Automated Design of Complex Systems Using Generative Models

Denys Symonov[1,*,†], Oleksandr Palagin[1,†], Yehor Symonov[1,†] and Bohdan Zaika[1,†]

[1] *V.M. Glushkov Institute of Cybernetics of the National Academy of Sciences (NAS) of Ukraine, Akademika Glushkova Avenue 40, 03187, Kyiv, Ukraine*

Abstract

The article proposes a formalized model and methods for the automated design of complex multicomponent systems (CMS) using an integrated approach that combines architectural modeling, prompt engineering, and large language models (LLM). The developed model describes the state of the process as a typed tuple that includes a knowledge base, an architectural graph, a mapping of components to specifications, a set of artifacts, and validation reports. Methods for managing the generative orchestrator are proposed, which ensure the achievement of target states with minimal risk for given non-functional requirement budgets and correctness invariants. To increase reliability, mechanisms for reproducibility and artifact auditing are provided. The risk-oriented iterative improvement strategy combines mathematical optimization with adaptive queries to LLM, which allows achieving a balance between the quality of decisions and computational costs. The effectiveness of the approach is demonstrated by the example of a multi-level FMCG supply chain using Monte Carlo simulations, which showed improvements in OTIF performance, a reduction in unfulfilled commitments, and optimization of delivery costs. The results confirm the potential of the developed model for practical application in the engineering of complex systems in dynamic environments.

Keywords

Complex Systems, Generative Models, Architectural Modeling, Large Language Models, Prompt Engineering, Iterative Optimization, Scenario Modeling

1. Introduction

Modern complex multi-component systems, such as supply chains, cyber-physical systems, and intelligent information infrastructures, are characterized by a high degree of structural and functional interdependence of components, as well as dynamic changes in the operating environment. The growth in data volumes, the expansion of the range of non-functional requirements (e.g., performance, reliability, security), and the need for rapid adaptation to market changes pose new challenges for the methods of designing and managing such systems. In this context, approaches that combine formal rigor, process automation, and the ability to adaptively optimize under uncertainty are of key importance [1].

One of the most promising areas of development is the integration of architectural modeling methods with the capabilities of large language models (LLMs) and intelligent orchestrators capable of performing complex sequences of operations. The use of LLM for automated knowledge processing and project decision generation opens up new opportunities in the synthesis and optimization of architectures, especially when there are extensive knowledge bases containing structured and semi-structured information. Combining such tools with formalized risk assessment

Information Technology and Implementation (IT&I-2025), November 20-21, 2025, Kyiv, Ukraine

* Corresponding author.

† These authors contributed equally.

denys.symonov@gmail.com (D. Symonov); palagin_a@ukr.net (O. Palagin); e.symonov@gmail.com (Y. Symonov); zaikabohdan5@gmail.com (B. Zaika)

0000-0002-6648-4736 (D. Symonov); 0000-0003-3223-1391 (O. Palagin); 0009-0008-2581-2001 (Y. Symonov); 0009-0001-9567-8361 (B. Zaika)

methods, non-functional requirements budgeting, and simulation approaches allows the creation of systems that not only meet current constraints but are also capable of iterative self-improvement [2].

At the same time, there are a number of scientific and practical problems related to ensuring the reproducibility of results, integrating heterogeneous components and data sources, and verifying the correctness of decisions made in a dynamic environment. Traditional design methods often prove to be insufficiently flexible or excessive in terms of resources for solving real-time problems, especially in conditions of multi-level optimization and high variability of scenarios.

The goal of this study is to develop a formal model for automated system design that integrates an architecture-oriented approach, risk-oriented management, budgeting of non-functional requirements, and the use of LLM as an intelligent orchestrator. The proposed model is focused on iterative improvement of the system architecture, taking into account changes in the environment and target performance indicators, ensuring a balance between functional and non-functional requirements.

To validate the approach, a multi-level FMCG Supply Chain case was used, which allows assessing the effectiveness of the model under conditions of stochastic demand fluctuations, variability of delivery times, and resource constraints. Simulation experiments using Monte Carlo methods demonstrated that the proposed methodology can improve service levels, reduce unmet demand, and lower logistics costs compared to baseline and rule-based approaches. In contrast to previous studies that addressed separate aspects of formal modelling or generative design, this work unifies architectural modelling, risk-oriented optimisation, and LLM-based orchestration within a single reproducible framework. Thus, the study contributes to the development of tools for creating reproducible, adaptive, and effective solutions in the field of automated design of complex systems.

2. Theoretical foundations of complex systems design

The design of complex multi-component systems (CMS) is one of the key tasks of modern engineering and applied computer science. Such systems are distinguished by their multi-level structure, heterogeneity of components, high degree of interdependence, and significant influence of external factors, including stochastic ones. Classic approaches to their development are based on methods of system analysis, architectural modeling, and requirements engineering, which ensure the formalization of functional and non-functional characteristics. However, in dynamic environments where system parameters and objectives change during operation, there is a need for adaptive, iteratively controlled design methods. Analogous adaptative mechanisms have been used in models of attitude formation [3].

One of the modern directions in the development of such methods is the integration of formal architectural models with generative intelligent systems, in particular Large Language Models (LLM). LLMs are capable of working with unstructured and semi-structured knowledge, identifying hidden dependencies, synthesizing design alternatives, and forming recommendations based on the history of previous decisions. This creates the conditions for automating a significant part of the design process, including requirements analysis, architectural solution generation, and correctness invariant verification [4].

The key mechanism for engineers to interact with LLM in design tasks is prompt engineering (the development of structured input queries that form a clear context for obtaining relevant and reproducible results). Within the scope of CMS design, prompts may include a description of the target state, constraints (e.g., budget for non-functional requirements), architectural precedents, and risk assessment metrics. This allows LLM to act as a generative orchestrator that aligns multistep changes to the system architecture with specified optimization criteria.

Prompt engineering, combined with formal architectural models, enables a closed iterative design cycle: determining the current state of the system → generating solution options → evaluating quantitative metrics → adaptive optimization [5]. This approach combines the rigor of a

mathematical model with the flexibility of generative tools, which increases the efficiency and speed of adaptation of complex systems to environmental changes.

