systems

MDPI

Article

LLM-Powered, Expert-Refined Causal Loop Diagramming via Pipeline Algebra

Kirk Reinholtz [1], Kamran Eftekhari Shahroudi [1,*] and Svetlana Lawrence [1,2]

[1] Department of Systems Engineering, Colorado State University, Ft Collins, CO 80523, USA; kirk.reinholtz@colostate.edu (K.R.)
[2] Idaho National Laboratory, 1955 Fremont Ave., Idaho Falls, ID 83415, USA
* Correspondence: kamran.eftekhari_shahroudi@colostate.edu

Abstract

Building a causal-loop diagram (CLD) is central to system-dynamics modeling but demands domain insight, the mastery of CLD notation, and the ability to juggle AI, mathematical, and execution tools. Pipeline Algebra (PA) reduces that burden by treating each step—LLM prompting, symbolic or numeric computation, algorithmic transforms, and cloud execution—as a typed, idempotent operator in one algebraic expression. Operators are intrinsically idempotent (implemented through memoization), so every intermediate result is re-used verbatim, yielding bit-level reproducibility even when individual components are stochastic. Unlike DAG (directed acyclic graph) frameworks such as Airflow or Snakemake, which force analysts to wire heterogeneous APIs together with glue code, PA's compact notation lets them think in the problem space, rather than in workflow plumbing—echoing Iverson's dictum that "notation is a tool of thought." We demonstrated PA on a peer-reviewed study of novel-energy commercialization. Starting only from the article's abstract, an AI-extracted problem statement, and an AI-assisted web search, PA produced an initial CLD. A senior system-dynamics practitioner identified two shortcomings: missing best-practice patterns and lingering dependence on the problem statement. A one-hour rewrite that embedded best-practice rules, used iterative prompting, and removed the problem statement yielded a diagram that conformed to accepted conventions and better captured the system. The results suggest that earlier gaps were implementation artifacts, not flaws in PA's design; quantitative validation will be the subject of future work.

Keywords: system dynamics; causal loop diagram; artificial intelligence; large language model; orchestration; Pipeline Algebra

Academic Editors: Wayne Wakeland and Federico Barnabè

Received: 28 May 2025
Revised: 17 August 2025
Accepted: 19 August 2025
Published: 7 September 2025

Citation: Reinholtz, K.; Shahroudi, K.E.; Lawrence, S. LLM-Powered, Expert-Refined Causal Loop Diagramming via Pipeline Algebra. *Systems* **2025**, *13*, 784. https://doi.org/10.3390/systems13090784

1. Introduction

Humans are notoriously poor at reasoning about feedback-dominated, nonlinear phenomena, a limitation captured in the observation that "systems thinking is not a natural act" [1]. Linear, one-way causal narratives routinely obscure the reinforcing and balancing loops that govern real-world behavior [2]. We humans have proven adept at creating systems exhibiting such non-linear behavior. Unfortunately, the real-world behavior is frequently undesirable, but it is self-evident that we are not as adept at predicting and mitigating the undesired behaviors as we are at creating the systems in the first place. In perhaps one of the greatest of vicious loops, we are creating ever more and ever-more-dangerous systems.

Systems thinking (ST) [3–5] is a general term for attempting to comprehend the behavior of feedback-laden non-linear systems. But ST is itself hard. Tools have been developed to aid our attempts at ST, in particular, the qualitative causal loop diagram and its quantitative system dynamics (SD) relative, the stock-flow diagram (SFD). But until now, most of the tools mechanized the processes but did not much aid the intellectual modeling effort itself. The challenge is often further exacerbated by a general lack of training in the methods of ST and SD (see, e.g., [6]).

Causal loop diagrams (ch5 of [7]) elucidate the structure that leads to the feedback-driven behavior of the system and help identify leverage points [2,8,9]. Constructing a high-quality CLD, however, can be labor-intensive (see, e.g., [10]): an expert must review source material, define variables with well-formed polarity-aware names, identify and close impactful feedback loops, and justify every link. The required effort and skills in wrangling domain specific, mathematical, and general software tools consequently limits CLD and SFD uptake among engineers and policy analysts.

But transformer-based large language models (LLMs) can alter that cost profile. The architecture that [11] underpins frontier systems such as GPT-4 [12] can extract, correlate, and synthesize content at great scale. When an LLM is paired by its developer [13] with retrieval utilities, deterministic calculators, and task breakdown and orchestration logic, it becomes capable of exploiting its hardwired knowledge base, as well as searching the web and various databases and even writing code to assist itself in its reasoning. We refer to all of these as a "TALM" (tool-augmented language model) in this paper, which is intended to cover what is also called "Agentic AI", "Agentic LLM", "Composite AI", "Multi-tool CoT" (Chain of Thought), "ReAct Agents", "Toolformer", "LLM+", "agentic tool use", and others.

Growing empirical evidence [14–16] indicates that large-language-model architectures such as ChatGPT can autonomously abstract and integrate heterogeneous textual inputs into semi-formal, graph-based knowledge structures—most notably causal loop diagrams and Toulmin-style argument schemata—while achieving accuracy that approaches expert baselines.

Such a system can deliver, for our present purposes, two complementary artifacts: (i) a source-constrained CLD whose links are justified solely by the focal text, and (ii) a knowledge-augmented CLD that extends that skeleton with variables, drivers, and risks obtained from the LLM knowledge base and external (e.g., the public Web) sources.

Automating CLD generation involves multiple coupled stages: text extraction, causal-relation mining, polarity assignment, loop synthesis, diagram layout, knowledge augmentation, validation, and iteration with human domain experts. General-purpose orchestration frameworks such as TFX [17], Airflow, and Kubeflow connect these steps but rely on loosely specified interfaces that can hide data mismatches and are optimized for execution at scale, not epistemics or pedagogic clarity. We propose that an executable, mathematical style of workflow formalism that enforces type safety, records provenance, guarantees reproducibility, integrates AI, mathematical, and general algorithmics, and communicates ideas in epistemically transparent, rather than programmatic, terms will improve CLD and SD uptake.

This study introduces Pipeline Algebra (PA), a category-theoretic pipelined workflow notation in which every stage is a typed morphism with explicit domain and codomain [18]. PA is, in effect, statically and strongly typed. All PA operators are capable of idempotency: executing a stage with identical inputs can always yield identical outputs, even when the underlying TALM is stochastic. Figure 1 provides an overview of the PA-orchestrated workflow, highlighting the hand-off between deterministic operators, TALM services, and subject-matter-expert review. In the spirit of Iverson's dictum that "Notation

is thought" [19], a PA script functions simultaneously as formal human-readable documentation and executable specification, producing a deterministic audit trail suitable for replication and peer review [20].

The PA morphisms (presented graphically in Figure 1) were used to perform the work described in this paper, though the figure represents a more general set of operations used to analyze documents in terms of the causal relationships described within the documents, and so some of the operations (e.g., CLD compare/contrast) were not used for this work. Each morphism "morphs" its inputs into its outputs in a formal manner that allows algebraic manipulation of the operations. This is why we refer to them as morphisms, rather than software-oriented functions. There are also domain-agnostic higher-order operators available (e.g., parallel operations such as map and comap and function compositions to apply morphisms in sequence)

Each morphism is specified by its input type(s), output type(s), implementation mechanism, and particular named morphism. For the work reported herein and as shown in Figure 1, mechanisms include conventional algorithms and computer math (e.g., scipy libraries), neural nets, chat LLMs, and even humans.

A workflow is constructed as a particular path through the available morphisms, or an algebraic manipulation of such a path (perhaps reordering or parallelizing the operations subject to the algebra).

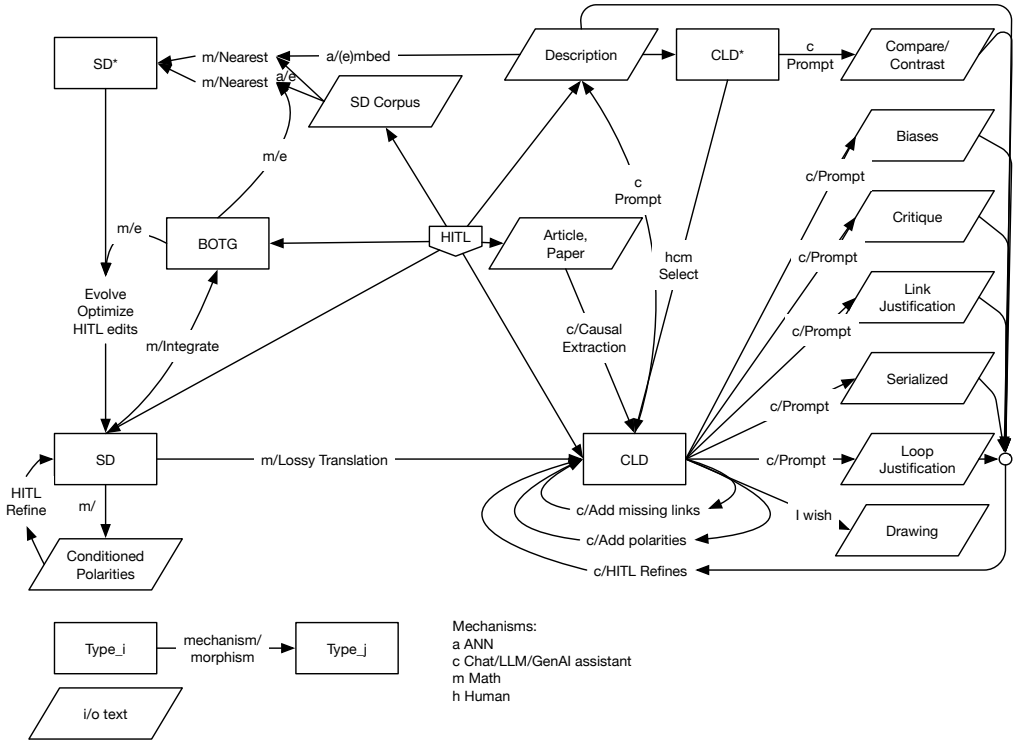

Figure 1. Pipeline Algebra, graphical workflow presentation form. BOTG: behavior over time graph. CLD: causal loop diagram. HITL: human in the loop. SD: system dynamics model (also known as a stock-flow diagram).

