

AGENTIC AI FOR AIRCRAFT PRELIMINARY DESIGN: PROOF OF CONCEPT

Ansgar Kirste [*,1], Lau Faltum[1] & Eike Stumpf[1]

[1]Institute of Aerospace Systems, RWTH Aachen University, Germany
[*]Corresponding author. kirste@ilr.rwth-aachen.de

Abstract

Agentic artificial intelligence enables autonomous code generation and iterative reasoning for complex engineering tasks. This work presents a proof of concept for agentic artificial intelligence in aircraft preliminary design. A dedicated multi-agent system is first benchmarked across eight language-model configurations against a reference process of the established UNICADO framework. An agentic application then operates the UNICADO workflow directly. The benchmark shows that the orchestrating role and not the discipline computation limits the agentic workflow, which is reliably handled only by the larger models. Operating the established UNICADO workflow with a sustainability optimization task for a conventional short-range aircraft, current flagship models autonomously reached a substantial reduction of the Average Temperature Response with liquid hydrogen at a low cruise altitude, in line with an in-house research reference. The results indicate that frontier models can operate aircraft preliminary design autonomously, while the specialization through orchestrated agents is more decisive than the raw model size.

Keywords: Aircraft Sizing, Agentic AI, Preliminary Design, LLM Benchmarking, UNICADO

1. Introduction

Aircraft preliminary design for conventional configurations relies on established semi-empirical methods documented in foundational works, such as Raymer [1] and Torenbeek [2]. These methods have been integrated into multidisciplinary design environments like UNICADO[1] [3], enabling systematic analysis across aerodynamics, structures, and performance disciplines. These frameworks automate the coupling and execution of the discipline tools. However, they require the user to define and script the workflow a priori [4]. Until 2024, the primary trend in preliminary aircraft design has been the integration of supervised machine learning methods, particularly neural network-based surrogate models, that replace computationally expensive high-fidelity analyses. Recent work demonstrates this approach: Zimmnau et al. [5] integrated Gaussian process regression surrogate models for aerodynamic analysis into UNICADO, combining low-fidelity methods with CFD simulations to improve accuracy while reducing computational cost. Similarly, Maruyama et al. [6] integrated an efficient global optimization framework based on Gaussian process surrogates into UNICADO. These surrogate methods share two common principles: First, they replace the expensive evaluation itself with a cheap learned approximation. Second, they trade fidelity for speed and leave the surrounding design loop with the engineer [7, 8].

In parallel to these domain-specific machine learning (ML) applications, agentic artificial intelligence (AI) has emerged as a fundamentally different paradigm since late 2023. Agentic AI systems extend large language models (LLMs) with autonomous planning, code generation and execution, and

[1]Available open-source at `https://unicado.io`

iterative refinement capabilities [9]. Recent benchmarks demonstrate their proficiency in software engineering tasks. State-of-the-art models achieve resolution rates exceeding 80 % on the software-engineering benchmark SWE-bench [10, 11], a benchmark evaluating real-world GitHub issue resolution. The key difference is the ability to autonomously plan task sequences, select tools, interpret results, and refine approaches based on feedback. This capability has been demonstrated in software engineering and analytical workflows [12–14].

Agentic AI adds a third form of automation. Surrogates replace the analysis and scripted frameworks execute a fixed workflow. An agent instead operates the toolchain itself. Rather than following a fixed sequence defined in advance, it decides the next step autonomously. It further automates the control layer that an engineer normally performs, while the underlying high-fidelity evaluation remains exact. An agent can in principle call a surrogate model as one of its tools, hence the two approaches can be considered complementary rather than competing. First domain-near studies apply this idea to engineering design tasks, yet they remain limited to individual disciplines, such as multidisciplinary optimization setup or airfoil aerodynamics [4, 15].
The rapid improvement in model accuracy and the expanding availability of specialized agentic AI tools suggest transformative potential for highly integrated domains such as aircraft preliminary design. Particularly for sustainable aviation concepts, including hydrogen-fueled aircraft, blended wing body configurations, and strut-braced wing designs, the design space is considerably larger than for conventional configurations. Continuous optimization across multiple interacting modules is decisive for identifying viable solutions. Agentic AI offers the potential to orchestrate the entire preliminary design workflow. This includes

- interpreting top-level requirements,
- coordinating aerodynamic, structural, and performance analyses,
- evaluating design trade-offs,
- and iteratively refining configurations towards design targets, such as a reduction of the Average Temperature Response.

This holistic integration capability, combined with demonstrated proficiency in code generation and multi-step reasoning, positions agentic AI as a candidate technology for automating the multi-disciplinary complexity of aircraft design analysis.

Despite the demonstrated capabilities of agentic AI in software engineering [10–14] and the established use of machine learning in aircraft design [3, 5, 9], no systematic validation exists so far of whether agentic AI can autonomously execute aircraft preliminary design with acceptable accuracy. Existing AI coding agents such as GitHub Copilot operate at the software development level and do not capture the domain-specific requirements of multidisciplinary aircraft sizing. This represents a critical knowledge gap. Hence, before agentic AI can be considered for advanced applications such as unconventional configurations or autonomous design space exploration, fundamental proof of concept is required for conventional aircraft designs where reference data is well-established and validation criteria are clear.

This work addresses this research gap in two steps. First, a multi-agent proof of concept is presented that operates within a UNICADO-like framework. The agent ingests top-level aircraft requirements (TLAR), manages the associated design parameter set, autonomously derives the lift-to-drag ratio (L/D_{max}) as a representative performance metric, and performs a multi-criteria evaluation of the resulting design for three aircraft classes: conventional narrow-body, wide-body, and hydrogen-fueled aircraft. The lift-to-drag ratio is deliberately chosen as a single discipline metric that represents the broader multidisciplinary process and keeps up with the proof-of-concept scope. Unlike software-focused coding agents, the proposed framework operates at the engineering design level. A coherence model calibration validates the agent output against UNICADO reference values. The benchmarking framework operates on two levels. It evaluates the aircraft design output on the one hand