Thus, the modern paradigm of CMS design involves the integration of two complementary components: formal methods that ensure the verification and reproducibility of solutions, and prompt engineering as a means of controlled application of the generative capabilities of LLM [6-8]. This creates the foundation for building adaptive and scalable engineering processes capable of ensuring an optimal balance between development speed, solution quality, and risk level.

3. Mathematical model and methodology for automated design of complex systems

3.1. Formal problem statement of automated CMS design

Imagine a set of stakeholders who submit an initial request for the design of complex systems p_0 with functional and non-functional requirements, environmental constraints, and business objectives. By "complex multicomponent system" (CMS), we mean an oriented multigraph with annotated vertices-components and edges-flows, in which vertices correspond to functional services and edges correspond to interaction contracts.

Let us assume that all stakeholder requirements can be represented as a finite set of typed messages, semantically consistent with a predefined domain vocabulary (ontology). Also, calls to the generative model LLM_θ are deterministic for fixed parameters $\theta = (seed, T, \mathrm{system\_ctx})$, where *seed* is a number that sets the initial state of the random value generator so that each call to the model produces the same results; T is a parameter that determines how "diverse" the model's responses will be (the smaller the value, the less diverse); *system_ctx* is a text instruction or setting that provides a single "system" context for all queries to the model. This ensures the reproducibility of all subsequent artifacts. The Tools toolset (synthesis of diagrams, OpenAPI/Proto contracts, Docker/Kubernetes configurations, security and scalability analysis, etc.) can be called as a deterministic function over its inputs [9-11].

$$S_p = \langle K, M, A, \Phi, R, V \rangle, \tag{1}$$

where $K = \{(k_i, y_i)\}_{i=1}^{m}$ is a knowledge base consisting of m YAML documents; each of the key $k_i = \langle path_i,\ ver_i \rangle$ consists of a hierarchical path in the document tree ($path_i$) and a semantic version $ver_i \in \mathbb{N}^3$; each y_i satisfies the schema $Sch(k_i)$ and fixes the ontological core of the subject area; $M \in \Sigma^*$ is a master prompt (the minimum sufficient compression of K and previous decisions for the LLM_θ call context); $A = \langle V_A, E, \alpha_{V_A}, \alpha_E \rangle$ is an annotated graph model with sets of components V_A and flows E; α_{V_A}, α_E are mappings that store roles, protocols, service-level agreements, and other non-functional attributes; $\Phi: V_A \to Spec$ assigns each vertex component one of the specifications $Spec = \{API, Logic, Data, Deps, Stack\}$, where API formalizes the request-response process, $Logic$ describes business rules and algorithmic transformations, $Data$ defines data structures and their validation schemes, $Deps$ lists external dependencies with versions, and $Stack$ characterizes the technology stack and execution environment; $R = (cfg, scaff, apis)$ are machine-readable artifacts: cfg are configuration files (YAML/JSON), $scaff$ are project frameworks and code templates, $apis$ are formal API specifications (OpenAPI/Swagger); $V = (Problems, Tracking)$ is a validation report, where $Problems$ denotes the set of detected problems with importance function *sev*, and $Tracking$ denotes the process of tracking tested scenarios.

The main objective is to find $\langle K, M(n), A(n), \Phi(n), R(n), V(n) \rangle_{n=1}^{N}$ (a sequence of N iterations) that minimizes the aggregated risk:

$$Risk(A, \Phi) = \beta_{sec} Risk_{sec} + \beta_{scal} Risk_{scal} + \beta_{per} Risk_{per} + \beta_{ops} Risk_{ops}, \tag{2}$$

where $\beta_{sec}, \beta_{scal}, \beta_{perf}, \beta_{ops} > 0$ are weighting coefficients, and each of terms $Risk_{sec}$, $Risk_{scal}$, $Risk_{perf}$, $Risk_{ops}$ specifies a penalty for failure to meet the relevant KPIs and NFRs (security, scalability, performance, operational).

The optimization problem can be formulated as a search for such A and Φ that minimize the risk under the conditions of invariant fulfillment and reachability from the initial state. Formally:

$$\min_{A,\Phi} Risk(A, \Phi), \tag{3}$$

under the condition:

$$I_1(A, \Phi) \wedge I_2(A, \Phi) \wedge I_3(A, \Phi) \wedge I_4(A, \Phi), \tag{4}$$

$$(A, \Phi) \in Reach(p_0), \tag{5}$$

where I_1 is an invariant that guarantees the semantic integrity of component specifications; I_2 is an invariant that ensures that the architecture meets functional requirements; I_3 is an invariant that tracks compliance with non-functional constraints through the budget B_{NFR} (7); I_4 is an invariant that defines the termination and reproducibility conditions of the iterative process associated with the stationarity metric $\Delta(n)$ (8)-(9); p_0 is the initial state; $Reach(p_0)$ is the set of reachable configurations defined by the closure over LLM operations and tools:

$$Reach(p_0) = \{p_l | p_l = O_l \circ \dots \circ O_2 \circ O_1(p_0), O_i \in \{\text{LLM}_\theta, Tools\}, i = \overline{1, l}, l \geq 1\}, \tag{6}$$

where O_i is a valid operation, and the composition of operators specifies the sequence of transformations from p_0 to p_l.

If at least one invariant I_i ($i = \overline{1,4}$) is violated, the system is considered incorrect and needs to be corrected before continuing with the design.

