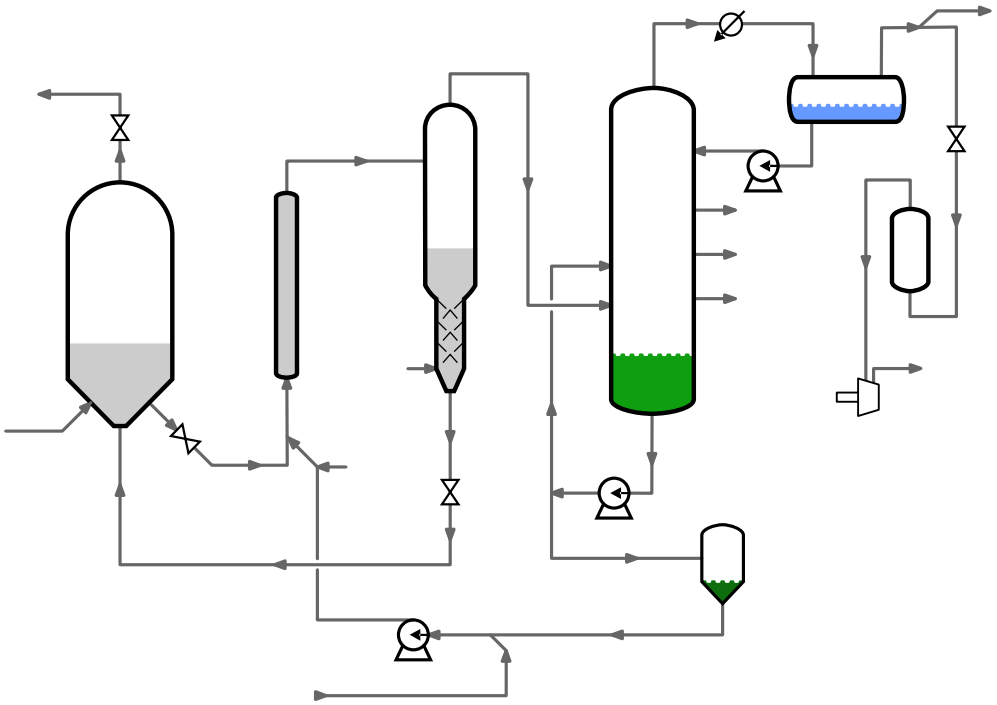
White Paper

Not Just Complicated: Why Digital Twins Struggle with Real-World System Behavior

By Sam Matic, P.Eng. — Director and Principal Consultant

aprocesr Ltd.

Asian Downstream Summit — October 2025



Executive Summary

Digital Twin (DT) initiatives across the process industries continue to grow in ambition and investment, yet their success rates remain uneven. Many projects deliver elegant models that fail to perform when confronted with the messy reality of plant operation. The reason is rarely the technology itself—it lies in a deeper mismatch between **how real industrial systems behave** and **how they are specified, modeled, and expected to behave**.

Most projects assume that the physical process is deterministic and therefore *fully knowable*: that with sufficient data, physics, and computation, a model can replicate reality under all conditions. In practice, however, once the model meets the operating plant, unmodeled interactions, nonlinear responses, and human interventions appear. These effects are **systemic rather than accidental**—they emerge from how the system is structured and controlled, not from any single fault.

This paper explores how **system complexity**, not complication, undermines Digital Twin realism and project Return on Investment (ROI). It introduces a practical framework for assessing and managing that complexity through **requirements engineering**, ensuring that expectations and model scope remain aligned with system behavior.

Because the word *complexity* carries different meanings in refining and management circles, it is important to clarify what it means here.

In this paper, *complexity* refers to **system behavior and interactions**, not to the **Nelson Complexity Index** or the **Cynefin framework**:

- The *Nelson Complexity Index (NCI)* measures **processing configuration and upgrading capability**—how many conversion steps a refinery has and what range of products it can make. It describes *process configuration*, not *system behavior*.
- The *Cynefin framework* classifies **decision environments** in organizations. It is a management sense-making tool, not a physical-system descriptor.
- *System complexity*, as used here, means the **degree of interdependence, feedback, and adaptation** within a process or control system—how strongly its parts interact to produce nonlinear or emergent behavior.

