Digital Design Network+
Case Study 1 Individual Report Template

Name	Johannes Josef Norheim
Email	johannes.norheim@strath.ac.uk
Institution	University of Strathclyde
Team	1
Focus	In Challenge 1, our research team worked towards a workflow that enables rapid iteration on design requirements through computational design methods and automated simulation-based verification and validation. The project documented in this report specifically focused on the application of Large Language Models (LLMs) to generate simulation-based tests that achieve coverage over the requirements. This involves processing the natural language embedded in requirements and ensuring that the generated test configurations can be executed by the simulator (generating the tests correctly), and that the executed simulations ultimately correctly test the requirement statements (generating the right tests). This will ultimately support faster design iteration while ensuring early discovery of unsatisfied requirements, speeding up requirements-based design workflows.
Design Process	This was targeting the requirements engineering part of the design process.
Tools Used	Digital tools: LLM models (through API usage) involved: OpenAI's GPT-4o, GPT-5-mini and Google's Gemma 4
Links	https://github.com/norheim/reqverify

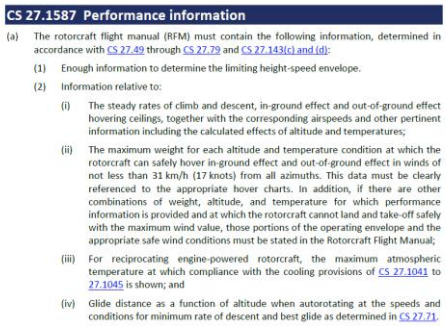
Executive Summary

The engineering of complex systems, such as drones designed to operate in challenging environments (e.g., Mars), is often requirements-driven, introducing a verification step in which the design is evaluated against natural-language-specified requirements for the functionality and expected performance. In the digital context, this introduces the opportunity to tighten the integration of the verification loop with digital design models across simulated environments, enabling comparison of expected performance against requirements.

This work focused on applying Large Language Models (LLMs) to generate simulation-based test plans from requirements. This involves processing the natural language embedded in requirements, ensuring that the generated test configurations can be executed by the simulator (ensuring the tests are generated correctly), and that the executed simulations ultimately correctly test the requirement statements (ensuring the right tests are generated). This ultimately supports faster design iteration while ensuring early discovery of unsatisfied requirements, speeding up requirements-based design workflows.

Problem

A widely adopted product development process (PDP) for complex systems, such as aerial drones, is the V-model of systems engineering [1], in which key functions, constraints, and performance targets are captured in requirements documents, such as the one shown in Figure 1.



CS 27.1587 Performance information

(a) The rotorcraft flight manual (RFM) must contain the following information, determined in accordance with CS 27.49 through CS 27.79 and CS 27.143(c) and (d):

(1) Enough information to determine the limiting height-speed envelope.

(2) Information relative to:

(i) The steady rates of climb and descent, in-ground effect and out-of-ground effect hovering ceilings, together with the corresponding airspeeds and other pertinent information including the calculated effects of altitude and temperatures;

(ii) The maximum weight for each altitude and temperature condition at which the rotorcraft can safely hover in-ground effect and out-of-ground effect in winds of not less than 31 km/h (17 knots) from all azimuths. This data must be clearly referenced to the appropriate hover charts. In addition, if there are other combinations of weight, altitude, and temperature for which performance information is provided and at which the rotorcraft cannot land and take-off safely with the maximum wind value, those portions of the operating envelope and the appropriate safe wind conditions must be stated in the Rotorcraft Flight Manual;

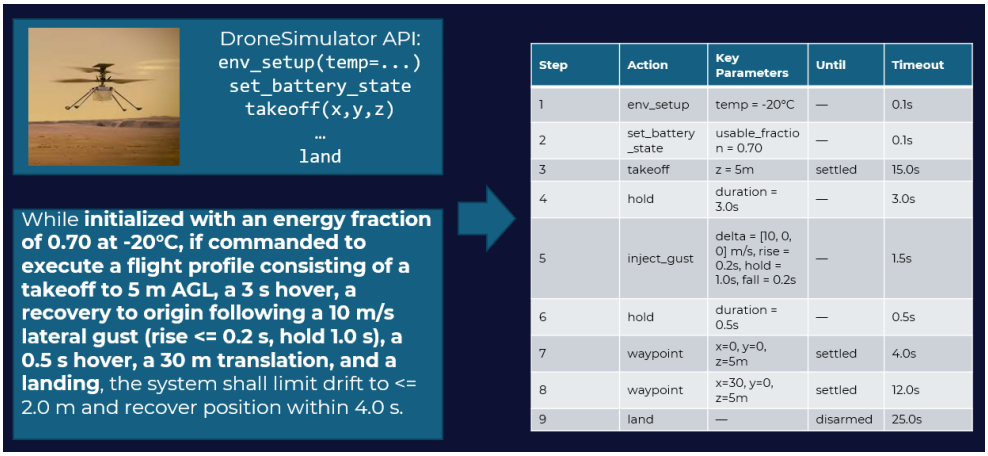
(iii) For reciprocating engine-powered rotorcraft, the maximum atmospheric temperature at which compliance with the cooling provisions of CS 27.1041 to 27.1045 is shown; and

(iv) Glide distance as a function of altitude when autorotating at the speeds and conditions for minimum rate of descent and best glide as determined in CS 27.71.