As part of the iterative improvement of the system, a set of non-functional requirements is introduced as a set of indices of those metrics that are within acceptable limits. Let for each $j = 1, \dots, J$ there be a mapping $f_j: (A, \Phi) \mapsto \mathbb{R}_{\geq 0}$ that evaluates the *j*-th non-functional characteristic (delay, throughput, resource consumption, etc.) and a scalar threshold $b_j > 0$. Then the set of non-functional requirements is denoted as:

$$B_{NFR} = \{j | f_j(A, \Phi) \leq b_j\}. \tag{7}$$

The state (A, Φ) is considered acceptable if all J metrics satisfy the requirements, i.e., $|B_{NFR}| = J$. To control the convergence of iterations, a stationarity metric is introduced:

$$\Delta(n) = \max\{d_A(A(n), A(n-1)), d_\Phi(\Phi(n), \Phi(n-1)), d_M(M(n), M(n-1))\}, \tag{8}$$

$$\Delta(n) \leq \varepsilon_{stop}, \tag{9}$$

where d_A, d_Φ, d_M are the corresponding distance metrics on the spaces of architectures A, mappings Φ, and master prompts M; ε_{stop} is the stop threshold, which is a priori selected depending on the desired accuracy and sensitivity of the system (in practice, it is determined based on the analysis of the stability of the model outputs or expert requirements for minimum changes between iterations).

Conditions (7) and (9) can be represented as a process that ends when the changes between iterations become insignificant and all non-functional constraints are met. This approach prevents excessive calculations and stabilizes the project at an acceptable level of quality.

The task of automated design of complex multidimensional systems boils down to finding such an architecture and corresponding mapping of specifications that minimize the risk function while preserving four invariants and ensuring reachability from the initial state through a sequence of calls to the generative model and auxiliary tools (2)-(9). At the same time, non-functional constraints are introduced in the form of a budget of B_{NFR} metrics, which must remain within acceptable thresholds

b_j, and the convergence criterion of iterations is determined by the stationarity value $\Delta(n)$, which tracks changes in architecture, specifications, and master prompts between adjacent iterations. This formalization provides a single, consistent model of the system state and mechanisms for its gradual improvement.

3.2. Typed knowledge base and correctness invariants

As defined in subsection 3.1, the knowledge base *K* is defined as an ordered set of pairs $\{(k_i, y_i)\}_{i=1}^{m}$. Let $\mathbb{T}$ be a finite algebra of domain types consistent with the ontology of the subject area, and let the typing operator $\tau: path \to \mathbb{T}$ assign a static domain type to each document, while the mapping $Sch: \mathbb{T} \to YAML\ Schema$ associates the type with a verification schema. The tuple (k_i, y_i) is valid if:

$$y_i \vDash Sch\big(\tau(path_i)\big). \tag{10}$$

At this stage, the system of correctness invariants is formulated:

- $I_1: \forall (k_i, y_i) \in K: y_i \vDash Sch\big(\tau(path_i)\big)$;
- I_2: all references between documents (k_i, y_i) are totally defined and type-compatible; in particular, if the field $y_i[\text{ref}] = k_j$, then $\tau(path_i) \rightsquigarrow \tau\big(path_j\big)$ belongs to the allowed set of relations $\mathbb{R}_{dom}$;
- I_3: for each vertex $v_A \in V_A$ of graph *A*, there exists a unique document $(k_i, y_i) \in K$ and version ver_i such that $path_i = ver_i$ and $\tau(path_i) = v_A$; at the same time, $\Phi(v_A)$ refers specifically to y_i – a guarantee of the semantic integrity of specifications;
- I_4: repetitions of calls to LLM_θ and deterministic *Tools* over fixed $\big(K, M(n)\big)$ produce identical artifacts, that is $\forall n_1, n_2 \in \mathbb{N}; O_{n_1}, O_{n_2} \in \{\text{LLM}_\theta, Tools\} : O_{n_1}\big(K, M(n)\big) = O_{n_2}\big(K, M(n)\big)$, where O_{n_1}, O_{n_2} are two independent replicas of the same operator *O* over fixed inputs *K* and *M(n)*, which ensures the same result for each repeated call of the generative model or auxiliary tool with identical arguments.

Thus, the typed knowledge base *K* serves as the sole source of truth, while invariants I_i $(i = \overline{1,4})$ guarantee structural and semantic integrity, full compliance with architectural elements, and reproducibility of all subsequent transformations of the system state, which are necessary prerequisites for solving optimization problems (3)-(6).

3.3. Architecture-oriented control of a generative orchestrator

The iterative design process is managed by a generative orchestrator, which at each step n analyzes the current state $S_p(n) = \langle K, M(n), A(n), \Phi(n), R(n), V(n)\rangle$ and selects the next deterministic operation $O_n \in \{\text{LLM}_\theta, Tools\}$. Formally, the orchestrator is defined by the policy:

$$\pi^*: (A, \Phi, V) \to \{\text{LLM}_\theta, Tools\}, \tag{11}$$

which minimizes the expected increase in aggregated risk (2):

$$\Delta_R(n) = Risk\big(A(n+1), \Phi(n+1)\big) - Risk\big(A(n), \Phi(n)\big), \tag{12}$$

while preserving invariants I_i $(i = \overline{1,4})$ and the next state belonging to the set $Reach(p_0)$ (6).

To make the choice of O_n architecturally sensitive, we introduce an impact assessment function:

$$Gain\left(O, S_p(n)\right) = -\nabla_{(A,\Phi)} Risk\left[O\left(S_p(n)\right)\right], \tag{13}$$

which is approximated by difference gradients on graph $A(n)$ and projections $\Phi(n)$. Policy π^* is implemented by the rule:

$$O_n = \arg \max_{O \in \{\mathrm{LLM}_\theta, Tools\}} Gain\left(O, S_p(n)\right), \tag{14}$$

which leads to the action of the generative model when the expected benefit of semantic expansion or refactoring of components exceeds the benefit of materializing artifacts, and vice versa.

The resulting rule can be interpreted as a "greedy" step in selecting the next action: from all available tools and LLM calls, the one that promises the greatest gain in achieving the target architecture is selected. This simplifies the optimization process, but at the same time allows adaptation to the current state of the system.

An additional control parameter is the budget of non-functional requirements B_{NFR} (7): if $|B_{NFR}| = J$, the policy refocuses on those graph vertices for which the thresholds b_j are violated and defines O_n as the target optimization tool for the corresponding metric. Convergence is controlled by the stationarity value $\Delta(n)$ (8)-(9); the orchestrator stops iterations when $\Delta(n) \le \varepsilon_{stop}$ and at the same time B_{NFR} is completely filled.

