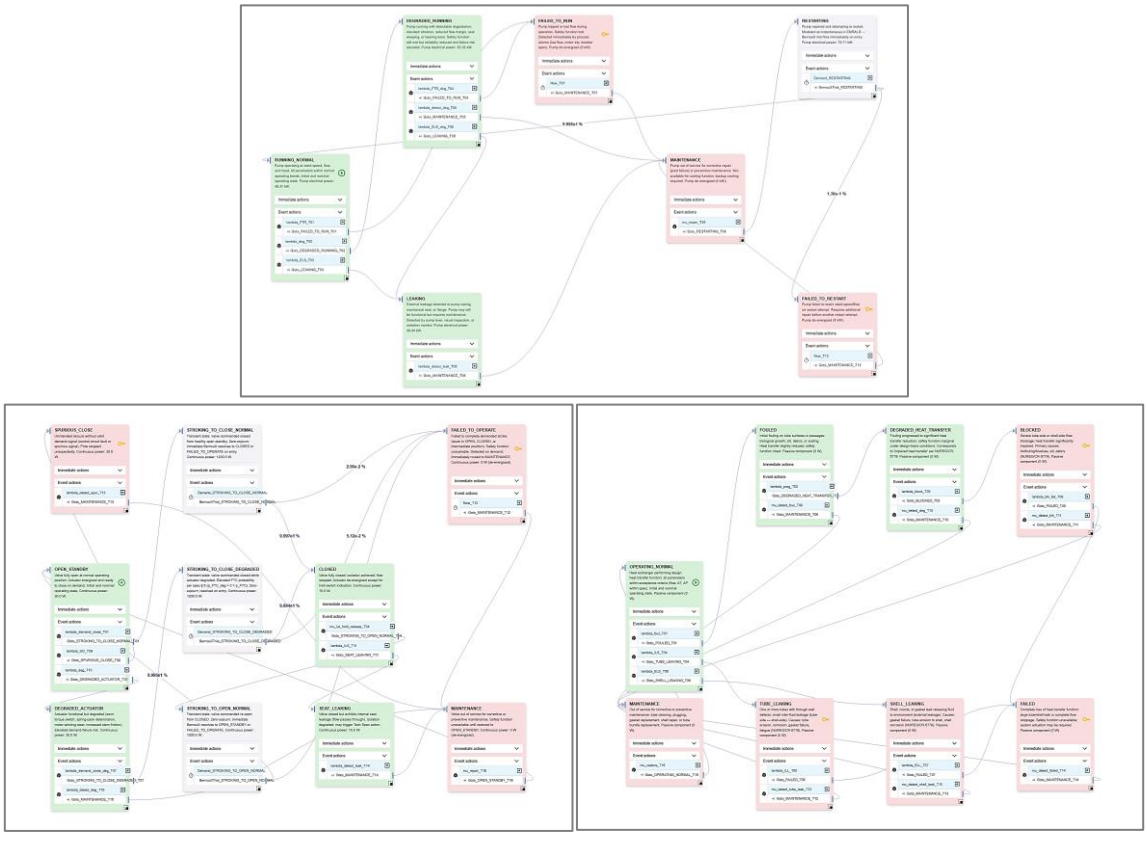

Figure 4: EMRALD-generated state diagram for the selected pump (top), valve (bottom left), and heat exchanger (bottom right).

Finally, the simulation agent ran Monte Carlo simulations for 1,000 trials, each representing a 10-year mission period. **Figure 5** and **Figure 6** are screen captures of the web interface, showing the simulation outputs. The estimates are based on NRC's industry data (where available) and the model's engineering judgement.

System reliability over a 10-year mission

METRIC	VALUE
System availability	**99.85 %** (90 % confidence range: 99.57 % – 100.00 %)
Average system downtime	134 hours (≈ 5.6 days)
Average time between system failures	2,080 hours (≈ 86.7 days)
Failure frequency	4.205 events / year

Estimated lifetime operating cost (10-year mission)

COST TYPE	TOTAL	PER YEAR
Energy (electricity)	$424,392	$42,439
Maintenance events	$121,110	$12,111
Combined total	**$545,502**	**$54,550**

Figure 5: Estimated system-level reliability metrics, annual operating cost, and 10-year operating cost (without inflation adjustment).

Component-level reliability

COMPONENT	AVAILABILITY	AVG. FAILURES	AVG. MAINTENANCE EVENTS
Pump	99.88 %	1.70	2.16
Valve	99.97 %	0.98	1.04
Heat exchanger	99.98 %	0.00	0.10

External events that affect multiple components at once

These are shared support-system disturbances (grid loss, fire, etc.) that trip more than one component simultaneously.