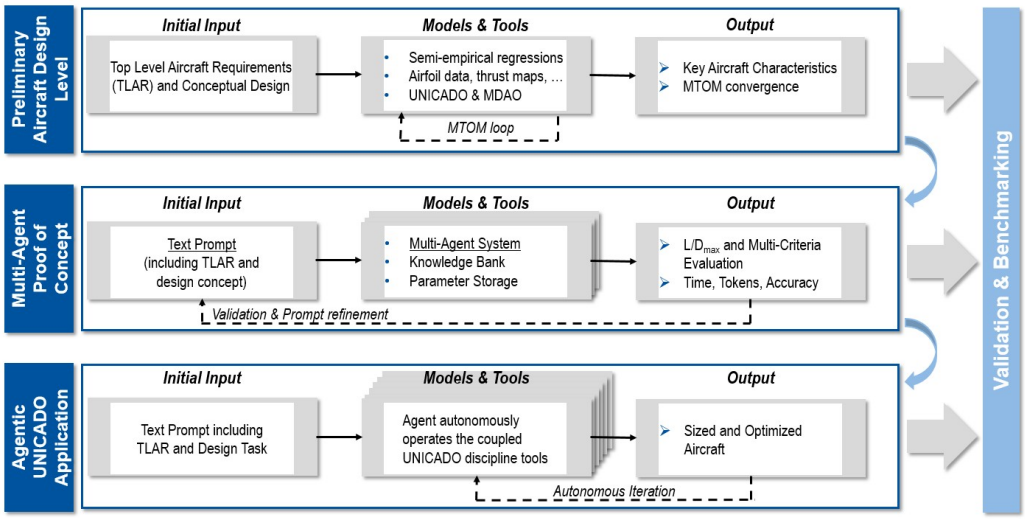


Figure 1 – Methodology of this work, comprising the conventional preliminary aircraft design level as reference, the multi-agent proof of concept, and the agentic UNICADO application, each assessed by coherence model calibration and two-level benchmarking.

and agentic AI performance metrics on the other, namely computation time, token consumption, iteration count, and accuracy versus reference. The framework is applied across a range of proprietary and open-source language models. Second, the transfer of this concept to the actual UNICADO MDAO toolchain is demonstrated, where an agent autonomously operates the established discipline tools to size and optimize the conventional reference aircraft across the coupled design chain. To the best of our knowledge, this is the first proof of concept in which agentic AI autonomously operates a complete and established conceptual aircraft design and optimization environment, in contrast to surrogate-based acceleration, conventional scripted MDAO automation, and existing single-discipline agents.

2. Methodology

To address the research incentive identified above, a structured and verifiable approach is required for the multidisciplinary design task. A proof-of-concept methodology is therefore adopted, as it is suited to assessing feasibility in emerging fields rather than delivering a finished solution [16, 17]. Elliott [16] structures such research into three pillars, namely the use of a prototype, its demonstration, and post-facto arguments that motivate further investigation. Antognolli and Petrillo [18] proposes a comparable structure and provides an explicit planning stage. These perspectives are combined in this work. A planning stage defines the objectives and the validation criteria. A prototype is then implemented and demonstrated. The results are then to be discussed with respect to the transferability of the approach.

The validation follows a coherence model calibration. Two independent systems that produce the same output for identical inputs can be considered coherent [16]. In this work, the multi-agent system and the established UNICADO environment both estimate the lift-to-drag ratio for identical top-level aircraft requirements. A close agreement indicates that the agentic approach reproduces the behavior of the reference process. The lift-to-drag ratio is selected as a representative performance metric. It can be derived from a small set of geometric and aerodynamic parameters. It also admits several solution methods, which makes it suitable for assessing the reasoning of an autonomous agent. This single metric is deliberately chosen at this stage to isolate the agentic behavior from the additional uncertainty of a full multidisciplinary workflow. Further discipline metrics, such as the thrust-specific fuel consumption and the operating empty mass, are introduced with the coupled UNICADO application in Section 3.5. The overall methodology is displayed in Figure 1. It comprises the conventional preliminary aircraft design level as reference, the multi-agent proof of concept, and the agentic UNICADO application. The following subsections describe the multi-agent architecture, the calibration against UNICADO, the benchmarking setup, and the transfer to the established UNICADO toolchain.

https://doi.org/10.71945/icas2026_0575

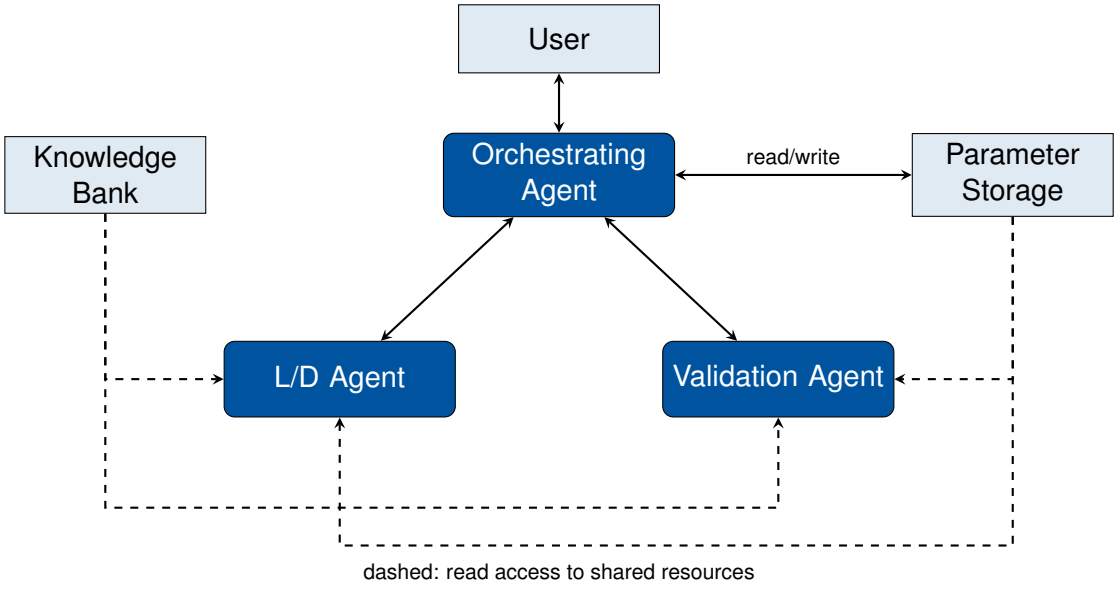


Figure 2 – Vertical architecture of the multi-agent system. The orchestrator controls the L/D and validation agents and is the only component permitted to write to the parameter storage.

2.1 Multi-Agent System Architecture

The multi-agent system follows a vertical architecture [9]. A supervising orchestrator agent controls two subordinate agents and communicates with the user. A vertical architecture is selected because the conceptual design process is sequential and strongly coupled. The orchestrator manages the information flow. It is the only component permitted to write to the shared parameter storage, which maintains a single source of truth across the system. The hierarchical arrangement is shown in Figure 2, and the corresponding information flow is given as an N^2 diagram in Figure 3.