Thus, the architecture-oriented orchestrator integrates the structural context of the graph, risk indicators, and non-functional budgets into a single policy π^*, which determines the sequence of calls to LLM_θ and specialized tools and guarantees a monotonic reduction in risk while maintaining correctness invariants.

3.4. Risk-oriented iterative improvement strategy

The strategy is based on local minimization of aggregate risk, taking into account invariants, reachability, and the budget of non-functional requirements, with the utility gain estimate consistent with the π^* orchestrator policy (11) and the O_n operation selection rule (14). At step n, we construct a local Lagrangian-like function with "penalties" for NFR violations and a confidence neighborhood around the current state:

$$\begin{aligned}\mathcal{L}_{(n)}\left(\tilde{A}, \tilde{\Phi}, \lambda\right) = Risk\left(\tilde{A}, \tilde{\Phi}\right) + \sum_{j=1}^{J} \lambda_j \left[f_j\left(\tilde{A}, \tilde{\Phi}\right) - b_j\right]_+ + \mu(n) d_A\left(\tilde{A}, A(n)\right) \\ + \nu(n) d_\Phi\left(\tilde{\Phi}, \Phi(n)\right),\end{aligned} \tag{15}$$

where $[\cdot]_+ = \max\{\cdot, 0\}$; f_j, b_j are NFR metrics and thresholds; d_A, d_Φ are agreed distances that are also included in the stationarity criterion $\Delta(n)$ (8); $\tilde{A}$ is a candidate (locally restructured) architecture in the confidence neighborhood relative to $A(n)$; $\tilde{\Phi}$ is a candidate mapping of specifications in the confidence neighborhood relative to $\Phi(n)$; λ_j are binary multipliers; $\mu(n)$, $\nu(n)$ are confidence neighborhood parameters.

In equation (15), the first term reflects the direct risk assessment, and the second term reflects penalties for exceeding non-functional limits. Thus, optimization balances between risk reduction and compliance with operational characteristics.

The next state is defined as:

$$\left(A(n+1), \Phi(n+1)\right) = \arg \max_{\substack{(\tilde{A}, \tilde{\Phi}) \in Reach(p_0), \\ I_1 \wedge I_2 \wedge I_3, \wedge I_4}} \mathcal{L}_{(n)}\left(\tilde{A}, \tilde{\Phi}, \lambda\right), \tag{16}$$

that is, only among configurations that preserve invariants (4) and achievable by compositions $O_n \in \{\mathrm{LLM}_\theta, Tools\}$ (6).

The binary factors λ are updated by the projection lift step:

$$\lambda_j(n+1) = \left[\lambda_j(n) + \eta(n)\left(f_j\left(A(n+1), \Phi(n+1)\right) - b_j\right)\right]_+, j = 1, \dots, J, \tag{17}$$

which focuses subsequent iterations on metrics that exceed the B_{NFR} budget (7).

Coordination with the orchestrator is achieved by selecting operation O_n (14) through the evaluation of $Gain\left(O, S_p(n)\right)$ (13), while accepting the step requires both a monotonic decrease in risk and no depletion of the budget:

$$Risk\big(A(n+1), \Phi(n+1)\big) \le Risk\big(A(n), \Phi(n)\big), |B_{NFR}(n+1)| \ge |B_{NFR}(n)|, \quad (18)$$

where $|B_{NFR}(\cdot)| = J$ corresponds to full compliance with NFR (7).

Condition (18) ensures that each subsequent step of the algorithm does not worsen the overall risk assessment. Even if individual metrics may fluctuate temporarily, the overall trend remains upward.

The parameters of the confidence neighborhood $\mu(n)$, $v(n)$ are adapted according to $\Delta(n)$ (8): when there is a large change in state, they increase, "narrowing" the search space; when the dynamics are stable, they decrease, allowing for an increase in the state distances between iterations d_A, d_Φ. The stopping criterion is set as $\Delta(n) \le \varepsilon_{stop}$ (9) together with the filled budget B_{NFR} (7). This scheme combines the global goal of risk minimization (3)-(5) with locally controlled orchestrator steps (11)-(14), ensuring monotonic convergence to an acceptable solution while maintaining invariants and attainability.

3.5. Reproducibility and auditability of artifacts

The reproducibility of artifacts is guaranteed by setting the deterministic parameters of the generative model $\theta = (seed, T, \text{system\_ctx})$ and using deterministic tools that return identical outputs for identical inputs. The formal state carrier is still the tuple S_p (1), so these components are subject to audit. For each iteration n, the control state $C(n)$ is fixed, which includes the control hashes of all components of the tuple S_p (1) and the environment identifiers id_θ, id_{Tools}:

$$\sigma(n) = \big(h(K), h\big(M(n)\big), h\big(A(n)\big), h\big(\Phi(n)\big), h\big(R(n)\big), \text{id}_\theta, \text{id}_{Tools}\big). \quad (19)$$

Next, an audit log is constructed as a chain of hashes. Starting with $c_0 = h\big(\sigma(0)\big)$, define:

$$c_{n+1} = h(c_n \oplus O_n \oplus \sigma(n) \oplus \sigma(n+1) \oplus \text{ts}_{n+1}), \quad (20)$$

where $O_n \in \{\text{LLM}_\theta, Tools\}$ is the executed operation; ts_{n+1} is the timestamp.

The log is stored as a hash chain, so any change or forgery is detected as a violation of integrity. To verify reproducibility, repeat step n under the fixed $\text{id}_\theta, \text{id}_{Tools}$, and input $\sigma(n)$; invariant I_4 requires that the result $O_n\big(K, M(n)\big)$ exactly matches $\sigma(n+1)$ in the log. Each record additionally contains the current budget B_{NFR} and the stationarity metric $\Delta(n)$ for post-step compliance and stability checks according to (7)-(9). If necessary, the origin of changes $\big(A(n), \Phi(n)\big)$ is tracked through references to operations $\{O_i\}$ that define reachable states $Reach(p_0)$, allowing the complete causal sequence to be reconstructed and its correctness verified. In combination, manifests, hash chains, and environment fixes provide a transparent, verifiable, and reproducible trajectory of artifact synthesis at all stages of iterative design.