CLDs produced via the pipeline are stored as attributed directed graphs (ADGs). An ADG extends the directed graph by attaching formal typed attributes to vertices, edges, and the graph as a whole. The same structure can also encode the SFD constructs of a quantitative system dynamics (SD) model. Using one uniform representation, therefore, enables a future, automated evolution from qualitative CLD to fully parameterized SFD while preserving provenance at every step.

A subject-matter expert (SME) remains essential. The TALM supplies breadth, text mining, global knowledge recall, and the ability to connect distant facts, whereas the SME provides depth: contextual judgment, domain heuristics, and strategic intent. Effective diagram construction proceeds iteratively: the SME frames a prompt or partial diagram, the AI expands it, and the SME validates or refines the output and/or the pipeline itself. This division of labor concentrates expert effort on conceptual issues while delegating repetitive extraction and pattern-matching tasks to the machine.

This paper pursues four objectives. First, we formalize Pipeline Algebra and show how its idempotent, typed operators convert a tool ensemble including TALM, symbolic and pattern matching, numeric mathematical capabilities, and local AI and ordinary algorithms into a dependable collaborator. Second, we demonstrate practical utility by generating both source-constrained and knowledge-augmented CLDs for the subject energy-transition case study, achieving turnaround times measured in minutes, rather than hours or days. Third, we report an SME evaluation that identifies weaknesses in the CLD that highlight the need for expert evaluation and refinement, as well as the value of pipeline improvement based upon SME analysis. Finally, the refinements are applied to the PA pipeline, resulting in an improved CLD and a pipeline that will reflect the SME input into any future instantiations.

Collectively, these results support the thesis that PA-orchestrated TALM lowers the effort barrier to feedback-centric reasoning without displacing expert oversight, and they lay the groundwork for future automated evolution from CLD to quantitative SD modeling. Future work will provide quantitative accuracy benchmarks against reference diagrams, extend the CLD→SD transformation to full simulation, and test the pipeline across multiple domains and SME reviewers while developing automated variable-naming and polarity checks. These extensions are essential to establish the general validity and practical scalability of the approach.

In the sections that follow, we detail the orchestration framework, implemented in Pipeline Algebra, used to generate and evaluate AI-constructed causal loop diagrams. We begin by describing the formalism and its implementation before presenting results from applying the pipeline to our case study.

2. Materials and Methods

The work reported herein is both described in and performed through the execution of Pipeline Algebra. Section 2.1 provides an overview of PA, and Section 2.8 presents the PA formulation that performed the work we report herein.

2.1. Pipeline Algebra

Pipeline Algebra (PA) is a category-theoretic workflow language [18] whose atomic unit is the typed morphism: a pure function declared with an explicit domain and codomain. Any morphism may invoke a large-language model, run deterministic code, launch a cloud batch job, or delegate to a human, yet it must honor its type contract—given an input of type τ_1, it yields an output of type τ_2, and it must be a pure function from a computational perspective, hence leaving the global compute state untouched. There is a subtlety in this purity: a TALM conversation may take place across several interactions in a session, which obviously modifies the global state. We treat this formally as a monadic construct with the session key representing the monad state that is mutated across prompt/response pairs. Comprehensive morphisms are constructed by composing operators using higher-order functional programming operators [21] such as composition, map, and comap, and functional transformation operators such as symbolic differentiation. Higher-order functions are generally intended to be useful for any application of PA, while atomic operators are expected to be a mix of general and domain-specific capabilities.

PA uses a state-passing style [22] where each operator has a single argument (an associative array) that contains a state that must include the named input instances for the operator, and the return from the operator is the input state with the operator result added to the state. For example, in Equations (1) and (5), this is shown explicitly.

Some operators, such as Comap, create a list of states (one from each comapped function). For notational convenience, we assume that the function return values can be merged without issue and do so automatically within the Comap operator. A richer option is available to merge explicitly, but we do not use that or further describe that capability here.

We use three equivalent notations. The most formal is the executable equational form shown in the PA Cheat Sheet (Equation (1)), typically used when completeness is foremost. The graphical form in Figure 1 is more suited to higher-level explanation. The detailed graphical form based on an inline arrow notation, $F \xrightarrow[\text{Causal Extraction}]{} +G$, is suitable for short morphism chains or the discussion of a particular morphism.

– NOTE Notional: Details elided or simplified for clarity.

– Some operator names changed for clarity.

– Explicit state threading pattern is used.

– State in Keyword/Value list e.g., $\{c_1 \mapsto 3\}$

$$
\begin{aligned}
&1 \quad (f \circ g)(x) \equiv f(g(x)) \\
&2 \quad \text{Map}(f)\{a, b, c, \ldots\} \equiv \{f(a), f(b), f(c), \ldots\} \\
&3 \quad \text{Comap}(\{f, g\})(x) \equiv \{f(x), g(x)\} \\
&4 \quad \text{f}(\ldots, rv \mapsto \cdot) \equiv f(\ldots) \to \cdot \\
&5 \quad (\text{Plus}(a_1 \mapsto 2, a_2 \mapsto \text{Key}(c_1)) \to p_1)(\{c_1 \mapsto 3\}) \to \{c_1 \mapsto 3, p_1 \mapsto 5\} \\
&\quad F : A \to B \equiv A \underset{\text{F}}{\to} B \\
&\quad \text{– } \{c_1 \mapsto 3\} \text{ is input state. } \{c_1 \mapsto 3, p_1 \mapsto 5\} \text{ is output state.} \\
&6 \quad (f)(x_1, x_2) = \text{g}(\text{F} \mapsto \text{x}_1, \text{G} \mapsto \text{x}_2)
\end{aligned}
\tag{1}
$$

– Hungarian notation used to improve readability. It has no formal semantics:

$A \mapsto$ kvmap for state passing e.g., $\{c_1 \mapsto 3, p_1 \mapsto 5\}$

$F_{name} \mapsto$ File name

$G_{name} \mapsto$ ADG

$J_{name} \mapsto$ JSON string or struct

$L_{name} \mapsto$ LaTeX

$M_{name} \mapsto$ Markdown native chat rv

$P_{name} \mapsto$ prompt

$S_{name} \mapsto$ String. May also be J | L | M | P | T

$T_{name} \mapsto$ Template

We designed PA as a high-density, mathematically oriented notation, on the premise that doing so would facilitate an epistemic exploration of the premise that "Notation is thought" [19] and "Civilization advances by extending the number of important operations which we can perform without thinking of them." [23]. A PA script functions simultaneously as formal, human-readable epistemic documentation, the pedagogic exposition of a method, and instantiates to generate the reported results.

The formal, mathematical syntax and semantics of PA facilitate meta-manipulations. Meta-algorithmic methods [24] that may be applied to optimize for real-time cost, speed, or accuracy targets.

We speculate that the PA compositional forms are relatively amenable to TALM comprehension and manipulation because the LLM loss rewards first capture n-gram-like continuities (because they are the strongest, most ubiquitous statistical cues in text) and hierarchal queues appear far less often and matter only many tokens later, so the gradient signal that would teach stable recursion (e.g., expression trees) is weaker and noisier. If true, then PA may facilitate novel TALM-based meta-algorithmic methods. It is future work to validate our speculation.

Functions can be composed as seen in Equation (1) line 6. Functions behave as substitutions: Any instance of the left-hand side (LHS) is, in effect, replaced with the right-hand side (RHS) with the argument patterns replaced by their actual values. This can be used to increase notation density, but it provides no fundamentally new semantics.

PA defines denotational semantics; any runtime that respects the data-dependency DAG and memoization invariants is sound. This facilitates two important design choices: DAG execution [25,26] (the basis of several large scale open source and commercial workflow systems such as Apache Airflow and Amazon Sagemaker) and memoization [27]. DAG execution enables effective cloud-scale execution by identifying the minimal sequencing necessary to insure causality (i.e., "compile" PA into Airflow or Sagemaker), and memoization provides deterministic results in the face of stochastic operators (e.g., a TALM prompt response). Deterministic results greatly simplify the scientific verifiability [20] of computed results, and they can tactically improve performance by performing make-like [28] optimizations. It is the topic of future work to perform such optimizations.

DAG execution is simpler to implement when the DAG is static and can be determined syntactically. All PA constructs allow this. It sometimes complicates the syntax (e.g., template expansion requires that the interpolated variables are mapped to state variables syntactically (i.e., not looked up at runtime).

Categorical semantics achieve more than tidy notation. Let P be the category whose objects are PA data types and whose morphisms are the pure functions we admit, and let **Exec** be the category of executable DAG nodes under composition. We define a semantics functor,

$$[\![-]\!] : P \longrightarrow \textbf{Exec},$$

that maps each atomic operator to its runtime task and preserves identities and composition. Because $[\![-]\!]$ is functorial, any equational law proved in P—for example,

$$\mathrm{Map}\, f \circ \mathrm{Map}\, g = \mathrm{Map}\,(f \circ g),$$

—carries over to the compiled DAG, enabling correctness-preserving optimizations such as fusion and parallel re-ordering. For effective steps, we work in the Kleisli category $\mathrm{Kl}(S)$ of the state monad S, so operator purity is maintained at the semantic level even while session state threads occur through the computation.

2.2. Implementation

PA was implemented in Wolfram Mathematica Version 14.2 [29]. The control flow was managed from within Mathematica: external capabilities such as ChatGPT (ChatGPT-4, accessed March–May 2025) were accessed via synchronous calls to the service. Parallelism was implemented using local processes.

2.3. Templates

PA uses a simple template notation, e.g.,

$$\begin{aligned} &\text{ExpandTemplate}(\\ &\quad T \mapsto \text{“Hello, ‘whoever‘!”}, \\ &\quad args \mapsto \{whoever \mapsto \text{Key}(a1)\} \\ &)(\{a1 \mapsto \text{"world"}\}) \to P_{prompt} \end{aligned} \tag{2}$$

Equation (2) will dereference $a1$ from A, yielding the string “world”, which is assigned to the template variable “whoever”, which is interpolated into the template where the variable is found enclosed in single quotes, “‘”. The result, in the state S_{prompt}, is “Hello, world!”.