For example, a Crude Distillation Unit (CDU) may rank higher than an Fluid Catalytic Cracking Unit (FCCU) on Nelson's index, yet behave far more predictably. In contrast, the FCCU—though "simpler" in Nelson terms—exhibits nonlinear feedback, self-reinforcing dynamics, and emergent instability. This behavioral complexity, not configurational complexity, is what challenges Digital Twin fidelity.

Through examples from refinery operations, particularly **CDU vs. FCCU**, this paper illustrates why seemingly identical units behave differently and why Digital Twins often diverge from reality when confronted with such behavior. It proposes a structured approach based on **complexity assessment and requirements engineering** to identify these challenges early and set realistic modeling objectives.

Key messages:

- Complex systems cannot be understood through component analysis alone; their behavior emerges from interactions.
- Most Digital Twin project failures originate from unrealistic or underspecified requirements, not from modeling tools.
- A structured **complexity assessment** and **requirements-engineering process** can prevent those failures.
- Matching the right modeling strategy to the system's complexity leads to realistic expectations, lower cost, and higher ROI.

The concepts and approach presented in this paper draw from **complexity science**—the interdisciplinary study of nonlinear, adaptive systems—and translate those principles into a practical framework for engineering decision-making. This provides a scientific basis for understanding why process systems often behave in ways that defy deterministic prediction, and how such behavior can be managed rather than eliminated.

This paper and its companion study, *"From Pilot to Plant: When Scale Breaks Your Digital Twin,"* form part of a broader research program on complexity-aware Digital Twin design. A forthcoming e-book, *Beyond the Hype: System Complexity Science – The Overlooked Constraint in Digital Twin Design,* will further expand on the theoretical foundations and practical frameworks introduced here.

1. Introduction – The Digital Twin Expectation Gap

The idea of a Digital Twin—a virtual replica that behaves exactly like the real system—promises near-limitless value: predictive maintenance, optimization, autonomous control, and continuous improvement. In marketing material, the model and the process are shown as two identical loops seamlessly exchanging data. In reality, that equality rarely exists beyond carefully calibrated test cases.

The gap between **expectation and realization** stems from a fundamental assumption: that the physical process is deterministic, and that with enough sensors, data, and equations, the model can be made to replicate it fully. In practice, once the model meets the real plant, unmodeled interactions appear. Operators override controls, the environment changes, and the system exhibits behaviors that are **internally generated, not externally forced**.

This mismatch becomes evident in pilot implementations that appear successful but fail to generalize, or in brownfield deployments where legacy systems, control philosophies, and human actions interact unpredictably. The result is frustration—*"the model doesn't match reality"*—and costly re-work to redefine the scope.

2. A Useful Analogy – The Phantom Traffic Jam

A familiar example helps frame the issue.

You are driving on a highway when traffic suddenly slows to a standstill. After a few minutes, it begins to move again—no accident, no obstruction. The jam was self-generated: small, random fluctuations in driver speed amplified through feedback between vehicles.

This phenomenon—studied extensively in control theory—is a form of **emergent behavior**: order arising from local interactions without central coordination. Each driver reacts rationally to nearby conditions, yet collectively the system produces an outcome (a traffic jam) that no one intended.

Industrial systems often behave in the same way. Each control loop, unit, or operator acts locally and correctly, but the aggregate outcome may oscillate, destabilize, or shift to an entirely new operating state. Recognizing such emergent behavior as normal, not exceptional, is the first step toward realistic modeling. (For a visual demonstration of how scale alone creates emergent complexity—using identical process rules at pilot vs. plant scale—see the companion paper 'From Pilot to Plant: *When Scale Breaks Your Digital Twin*.')

The analogy also highlights the **limits of reductionism**. A model based solely on individual vehicle dynamics cannot reproduce a phantom jam unless it captures the interactions between vehicles. Similarly, a Digital Twin that perfectly represents equipment physics but omits interaction dynamics will diverge from reality the moment those interactions dominate.

3. Case Study – Identical Design, Different Behavior

Refinery operators often assume that two units built to identical specifications will behave identically. Experience shows otherwise.

Consider first two **Crude Distillation Units (CDUs)**. Both are steady, predictable systems whose performance depends mainly on physical configuration and heat-integration efficiency. Their first-principles models—based on thermodynamics and material balances—track reality closely, and any small deviations can usually be corrected with parameter tuning or calibration.

From a systems-thinking perspective, these units are complicated: they contain many components, yet their interactions remain largely linear and predictable. Operating behavior is well understood, and changes in one variable seldom trigger unexpected responses elsewhere.