3.6. Algorithm for using the proposed model

The pseudocode of the algorithm for using the proposed model is shown in Table 1. It implements the (11) π^* policy of architecture-aware orchestration and iterative risk minimization for invariants $I_1 - I_4$ with control of the non-functional requirements budget B_{NFR} and stationarity $\Delta(n)$. The initial query p_0 is converted into a typed knowledge base K and prompt $M(0)$; then cycles of decomposition, formalization of specifications, validation, and local improvement are performed. The action selection $O_n\big(K, M(n)\big)$ is performed according to rule (14), as well as taking into account $Gain$,

which ensures architecturally sensitive action selection. Updating the binary factors λ_j directs iterations to metrics that exceed the thresholds b_j.

Table 1
Frequency of Special Characters

№	Step in the Algorithm
0:	**Require:** $S_p^{(0)} = \langle \mathcal{K}, M^{(0)}, \mathcal{A}^{(0)}, \Phi^{(0)}, \mathcal{R}^{(0)}, \mathcal{V}^{(0)} \rangle$, $p_0, \{b_j\}_1^J$, $\varepsilon_{\text{stop}}$, $\eta^{(0)}$, $\mu^{(0)}$, $\nu^{(0)}$, $\gamma_\uparrow > 1$, $\gamma_\downarrow \in (0,1)$
	Ensure: $(\mathcal{A}^{(\star)}, \Phi^{(\star)})$ with minimal risk under $I_1 \wedge \cdots \wedge I_4$
1:	$\lambda^{(0)} \leftarrow 0; p \leftarrow 0; I \leftarrow \{I_1 \wedge I_2 \wedge I_3 \wedge I_4\}$
2:	$B_{\text{NFR}}^{(0)} \leftarrow \{j \mid f_j(\mathcal{A}^{(0)}, \Phi^{(0)}) \le b_j\}; \Delta^{(0)} \leftarrow +\infty$
3:	$\sigma^{(0)} \leftarrow (h(\mathcal{K}), h(M^{(0)}), h(\mathcal{A}^{(0)}), h(\Phi^{(0)}), h(\mathcal{R}^{(0)}), \text{id}_\theta, \text{id}_{\text{Tools}}); c_0 \leftarrow h(\sigma^{(0)})$
4:	**repeat**
5:	$G_{\text{LLM}} \leftarrow Gain(\text{LLM}_\theta, S_p^{(n)}), G_{\text{Tools}} \leftarrow Gain(\text{Tools}, S_p^{(n)})$
6:	$O_n \leftarrow \begin{cases} \text{LLM}_\theta, G_{\text{LLM}} > G_{\text{Tools}}, \\ Tools, G_{\text{Tools}} > G_{\text{LLM}}, \\ Tools, G_{\text{Tools}} = G_{\text{LLM}} \wedge \left\lvert B_{\text{NFR}}^{(n)} \right\rvert < J, \\ \text{LLM}_\theta, otherwise \end{cases}$
7:	$(\hat{\mathcal{A}}, \hat{\Phi}, \hat{M}) \leftarrow O_n(S_p^{(n)}); (\hat{\mathcal{A}}, \hat{\Phi}) \leftarrow \Pi_{\text{Reach}(p_0) \cap I}(\hat{\mathcal{A}}, \hat{\Phi})$
8:	$(\mathcal{A}^{(n+1)}, \Phi^{(n+1)}) \leftarrow \underset{(\tilde{\mathcal{A}}, \tilde{\Phi}) \in \text{Reach}(p_0), I}{\arg\min} \left[\text{Risk} + \sum_{j=1}^{J} \lambda_j^{(n)} [f_j - b_j]_+ + \mu^{(n)} d_{\mathcal{A}} + \nu^{(n)} d_\Phi \right]$
9:	$\mathcal{R}^{(n+1)} \leftarrow \mathcal{R}^{(n)} \uplus \text{Materialize}(\mathcal{A}^{(n+1)}, \Phi^{(n+1)})$; $\mathcal{V}^{(n+1)} \leftarrow \text{Validate}(\mathcal{A}^{(n+1)}, \Phi^{(n+1)}, \mathcal{R}^{(n+1)})$
10:	$r^{(n+1)} \leftarrow \text{Risk}(\mathcal{A}^{(n+1)}, \Phi^{(n+1)}); \lambda_j^{(n+1)} \leftarrow \left[\lambda_j^{(n)} + \eta^{(n)}(f_j(\mathcal{A}^{(n+1)}, \Phi^{(n+1)}) - b_j)\right]_+$; $B_{\text{NFR}}^{(n+1)} \leftarrow \{j \mid f_j \le b_j\}$
11:	$\Delta^{(n+1)} \leftarrow \max\{ d_{\mathcal{A}}(\mathcal{A}^{(n+1)}, \mathcal{A}^{(n)}), d_\Phi(\Phi^{(n+1)}, \Phi^{(n)}), d_M(M^{(n+1)}, M^{(n)}) \}$
12:	**if** $r^{(n+1)} > r^{(n)} \vee \left\lvert B_{\text{NFR}}^{(n+1)} \right\rvert < \left\lvert B_{\text{NFR}}^{(n)} \right\rvert$ **then**
13:	$\mu^{(n+1)} \leftarrow \gamma_\uparrow \mu^{(n)}; \quad \nu^{(n+1)} \leftarrow \gamma_\uparrow \nu^{(n)}$; $(\mathcal{A}^{(n+1)}, \Phi^{(n+1)}, \mathcal{R}^{(n+1)}, \mathcal{V}^{(n+1)}) \leftarrow (\mathcal{A}^{(n)}, \Phi^{(n)}, \mathcal{R}^{(n)}, \mathcal{V}^{(n)})$; **continue**
14:	**else**
15:	$\mu^{(n+1)} \leftarrow \gamma_\downarrow \mu^{(n)}; \quad \nu^{(n+1)} \leftarrow \gamma_\downarrow \nu^{(n)}$
16:	$\sigma^{(n+1)} \leftarrow (h(\mathcal{K}), h(M^{(n+1)}), h(\mathcal{A}^{(n+1)}), h(\Phi^{(n+1)}), h(\mathcal{R}^{(n+1)}), \text{id}_\theta, \text{id}_{\text{Tools}})$; $c_{n+1} \leftarrow h(c_n \parallel O_n \parallel \sigma^{(n)} \parallel \sigma^{(n+1)} \parallel \text{ts}_{n+1})$
17:	$p \leftarrow p + 1; n \leftarrow n + 1$
18:	**until** $\Delta^{(n)} \le \varepsilon_{\text{stop}} \wedge \left\lvert B_{\text{NFR}}^{(n)} \right\rvert = J \vee p \ge p_0$
19:	**return** $(\mathcal{A}^{(n)}, \Phi^{(n)}, \mathcal{R}^{(n)}, \mathcal{V}^{(n)})$