Expansion is iterated until a fixed point is achieved. Interpolated variables may, therefore, themselves contain interpolated variables. Loops are fatal.

2.4. Causal Loop Diagram (CLD)

A causal loop diagram [30], Figure 2, is a conceptual diagram made up of variables connected via arrows with polarity signs $(+/-)$ that indicate the direction and nature of causality, forming feedback loops that represent the structure of a dynamic system.

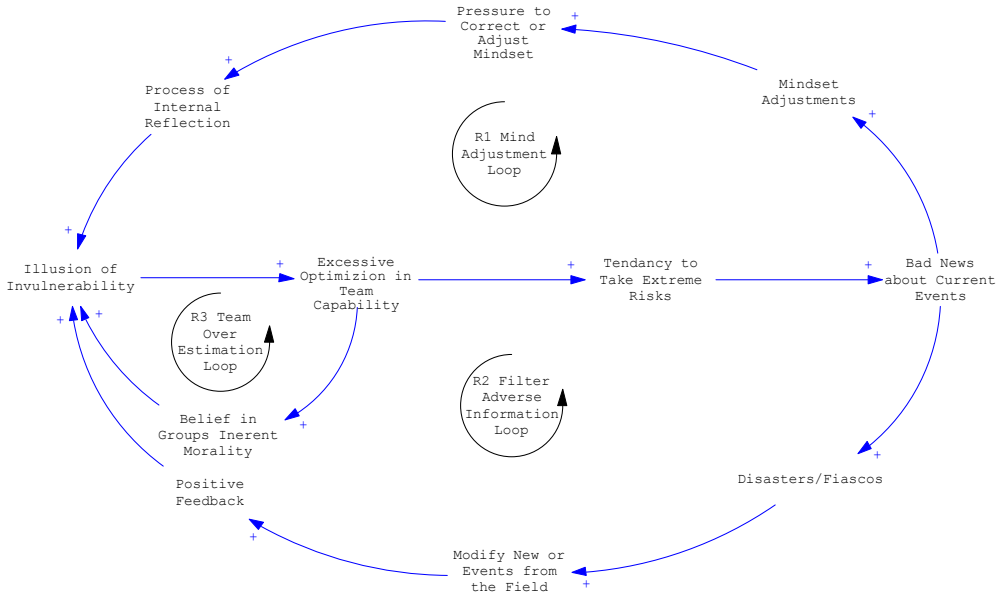

Figure 2. Notional causal loop diagram. The CLD is distinguished by the causal links with polarity markings $(+/-)$, labeled as reinforcing (R) or balancing (B) loops of interest (R1, R2, ...) and a rich feedback structure.

We represent a CLD formally as an attributed directed graph (ADG) (Section 2.5), where the variables are vertices and links are directed edges. In the case of CLD, the edge attributes certainly include the link polarity. We also use various augmented CLD representations, in which case the attribute might include an indication of link delay, magnitude of effect, and other link or vertex-specific attributes.

The semantics of the CLD are quite simple: an arrow from variable A to variable B with a polarity sign “$+$” or “$-$” indicates that, if the “value” of variable A is incrementally increased, the “value” of B will incrementally either increase (“$+$”) or decrease (“$-$”), ceteris paribus. “Increased” generally means “towards $+\infty$”, not “away from zero”. A loop is defined as (B)alancing if it has an odd number of “$-$” links because, ultimately, a disturbing

force will tend to be counteracted, or else it is (R)einforcing, and it will tend to amplify a disturbing force.

From a semantic standpoint, each variable in the diagram denotes an attribute or condition that can vary in magnitude (more on this restriction to scalar values later), and each link's polarity determines whether an incremental increase (or decrease) in one variable leads to a corresponding increase (or decrease) in another. Reinforcing loops (Rn) encapsulate processes that accelerate growth or decline by sending changes back through the system in a manner that compounds their initial effect. In contrast, balancing loops (Bn) are characterized by self-correcting tendencies, whereby shifts in one direction eventually produce counteractions that moderate or reverse the original trend. Delays add further complexity by introducing lags between cause and effect, an essential consideration for accurately interpreting feedback phenomena and anticipating system behavior.

Causal loop diagrams are limited because variables without a clear scalar algebraic meaning, such as those that aggregate their input, make polarity hard to define, which can blur the distinction between accumulation and direct causation and lead to misinterpretation. For example, while water flow rate into a bucket from a faucet might go up or down, the level of the water in the bucket will never go down [31]. This is the origin of the "bathtub paradox" [32].

Refs. [33,34] has good guidelines for manual CLD construction, as well as a number of behavioral archetypes. Kim has written a number of useful books on systems thinking and the use of the CLD.

2.5. Attributed Directed Graph (ADG)

In our Pipeline Algebra (PA) formalism, every model is carried in an *attributed directed graph* (ADG, Hungarian prefix *G*) (link: ADG Formal Definition). An ADG is a directed graph whose vertices, edges, and whole-graph object each possess a statically declared *attribute set*. Attribute types—real, string, Boolean, enumerated, or user-defined domain classes—are registered in the same type system that governs pipeline morphisms; their presence and type correctness can then be mechanically verified. The attribute schema is fixed by the formalism, not supplied ad hoc.

This design permits a single ADG instance to represent multiple modeling layers. For Causal-Loop Diagram Section 2.4, the attributes record variable names and link polarities; for a System-Dynamics SFD model, additional attributes hold equations, units, and parameter values. Because every attribute is formally specified, downstream operators—such as CLD→SD transformers or provenance checkers—can rely on guaranteed field semantics, rather than ad hoc tag parsing. The ADG serves chiefly as an in-memory representation, but it can be losslessly exported to, e.g., GraphML [35], should persistent storage or interchange be required.

The ADG is described formally and in the vernacular, as shown in Equation (3)

$$\begin{aligned}
&\text{Let } \mathcal{N} \text{ be a set of named attributes,}\\
&\mathcal{P}(\mathcal{N}) = \{S \mid S \subseteq \mathcal{N}\}\\
&\mathcal{V} \text{ be the set of vertices,}\\
&\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \text{ be the set of directed edges.}\\
&\text{An ADG is a 5-tuple } \mathcal{G} = (\mathcal{V}, \mathcal{E}, \phi_V, \phi_E, \phi_G),\\
&\phi_V : \mathcal{V} \to \mathcal{P}(\mathcal{N}),\\
&\phi_E : \mathcal{E} \to \mathcal{P}(\mathcal{N}),\\
&\phi_G : \{\bullet\} \to \mathcal{P}(\mathcal{N}),
\end{aligned} \tag{3}$$

where

ϕ_V maps each vertex $v \in \mathcal{V}$
to a subset of named attributes in $\mathcal{N}$,
ϕ_E maps each edge $e \in \mathcal{E}$
to a subset of named attributes in $\mathcal{N}$,
ϕ_G assigns a set of global attributes
to the graph as a whole (indicated by $\{\bullet\}$).

2.6. CLD Variable Naming Guidelines

1. Unambiguous variable phrasing. Name each variable so that an increase (↑) or decrease (↓) is clear—e.g., *Inventory Level*, rather than *Inventory* [34] (Guideline table).
2. Polarity on every link. Mark each causal arrow, + or −, and confirm with the "if . . . then . . . " test [7], ch 5, Section 2.7.
3. Close the feedback loops. Convert open chains into closed loops to give the diagram dynamic meaning [36].
4. Show time delays. Denote delayed effects explicitly so that slow feedback is not mistaken for instantaneous response [37,38].
5. Separate stocks from flows. Depict accumulations and rates as distinct nodes; conflating them yields the "bathtub" reasoning errors documented by [32]. See also [34].
6. Avoid "starburst" nodes. More than about four incoming or outgoing arrows often signals an aggregate that should be decomposed.
7. Flag exogenous drivers. Identify variables outside the boundary so the endogenous structure is unambiguous.
8. Label loops R or B and verify sign counts. Count negative links: with odd ⇒ balancing and even ⇒ reinforcing.
9. Check sign consistency around each loop. A mis-signed link flips an R loop to B or vice-versa [39].
10. Provide a causal narrative. A short prose explanation surfaces missing links or variables before simulation [40].
11. Use neutral, quantity-like names. Avoid evaluative labels that obscure polarity (e.g., *job stress*, not *stressed employees*) [34].
12. No bidirectional arrows. If causation runs both ways, insert an intermediate variable, rather than two opposing arrows [34].
13. Keep each variable conceptually homogeneous. Do not bundle disparate constructs in one node; split or rename as needed [41].
14. Eliminate duplicate or synonym variables. After drafting, run a *uniqueness check* and merge nodes that are merely different labels for the same concept [36].

We have also noted that a few more guidelines are suggested when interacting with AI:

- Glossary definitions. Supply a one-sentence definition for every variable because clear, localized semantics help the LLM disambiguate synonyms and anchor embeddings, improving automated vertex alignment and question-answering.
- Descriptive multi-word names. Use self-contained phrases such as *capacity factor*, rather than abbreviations like *CF*, because the richer surface forms boost embedding quality and reduce alias collisions in LLM-based matching.
- Edge-level provenance. Store a source sentence, citation, or SME note with every causal link because provenance metadata lets downstream reflection or explanation prompts trace reasoning paths and focus internal or external expert review where confidence is low.

- Loop narrative tags. Provide a plain-language, one-line description for each feedback loop because concise narratives help both humans and language models verify sign counts, detect missing links, and generate coherent summaries [40].

2.7. CLD Link Polarity Determination Rule

These rules are, for the most part, paraphrased from [7], ch5.

Applicability. This rule determines the polarity (+ or −) of a direct causal link from a source, Variable X, to a destination, Variable Y ($X \rightarrow Y$), in a causal loop Diagram (CLD).

Principle (ceteris paribus). To apply the test, consider only the direct influence of X on Y. Mentally change Variable X, and determine how Variable Y would respond *directly* to that change, assuming that *all other variables in the system remain momentarily constant* (*ceteris paribus*).