Now compare a pair of **Fluid Catalytic Cracking Units (FCCUs)** at the same site. They share design, control logic, and feedstock, yet their operating behavior diverges dramatically. One remains stable during feed transitions; the other periodically destabilizes with no clear external trigger. Catalyst-activity decay, regenerator-temperature drift, and intermittent emissions variability appear without consistent cause.

Traditional engineering analysis looks for single explanations—instrument drift, controller tuning, or data error. Even after such issues are corrected, differences persist. The explanation lies not in hardware or human error but in **system interactions**.

Here the unit's behavior becomes **complex** in the systems-behavior sense introduced earlier: multiple feedback loops interact, amplifying or damping one another, and small timing or gain differences can tip the entire system into new equilibrium states or oscillations.

FCCUs are governed by multiple coupled feedback loops: the heat balance between reactor and regenerator, the coke-formation and combustion cycle, fractionator cut-point adjustments, and emissions control. Each loop can reinforce or counteract others depending on local conditions. Small timing or gain differences in one loop alter the overall feedback structure, producing new equilibrium states or oscillations. The unit's behavior is therefore an **emergent property of interactions**, not of its components.

This is the essence of **system-behavioral complexity** introduced earlier. A CDU's stability arises from largely one-way causal chains; an FCCU's dynamics emerge from circular causality. The CDU's predictability reflects its structural complication, whereas the FCCU's variability reflects its **interactional complexity**.

In practical terms, even if two FCCUs share identical blueprints, they occupy different points in the multidimensional interaction space defined by feed composition, catalyst condition, control response, and operator adaptation. They are, in effect, **different organisms sharing the same DNA**—a reminder that configurational similarity (the domain of the Nelson Index) does not imply behavioral equivalence.

Such differences illustrate why Digital Twins calibrated under one set of conditions often fail when deployed on another, and why early **complexity assessment**—identifying where interactions dominate—should precede model development rather than follow it.

---

4. Behavior Emerges from Interactions

The contrasting behavior of the two FCCUs can be explained only by looking beyond individual components and examining **how their feedback loops interact**.

In a reductionist approach, each component is analyzed in isolation and the whole is assumed to be the sum of its parts. That logic works for *complicated* systems such as CDUs, where relationships are stable and largely one-way.

In *complex* systems, however, the **interactions themselves drive behavior**, and those interactions change with context.

Within the FCCU, several loops operate simultaneously.

Three dominant examples illustrate how reinforcing and balancing mechanisms compete to define stability:

- **Reinforcing loop (R1 – Coke / Heat / Severity):**
  Feed heaviness ↑ → coke on catalyst ↑ → regenerator temperature ↑ → heat to riser ↑ → conversion ↑ → coke ↑.
  This loop amplifies deviations; small changes in feed composition or catalyst activity can cascade into runaway severity.
- **Reinforcing loop (R2 – Cut Points & Recycle):**
  Tighter fractionator cuts → LCO recycle ↑ → effective feed heaviness / Conradson carbon residue (CCR) ↑ → coke formation ↑ → regenerator temperature ↑ → riser severity ↑ → conversion ↑.
  This secondary loop reinforces R1 by increasing effective feed severity through internal

recycles. Adjusting cut points for short-term yield optimization can therefore strengthen internal feedback and accelerate instability.

- **Balancing loop (B1 – CO / Temperature Control):**
  CO in flue gas ↑ → air-flow controller ↑ → regenerator temperature ↑ → CO conversion ↓.
  This loop stabilizes operation—until interaction delays or tuning mismatches shift the balance.

When operating conditions change, the relative strength of these loops changes as well.

A control action that dampens one disturbance may reinforce another.