The orchestrating agent coordinates the process. It interprets the user request, delegates the discipline tasks, and is the only agent permitted to write to the parameter storage. Two subordinate agents perform the domain tasks. The L/D agent reads the available parameters and the knowledge bank. It then generates executable Python code based on this knowledge and computes the lift-to-drag ratio. The validation agent assesses the resulting configuration with respect to physical correctness, fuel efficiency, and design innovation, and further proposes targeted improvements. The knowledge bank adds to the underlying conceptual design literature and is described below. The parameter storage shares the parameter values across all agents and holds up to fifteen design parameters, hence maintaining a single source of truth.

The L/D agent serves as a placeholder for an arbitrary discipline module. The architecture can therefore be extended to further design parameters without structural changes. The system is implemented in *n8n*, a node-based workflow automation platform. The generated code is executed on a local Python server, and the returned value is written back to the parameter storage by the orchestrator.

2.2 Coherence Model Calibration

The calibration compares the agent output against UNICADO reference values. Three aircraft classes are considered, namely a narrow-body, a wide-body, and a hydrogen-fueled configuration. The corresponding top-level aircraft requirements are taken from the open-source available UNICADO reference aircraft. This enables a direct comparison between the two systems.

Two knowledge banks are considered. The first contains classical conceptual design literature [1, 2]. The second contains a selection of recent published studies on conceptual design and aerodynamics. This tests whether the knowledge the agent acts upon can be changed by exchanging the knowledge bank rather than modifying the system itself.

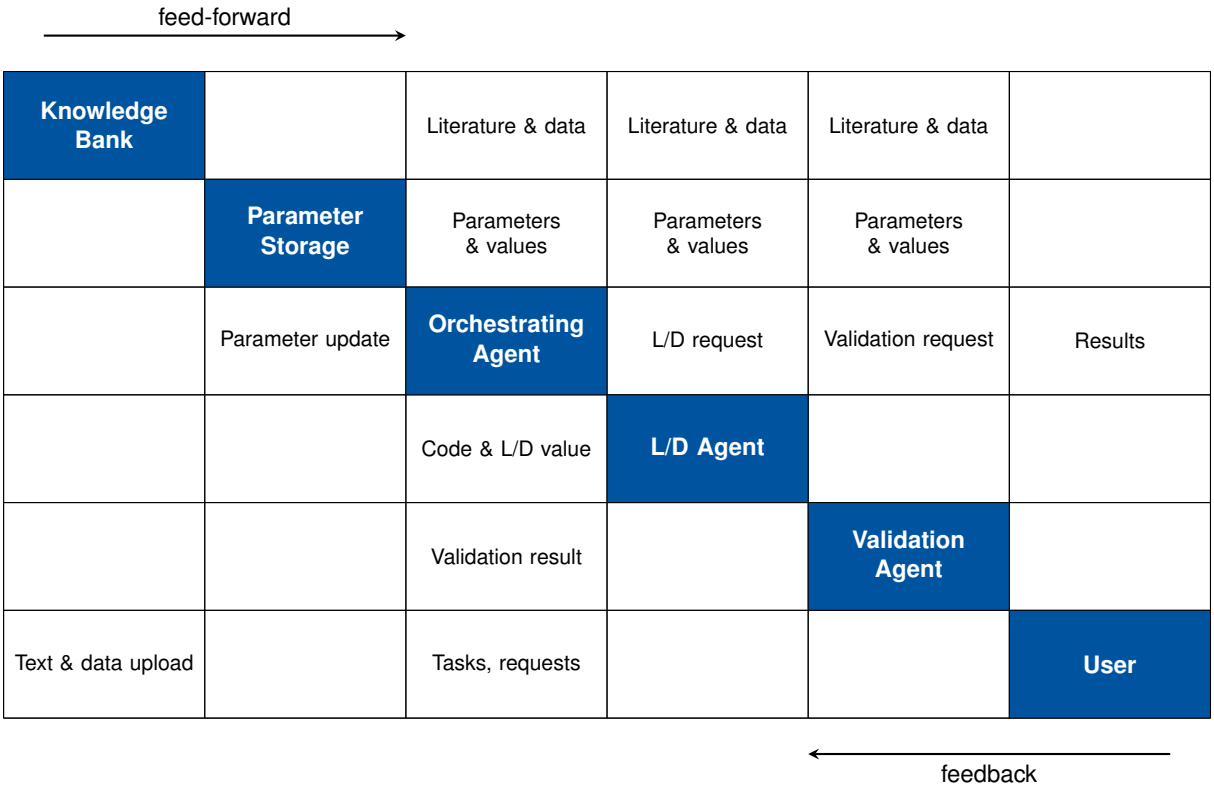


Figure 3 – N^2 diagram of the multi-agent system. Diagonal cells denote the components. Off-diagonal cells denote the information passed between them, with feed-forward couplings in the upper and feedback couplings in the lower triangle.

2.3 Two-Level Benchmarking

The benchmarking records metrics at two levels, following the agentic metric framework of Massoudi and Fuge [19]. At the aircraft design level, the requirements satisfaction and the physical correctness are assessed. Requirements satisfaction measures the deviation of the computed lift-to-drag ratio from the reference value. Physical correctness assesses whether the agent output is dimensionally and physically consistent. At the agentic level, the computation time, the token consumption, the number of agent invocations, and the task completion are recorded.

A total of eight language model configurations are evaluated. The configurations combine proprietary and open-source models. They include configurations in which a single model drives all agents as well as mixed configurations. The single-model configurations apply GPT-5.2, GPT-5.4 mini, Qwen3.6 27B, Gemma4 31b, Mistral 119b, and GPT-oss 120b to all three agents. The two mixed configurations combine a Qwen3.6 27B or a Gemma4 31b orchestrator with GPT-5.2 as the L/D and validation agent. The GPT-5.2, GPT-5.4 mini, Mistral 119b, and GPT-oss 120b models are accessed through the RWTH KI:connect platform using API. The Qwen3.6 27B and Gemma4 31b models are hosted locally on a GPU virtual machine. Each configuration is tested for the two knowledge banks and the three aircraft classes. Each case is repeated three times to account for the stochastic behavior of the models. This design allows conclusions on the influence of the model, the role of the knowledge bank, and the reliability of the agentic system.

2.4 Agentic UNICADO Application

The same agentic principle is applied to the established UNICADO MDAO workflow itself. In contrast to the dedicated multi-agent system, the agent here operates the actual discipline tools of a coupled conceptual design environment. The workflow is executed in headless batch mode of RCE within a virtual machine environment. Here, the agent configures the design case, launches the execution, interprets the returned results, and iterates autonomously.