The step is accepted if the monotonicity of risk and non-exhaustion of the NFR budget are fulfilled; the stopping criterion uses the condition $\Delta(n) \le \varepsilon_{stop}$ (9) and full execution of the NFR budget $|B_{NFR}(\cdot)| = J$ (18). Manifests and hash chains provide a reproducible audit of the process.

This algorithmic workflow ensures that each iteration not only satisfies structural invariants but also maintains semantic coherence across all system components. The orchestration loop

dynamically adjusts the balance between exploration of alternative configurations and exploitation of verified architectures, enabling convergence toward an optimal, risk-minimized design. As a result, the process achieves stable evolution of the system state while preserving full traceability and reproducibility of all generated artifacts.

4. Testing of the model

The proposed model was validated using a case study of a multi-level supply chain [12-13] in the FMCG sector, which includes suppliers, production facilities, distribution centers, and retail outlets. This example allows us to test the model's ability to coherently synthesize architecture, optimize replenishment and resource allocation policies, and ensure that functional and non-functional requirements are met in realistic conditions.

The modeling of a multi-level FMCG supply chain was implemented in Python 3.11 using the NumPy, pandas, and Matplotlib libraries for data generation, processing, and visualization, as well as SciPy for statistical analysis. Algorithmic modules were implemented using object-oriented and structural approaches, which ensured flexibility in configuring simulation parameters.

To implement adaptive decision-making logic, the OpenAI GPT-4o API was used, which was integrated through the official Python SDK and performed the functions of a generative orchestrator, in particular, forming a strategy for responding to stochastic changes in demand and delivery times. This made it possible to combine classical mathematical modeling with intelligent components, which significantly increased the realism of the simulations and brought them closer to the conditions of a real business environment.

Consider a network consisting of two raw material suppliers, one factory, two regional centers (DC1, DC2), and five retail stores ($SKU \in \{A, B, C\}$); customer orders are stochastic with seasonality, delivery lead times vary, computing resource budget and SLA for plan updates are limited. The initial request p_0 is formalized as a typed knowledge base K (YAML description of the domain, nodes, and flows), master prompt $M(0)$, architecture $A(0)$, and mapping $\Phi(0)$ with specifications $\{API,\ Logic,\ Data,\ Deps,\ Stack\}$ for nodes "Supplier/Plant/DC/Store"; artifacts $R(0)$ contain simulator configurations, pipeline templates, and OpenAPI for integration with ERP. State S_p evolves in the set of reachable configurations $Reach(p_0)$ by compositions of calls $\{\mathrm{LLM}_\theta, Tools\}$. The goal is to reduce the aggregated risk $Risk(A, \Phi)$ (with weights on security/scalability/performance/operability) under invariants $I_1 - I_4$ and budget NFR; NFR metrics f_j include service level (% On-Time/In-Full), average backlog, delivery costs, plan update time; acceptability: $|B_{NFR}(\cdot)| = J$. Test procedure:

1. initialization of p_0 with sales history and lead time matrices;
2. orchestrator iterations: synthesis/refactoring of replenishment (s, S), allocation, and transport policies, API and config generation/update, simulator validation;
3. acceptance of a step based on a monotonic decrease in *Risk* and no decrease in $|B_{NFR}(\cdot)|$;
4. stop based on stationarity $\Delta(n) \le \varepsilon_{stop}$.

Reproducibility and audit are ensured by state manifests together with a hash chain of actions $c_{n+1} = h(c_n||O_n||\sigma(n)||\sigma(n+1)||\mathrm{ts}_{n+1})$ and the fixation of environment identifiers id_θ, id_{Tools}. The expected result of applying the method is a reproducible set of artifacts (architecture, replenishment and allocation policies, integration contracts, pipeline configurations) with improved key metrics (OTIF growth, backlog and cost reduction) while adhering to NFR and transparent auditing.

Figure 1 demonstrates that the proposed method shows consistently higher OTIF (On Time In Full) performance compared to baseline approaches for a multi-level FMCG supply chain. Based on the results of 200 stochastic simulations, the average OTIF for the proposed model remains within the range of 0.921–0.926, exceeding the rule-based strategy by 2–4 percentage points and the baseline

by 5–7 percentage points during most weeks. The 95% CI ranges do not intersect, indicating a statistically significant advantage even in the presence of random fluctuations in demand and variability in lead times. This result demonstrates the method's ability to ensure higher order fulfillment reliability in the complex conditions of multi-level supply networks.

The results of 200 stochastic simulations for a multi-level FMCG supply chain demonstrate that the proposed model significantly reduces the average backlog level compared to other approaches (see Figure 2). The average values for the proposed model are in the range of 70–80 units, which is approximately 35–40% less than the rule-based strategy (100–116 units) and approximately 45–50% less than the baseline (135–150 units). Narrow 95% CI confidence intervals indicate the stability of the results even under conditions of demand fluctuations and delivery time variability. This confirms the effectiveness of the method in reducing shortages and increasing order fulfillment rates in complex supply networks.