Test and Assignment

1. Consider a hypothetical *increase* in Variable X.
   - If this increase in X causes Variable Y to *increase*, the link $X \rightarrow Y$ is *positive (+)*.
   - If this increase in X causes Variable Y to *decrease*, the link $X \rightarrow Y$ is *negative (–)*.
2. (As a check, the opposite change in X must yield a consistent polarity): Consider a hypothetical *decrease* in Variable X.
   - If this decrease in X causes Variable Y to *decrease*, the link $X \rightarrow Y$ is *positive (+)*.
   - If this decrease in X causes Variable Y to *increase*, the link $X \rightarrow Y$ is *negative (–)*.

Conclusion. The link polarity is positive (+) if X and Y change in the *same* direction and negative (–) if X and Y change in *opposite* directions, based only on the direct, isolated impact of X on Y.

2.8. Pipeline Specification of Work Reported Herein

The PA morph that computed the results we report upon here is shown in Equations (4) and (5).

Human-in-the-loop (HITL) interactions are encoded in the same manner as TALM chat interactions: a function, Doprompt(), is given a model (e.g., "chatgpt-o3" or "sme") to which the prompt, as a string, is sent and from which the response, as a string, is received. The prompt itself is typically a template (Section 2.3) but may be a simple string literal.

A few other operators (e.g., ToPDF() may be HITL as well. The PA specification stands, no matter the method of implementation.

$$
\begin{aligned}
&(\\
&\quad \circ \text{ExportToVensim}(G \mapsto KeyG_{aicld1}, F \mapsto \text{"/path/to/aicld1.mdl"})\\
&\quad \circ \text{CloseSession}()\\
&\quad\quad \circ \text{Doprompt}(SID \mapsto \text{Key}(SID_{sme}), F \mapsto \text{"/path/to/aicld1.mdl"})\\
&\quad \circ \text{OpenSession}()\\
&\quad \circ \text{ExportToVensim}(G \mapsto G_{aicld1}, F \mapsto \text{"/path/to/aicld1.mdl"})\\
&\quad \circ \text{CloseSession}(SID \mapsto SID_{mid1})\\
&\quad\quad \circ \text{AIMakesCLD}(SID \mapsto SID_{mid1} abstract \mapsto S_{abstract}, problemstmt \mapsto S_{problemstmt}) \to G_{aicld1}\\
&\quad\quad \circ \text{Comap}(\\
&\quad\quad\quad \text{Scrape}(\text{"abstract"}, F \mapsto Key(F_{article})) \to S_{abstract},\\
&\quad\quad\quad \text{SIDmid1 AI extract problem statement from file to Sproblemstmt}\\
&\quad\quad)\\
&\quad \circ \text{OpenSession}(MID \mapsto \text{Key}(S_{mid1})) \to SID_{mid1}\\
&\quad \circ \text{Comap}(\\
&\quad\quad \text{Set}(F_{article} \mapsto \text{"/path/to/smearticle.pdf"})\\
&\quad\quad \text{Set}(F_{smecld} \mapsto \text{"/path/to/smecld.mdl"})\\
&\quad\quad \text{Set}(S_{mid1} \mapsto \text{"chatgpt-o1"})\\
&\quad\quad \text{Set}(S_{mid2} \mapsto \text{"chatgpt-o3"})\\
&\quad\quad \text{Set}(S_{sme} \mapsto \text{"human-sme"})\\
&\quad\quad \text{Set}(S_{author} \mapsto \text{"human-author"})\\
&\quad\quad \text{Set}(S_{jsoncldschema} \mapsto \text{"..."})\\
&\quad)\\
&)(\{\})
\end{aligned}
\tag{4}
$$

Equation (4) creates the AI CLD, given the reference article abstract and AI-extracted problem statement.

$$
\begin{aligned}
&(\\
&\quad \text{Comap}(\\
&\quad\quad \text{ToPDF}(F_{vensim} \mapsto \text{"/path/to/aicld1.mdl"}, F_{PDF} \mapsto \text{"/path/to/aicld1.pdf"}),\\
&\quad\quad \text{ToPDF}(F_{vensim} \mapsto \text{"/path/to/aicld2.mdl"}, F_{PDF} \mapsto \text{"/path/to/aicld2.pdf"})\\
&\quad)\\
&\quad \circ \text{Comap}(\\
&\quad\quad \circ \text{ExportToVensim}(G \mapsto \text{Key}(G_{aicld1}), F \mapsto \text{"/path/to/aicld1.mdl"}),\\
&\quad\quad \circ \text{ExportToVensim}(G \mapsto \text{Key}(G_{aicld2}), F \mapsto \text{"/path/to/aicld2.mdl"})\\
&\quad)\\
&\quad \circ \text{JSONtoADG}(J \mapsto \text{Key}(J_{aicld2})) \to G_{aicld2}\\
&\quad \circ \text{CloseSession}(SID \mapsto SID_{mid2})\\
&\quad\quad \circ \text{Doprompt}(SID \mapsto \text{Key}(SID_{mid2}), \ldots, T \mapsto \ldots \text{export CLD in JSON}) \to J_{aicld2}\\
&\quad\quad \circ \text{Doprompt}(SID \mapsto \text{Key}(SID_{mid2}), \ldots, T \mapsto \ldots \text{for review})\\
&\quad\quad \circ \text{Doprompt}(SID \mapsto \text{Key}(SID_{mid2}), \ldots, T \mapsto \ldots \text{for loops})\\
&\quad\quad \circ \text{Doprompt}(SID \mapsto \text{Key}(SID_{mid2}), \ldots, T \mapsto \ldots \text{for links})\\
&\quad\quad \circ \text{Doprompt}(\\
&\quad\quad\quad SID \mapsto \text{Key}(SID_{mid2}),\\
&\quad\quad\quad args \mapsto \{abstract \mapsto \text{Key}(S_{abstract}), cldguidelines \mapsto \text{Key}(S_{cldguidelines}))\}\\
&\quad\quad\quad T \mapsto \ldots \text{for variables} \ldots\\
&\quad\quad)\\
&\quad \circ \text{OpenSession}(MID \mapsto \text{Key}(S_{mid2})) \to SID_{mid2}\\
&\quad \circ \text{CloseSession}(SID \mapsto \text{Key}(SID_{midauthor}))\\
&\quad\quad \circ \text{Doprompt}(SID \mapsto \text{Key}(SID_{midauthor}), \ldots) \to S_{cldguidelines}\\
&\quad \circ \text{OpenSession}(MID \mapsto \text{Key}(S_{midauthor})) \to SID_{midauthor}\\
&) \circ (Equation(4))
\end{aligned}
\tag{5}
$$

Equation (5), derived during the execution of this work based upon SME feedback [42], derives an AI-generated output, given the abstract of the reference article and guidelines on creating a well-formed CLD.

3. Results

This section presents the outputs of our pipeline and their evaluation, moving from the initial AI-generated causal loop diagram to expert critique, the categorization of issues, and subsequent pipeline refinements. The subsections are organized to follow the flow of the experiment, from generation through assessment and iteration.

3.1. AI-Generated CLD

We next assess these outputs against SME expertise to identify strengths and weaknesses in the AI-generated structure (Figure 3) relative to the SME-generated CLD shown in Figure 4.

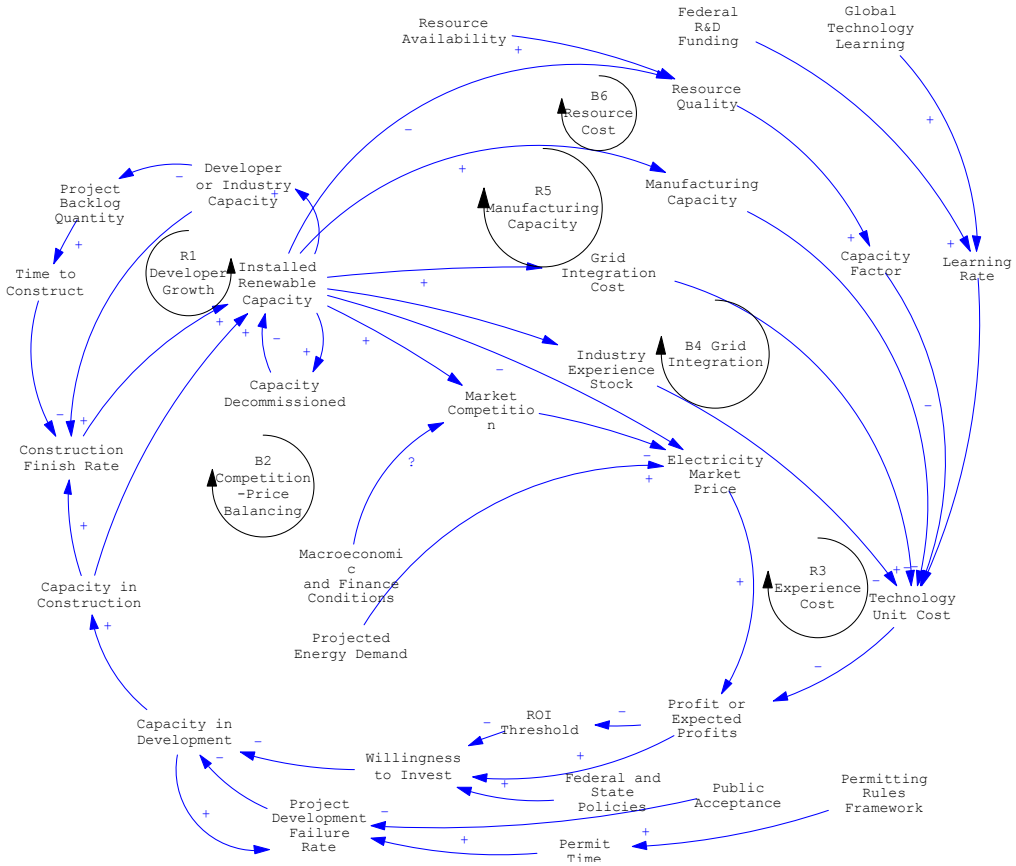

Figure 3. AI-generated CLD showing dynamics of deployment of novel energy systems. CLD derived from abstract and AI-derived problem statement of [43] using PA pipeline shown in Equation (4). This CLD is critiqued using the SME in Section 3.2. A variable glossary is presented in Appendix A.1.