Two aspects are examined. First, the dependence on the underlying model is assessed by a bench-

mark of several locally hosted open-source models, consistent with the benchmarking of the proof of concept. Second, the agent autonomously sizes and optimizes a reference aircraft, using the Average Temperature Response (ATR) as the climate objective. The resulting configuration is compared against the corresponding UNICADO reference optimization of recent work within the ZeroCloud project. This demonstrates that an agent can operate a complete and coupled design environment, and can autonomously diagnose and resolve issues during the optimization, rather than only execute an isolated discipline module.

3. Results

The proof of concept is evaluated along the two-level benchmarking defined in Section 2.3. The evaluation comprises the eight model configurations introduced in Section 2.3. Three configurations are tested across the full matrix of two knowledge banks, three aircraft classes, and three repetitions, which results in 18 runs each. The three repetitions represent the run-to-run variability of the stochastic models with the proof-of-concept scope rather than provide a complete statistical distribution. These three configurations are GPT-5.2, Gemma4 31b, and the mixed Qwen3.6 27B orchestrator with GPT-5.2 subordinate agents. The remaining five configurations are tested on the narrow-body class only, after they failed reproducibly on the narrow-body cases with a run score of 0. These configurations were not evaluated on the full matrix, since a subordinate agent that cannot read the parameter storage fails identically across all aircraft classes. Therefore, the reported narrow-body results represent their best case rather than a reduced test scope. Each agent receives a completion score of 1, 0.5, or 0 per run. A score of 1 denotes correct execution, a score of 0.5 a non-terminal deviation such as an incomplete parameter transfer, and a score of 0 a terminal failure or a physically incorrect output. The run score is the lowest score among the three agents, following the principle that no information can be considered preferable to wrong information. The lift-to-drag ratio is the requirements-satisfaction metric. Computation time, token consumption, and agent completion are recorded at the agentic level, following Massoudi and Fuge [19]. The results of the multi-agent proof of concept are presented in Sections 3.1 to 3.4. The transfer to the agentic UNICADO application is presented in Section 3.5.

3.1 Benchmarking across Models and Agent Roles

The agent completion success rates are shown in Figure 4. GPT-5.2 is the strongest configuration within the test set. It reaches a completion score of 1.0 for the L/D agent and the validation agent across all 18 runs. The orchestrating agent reaches 0.639 and remains the limiting role. The highest full-run success, defined as all three agents scoring 1.0 within the same run, is 7 of 18 runs and is also achieved by GPT-5.2. Only three configurations reach a non-zero full-run success, namely GPT-5.2 with 38.9 %, the Qwen3.6 27B orchestrator combined with GPT-5.2 subordinate agents with 33.3 %, and GPT-5.4 mini with 28.6 %.

The two mixed configurations isolate the orchestrating role. The L/D agent and the validation agent are fixed to GPT-5.2, and only the orchestrator is exchanged. The orchestration success then drops from 0.639 for GPT-5.2 to 0.527 for the Qwen3.6 27B orchestrator and to 0 for the Gemma4 31b orchestrator. The Gemma4 31b orchestrator discards the correct L/D values of the GPT-5.2 subordinate agent and substitutes its own incorrect values in all but one run. This indicates that the orchestrating role rather than the discipline computation is the systematic bottleneck of the multi-agent system. The locally hosted open-source models fail the subordinate roles in particular. The L/D and validation agents based on these models do not reliably read the parameter storage. All models operate the same tools correctly in the orchestrating role, which suggests a capability limit rather than a configuration error.

The requirements satisfaction is assessed against the UNICADO reference. The reference values of the maximum lift-to-drag ratio are 17.64 for the narrow-body (SMR-01), 20.63 for the wide-body (LR-01), and 16.56 for the hydrogen (SMR-LH2-01) configuration. UNICADO derives the lift-to-drag ratio from a component-wise drag build-up. It resolves the induced drag with a lifting-line method, estimates the viscous drag with handbook methods after Raymer, and models the wave drag with the Korn-Mason relation. For the comparison, a simplified closed-form relation is used instead. It represents the induced drag through an Oswald factor and combines the remaining contributions

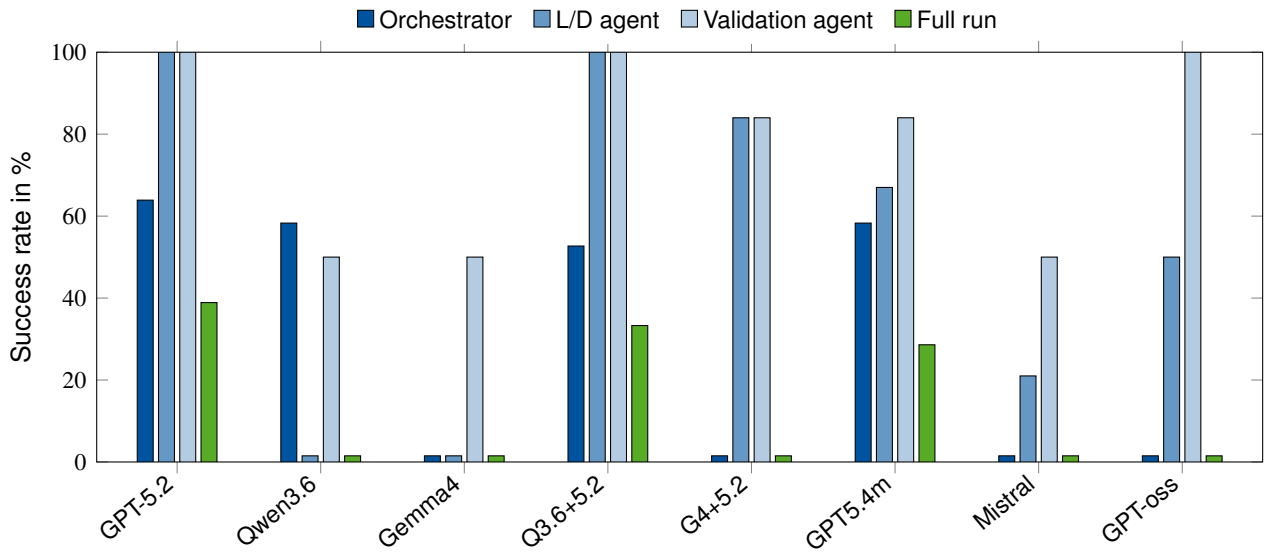


Figure 4 – Agent completion success rates by model configuration. Five configurations are evaluated on the narrow-body class only.