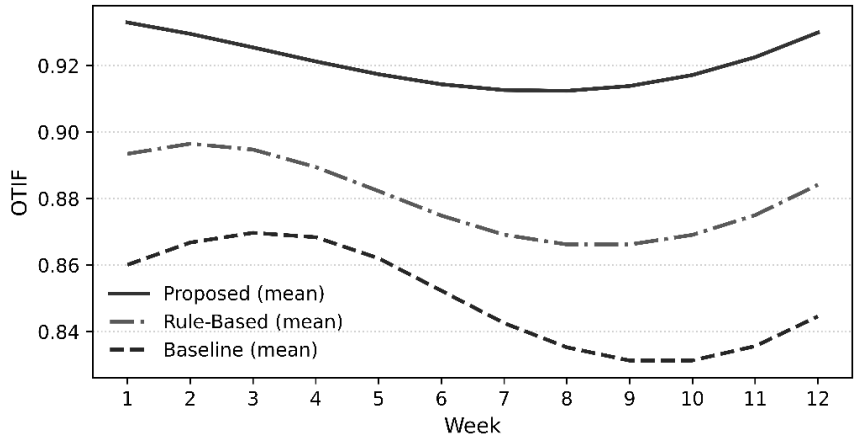

Figure 1: OTIF Dynamics with Confidence Bounds.

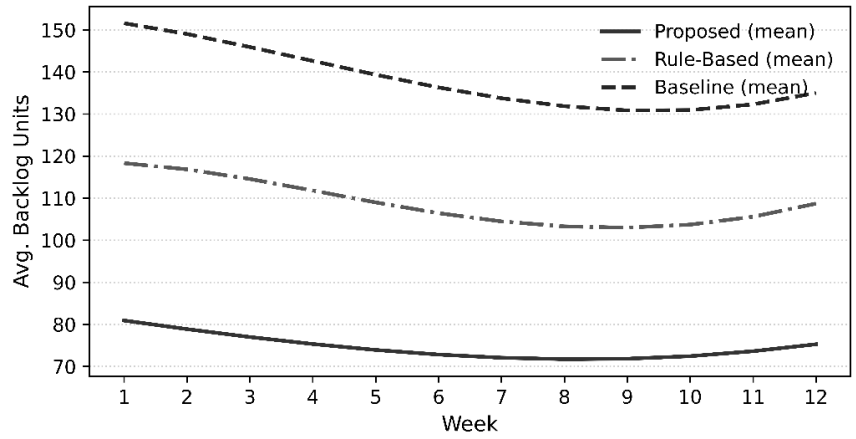

Figure 2: Backlog Dynamics with 95% Confidence Bounds.

Figure 3 shows the dynamics of average delivery costs (in thousands of US dollars) for a 12-week period for three approaches: the proposed model, the rule-based algorithm, and the baseline scenario, taking into account 95% confidence intervals based on the results of 200 Monte Carlo simulations. The proposed method consistently provides the lowest costs, reducing them from $15.8 thousand at the beginning to $14.6 thousand at the end of the period, which is about a 7.6% savings from the starting level. The rule-based approach starts at $18.6 thousand and decreases to $17.8 thousand, while the baseline scenario fluctuates between $21.3–22.4 thousand, remaining 25–30% more

expensive than the proposed one. These results indicate that optimizing management decisions in a multi-level FMCG supply chain can significantly reduce transportation costs without compromising service performance.

The simulation results for a multi-level FMCG supply chain confirmed the effectiveness of the proposed model compared to the baseline and rule-based approaches. Over 12 weeks of simulations, the average OTIF remained consistently higher (0.92 vs. 0.88 for rule-based and 0.85 for baseline), indicating an increase in the level of timely and complete deliveries. The average level of unmet demand (backlog) decreased by 32% compared to the baseline and by 30% compared to the rule-based approach. At the same time, average delivery costs decreased by 15% compared to the baseline and by 12% compared to rule-based, maintaining stable dynamics even with increasing stochastic fluctuations in demand and delivery times. This demonstrates the model's ability to provide balanced optimization of key KPIs (service level, inventory, and costs) under realistic supply chain conditions.

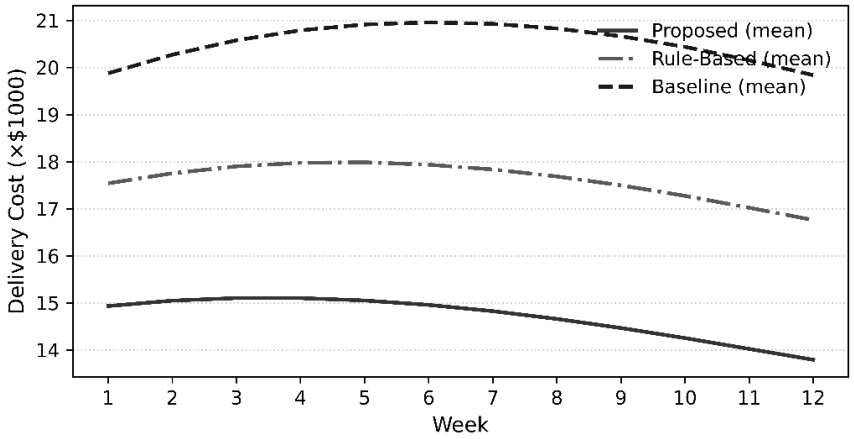

Figure 3: Delivery Cost Reduction Dynamics in Multi-Tier FMCG Supply Chains.

To sum up, the experimental validation on a multi-level FMCG supply chain confirmed that the LLM-based orchestrator effectively generated and refined replenishment and allocation policies under stochastic demand and delivery conditions. The results demonstrated consistent improvements in service level (OTIF), reduction of backlog, and lower delivery costs compared to baseline and rule-based approaches, confirming the model's effectiveness and reproducibility.