Figure 1 Excerpt of EASA certification requirement for rotorcraft [2]

To verify these requirements, system and/or test engineers must interpret them, manually create test plans and procedures, and run them against the design to verify and validate the requirements. This is normally not carried out until the system has been fully implemented; the digital environment offers an opportunity to carry out such activities earlier [3]. The work described in this report focused only on **verification** (the ability to perform as specified in the requirements) rather than validation (ensuring the design performs as expected by the stakeholder).

Note that this verification loop might be trivial when we have a mathematical model that has been developed specifically to evaluate a performance metric of the design (e.g. a mass model, or an aerodynamic CFD code to predict the lift); the major technical challenge addressed in this work focuses on the more general case; when a general purpose digital tool (e.g. a simulation tool) with which we can only interact through a high-level API (application programming interface), can be used to test the design under different conditions, mimicking the way different functionalities might be triggered in the system externally. For a Martian drone, an illustrative requirement, an example API, and a possible test plan for testing the design are shown in Figure 2.



Step	Action	Key Parameters	Until	Timeout
1	env_setup	temp = -20°C	—	0.1s
2	set_battery_state	usable_fraction = 0.70	—	0.1s
3	takeoff	z = 5m	settled	15.0s
4	hold	duration = 3.0s	—	3.0s
5	inject_gust	delta = [10, 0, 0] m/s, rise = 0.2s, hold = 1.0s, fall = 0.2s	—	1.5s
6	hold	duration = 0.5s	—	0.5s
7	waypoint	x=0, y=0, z=5m	settled	4.0s
8	waypoint	x=30, y=0, z=5m	settled	12.0s
9	land	—	disarmed	25.0s

Figure 2 Illustrative requirement, as well as simulation interface, and expected output plan structure

The vision is that, whereas today test plans and execution involve significant manual effort and a risk of late discovery and rework, tomorrow we will be able to swiftly generate tests and run them to verify requirements or help iterate, without the resource-intensive processes of today, enabling faster iteration of the systems engineering V-model.

Digital Design Solution

The key solution adopts the LLM as the generator of the test scripts; in order to ensure executable test scripts are generated, we adopt a minimal data structure that forces the LLM to output a test as a list of actions, where actions are either a change to the environment (e.g. changing the wind, or other boundary conditions) or high-level activities enabled by the system (e.g. take-off, waypoint navigation, etc...). With each action follows any parameters that the action might take (some actions don't have parameters), and for some actions, an until condition can be specified (which checks for the system reaching a target state), as well as a timeout for which the effect is action-specific. The overall methodology is shown in Figure 3:

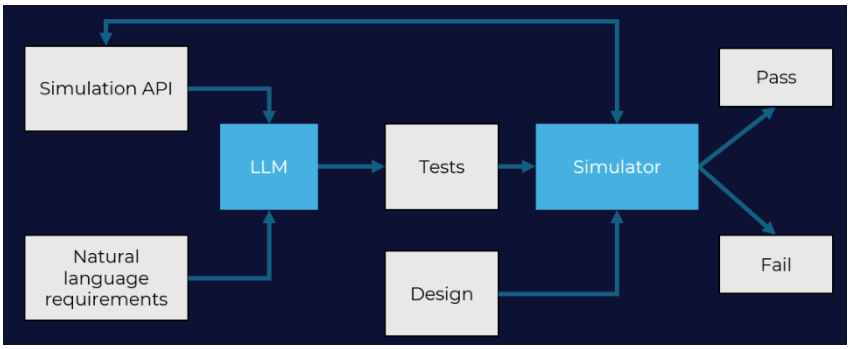


Figure 3 Methodology introduced in this project

The LLM-generated script is compared with the benchmark script, and any discrepancies in the generated trajectories are noted. We also evaluate the simulated trajectories against the requirements criteria to determine whether the design passes or fails them. We test across four LLMs: Google's Gemini model, OpenAI's GPT-4o (non-reasoning model), GPT-5-mini (reasoning model), and a recently developed open-source model, Google's Gemma-4-31B. Different flight trajectories resulting from the LLM applied to an excerpt of some of the requirements used are shown in Figure 4.