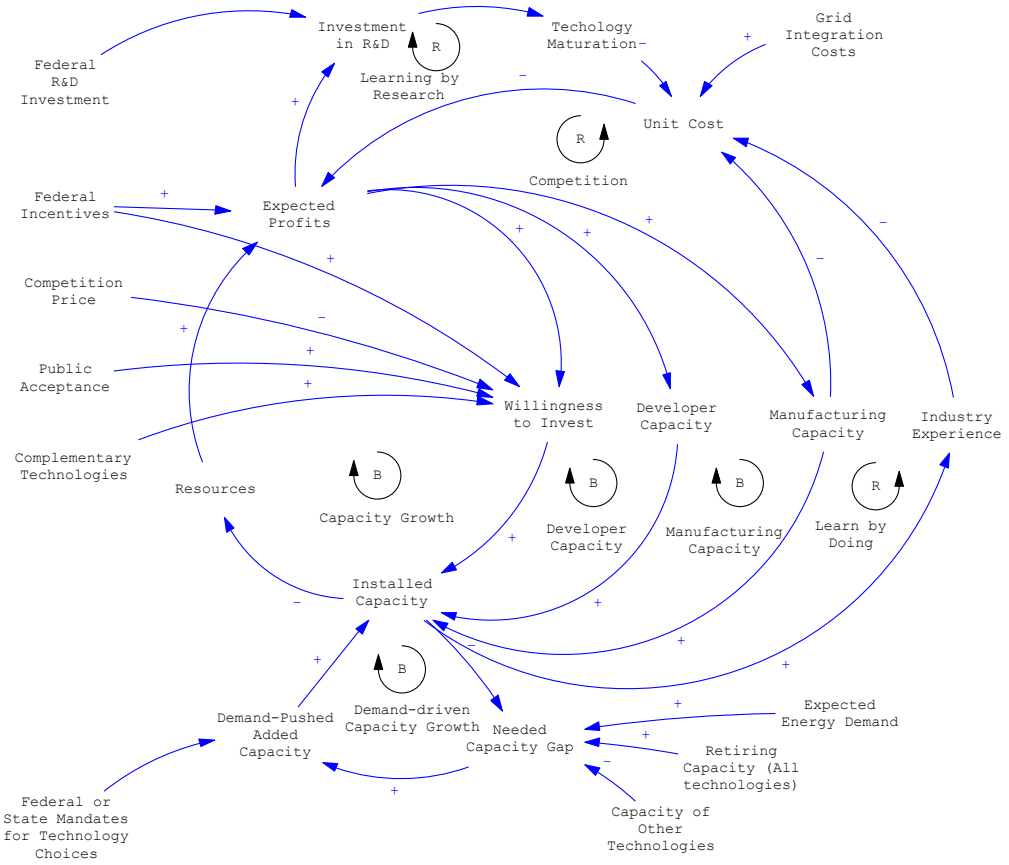

Figure 4. SME-generated CLD showing dynamics of deployment of novel energy systems. This is the reference CLD for our work from Figure 3 of [43]. Variables defined in [43].

3.2. SME Critique of AI-Generated CLD

The SME evaluated the AI-generated CLD Section 3.1, with the results shown below. The original SME comment may have been lightly edited. A classification of the errors reported via the SME is shown in the classification of SME-assessed CLD issues section, Section 3.3.

Note that these are quoted from the SME.

- Global technology learning—this is too vague. What does this mean? See the comment for learning rate, it applies to this variable as well.
- Learning rate—this is too vague. Learning rates are applied to capital expenses, O&M expenses, and capacity factor. In other words, by improving the technology, we make it cheaper to make, cheaper to maintain, and we extract more power (capacity factor).
- Macroeconomic and finance conditions—this is an odd variables because it can mean anything. The ? instead of +/− shows just that—it is unclear how financial condition will affect the market competition.
- Projected energy demand—projected demand does not increase the market price; the market price is the "today" measure, “projected demand” is the possible future measure. The projected demand affects the willingness to invest though.
- Profit or expected profit—two profits in the name of this variable. Odd that AI labeled it this way, just a note.
- ROI threshold—return on investment threshold is a predetermined desired value, e.g., min 10%. There is already a dependency between the expected

profit and willingness to invest, no reason to have another dependency on ROI threshold.

- Federal and state policies—this is too vague. Policies could push willingness to invest both ways. EPA is a policy, and stricter EPA rules will result in greater portion of declined permit approvals and reduced willingness to invest while tax incentives increase willingness to invest.
- Permitting rules framework—the permitting rules framework affects the project permit rate which in turn affect the project development failure rate. The permitting rules don't affect permit time; the number of applications does. Lastly, the permit time does not directly affect the project development failure rate. It just takes longer to get from "development" to "construction".
- The SME also noted that significant knowledge is required in order to construct a prompt and to validate the AI response.

3.3. Categorization of SME Critiques

We categorized the SME-identified errors/oddities as shown in Table 1. The categories were derived based on the SME analysis, not a priori.

The categorized errors shown in Table 2 suggest two systemic errors: the AI did not perform well in giving CLD variables well-formed names, and the AI did not perform well in relating link polarities with the semantics implied by the variable name. The pipeline was subsequently improved as shown in Section 3.4.

In response to the categorized SME critique, we added both CLD Link Naming Guidelines Section 2.6 and a CLD Link Polarity Determination Rule Section 2.7 to the CLD formulation prompt Equation (5). The resulting CLD is shown in Figure 5. We did not further analyze the results: that is the topic of future work.

Table 1. Categories for CLD flaws.

Code	Category (Focus)	Typical Symptom
A	Ambiguous/overly broad variable	Name fails to convey a single, well-bounded concept or its sign.
B	Duplicate/syntactically odd label	Internal redundancy or repeated wording in the variable name.
C	Mis-specified causal linkage or polarity	Link direction or sign is inconsistent with domain logic.
D	Redundant/unnecessary element	Adds no new information; duplicates an existing construct.
E	Process-level limitation	The issue lies in the human-AI workflow, not in the diagram content.

Table 2. SME-identified issues mapped to category codes.

Variable	Code(s)	Rationale
Global technology learning	a	Label too vague; concept unclear
learning rate	A	Same vagueness; unspecified which rates apply.
Macroeconomic and finance conditions	A, C	Broad label; polarity unknown ("?").
Projected energy demand	C	Incorrect causal path to market price.
Profit or expected profit	B	Redundant wording inside the label
ROI threshold	D	Adds a second dependency already implied by expected profit
Federal and state policies	A	Over-general; can push willingness both ways
Permitting rules framework	C,D	Mis-states drivers of permit time and failure; adds extra link
Prompting/validation effort	E	Workflow limitation, rather than a diagram variable

3.4. Pipeline Updates in Response to SME Critique

The SME determined that the AI did not perform well in giving CLD variables well-formed names, and the AI did not perform well in relating link polarities with the semantics implied by the variable name. In response to the categorized SME critique, we added both the CLD Link Naming Guidelines (Section 2.6) and a CLD Link Polarity Determination Rule (Section 2.7) to the CLD formulation prompt. The pipeline specification section, Section 2.8, was extended to include the second round of prompting with the improved prompt so that both the phase I and phase II CLDs can be recreated from the pipeline. The resulting CLD, not further analyzed in this article, is shown in Figure 5.