Table 1 – Requirements satisfaction of the multi-agent system. The agent column denotes the strongest configurations, which reproduce the analytical reference exactly.

Aircraft	UNICADO	Closed-form	Agent	Deviation
Narrow-body (SMR-01)	17.64	18.39	18.39	+4.3 %
Wide-body (LR-01)	20.63	21.13	21.13	+2.4 %
Hydrogen (SMR-LH2-01)	16.56	17.50	17.50	+5.7 %

into a single zero-lift drag coefficient. The closed-form relation used here is the parabolic drag polar $C_D = C_{D_0} + C_L^2/(\pi AR\, e)$, with aspect ratio AR and Oswald factor e, from which the maximum lift-to-drag ratio follows as $(L/D)_{max} = \frac{1}{2}\sqrt{\pi AR\, e/C_{D_0}}$. It combines the zero-lift drag contributions into a single zero-lift drag coefficient C_{D_0} and does not include a wave drag term. The same top-level requirements evaluated with the closed-form relation yield 18.39, 21.13, and 17.50. The strong configurations reproduce these analytical values without deviation. This results in a systematic offset against UNICADO of 4.3 %, 2.4 %, and 5.7 % as summarized in Table 1. The offset can therefore be attributed to the difference in the drag estimation method and not to the agent. The agent reproduces the analytical reference exactly. This confirms the correct execution of the closed-form relation rather than the physical fidelity against UNICADO. The residual deviation reflects the difference between the closed-form relation and the UNICADO drag build-up, not an error of the agent. It should be noted that only GPT-5.2 and the Qwen3.6 27B orchestrator combined with GPT-5.2 subordinate agents deliver the expected values reliably across all conditions. The accuracy result is consistent across both knowledge banks. The larger knowledge bank increases token consumption without changing the L/D value, which indicates that the closed-form task does not benefit from the additional literature.

3.2 Failure Mode Analysis

The run outcomes are classified by the lowest agent score and shown in Figure 5. A run is clean when all three agents score 1.0, degraded when the lowest score is 0.5, and failed when the lowest score is 0. Only the GPT-5.2 driven configurations produce clean and degraded runs. GPT-5.2 reaches 39 % clean and 50 % degraded runs, and the Qwen3.6 27B orchestrator combined with GPT-5.2 reaches 33 % clean and 39 % degraded runs. All configurations that use an open-source model in a subordinate role fail every completed run. This pattern is consistent across the narrow-body, wide-body, and hydrogen cases.

The dominant failure modes are listed in Table 2. The decisive failure of the open-source subordinate

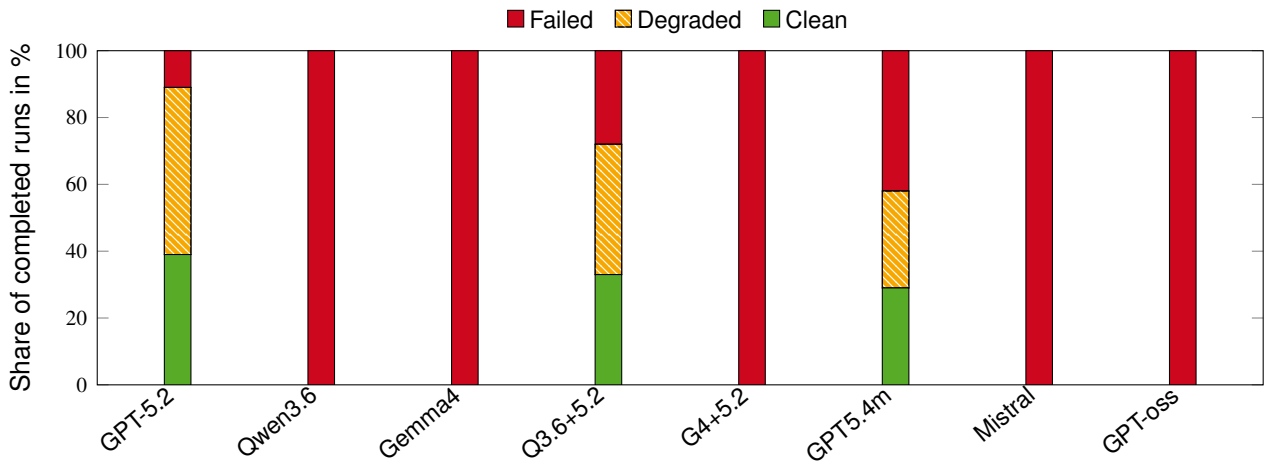


Figure 5 – Run outcome distribution by configuration, classified by the lowest agent score. Five configurations are evaluated on the narrow-body class only.

Table 2 – Dominant failure modes and affected configurations.

Failure mode	Score	Affected configurations
Parameter storage not read (sub-agent)	0.5	Gemma4, Qwen3.6, Mistral, GPT-oss
Generated code not executable	0.5	open-source subordinate agents
Orchestrator substitutes own content	0	all except GPT-5.2
Physically incorrect L/D relation	0	Mistral, GPT-5.4 mini
Incomplete parameter transfer	0.5	GPT-5.2, mixed configurations
Incorrect chat formatting	1	GPT-5.2

agents is the inability to read the parameter storage through the spreadsheet tool. This failure affects 17 of the Gemma4 31b runs and recurs for Qwen3.6 27B, Mistral 119b, and GPT-oss 120b. The spreadsheet tool was successfully read by an open-source subordinate agent in only three runs in total. A second decisive failure is the orchestrator substituting its own content for the output of the subordinate agents. This failure is observed for every configuration except GPT-5.2 and is most pronounced for Gemma4 31b. The substituted content includes lift-to-drag values, Python code, and validations, which are frequently incorrect. A third failure mode concerns physical correctness. Mistral 119b applied an incorrect form of the closed-form relation in two runs by omitting the factor of one half. GPT-5.4 mini returned a lift-to-drag ratio without generating executable code in two runs. The value was asserted directly rather than computed, and it was incorrect in both cases. The primary deviations of GPT-5.2 are non-terminal. They consist of incorrect chat formatting and incomplete parameter transfer, in particular for text-based parameters such as the cabin layout. These entries are part of the shared parameter set but do not enter the lift-to-drag computation directly.
The capability threshold is visible across all metrics. Here, the number of parameters refers to the size of the language model in billions of trainable weights, not to the aircraft design parameters in the storage. The configurations below roughly 27 billion parameters (trainable weights) fail the subordinate roles reproducibly, while the strong proprietary models complete the full matrix. This is in line with the agent benchmarks reported for software engineering tasks, where long-horizon orchestration rather than the isolated subtask is the limiting capability [10, 11].

