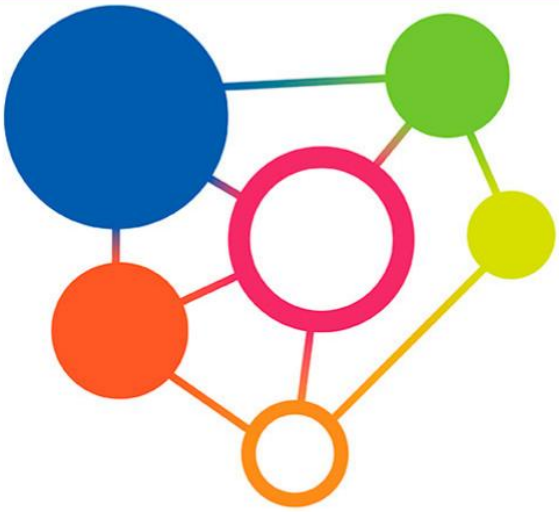

Engineering Professors COUNCIL
The voice of engineering academics

QUANSER
INNOVATE·EDUCATE

COMPLEX SYSTEMS

Teaching activity: LLM-driven interactive simulation for complex sociotechnical systems

Author: Dr. Stuart Grey, SFHEA (University of Glasgow).

Topic: Student created interactive simulation for complex sociotechnical systems.

Related INCOSE Competencies: Toolkit resources are designed to be applicable to any engineering discipline, but educators might find it useful to understand their alignment to competencies outlined by the International Council on Systems Engineering (INCOSE). The INCOSE Competency Framework provides a set of 37 competencies for Systems Engineering within a tailorable framework that provides guidance for practitioners and stakeholders to identify knowledge, skills, abilities and behaviours crucial to Systems Engineering effectiveness. A free spreadsheet version of the framework can be downloaded. This resource relates to the **Systems Thinking, Life Cycle, Configuration Management, Requirements Definition, Verification,** and **Validation** INCOSE Competencies.

AHEP mapping: This resource addresses several of the themes from the UK's Accreditation of Higher Education Programmes fourth edition (AHEP4): **Analytical Tools and Techniques** (critical to the ability to model and solve problems), and **Integrated / Systems Approach** (essential to the solution of broadly-defined problems). In addition, this resource addresses AHEP themes of **Ethics** and **Communication.**

Education level: Intermediate.

Learners have the opportunity to:

- Design and run a text based, turn based simulation that operationalises systems thinking (stakeholders, boundaries, feedback loops, delays, uncertainty and emergence).
- Debug their simulation through playtesting, documenting issue → fix → retest cycles and demonstrating how changes improve coherence.
- Explore trade-offs and justify decisions in ethics (e.g. consequences and equity) and complex systems (e.g. resilience vs cost vs emissions).
- Evidence learning with transparent artefacts: initial prompt, changes via tracked changes or before/after snippets, tester feedback, and final prompt.
- Reflect critically on validity, bias and the limitations of LLMs as simulators, including how to handle unsafe/poor choices by surfacing realistic consequences.
- Communicate findings clearly to technical and nontechnical audiences.

Teachers have the opportunity to:

- Use this as either a studio activity (3–5 sessions) or a compact assessment only task (1–2 sessions), with clear rubrics for each.
- Standardise scope by offering a predefined scenario (e.g., Urban Heatwave Response, UK city), or permit student proposed topics.
- Scale marking via artefact-based evidence (prompt, change log, feedback, final prompt) rather than long reports.
- Deliver with institutional Microsoft Copilot licences or any free web LLM; require students to disclose model and version used.
- Adapt quickly to different disciplines by swapping the scenario pack (microgrids, water networks, medical device supply chains, etc.).

Overview:

This resource enables engineering students to create, run, and debug a text-based, interactive simulation of a complex sociotechnical system using a Large Language Model (LLM). It is intentionally flexible and may be delivered as a multi-session studio activity (including assessment) or used solely as a compact assessment.