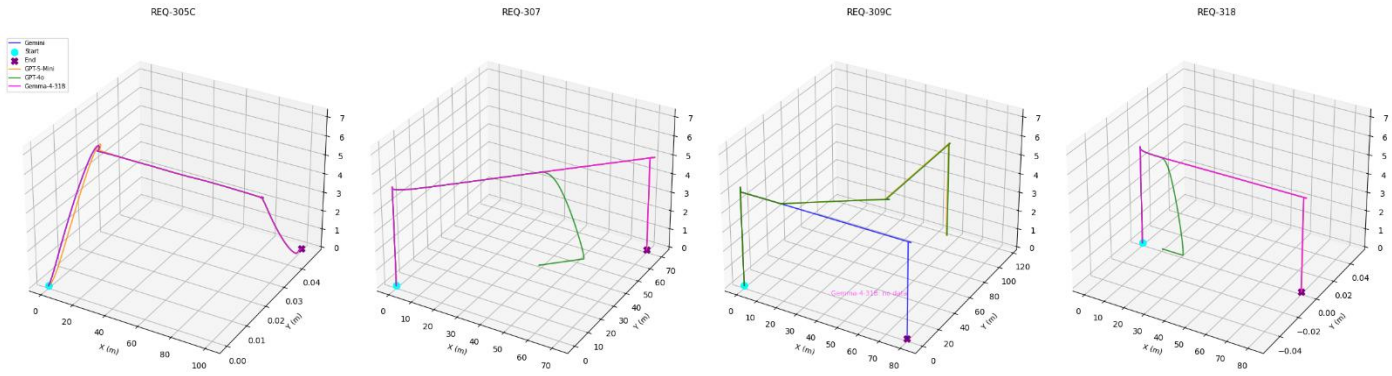


Figure 4 Excerpt of drone trajectories flown resulting from test sequences generated by LLM

A key observation is that all LLMs successfully generated valid flight sequences encoded in a data structure compatible with the minimal simulation API developed for this project. As a result, executable sequences were generated, and the design could be tested against the requirements. However, one key result of this project is that the LLMs' trajectories were not always identical, as shown in Figure 4. This highlights that LLMs will not always produce consistent results and further reinforces the need for robust benchmarks to evaluate the performance of different models for these use cases, to better characterise failure modes and any risks introduced by automation. This aligns with the existing literature [4].

How it makes Digital Design Better, Faster and/or With Less

Today, test plans and execution involve significant manual effort and a risk of late discovery and rework. This solution enables earlier integration of verification efforts, ultimately enabling the swift generation of tests to verify requirements and faster iteration of the systems engineering V-model.

Reflections

The major takeaway from this project is that generative AI technology (LLMs) can successfully generate test scripts under controlled conditions, such as well-written requirements and a limited API/simulation tool environment. However, we still have limited insight into all failure modes and into benchmarking the method against the state of practice. A few challenges were encountered:

- Ambiguity in requirements leads to test case degeneracy (more than one test possible per one requirement); this is related to the challenge of requirements coverage (i.e. can one test check for all cases implied by the requirement).
- When automating the test plan/procedure generation stage, we introduce a new requirements verification failure mode, where the design passes because the test plan was trivial, or did not correctly test the design. Uncovering such failure modes could arise from designs that we know should fail yet still pass; however, generating "failing" designs poses its own set of challenges.
- There are non-LLM-based requirements verification methods, like model verification, where requirements can be exhaustively tested in a formal sense; it might be interesting to compare the performance of the LLM to the formal methods; however, this is a very limited subset of the cases.
- Ultimately, we would also want to investigate how this work integrates with the design iteration loop and use the results from tests to inform the next design
- Future work for validating methodology: Gather information on what subject matter experts (SME) on test procedures, and generate a manual benchmark that reflects real industrial use

References

[1] INCOSE, *Systems Engineering Handbook*, 5th ed. Hoboken, NJ, USA: Wiley, 2023.
[2] European Union Aviation Safety Agency, *Certification Specifications for Small Rotorcraft (CS-27)*, Amendment 10, 2023.
[3] Z. Bai, B. Zhang, M. Song, and Z. Tian, "Rapid integrated design verification of vertical take-off and landing UAVs based on modified model-based systems engineering," *Drones*, vol. 8, no. 12, p. 755, 2024.
[4] J. J. Norheim, E. Rebentisch, D. Xiao, L. Draeger, A. Kerbrat, and O. L. de Weck, "Challenges in applying large language models to requirements engineering tasks," *Design Science*, vol. 10, p. e16, 2024.