3.3 Runtime and Token Consumption

The agentic-level metrics are summarized in Table 3 and the completion time per run is shown in Figure 6. The locally hosted open-source models are the slowest. The absolute run times are not directly comparable, since the local models run on a single GPU virtual machine while the proprietary models are served through a scaled API infrastructure. Therefore, the time is an indicative reference value rather than a controlled benchmark. The Qwen3.6 27B configuration averages 52.4 min per

Table 3 – Agentic-level metrics per configuration, averaged over the completed runs. Tokens and cost denote the input side.

Configuration	Time min	Tokens 10^6	Cost USD	Clean %
GPT-5.2	5.6	2.64	4.60	39
Qwen3.6 27B	52.4	4.70	1.25	0
Gemma4 31b	17.3	1.60	0.15	0
Qwen3.6 27B + GPT-5.2	17.9	1.95	1.65	33
Gemma4 31b + GPT-5.2	14.0	1.95	1.40	0
GPT-5.4 mini	3.9	1.55	1.40	29
Mistral 119b	4.2	2.80	0.40	0
GPT-oss 120b	9.7	5.40	0.20	0

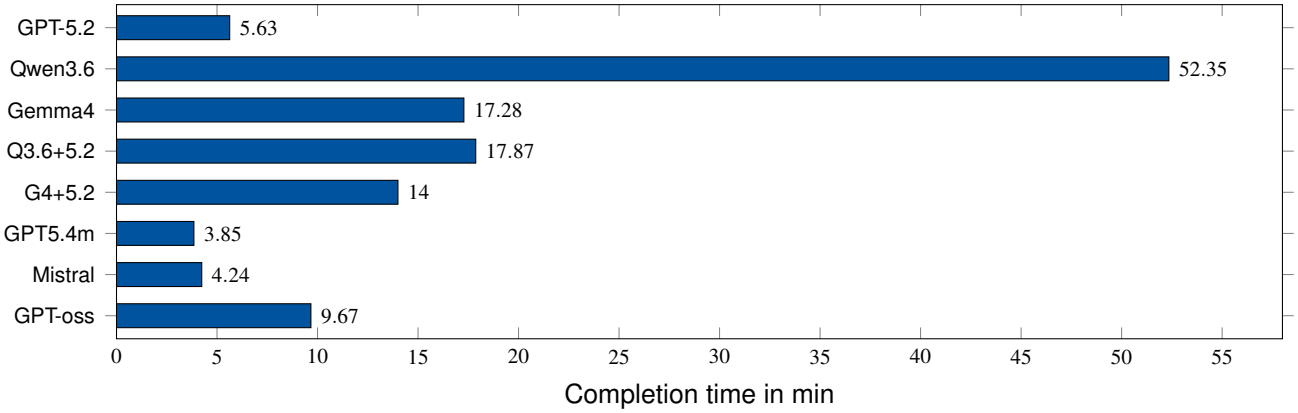


Figure 6 – Average completion time per run by configuration.

run, because the orchestrator recalls the L/D agent repeatedly when the agent cannot return a value. The completion time drops to 17.9 min when Qwen3.6 27B operates only the orchestrator and GPT-5.2 operates the subordinate agents. The proprietary models are faster, with GPT-5.4 mini at 3.9 min and GPT-5.2 at 5.6 min. Run time is not a reliable comparison metric, since the smaller Mistral 119b model completes faster than GPT-5.2 at 4.2 min while failing every run.

The input token consumption dominates the total token count in every configuration, as shown in Figure 7. The orchestrating agent accounts for the largest share. The Qwen3.6 27B and GPT-oss 120b configurations consume the most input tokens, at 4.7 and 5.4 million per run, because the orchestrator repeats failed tool calls. The input cost ranges from 0.15 USD for Gemma4 31b to 4.60 USD for GPT-5.2. The high cost of GPT-5.2 results from its per-token price rather than its token count. The larger knowledge bank roughly doubles the token consumption of GPT-5.2, from 1.42 to 3.83 million input tokens per run, without changing the computed lift-to-drag ratio. This indicates that the knowledge bank affects the cost but not the accuracy of the closed-form task. The result also indicates a structural limit. Large input contexts approach the maximum input token count of the smaller models, which motivates a structured retrieval of only the task-relevant knowledge in a larger design environment.

3.4 Extension toward a Multidisciplinary Process

As a proof of concept, the multi-agent system computes a single discipline metric. The conceptual aircraft design process couples several disciplines. The transferability of the architecture is therefore assessed by an extension. A replica of the L/D agent is added as an additional discipline agent and is tasked with estimating the maximum take-off mass and the aspect ratio. The orchestrator system prompt is adjusted to call the additional agent before the L/D agent. No structural change of the architecture is required, which confirms that the L/D agent can be considered a placeholder for an

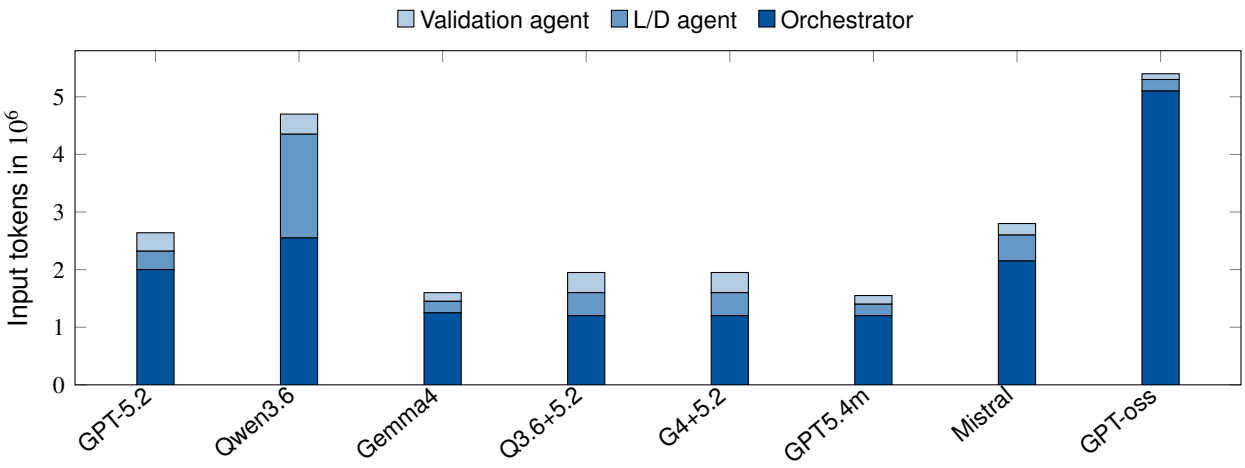


Figure 7 – Average input token consumption per run, separated by agent.