Purpose and use:

In both modes, students design a robust text prompt, test it with a user, document changes, and submit auditable artefacts that evidence learning. The key activity is interrogating their own thinking on how complex systems should be modelled by making judgements as to how their game does and does not capture the system dynamics.

Why and how:

The approach aims to give students hands-on experience in putting systems thinking into practice. Concepts such as stakeholders, feedback loops, delays, uncertainty, and emergent behaviour can be implemented and interrogated without heavy tooling.

The submission is a text LLM prompt with tracked changes, which allows students to demonstrate system design and debugging, produce transparent process evidence, and scale to large cohorts with minimal infrastructure.

Delivery options at a glance:

Audience	Undergraduate Years 2–4 and taught MSc, any engineering discipline
Modes	Studio activity (3–5×2 h + independent study) or Assessment-only (prompt-only; 1–2×2 h + 4–6 h)
Teams	3–4 students (solo permitted for assessment-only)
Assessment	Portfolio (studio) or prompt-plus-change-log (assessment-only)
Platforms	Institutional Copilot licences successful; encourage exploration of free tools (students record model/version)

Materials and software:

- LLM access: institutional Microsoft Copilot licences (proven) or any reputable free web-based tool. Students disclose the model and version.

Delivery modes:

Mode A — Studio activity (3–5 sessions)

- Session 1: Frame the system — boundary, stakeholders, conflicting goals; sketch a Causal Loop Diagram (CLD) with at least two reinforcing and two balancing loops.
- Session 2: Make it playable — define 4–8 state variables and KPIs; draft the prompt (based on Appendix A); specify commands, turn length and stop conditions; add debug controls (`trace`, `why`, `show variables`, `revert`).
- Between sessions: Prototype v1 — run 10–15 turns; capture a transcript; log defects (e.g. inconsistent updates, missing delays, moralising responses).
- Session 3: Play-test and iterate — exchange prototypes across teams or test with an external user; record issue → fix → re-test cycles with evidence (make sure edits are captured in tracked changes).
- Session 4: Present and reflect — short demo (6–8 turns); explain how feedback/delays manifest; discuss surprises and limits.

Mode B — Assessment-only (prompt-only; 1–2 sessions)

- Session 1: Brief and rapid scoping — select a scenario (student-chosen or predefined); write a one-paragraph boundary and stakeholders note; draft the initial prompt (based on Appendix A) with role choices, 4–6 state variables, simple commands, and a 12–15 turn cap.
- Independent work: Debugging loop — run the prompt; identify faults; edit the prompt (make sure edits are captured in tracked changes); re-run and capture short snippets demonstrating fixes; test with one peer and collect written feedback.
- Session 2: Submission — students submit a single document with the initial prompt, change log (before/after snippets), tester feedback, the final prompt, and a short rationale of innovative choices.

In both modes, module leaders may supply a predefined scenario(s) to standardise scope and simplify marking. A ready-to-use example is provided in Appendix C.

Assessment:

Studio portfolio — rubric (suggested weighting):

Criterion	A	B	C	D-E	%
Complexity modelling	Clear boundary; rich stakeholders; ≥4 correct loops; delays explicit; coherent KPIs	Mostly sound	Basic map	Superficial	25
Simulation design and prompt quality	Consistent state logic; visible feedbacks/delays; non-linearity; negative choices allowed with consequences; clear commands	Mostly coherent	Playable but brittle	Confusing/linear	25
Debugging evidence	Systematic play-tests; clear issue → fix → re-test artefacts	Some iteration	Minimal	None	20
Insight and reflection	Deep analysis of emergence, trade-offs, equity, uncertainty, and LLM limits	Good	Descriptive	Vague	20
Communication and referencing	Clear, concise, correct Harvard referencing	Minor issues	Adequate	Disorganised	10

Assessment-only (prompt-only) — compact rubric:

- Novelty and engagement (role-play/game elements; authentic decision-making) – 35%
- Ability to provoke ethical and/or complex systems trade-offs – 30%
- Debugging quality (change log with before/after snippets; tester feedback) – 25%
- Clarity of prompt and rationale; Harvard referencing where used – 10%

Scenario options:

Students may propose their own topic or the module leader may supply a predefined scenario. Options suited to UK engineering contexts include:

- Community microgrid integration — resilience vs affordability vs emissions vs public buy-in.
- Urban heatwave response — emergency measures vs long-term urban design; uneven impacts.
- Water network nitrates — compliance, farm livelihoods, treatment costs, ecological outcomes.
- Critical medical device supply chain — redundancy vs cost; equitable allocation.

Appendix A — Prompt template (simulation + debug-ready):

Title: Complex Systems Simulator — [Scenario]

Purpose: Run a turn-based interactive simulation of a complex sociotechnical system. Track named state variables, apply feedback and delays, and let the player's decisions drive non-linear outcomes.

Setup:

1) Offer three roles (distinct authority/constraints).

2) Introduce 3–5 NPCs with clear goals and plausible interventions.

3) Show a dashboard of [STATE_VARIABLES] each turn with short context.

State rules:

- Track only these variables (with units/ranges): [list 5–8].
- Maintain at least two feedback loops and one delay; keep hidden rule notes consistent across turns.
- Each turn: recap; propose 3–5 options (plus free-text); explain updates; show dashboard; request the following action.
- Time step: 5 minutes to 1 week; end after 20–30 turns or on stop conditions.

Commands: status, talk [npc], inspect [asset], implement [policy], pilot [intervention], advance time, review log.

Debug commands (for testing): trace on/off (print update logic), why (state which loops/delays drove the change), show variables (print current state table), revert (roll back one turn), reseed (slight exogenous shock).

Realism and ethics: Allow all plausible actions and report consequences neutrally. If unsafe in the real world, refuse, propose safer alternatives, and continue with plausible systemic effects.

LLM pitfalls to avoid: Do not invent new variables; ask clarifying questions rather than guessing; keep outputs concise; summarise trajectory every five turns.

Begin: Greet the player, state the scenario, ask for a role, and wait.

Appendix B — Debugging and play-test checklist:

Functional coherence

- Do state variables update consistently with declared logic?
- Are delays visible (policy today → trust gradually → adoption later)?
- Are reinforcing and balancing feedback identifiable in play?

Robustness

- Does the simulation permit negative choices with realistic consequences?
- Do trace/why explanations match outcomes?
- Are stop conditions respected?

User experience and clarity

- Are commands clear? Is turn pacing appropriate?
- Are dashboards concise and informative?

Report

- Provide three concrete defects with turn numbers, the prompt edits that fixed them, and evidence of the re-run.

Appendix C — Predefined scenario (Urban Heatwave Response, UK city):

Boundary: One UK local authority area during the July–August heatwave period. Focus on public health, energy demand, and community resilience.

Roles: (1) Local Authority Resilience Lead; (2) NHS Trust Capacity Manager; (3) Distribution Network Operator (DNO) Duty Engineer.

Stakeholders: Residents (with a focus on vulnerable groups), care homes, schools, SMEs, DNO, local NHS Trust, emergency services, voluntary/community groups, Met Office (for alerts), and local media.

State variables (examples): Heat-health alert level (0–4); Emergency Department occupancy (%); Electricity demand/capacity (% of peak); Indoor temperature exceedance hours (hrs > 27 °C); Public trust (0–100); Budget (£); Equity index (0–100).

Events/shocks: Red heat alert; substation fault; procurement delay; misinformation spike on social media; transport disruption; community centre cooling failure.

KPIs and stop conditions: Heat-related admissions; unserved energy; cost variance; equity gap across wards. Stop if alert level 4 persists >3 days, budget overspends >10%, or trust <25.

Notes for assessors: Using a standard, predefined scenario simplifies marking and ensures comparable complexity across teams, while still allowing for diverse strategies and outcomes.