EVENT	AVG. OCCURRENCES IN 10-YR MISSION	WHAT IT MEANS
Loss of offsite power	0.86	Loss of offsite power; trips electrically-driven components.
Loss of service water	0.46	Loss of service water; pump motor cooling lost.
Local fire / electrical fault	0.09	Local fire / electrical fault in shared cabinet.
Beyond-design-basis seismic event	0.01	Beyond-design-basis seismic event; all components offline.

Figure 6: Estimated component-level reliability metrics and common cause failures.

Upon manual verification, the reliability and availability metrics seem reasonable, but there appears to be a significant underestimation of the cost metrics by both the optimization agent and the simulation agent. This points to a probable error in the agent prompts and calculation logic rather than a hallucination and warrants thorough review of the corresponding agents.

4. CONCLUSION

The work presented in this paper is the initial proof of concept for the AutoDONE application. The overview and demonstration presented here highlight some of key advantages and strengths:

- **Speed of design:** A single run takes a design task from raw process requirements through component search, cost-based ranking, operational modeling, and reliability simulation in minutes, as opposed to the traditional manual steps that normally require weeks or months.
- **Balanced use of AI models and deterministic processes:** Numerical engineering formulas (cost, feasibility) and state-machine structures are derived from vetted sources, while LLMs are used for search and data extraction, keeping the application's quantitative outputs reproducible and auditable.

- **Modular, extensible agent design:** Because the pipeline is registry-driven and agents use standardized protocols, new component categories or analysis steps can be added without modifying the orchestrator. This lets the same architecture generalize from the cooling-loop example to other multicomponent plant systems. Additionally, the underlying LLM provider is user-configurable although the application's performance and output quality will differ based on the model used.
- **Built-in data-quality awareness:** The optimization agent applies a penalty to combinations that rely on estimated rather than manufacturer-confirmed specifications, and flags results accordingly. This gives users an explicit, quantified signal of result trustworthiness rather than a false sense of precision.
- **Human in the loop:** AutoDONE surfaces ranked candidate designs for a human to review and choose before committing to detailed modeling and simulation. The user can also request simulations for a different combination or prompt the application to provide additional information or make corrections. This preserves engineering judgment where consequences are highest, while automating the labor-intensive steps around it.

The AutoDONE application is a work in progress and is therefore continuously improving. Hallucinations are a known risk in AI applications, including AutoDONE. However, with the right combination of prompts and guardrails, hallucinations can be significantly reduced. It is also important for users to find a balance between sufficient context and overtly prescriptive prompts, which may lead to overfitting. Apart from these fundamental changes, the next steps for AutoDONE include a review of the logic, parameters, and calculations for all the agents, inclusion of an additional independent verification agent in the workflow, and generalization of the application to handle more complex systems, including those with redundancies and customized components. AutoDONE will be validated using a real-world system and independently evaluated by nuclear facility design engineers. In summary, AutoDONE has the potential to accelerate the design process for nuclear energy use cases through semi-autonomous agentic design, thereby facilitating faster deployment and better operation of power generation facilities.

Acknowledgements

This work was supported through the Idaho National Laboratory (INL) Laboratory Directed Research & Development (LDRD) Program under U.S. Department of Energy (DOE) Idaho Operations Office Contract DE-AC07-05ID14517.

References

[1] P. Eash-Gates, M. M. Klemun, G. Kavlak, J. McNerney, J. Buongiorno and J. E. Trancik, "Sources of cost overrun in nuclear power plant construction call for a new approach to engineering design," *Joule,* vol. 4, no. 11, pp. 2348-2373, 2020.

[2] K. T. Yeo and J. Ning, "Managing uncertainty in major equipment procurement in engineering projects," *European Journal of Operational Research,* vol. 171, no. 1, pp. 123-134, 2006.

[3] C. Stryker and E. Kavlakoglu, "What is AI?," IBM, 2024. [Online]. Available: https://www.ibm.com/think/topics/artificial-intelligence. [Accessed June 2026].

[4] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM,* vol. 55, no. 10, pp. 78-87, 2012.

[5] K. Chowdhary, "Natural Language Processing," in *Fundamentals of Artificial Intelligence*, New Delhi, Springer, 2020.

[6] S. Feuerriegel, J. Hartmann, C. Janiesch and P. Zschech, "Generative AI," *Business & Information Systems Engineering,* vol. 66, no. 1, pp. 111-126, 2024.

[7] D. Caffagni, F. Cocchi, L. Barsellotti, N. Moratelli, S. Sarto, L. Baraldi, M. Cornia, and R. Cucchiara. "The revolution of multimodal large language models: A survey," *Findings of the association for computational linguistics: ACL 2024*, pp. 13590-13618, 2024.