arbitrary discipline module. The extension is conducted with GPT-5.2 for all agents across three runs. The extension operates the additional task adequately. All three runs reach a completion score of 1.0 across all agents. The additional agent estimates the maximum take-off mass from fuel fractions, derives the wing area from an assumed wing loading, and computes the aspect ratio. The aspect ratios of the three runs agree within 5 %. However, the estimated maximum take-off mass is more than 20,000 kg below the UNICADO reference in all three runs, which raises the lift-to-drag ratio to values between 22.4 and 22.9. The deviation can be reasoned by the absence of design iterations, which are an essential part of any conceptual design study. The extension also increases the average cost by 2.29 USD per run, which scales with the number of agents. The extension therefore demonstrates the modularity of the architecture rather than the fidelity of a complete design tool. The fidelity gap is the central reason for operating the established UNICADO process directly, which is presented in the following section.

3.5 Agentic UNICADO Application

The same agentic principle is applied to the established UNICADO multidisciplinary workflow. The agent operates the actual discipline tools in headless batch mode within a virtual machine. The application proceeds in three steps of increasing autonomy, namely the reliable operation of the workflow, the autonomous optimization of the aircraft, and the modification of a discipline source code.

3.5.1 Reliable Operation of the Workflow

The first step tests which models operate the coupled toolchain reliably. The agent is prompted to change the design range to 4,000 km, execute the sizing workflow, and report the resulting key performance indicators. These are the maximum take-off mass, the maximum zero-fuel mass, and the design-mission fuel mass. Eight locally hosted open-source models are evaluated. Four models with at least 27 billion parameters complete the task and return bit-identical results, with a maximum take-off mass of 75,806 kg. The four smaller models below this threshold fail and return fabricated values. This agreement confirms the determinism of the workflow. At the same time, per-model sandboxing is essential, because a single malformed command destroyed the shared aircraft exchange file in an unsandboxed pre-test. This capability threshold matches the one found in the multi-agent benchmark.

3.5.2 Autonomous Optimization

The second step tests whether an agent can optimize the aircraft within the workflow. The agent is prompted to minimize the Average Temperature Response (ATR) of a 180-passenger reference aircraft within at most 100 converged runs and without source-code changes. The design range is fixed as the top-level requirement, while all other parameters are free. These include the cruise

Table 4 – Autonomously optimized aircraft configurations within the UNICADO workflow. The kerosene baseline is compared against the kerosene local optimum and the liquid hydrogen global optimum found by Claude Fable 5. ATR denotes the Average Temperature Response.

Configuration	Cruise FL / M	MTOM kg	ATR K	Reduction %
Baseline (kerosene)	330 / 0.78	77 888	1.96×10^{-6}	—
LocalOptimum (kerosene)	190 / 0.50	76 757	4.67×10^{-7}	76.2
Optimum (LH_2)	200 / 0.57	76 452	1.03×10^{-7}	94.7

altitude, the cruise Mach number, the wing geometry, the energy carrier, and the tank configuration. The agent may only use the engines available in the database and may not scale them freely. The kerosene baseline reaches an ATR of 1.96×10^{-6} K at flight level 330 and Mach 0.78.
Claude Opus 4.7 operates and optimizes the workflow by iteratively adjusting the free parameters, running the sizing loop, and reading back the output and resulting ATR. Even with additional guidance towards a possibly better optimum, it reaches only a conservative local optimum with a reduction of 10 % at flight level 370 and Mach 0.76 with kerosene. Claude Fable 5 identifies the global optimum autonomously. It reaches a reduction of 94.7 % at flight level 200 and Mach 0.57 with liquid hydrogen. The optimum was confirmed in a repeated run. The best kerosene configuration reaches a reduction of 76.2 % at flight level 190 and Mach 0.50. The configurations are summarized in Table 4.
Claude Fable 5 performs the change to liquid hydrogen consistently across the coupled disciplines. The hydrogen optimum carries a 16 % higher operating empty mass from the tank installation and requires 61 % less block fuel mass. The agent identifies the dominant lever, namely the collapse of the aviation-induced cloudiness forcing below flight level 215. It separates the sizing condition from the operating condition, which outperforms a redesign for the low cruise condition. The global optimum is in line with the reduction potential of approximately 95 % of the ZeroCloud project. The agent therefore reproduces a currently unpublished research result and operates a complete and coupled design environment rather than an isolated discipline module.

3.5.3 Source-Code Modification

The third step tests whether an agent can modify a discipline method. The agent is prompted to adapt the wing structural weight computation in the wing design module. It locates the relevant routine in the compiled C++ module, applies a valid modification, and rebuilds the tool so that the change takes effect in the coupled workflow. This indicates that the agent operates beyond the execution of the fixed workflow and can extend the discipline methods themselves, which is the prerequisite for the autonomous exploration of unconventional configurations.

4. Discussion

The results allow a discussion in three aspects.

- First, the proof of concept shows that a dedicated agent per discipline module is the productive architecture. Larger models are on average more capable, and a capability threshold near 27 billion parameters separates the models that complete the agentic workflow from those that fail the subordinate roles in this study. This threshold is specific to the present tasks and model set, and its transferability to other tasks remains to be further assessed. The orchestrating role and not the discipline computation determines the success. The specialization through multiple agents can therefore be considered more important than the raw model size.

- Second, the approach scales to the established UNICADO process. A single agent operates and optimizes the complete and coupled workflow and reproduces a currently unpublished in-house optimum, which provides an independent validation. Modifications of the discipline source code are already possible. The dedicated multi-agent system demonstrates the modularity, whereas a high fidelity requires the underlying framework. A full multidisciplinary process is therefore achieved through the orchestrated addition of further specialized agents and

not through a re-implementation of the disciplines. The workflow integration matters for the efficiency, because the input tokens dominate the cost and favor a selective retrieval of the task-relevant knowledge.

- Third, the main limitations concern the operational reliability and the scalability. Per-model sandboxing and a strict version control of the baseline are essential, because a single malformed command can destroy the shared design state. It should be noted that the agent must not be biased by access to failed runs or earlier results. A controlled memory across the runs is therefore the central lever for scaling to larger campaigns. The data handling is a further consideration, because the cloud-hosted proprietary models process non-open-source data externally, in contrast to the local open-source models.