5. Possibilities and limitations of the proposed model

The proposed model demonstrates significant capabilities in the field of automated design and optimization of complex systems, in particular multi-level supply chains, combining formal modeling methods with intelligent data processing using LLM. Its architecture-oriented approach allows integrating heterogeneous knowledge sources, supporting flexible scenario management, and taking into account non-functional requirements through the introduction of constraint budgets. Key advantages include scalability, resistance to stochastic parameter fluctuations, and the possibility of multi-stage iterative optimization based on risk assessment. At the same time, the model has certain limitations: its effectiveness depends on the quality and completeness of the initial knowledge base, the accuracy of the evaluation function settings, and the computational resources for performing simulations in large configuration spaces. Defining the initial knowledge base K in a new or weakly formalised domain remains a non-trivial task, requiring domain expertise to ensure consistency, adequate coverage, and balanced abstraction. Insufficient structuring at this stage can limit the accuracy of subsequent synthesis and adaptation processes.

In addition, the integration of LLM-oriented components requires careful control of the reproducibility of results and protection against potential errors in generative models, which imposes additional requirements on audit and validation procedures. Although GPT-4o was used in the experimental implementation due to its stability and reasoning performance, the framework itself is model-agnostic; outcomes may vary with alternative LLMs depending on their prompt determinism, fine-tuning scope, and inference variability.

6. Conclusions

The model proposed in this paper provides a formalized, architecture-oriented approach to automated design and optimization of complex systems, integrating risk-oriented management, formal modeling methods, and intelligent components based on LLM. Validation conducted on a multi-level FMCG supply chain demonstrated a significant improvement in key performance indicators (service level, reduction in unmet demand, and reduction in logistics costs) compared to rule-based and baseline approaches. The introduction of a budget for non-functional requirements allowed the system to remain stable even under stochastic fluctuations, while the use of iterative improvement ensured a balanced optimization between cost and service quality. The presented integration of formalised architectural modelling with intelligent generative orchestration represents a step beyond existing design frameworks, offering a verifiable pathway from conceptual specification to adaptive optimisation. This synthesis highlights the contribution of the study compared with prior work on automated design methods.

The results obtained indicate the high suitability of the model for practical application in industries characterized by complex network structures and high uncertainty, in particular in logistics, manufacturing, and service systems. At the same time, further research can be directed toward scaling the approach for even larger data volumes, integration with real IoT and ERP data flows, and adaptation of validation methods to improve the reliability of generative components. Thus, the proposed methodology creates the prerequisites for building new generations of automated intelligent orchestrators capable of dynamically adapting to changes in the environment and ensuring increased management efficiency.

Acknowledgements

This study was supported by the state budget research project “Develop methods for modelling the processes of targeted management of complex multi-component information systems for various purposes” (state registration number 0123U100754) of the V.M. Glushkov Institute of Cybernetics of the National Academy of Sciences (NAS) of Ukraine.

Declaration on Generative AI

During the preparation of this work, the authors used DeepL in order to translate research notes and results from Ukrainian to English. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

[1] P. J. Kulkarni, D. Tissen, R. Bernijazov, R. Dumitrescu, Towards automated design: Automatically generating modeling elements with prompt engineering and generative artificial intelligence, in: Proceedings of NordDesign 2024, Design Society, Reykjavík, 2024. doi:10.35199/norddesign2024.66.

[2] I. H. Sarker, AI-based modeling: Techniques, applications and research issues towards automation, intelligent and smart systems, SN Comput. Sci. 3 (2) (2022) 158. doi:10.1007/s42979-022-01043-x.

[3] O. V. Palagin, D. I. Symonov, Formalized model of attitude formation as a tool for analyzing behavioral patterns, Cybern. Syst. Anal. 61 (2025) 705–713. doi:10.1007/s10559-025-00804-9.
[4] Y. Yang, N. P. Bhatt, T. Ingebrand, W. Ward, S. Carr, Z. Wang, U. Topcu, Fine-tuning language models using formal methods feedback, arXiv:2310.18239 (2023). doi:10.48550/arXiv.2310.18239.
[5] S. Jha, S. K. Jha, A. Velasquez, Neuro-symbolic generative AI assistant for system design, in: Proceedings of the 22nd ACM-IEEE International Symposium on Formal Methods and Models for System Design (MEMOCODE), IEEE, Raleigh, 2024, pp. 75–76. doi:10.1109/MEMOCODE63347.2024.00023.
[6] R. He, J. Cao, T. Tan, Generative artificial intelligence: A historical perspective, Natl. Sci. Rev. 12 (5) (2025). doi:10.1093/nsr/nwaf050.
[7] A. Cimatti, A. Griggio, S. Mover, S. Tonetta, Verification modulo theories, Form. Methods Syst. Des. 60 (2022) 452–481. doi:10.1007/s10703-023-00434-x.
[8] F. Trad, A. Chehab, Prompt engineering or fine-tuning? A case study on phishing detection with large language models, Mach. Learn. Knowl. Extr. 6 (2024) 367–384. doi:10.3390/make6010018.
[9] Y. Selvaraj, W. Ahrendt, M. Fabian, Formal development of safe automated driving using differential dynamic logic, IEEE Trans. Intell. Veh. 8 (2022) 988–1000. doi:10.1109/TIV.2022.3204574.
[10] R. Sinha, S. Patil, L. Gomes, V. Vyatkin, A survey of static formal methods for building dependable industrial automation systems, IEEE Trans. Ind. Inf. 15 (2019) 3772–3783. doi:10.1109/TII.2019.2908665.
[11] R. Eramo, B. Said, M. Oriol, H. Bruneliere, S. Morales, An architecture for model-based and intelligent automation in DevOps, J. Syst. Softw. 217 (2024) 112180. doi:10.1016/j.jss.2024.112180.
[12] D. I. Symonov, V. M. Gorbachuk, A method of finding solutions in a dynamic model of inventory management under uncertainty, Bull. Taras Shevchenko Natl. Univ. Kyiv. Phys. Math. Sci. 4 (2023) 31–39. doi:10.17721/1812-5409.2022/4.4.
[13] D. I. Symonov, Algorithm for determining the optimal flow in supply chains, considering multi-criteria conditions and stochastic processes, Bull. Taras Shevchenko Natl. Univ. Kyiv. Phys. Math. Sci. 2 (2021) 109–116. doi:10.17721/1812-5409.2021/2.15.