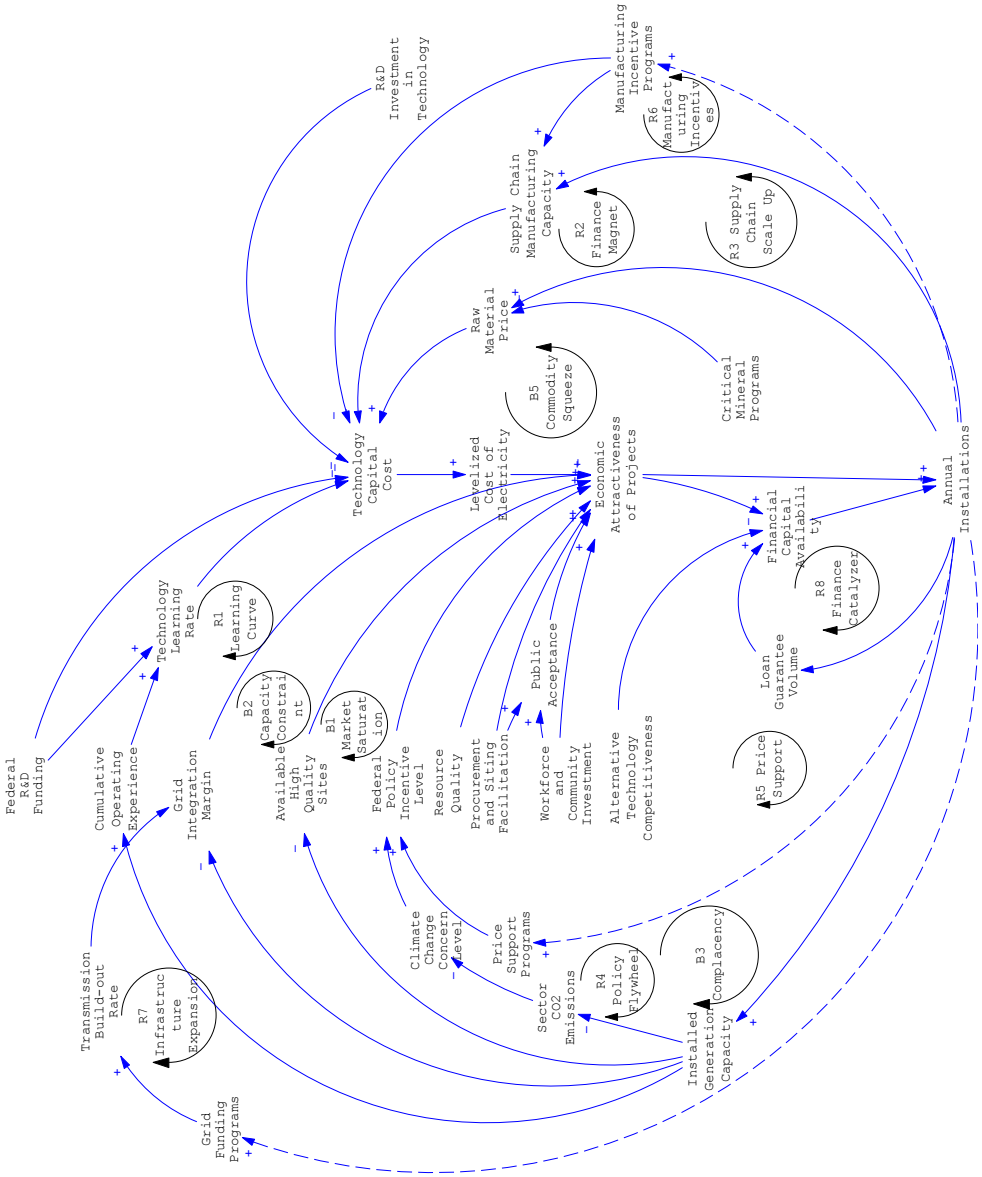

Figure 5. AI-generated CLD showing dynamics of deployment of novel energy systems. CLD derived in response to Section 3.2 from abstract of [43] and CLD guidelines in Section 2.6 using PA pipeline shown in Equation (5) onto Equation (4). Associated AI-generated CLD variable glossary in Appendix A.1. Dashed lines are implied by loop definitions in Appendix A.3 but not defined in Appendix A.2. "B4" is missing because AI went from B3 to B5 in Appendix A.3. All loops as identified by AI in Appendix A.3 but added manually to figure.

4. Discussion

The SME Assessment of AI-generated CLD Section 3.1, especially when distilled into a classification of SME-assessed CLD issues Section 3.3, strongly suggested two weaknesses in our first experiment: (1) we failed to include rules for a well-formed CLD in our prompt; and (2), by depending on the extracted problem statement, we tacitly used much of the SME labor as input to the algorithms, which we claim reduces the need for that labor.

We substantiated these implementation deficiencies and treated them as process feedback. Our response to this feedback was to achieve the following: (1) develop the CLD Authoring Guidelines (Section 2.6) with an eye towards utility for both human and AI use; (2) add them to the persona prompt; (3) use iterative prompting to increase probability that sufficient TPUs are allocated to our experiment; and (4) provide only the abstract of [43] for task definition, rather than the full Extracted Problem Statement.

Our preliminary results, after first prompting for AI-generated variable glossary, then for AI-generated links info, then for AI-generated loop info, and only then for the AI-generated CLD, suggest that the improved prompting based on SME feedback substantially improved the quality of the resultant CLD; see Figure 5. It is the subject of future work to formally analyze the CLD and further refine the feedback processes as part of the experiment PA setup, rather than external to the PA, as conducted done herein.

In addition, we prompted for AI-generated standard business loops embedded in the CLD Appendix A.3, which, if future analysis indicates the loops are well formed and significant, would provide further support for our claim that properly orchestrated AI can significantly aid in ST and SD, as well as specifically highlight the capability of the TALM to comprehend the structure and behavior of a CLD. The results strongly suggest this is worthy of further exploration and suggestive of the potential of explicitly assembling CLD's (or perhaps even SD models) from well-known molecules that embody the standard business loop behaviors on the one hand or mapping observed behaviors into standard business loops from which the model is inferred on the other hand.

While these findings highlight promising directions, they arise within a specific experimental framing that constrains their generality, as discussed in the following section on limitations.

5. Limitations

While these findings are encouraging, they should be interpreted in light of several important limitations in the experimental design.

Our principal limitation is a matter of framing. The study treated the LLM as a classical tool applied post hoc to a human-defined and human-assessed problem statement. This assumption shaped the evaluation protocol, prompt design, and role of the subject-matter expert (SME), and thereby constrained what could be learned about the model's independent framing capacity, its robustness, and its potential for generative novelty. In effect, the methodology is optimized for fidelity to the authors existing human conceptualization of how this activity should be performed, rather than for discovery, divergence, or triangulated validity.

5.1. Structural Consequences of This Framing

This tool-centric design led to several structural limitations:

1. Single-rater, unblinded evaluation. Quality was judged by a single SME without blinding or inter-rater reliability [44], leaving the results vulnerable to anchoring and confirmation effects.

2. Post hoc rubric construction. Error categories were derived after observing the outputs, rather than being specified a priori, increasing researchers' degrees of freedom and increasing vulnerability to experimenter bias.
3. Lack of quantitative, comparative metrics. We did not predefine or report task-grounded metrics (e.g., variable/edge/loop recovery, sign and delay accuracy, and agreement with a ground truth); nor did we conduct statistical comparisons against baselines or ablations.
4. Insufficient control of stochasticity and drift. We did not perform replicate generations under fixed hyperparameters (e.g., temperature, seed) or formally analyze variance across runs/models/time, limiting claims about stability and sensitivity. Reproducibility [20] is suspect, though it is inherent to contemporary AI systems that the bit-for-bit full reproducibility one might wish for is not plausible [45].
5. Adaptive tuning without hold-out re-evaluation. Pipeline updates were informed via an SME critique on the same case, but improvements were not re-assessed on held-out problems, inviting overfitting to evaluator preferences.
6. Narrow scope and inputs. Evaluation centered on a small number of cases and abstract-level inputs, which restricts external validity and overlooks information present in full texts or diverse domains.
7. Subjective outcome emphasis. Findings rely primarily on qualitative critique, rather than on executable tests (e.g., translation to stock–flow models and reproduction of stylized behaviors) that would tie structure to dynamics.

5.2. Why the Framing Matters

These limitations arise naturally from the subordinate, *post hoc* role in which the LLM was placed:

- The LLM operated within a human-preframed conceptual space, so evaluation emphasized *conformity* to that frame, rather than capacity for independent problem framing or justification.
- Outputs were judged against human-established representations, biasing assessment toward rubric-like fidelity, rather than internal coherence, novelty, or transfer.
- The overall design resembled classic "tool validation" (single case, single rater), leaving little incentive to incorporate replication, exploration of alternative framings, or cross-domain generalization.

The limitations outlined above stem directly from the subordinate, post hoc role in which the LLM was placed.

5.3. What a Different Framing Would Enable

Had we instead approached the LLM as an *alien intelligence*, i.e., as an agent capable of its own conceptual framing, the design could have addressed many of the above issues:

- Independent problem framing. Permit the model to extract and synthesize its own problem statement from raw sources (and multiple sources), with provenance logging and a priori evaluation criteria.
- Parallel, controlled replications. Run multiple independent instantiations (and, where relevant, models/personas) under (if possible) fixed hyperparameters and seeds to measure variance and convergence.
- Multi-criteria evaluation. Complement fidelity-to-human metrics with (i) internal dynamic coherence (e.g., translation to stock–flow and simulation checks), (ii) novelty/coverage relative to human CLDs, and (iii) transfer to related domains.

- Co-evolution with audit trails. Structure human–AI critique–revise cycles with full edit logs to quantify contributions, learning effects, and the marginal value of guidance artifacts.
- Baselines and ablations. Compare human-only, AI-only, and hybrid pipelines; ablate orchestration components (e.g., guidelines, session resets, HITL) to attribute effects and estimate effect sizes.
- Full reproducibility instrumentation. Release prompts, artifacts, and code; measure HITL time/cost; and pin model/version settings to mitigate drift and facilitate independent replication.

In sum, the present study should be interpreted as an initial, tool-framed exploration of LLM-assisted CLD construction. Reframing future experiments toward *AI-as-alien-intelligence* would allow us to test not only the model's ability to reproduce human-defined structures but also its capacity to generate novel, coherent causal representations that augment or challenge human conceptualizations, with stronger claims about validity, robustness, and generality.

Moreover, this reframing positions the work within the broader study of the behavior of intelligences—human or artificial—allowing future experiments to draw upon established methods and validity frameworks from the behavioral and cognitive sciences, human–computer interaction, and related fields.

These considerations inform our concluding assessment of the present work's contributions and highlight the most promising paths forward.

6. Conclusions

Our work has established the Causal Loop Diagram as a representational formalism, well suited to bridging human-centric qualitative reasoning and machine-driven manipulation via large language models (LLMs). For human stakeholders, its visual, graph-based syntax offers an intuitive method for mapping the feedback structures and interdependencies that govern complex systems, thereby facilitating a shared understanding. Concurrently, its formal structure, readily encoded as an attributed directed graph (ADG), renders it computationally tractable for an LLM. The diagram's semantics, defined by simple, polarity-based causal rules, proved sufficiently constrained for an AI to parse, critique, and transform. The CLD's ability to serve as an interface between human interpretation and algorithmic processing is a key finding of this research, positioning it as an enabler of the AI-accelerated workflows that can lower the barrier to entry for complex systems analysis.

Pipeline Algebra was presented as a framework that combines the mathematical rigor of category theory with the practical needs of modern, AI-driven data science. It is designed to make complex analytical workflows more robust, reproducible, scalable, and transparent by offering several key advantages:

- Formal rigor and reproducibility: PA provides formal rigor and reproducibility through the use of typed and auditable morphisms ($f : A \to B$) that prevent data mismatches, deterministic replay via memoization to ensure identical outputs even with stochastic components like LLMs, and complete provenance from its state-passing discipline, which creates a transparent audit trail.
- Conciseness and epistemic clarity: the framework offers conciseness and epistemic clarity by functioning as a high-density "notation as thought" that forces analysts to focus on the structure of their investigation and by replacing verbose configuration files with a compact algebraic notation that is easier to grasp.

- Unification and flexibility: it achieves unification and flexibility by integrating multimodal components, including LLM calls, symbolic math, and human-in-the-loop edits, under a single formal structure.
- Advanced capabilities for AI workflows: PA enables advanced capabilities for AI workflows, including a "falsifiable prompting" protocol that treats LLM outputs as testable contracts to prevent hallucinations, a robust foundation for agentic AI to plan and verify its own pipelines, and support for meta-algorithmic control to dynamically optimize workflows for cost, speed, or accuracy.