5. Summary

This work presents a proof of concept for agentic artificial intelligence in aircraft preliminary design, following a staged approach of increasing autonomy. A dedicated multi-agent system and an agentic application of the established UNICADO workflow are evaluated. The benchmark identifies the orchestration as the limiting capability, with a capability threshold near 27 billion parameters, and shows that the specialization through orchestrated agents is more decisive than the raw model size. Operating the established workflow, current flagship models autonomously reach a 94.7 % reduction of the Average Temperature Response with liquid hydrogen at flight level 200, in line with an in-house reference of approximately 95 %. The approach therefore scales from an isolated discipline metric to the operation and optimization of a complete and coupled design environment. Based on this proof of concept for conventional configurations, the approach appears particularly relevant for the design space exploration of innovative aircraft concepts. The proof of concept motivates the orchestrated coupling of several discipline agents and a structured memory.

CRediT Authorship Contribution Statement

Ansgar Kirste: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Lau Faltum:** Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization. **Eike Stumpf:** Conceptualization, Funding acquisition, Supervision, Validation, Writing – review & editing.

Funding

This work has been prepared within the project "UniSelect", supported by the funding scheme "LuFo VI-3" by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) (Grant no. 20E2228A). Besides the financial support, the funding source had no involvement in this research.

Competing Interests

The authors declare no competing interests.

Acknowledgements

The authors acknowledge the use of the RWTH Aachen KI:connect service for access to the language models.

Data Availability

Data is available upon reasonable request from the corresponding author.

Contact Author Email Address

Ansgar Kirste: ansgar.kirste@ilr.rwth-aachen.de
Institute of Aerospace Systems: research@ilr.rwth-aachen.de

Copyright Statement

The authors confirm that they, and/or their company or organization, hold copyright on all of the original material included in this paper. The authors also confirm that they have obtained permission, from the copyright holder of any third party material included in this paper, to publish it as part of this paper. The authors confirm that they give permission, or have obtained permission from the copyright holder of this paper, for the publication and distribution of this paper as part of the ICAS proceedings or as individual off-prints from the proceedings.

References

[1] D. Raymer. *Aircraft Design: A Conceptual Approach*. American Institute of Aeronautics and Astronautics, Reston, VA, 6th edition, 2018.

[2] E. Torenbeek. *Advanced Aircraft Design: Conceptual Design, Analysis and Optimization of Subsonic Civil Airplanes*. John Wiley & Sons, Chichester, UK, 1st edition, 2013.

[3] M. Zimmnau, F. Schültke, and E. Stumpf. UNICADO: Multidisciplinary analysis in conceptual aircraft design. *CEAS Aeronautical Journal*, 14(1):75–89, 2023. doi: 10.1007/s13272-022-00620-3.

[4] Justin S. Gray, John T. Hwang, Joaquim R. R. A. Martins, Kenneth T. Moore, and Bret A. Naylor. OpenMDAO: an open-source framework for multidisciplinary design, analysis, and optimization. *Structural and Multidisciplinary Optimization*, 59(4):1075–1104, 2019. doi: 10.1007/s00158-019-02211-z.

[5] M. Zimmnau, S. Cremer, A. Uhl, and E. Stumpf. Towards the integration of an aerodynamic variable fidelity surrogate model into a conceptual aircraft design environment. In *AIAA SciTech 2025 Forum*, 2025. doi: 10.2514/6.2025-0003.

[6] Daigo Maruyama, Florian Schültke, Eike Stumpf, and Ali Elham. Efficient global multidisciplinary design optimization to support aircraft conceptual design. In *Proceedings of AeroBest 2021*, pages 119–139, Lisbon, Portugal, 2021.

[7] Alexander I. J. Forrester and Andy J. Keane. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45(1–3):50–79, 2009. doi: 10.1016/j.paerosci.2008.11.001.

[8] Nestor V. Queipo, Raphael T. Haftka, Wei Shyy, Tushar Goel, Rajkumar Vaidyanathan, and P. Kevin Tucker. Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, 41(1):1–28, 2005. doi: 10.1016/j.paerosci.2005.02.001.

[9] Ranjan Sapkota, Konstantinos I. Roumeliotis, and Manoj Karkee. AI agents vs. agentic AI: A conceptual taxonomy, applications and challenges. *Information Fusion*, 126:103599, 2026. doi: 10.1016/j.inffus.2025.103599.

[10] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. R. Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *Proc 12th International Conference on Learning Representations (ICLR Oral)*, Vienna, Austria, 2024.

[11] J. Yang, C. E. Jimenez, A. Wettig, K. Lieret, S. Yao, K. R. Narasimhan, and O. Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *38th Annual Conference on Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada, 2024.

[12] X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang, S. Zhang, X. Deng, A. Zeng, Z. Du, C. Zhang, S. Shen, T. Zhang, Y. Su, H. Sun, M. Huang, Y. Dong, and J. Tang. AgentBench: Evaluating llms as agents. In *12th International Conference on Learning Representations (ICLR)*, Vienna, Austria, 2024.

[13] S. Hong, X. Zheng, J. Chen, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, and C. Wu. MetaGPT: Meta programming for a multi-agent collaborative framework. In *12th International Conference on Learning Representations (ICLR Oral)*, Vienna, Austria, 2024.

[14] J. He, C. Treude, and D. Lo. LLM-based multi-agent systems for software engineering: Literature review, vision and the road ahead. *ACM Transactions on Software Engineering and Methodology*, 34(5):1–30, 2025. doi: 10.1145/3712003.

[15] Yi Fan, Hao Zhan, Mingxuan Zhang, and Baigang Mi. AirfoilAgent: Airfoil aerodynamics optimization design via large language model multi-agent collaborations. *Advanced Engineering Informatics*, 71:104246, 2026. doi: 10.1016/j.aei.2025.104246.

[16] Steve Elliott. Proof of concept research. *Philosophy of Science*, 88(2):258–280, 2021. doi: 10.1086/711503.

[17] Catherine Elizabeth Kendig. What is proof of concept research and how does it generate epistemic and ethical categories for future scientific practice? *Science and Engineering Ethics*, 22 (3):735–753, 2016. doi: 10.1007/s11948-015-9654-0.

[18] Bruno Fernando Antognolli and Fabio Petrillo. Proof of concept as a first-class architectural decision instrument, 2026. arXiv preprint 2604.05835.

[19] Soheyl Massoudi and Mark Fuge. Agentic large language models for conceptual systems engineering and design. In *Volume 3B: 51st Design Automation Conference (DAC)*. American Society of Mechanical Engineers, August 2025. doi: 10.1115/DETC2025-168856.