[8] Y. K. Dwivedi, M. Y. I. Helal, I. A. Elgendy, R. Alahmad, P. Walton, A. Suh, V. Singh, and I. Jeon. “Agentic AI Systems: What It Is and Isn't.” *Global Business and Organizational Excellence,* 45, no. 3, pp. 253-263, 2026.

[9] R. Giachetti, “Digital Engineering,” in *Guide to the Systems Engineering Body of Knowledge (SEBoK)*, 2024.

[10] A. M. Madni and M. Sievers, “Model-based systems engineering: Motivation, current status and research opportunities,” *Systems Engineering,* vol. 21, pp. 172-190, 2018.

[11] H. Kaitlin and S. Alejandro, “Value and benefits of model-based systems engineering (MBSE): Evidence from the literature,” *Systems Engineering,* vol. 24, pp. 51-66, 2021.

[12] P. Duddu, “Boeing T-X Trainer Aircraft, USA,” *Airforce Technology*, August 8, 2024, Available: https://www.airforce-technology.com/projects/boeing-t-x-trainer-aircraft/. [Accessed: July 2026].

[13] Teradata, “Revolution on the Siemens Factory Floor,” *Forbes Insights,* 2019, Available: https://www.forbes.com/sites/insights-teradata/2019/07/08/revolution-on-the-siemens-factory-floor/. [Accessed May 2026].

[14] J. Hartley, F. Jolevski, V. Melo, and B. Moore, “The Labor Market Effects of Generative Artificial Intelligence,” December 18, 2024, Available: https://dx.doi.org/10.2139/ssrn.5136877. [Accessed May 2026].

[15] N. Conte, Ed. G. Bhutada, “Charted: Productivity Gains from Using AI,” Visual Capitalist, June 25, 2025, Available: https://www.visualcapitalist.com/charted-productivity-gains-from-using-ai/. [Accessed May 2026].

[16] R. Jyoti and D. Schubmehl, “The Business Opportunity of AI,” The Official Microsoft Blog, 2023. [Online]. Available: https://blogs.microsoft.com/wp-content/uploads/prod/2023/11/US51315823-IG-ADA.pdf

[17] SAS, “Modern manufacturing's triple play: Digital twins, analytics & IoT,” SAS Insights, Big Data, n.d. Available: https://www.sas.com/en_us/insights/articles/big-data/modern-manufacturing-s-triple-play-digital-twins-analytics-iot.html.

[18] Anthropic, “Anthropic Economic Index: AI’s impact on software development,” Apr. 28, 2025. [Online]. Available: https://www.anthropic.com/research/impact-software-development. [Accessed: June 2026]

[19] A. Ghose, A. B. Kahng, S. Kundu, and B. Pramanik. “Agentic AI for Physical Design R&D: Status and Prospects,” *Proceedings of the 2026 International Symposium on Physical Design*, pp. 133-141. 2026.

[20] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, “ReAct: Synergizing reasoning and acting in language models,” arXiv:2210.03629, 2022.

[21] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, “Toolformer: Language models can teach themselves to use tools,” arXiv:2302.04761, 2023.

[22] V. Trivedy, “The Anatomy of an Agent Harness,” *LangChain Blog*, Mar. 10, 2026. [Online]. Available: https://www.langchain.com/blog/the-anatomy-of-an-agent-harness. [Accessed: May 2026]

[23] H. Derouiche, Z. Brahmi, and H. Mazeni, “Agentic AI Frameworks: Architectures, Protocols, and Design Challenges,” *arXiv preprint* arXiv:2508.10146, Aug. 2025. [Online]. Available: https://arxiv.org/abs/2508.10146

[24] Model Context Protocol, “Specification,” *Model Context Protocol Documentation*, Nov. 25, 2025. [Online]. Available: https://modelcontextprotocol.io/specification/2025-11-25. [Accessed: May 2026].

[25] A2A Protocol Working Group, “Agent2Agent (A2A) Protocol Specification,” *Agent2Agent (A2A) Protocol Documentation*. [Online]. Available: https://a2a-protocol.org/latest/specification/. [Accessed: May 2026].

[26] LangChain, “LangGraph,” *LangChain Documentation*. [Online]. Available: https://www.langchain.com/langgraph. [Accessed: May 2026].

[27] Idaho National Laboratory, “EMRALD,” [Online]. Available: https://emrald.inl.gov/SitePages/Overview.aspx. [Accessed July 2026].

[28] Z. Ma, T. E. Wierman, and K. J. Kvarfordt. *Industry-Average Performance for Components and Initiating Events at US Commercial Nuclear Power Plants: 2020 Update*. No. INL/EXT-21-65055-Rev000. Idaho National Laboratory, Idaho Falls, ID (United States), 2021.