A core advantage of Pipeline Algebra (PA) is its compositional flexibility, which allows a single set of fundamental morphisms (see Figure 1) to orchestrate multiple distinct workflows. The power lies not in creating bespoke tools for every task but in reusing and recombining a common set of operators. For instance, extracting a CLD from a document and deriving one from a problem statement utilize many of the same underlying morphisms, such as those for prompting an LLM or parsing a graph structure. The distinction between the two workflows is defined simply by how these morphisms are sequenced and what initial data they are given: a document's text, in one case, and a high-level problem statement, in the other. This modularity means that the same algebraic toolkit can be dynamically configured to support a range of analytical tasks, from data extraction to creative synthesis, without altering the core operators.

We also recognize that full validation of the pipeline's technical correctness will benefit from broader community engagement. Making the workflow available for replication and critique by a wide audience will, we hope, help surface any critical rejoinders, drive iterative refinement, and accelerate collective progress in applying AI to complex systems reasoning.

7. Future Work

We believe it would be useful to develop a comprehensive prompt for CLD creation that covers the complete process, perhaps along the lines of [7] ch5 and [36] ch2. This would keep the TALM focused on creating a comprehensive and well-formed CLD. Key points probably include the inputs to the TALM, including reference modes of interest, the SOI boundary, and the time horizon, as well as the features of a well-formed CLD.

Further case studies should be conducted (healthcare, supply chain, and ecology) to test domain versatility. Each study should involve multiple SMEs so that some notion of the statistical validity of the results can be formed. With that being said, the rapid rate of technology advancement is, in our opinion, likely to render obsolete detailed statistical results before they can be published and acted upon.

Comparative and criterion-based methods of CLD evaluation should be developed, perhaps based on reconciliation of variable names, and then a confusion matrix should be used to compare variables, links, and loops (after lexicographic rotation for normalization) [46].

A meta-algorithmic optimization of PA should be exercised, with particular attention paid to LLM-based approaches. We speculate that the linear compositional structure of PA will enhance the ability of the LLM to comprehend and manipulate the PA.

A primary direction for future work is the inversion of Pipeline Algebra's control flow, transitioning orchestration from a local engine to the agentic large language model (LLM) itself and moving into a "vibe coding" epoch of AI-assisted engineering. We believe that this paradigm shift is now practical due to recent advances that have transformed LLMs into agents capable of tool use and reflective planning. This evolution allows the LLM to act as the central dispatcher for a workflow, a role we had previously reserved for our bespoke software. The feasibility of this approach hinges on PA's compositional, non-nested structure, which serves as a formal language comprehensible to both humans and

AI. Unlike deeply nested code, a PA script's compositional form of shallow nesting of linear chains of morphisms can be collaboratively defined and manipulated within a chat-based interface, with the LLM interpreting the formal structure and delegating tasks to the most appropriate resource. For intensive workloads, this framework can utilize self-contained computational packages, essentially PA fragments, that bundle a PA operator with its inputs (or references to inputs) for remote execution. The LLM can then offload these entire PA fragments to distributed environments, such as cloud platforms or on-premises clusters, with a single, high-level command. This evolution would transform PA from a static library of operators into a dynamic, interactive framework where the LLM can reason about, schedule, and optimize complex analytical pipelines in real time, truly realizing the potential of human–AI collaboration in systems engineering. The PA could be iteratively vibe-coded, validated, inspected, and executed within such a framework.

Author Contributions: Conceptualization, K.R. and K.E.S.; methodology, K.R.; software, K.R.; validation, K.R. and S.L.; formal analysis, K.R. and S.L.; investigation, K.R.; data curation, K.R.; writing—original draft preparation, K.R.; writing—review and editing, K.R.; visualization, K.R.; supervision, K.E.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Phase II CLD Results

Appendix A.1. AI-Generated Variable Glossary

Table A1. AI-generated variable glossary for the wind-adoption causal loop diagram.

#	Variable Name	One-Sentence Definition	Role
		THIS TABLE IS AI-GENERATED	
1	Installed Wind Generation Capacity	Cumulative nameplate megawatts of grid-connected land-based wind turbines.	Stock
2	Annual Wind Installations	Additional wind-turbine megawatts that achieve commercial operation each year (flow into Installed Capacity).	Flow
3	Cumulative Operating Experience	Total turbine-years logged by the operating wind fleet, basis for learning-by-doing effects.	Stock
4	Wind Turbine Capital Cost	Overnight capital cost ($/kW) for a new wind project, including turbines, balance-of-station, and construction.	Intermediate
5	Levelized Cost of Wind Electricity	Present-value cost of delivering 1 MWh from a new wind project over its lifetime (LCOE).	Intermediate
6	Economic Attractiveness of Wind Projects	Composite metric (e.g., IRR gap vs. hurdle rate) indicating how appealing wind projects are to investors.	Intermediate
7	Federal Production Tax Credit Level	Statutory $/MWh incentive available to qualifying U.S. wind projects.	Exogenous policy lever
8	Available High-Quality Wind Sites	Fraction of land with adequate wind speeds, grid access, and low siting conflict that remains undeveloped.	Depletable resource
9	Grid Integration Margin	Remaining transmission capacity and ancillary-service headroom that can absorb additional wind generation without curtailment.	Capacity buffer
10	Supply-Chain Manufacturing Capacity	Annual gigawatts of wind-turbine equipment that the domestic/global supply chain can deliver.	Capacity buffer
11	Steel & Critical-Mineral Prices	Indexed cost of steel, copper, and rare-earth materials relevant to turbine production (exogenous driver).	Exogenous cost driver
12	Public Acceptance of Wind Farms	Indicator of social license--permitting success rates and community attitudes toward wind development.	Soft variable
13	Wind Resource Quality	Mean wind-power density at prospective sites (exogenous natural condition).	Exogenous natural
14	CO_2 Emissions from Electricity	Annual million tonnes of CO_2 emitted by the national power sector.	Stock
15	Climate-Change Concern Level	Salience of climate risks in public and policymaker discourse.	Soft driver
16	R&D Investment in Wind Technology	Annual dollars devoted to turbine, control-system, and O&M innovation for wind power.	Flow
17	Learning-by-Doing Rate	Empirical percentage cost decline observed for each doubling of cumulative operating experience.	Parameter
18	Financial Capital Availability	Volume of project-finance and tax-equity capital actively seeking wind deals.	Capacity buffer
19	Alternative-Technology Competitiveness	Relative attractiveness of solar PV, storage, and other low-carbon substitutes competing for investment.	Competitive pressure

Appendix A.2. AI-Generated Link Info

Table A2. Causal links, their polarity, delays, and provenance notes. A check mark in the *Delay?* column indicates an explicit time delay for the link.

THIS TABLE IS AI-GENERATED			
From → To	+/-	Delay?	Provenance / note
Annual Wind Installations → Installed Wind Generation Capacity	+	-	definitional
Installed Wind Generation Capacity → Cumulative Operating Experience	+	-	accounting
Cumulative Operating Experience → Learning-by-Doing Rate	+	✓ (years)	classic Wright curve
Learning-by-Doing Rate → Wind Turbine Capital Cost	−	✓	empirically 10-20% per doubling
Wind Turbine Capital Cost → Levelized Cost of Wind Electricity	+	-	LCOE formula
Levelized Cost of Wind Electricity → Economic Attractiveness of Wind Projects	−	-	lower LCOE boosts IRR
Federal Production Tax Credit Level → Economic Attractiveness of Wind Projects	+	-	policy incentive
Economic Attractiveness of Wind Projects → Annual Wind Installations	+	✓ (planning/construction lag)	standard investment pipeline
Installed Wind Generation Capacity → Available High-Quality Wind Sites	−	-	site depletion
Available High-Quality Wind Sites → Economic Attractiveness of Wind Projects	+	-	siting quality affects yield
Installed Wind Generation Capacity → Grid Integration Margin	−	✓	increasing curtailment
Grid Integration Margin → Economic Attractiveness of Wind Projects	+	-	curtailment lowers revenue
Supply-Chain Manufacturing Capacity → Wind Turbine Capital Cost	−	✓	economies of scale
Annual Wind Installations → Supply-Chain Manufacturing Capacity	+	✓	capacity expansion investment
Steel & Critical-Mineral Prices → Wind Turbine Capital Cost	+	-	materials pass-through
Public Acceptance of Wind Farms → Economic Attractiveness of Wind Projects	+	✓	permitting ease
Climate-Change Concern Level → Federal Production Tax Credit Level	+	✓	policy feedback
Installed Wind Generation Capacity → CO_2 Emissions from Electricity	−	✓	displacement of fossil
CO_2 Emissions from Electricity → Climate-Change Concern Level	−	✓	success can erode urgency (risk)
R&D Investment in Wind Technology → Wind Turbine Capital Cost	−	✓ (tech lag)	innovation
Economic Attractiveness of Wind Projects → Financial Capital Availability	+	✓	investors follow returns
Financial Capital Availability → Annual Wind Installations	+	✓	funding constraint
Alternative-Technology Competitiveness → Financial Capital Availability	−	✓	capital allocation

Appendix A.3. AI-Generated Standard Business Loops

Table A3. Feedback-loop catalog with sign verification. The odd–even negative-link rule is used.

THIS TABLE IS AI-GENERATED			
Loop ID	Type	Core causal path	Short narrative
R1 Learning Curve	R	Annual Installations → Cumulative Operating Experience + → Technology Learning Rate − → Technology Capital Cost + → LCOE − → Economic Attractiveness + → Annual Installations	Experience lowers cost, cost boosts demand, and demand adds more experience.
R2 Finance Magnet	R	Economic Attractiveness + → Financial Capital Availability + → Annual Installations → (lower costs & higher returns feed back to Attractiveness)	High returns draw bigger capital pools that fund still more projects.
R3 Supply-Chain Scale-Up	R	Annual Installations + → Supply-Chain Capacity − → Technology Capital Cost + → LCOE − → Economic Attractiveness + → Annual Installations	Factory scale spreads overhead and lowers cost, fuelling demand that keeps factories full.
R4 Policy Flywheel	R	Climate Concern + → Policy Incentive Level + → Economic Attractiveness + → Annual Installations − → Sector CO_2 Emissions − → Climate Concern	Public worry drives subsidies; rapid build cuts emissions, sustaining concern long enough for deep penetration.
R5 Price-Support	R	Price Support Programs + → Policy Incentive Level + → Economic Attractiveness + → Annual Installations → (success sustains Price Support)	Fiscal credits directly raise ROI and create political momentum to preserve the credits.
R6 Manufacturing Incentive	R	Manufacturing Incentives + → Supply-Chain Capacity − → Technology Capital Cost + → Economic Attractiveness + → Annual Installations → (utilisation validates further incentives)	Production rebates seed domestic plants; lower costs boost volumes that harvest more rebates.
R7 Infrastructure Expansion	R	Grid Funding + → Transmission Build-out Rate + → Grid Margin + → Economic Attractiveness + → Annual Installations → (stakeholders lobby for more Grid Funding)	Public infrastructure spend lifts capacity limits, enabling growth that justifies further spend.
R8 Finance Catalyser	R	Loan Guarantees + → Financial Capital Availability + → Annual Installations → (portfolio success underwrites new guarantees)	Risk transfer lowers borrowing cost; successful builds validate and expand guarantee programmes.
B1 Market Saturation	B	Annual Installations + → Installed Capacity − → Available High-Quality Sites + → Economic Attractiveness − → Annual Installations	Finite prime sites mean economics deteriorate as easy opportunities are used up.
B2 Capacity Constraint	B	Annual Installations + → Installed Capacity − → Grid Margin + → Economic Attractiveness − → Annual Installations	Shared-asset congestion or curtailment risk slows further builds until capacity is expanded.
B3 Complacency	B	Annual Installations − → Sector CO_2 Emissions − → Climate Concern − → Policy Incentive Level − → Economic Attractiveness − → Annual Installations	Early success reduces perceived urgency, eroding policy support and cooling growth.
B5 Commodity Squeeze	B	Annual Installations + → Raw Material Prices + → Technology Capital Cost + → LCOE − → Economic Attractiveness − → Annual Installations	Rapid scale-up tightens commodity supply, raising costs and damping demand unless countered by diversification programmes.

References

1. Valerdi, R.; Rouse, W.B. When systems thinking is not a natural act. In Proceedings of the 2010 IEEE International Systems Conference, San Diego, CA, USA, 5–8 April 2010; pp. 184–189. [CrossRef]
2. Forrester, J.W. Counterintuitive behavior of social systems. *Theory Decis.* **1971**, *2*, 109–140. [CrossRef]
3. Kim, D.H. *Introduction to Systems Thinking*; Pegasus Communications: Waltham, MA, USA, 1999.
4. Meadows, D.H.; Wright, D. *Thinking in Systems: A Primer*; Wright, D., Ed.; Chelsea Green Pub.: White River Junction, VA, USA, 2008.
5. Monat, J.P.; Gannon, T.F. What is systems thinking? A review of selected literature plus recommendations. *Am. J. Syst. Sci.* **2015**, *4*, 11–26.
6. Yan, H.; Wang, L.; Goh, J.; Shen, W.; Richardson, J.; Yan, X. Towards Understanding the Causal Relationships in Proliferating SD Education—A System Dynamics Group Modelling Approach in China. *Systems* **2023**, *11*, 361. [CrossRef]
7. Sterman, J. *Business Dynamics: Systems Thinking and Modeling for a Complex World*; Irwin/McGraw-Hill: Boston, MA, USA, 2000; p. xxvi, 982p.
8. Goodman, M.R. Elementary System Dynamics Structures. Ph.D. Thesis, MIT, Cambridge, MA, USA, 1972.
9. Sterman, J.D. Learning in and about complex systems. *Syst. Dyn. Rev.* **1994**, *10*, 291–330. [CrossRef]
10. Taha, H.; Durham, J.; Smith, C.; Reid, S. Qualitative Causal Loop Diagram: One Health Model Conceptualizing Brucellosis in Jordan. *Systems* **2023**, *11*, 422. [CrossRef]
11. Vaswani, A. Attention is all you need. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
12. OpenAi. *GPT-4 Technical Report*; OpenAI: San Francisco, CA, USA, 2023.
13. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv* **2022**, arXiv:2210.03629. [CrossRef]
14. Liu, N.Y.G.; Keith, D.R. Leveraging Large Language Models for Automated Causal Loop Diagram Generation: Enhancing System Dynamics Modeling through Curated Prompting Techniques. *arXiv* **2025**, arXiv:2503.21798. [CrossRef]
15. Arndt, H. AI and education: An investigation into the use of ChatGPT for systems thinking. *arXiv* **2023**, arXiv:2307.14206. [CrossRef]
16. Gupta, A.; Zuckerman, E.; O'Connor, B.T. Harnessing Toulmin's theory for zero-shot argument explication. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, Bangkok, Thailand, 11–16 August 2024.
17. Baylor, D.; Breck, E.; Cheng, H.T.; Fiedel, N.; Foo, C.Y.; Haque, Z.; Haykal, S.; Ispir, M.; Jain, V.; Koc, L. Tfx: A tensorflow-based production-scale machine learning platform. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NC, Canada, 13–17 August 2017; pp. 1387–1395.
18. Spivak, D.I. *Category Theory for the Sciences*; MIT Press: Cambridge, MA, USA, 2014.
19. Iverson, K.E. Notation as a tool of thought. *Commun. ACM* **1980**, *23*, 444–465. [CrossRef]
20. Peng, R.D. Reproducible Research in Computational Science. *Science* **2011**, *334*, 1226–1227. [CrossRef]
21. Hughes, J. Why Functional Programming Matters. *Comput. J.* **1989**, *32*, 98–107. [CrossRef]
22. Moggi, E. Notions of computation and monads. *Inf. Comput.* **1991**, *93*, 55–92. [CrossRef]
23. Whitehead, A.N. *An Introduction to Mathematics*; H. Holt and Company: New York, NY, USA, 1911; p. 256.
24. Simske, S.J. *Meta-Algorithmics: Patterns for Robust, Low Cost, High Quality Systems*; John Wiley & Sons: Hoboken, NJ, USA, 2013.
25. Lee, E.A.; Parks, T.M. Dataflow process networks. *Proc. IEEE* **1995**, *83*, 773–801. [CrossRef]
26. Mac Lane, S. *Categories for the Working Mathematician*, 2nd ed.; Graduate texts in Mathematics; Springer: New York, NY, USA, 1998; p. xii, 314p.
27. Michie, D. "Memo" Functions and Machine Learning. *Nature* **1968**, *218*, 19–22. [CrossRef]
28. Feldman, S.I. Make—A program for maintaining computer programs. *Softw. Pract. Exp.* **1979**, *9*, 255–265. [CrossRef]
29. Wolfram Research, Inc. *Wolfram Mathematica*, Version 14.2; Wolfram Research, Inc.: Champaign, IL, USA, 2023. Available online: https://www.wolfram.com/mathematica/ (accessed on 18 August 2025).
30. Goodman, M. *Causal Loop Diagramming (D-1755-2)*; Report; MIT: Cambridge, MA, USA, 1973.
31. Richardson, G.P. Problems with causal-loop diagrams. *Syst. Dyn. Rev.* **1986**, *2*, 158–170. [CrossRef]
32. Sweeney, L.B.; Sterman, J.D. Bathtub dynamics: Initial results of a systems thinking inventory. *Syst. Dyn. Rev.* **2000**, *16*, 249–286. [CrossRef]
33. Kim, D.H.; Senge, P.M. Putting systems thinking into practice. *Syst. Dyn. Rev.* **1994**, *10*, 277–290. [CrossRef]
34. Kim, D.H. *Systems Thinking Tools: A User's Reference Guide*; Pegasus Communications: Waltham, MA, USA, 1995.
35. Brandes, U.; Eiglsperger, M.; Lerner, J.; Pich, C. Graph Markup Language (GraphML). *Handbook of Graph Drawing and Visualization*; Chapman & Hall: London, UK, 2013.
36. Richardson, G.P.; Pugh, A.L. *Introduction to System Dynamics Modeling with DYNAMO*; MIT Press: Cambridge, MA, USA, 1981; Wright-Allen Series in System Dynamics; p. xi, 413p.

37. Forrester, J.W. *Industrial Dynamics*; MIT Press: Cambridge, MA, USA, 1961; 464p.
38. Lane, D.C. The emergence and use of diagramming in system dynamics: A critical account. *Syst. Res. Behav. Sci.* **2008**, *25*, 3–23. [CrossRef]
39. Goodman, M.R. *Study Notes in System Dynamics*; Wright-Allen Press: Cambridge, MA, USA, 1974; p. xiv, 388p.
40. Vennix, J.A.M.; Akkermans, H.A.; Rouwette, E.A.J.A. Group model-building to facilitate organizational change: An exploratory study. *Syst. Dyn. Rev.* **1996**, *12*, 39–58. [CrossRef]
41. Richardson, G.P. Problems in causal loop diagrams revisited. *Syst. Dyn. Rev.* **1997**, *13*, 247–252. [CrossRef]
42. Lawrence, S. RE: Lana's Paper [Personal communication]. E-mail, To: Reinholtz, K.; Colorado State University, Ft Collins, CO, USA, 1 April 2025.
43. Lawrence, S.; Herber, D.R.; Shahroudi, K.E. Leveraging System Dynamics to Predict the Commercialization Success of Emerging Energy Technologies: Lessons from Wind Energy. *Energies* **2025**, *18*, 2048. [CrossRef]
44. Gwet, K.L. *Handbook of Inter-Rater Reliability: The Definitive Guide to Measuring the Extent of Agreement Among Raters*; Advanced Analytics, LLC: Gaithersberg, MD, USA, 2014.
45. Heil, B.J.; Hoffman, M.M.; Markowetz, F.; Lee, S.I.; Greene, C.S.; Hicks, S.C. Reproducibility standards for machine learning in the life sciences. *Nat. Methods* **2021**, *18*, 1132–1135. [CrossRef] [PubMed]
46. Kohavi, R. Glossary of terms (incl Confusion Matrix - Kirk). *Mach. Learn.* **1998**, *30*, 271–274.