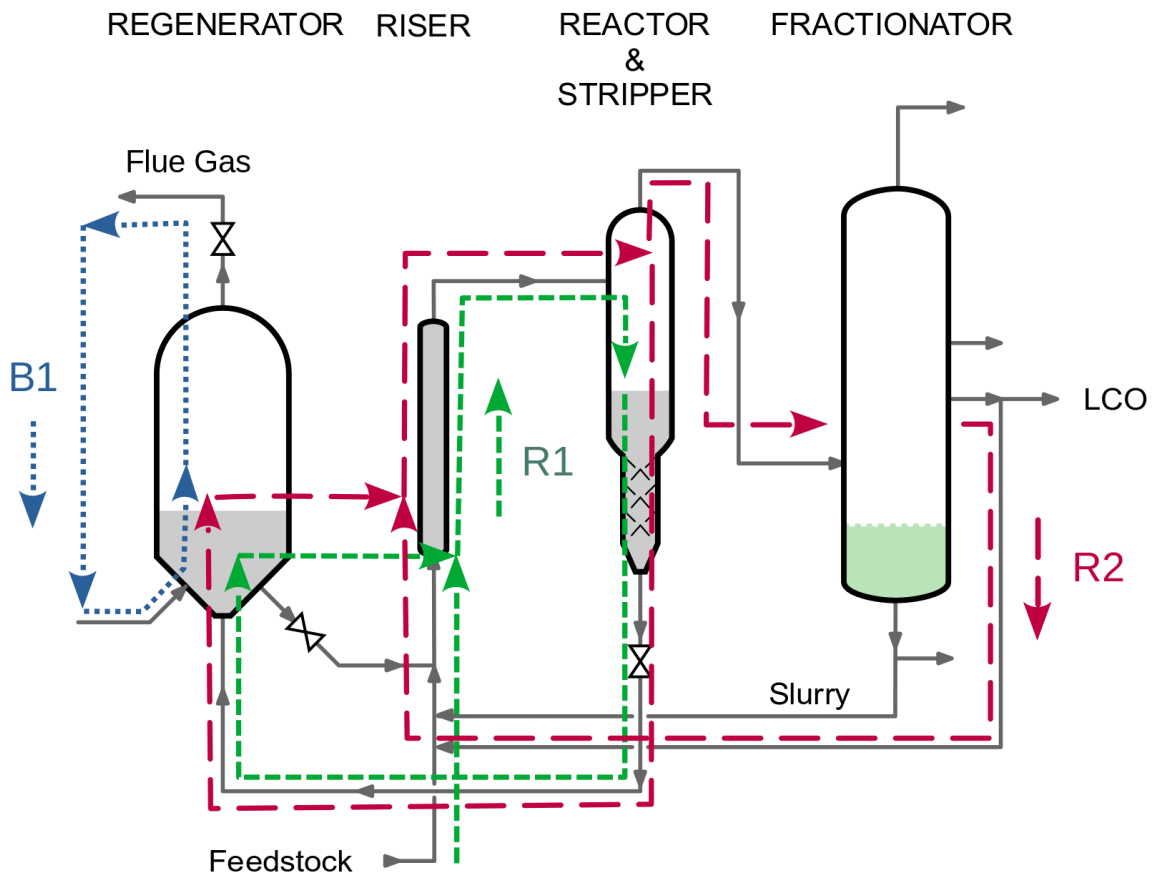


Figure 1: Generic FCCU Diagram with feedback loop interactions

The **dominant feedback path** can therefore migrate, causing the unit to behave differently under seemingly identical conditions.

These patterns are characteristic of what **complexity science** describes as *emergent behavior*—system-level outcomes produced by local interactions rather than by any central control. Originating from systems theory and the study of nonlinear, adaptive processes (Prigogine, Holland, Bar-Yam, Mitchell), this perspective provides the conceptual foundation for understanding why identical units can exhibit divergent dynamics and why Digital Twin models must represent interactions as much as components.

This context-dependence is the hallmark of complexity. The system's behavior is not merely *complicated*—it is **adaptive**. Local interactions between process dynamics, control logic, and human interventions continually reshape the feedback network itself.

These insights echo a body of research on complex system behavior across many high-risk domains. One of the most influential contributions is **Richard I. Cook's** essay *"How Complex Systems Fail"* (2000), written originally for medicine but profoundly applicable to industrial operations. Cook observed that failures in complex systems do not arise from single causes but from **combinations of normal interactions** that align in unanticipated ways. His observations illustrate why conventional root-cause logic breaks down when multiple small factors compound into emergent failure modes.

Selected Principles from Cook (2000)	Implications for Digital Twins
Complex systems are intrinsically hazardous systems.	Failure cannot be eliminated by design or modeling—it is inherent to the system.
Complex systems are heavily defended against failure.	Automation and control layers hide latent interactions until multiple defenses align.
Catastrophe requires multiple small failures aligning in time and space.	Predictive models trained on normal data rarely see combinations leading to collapse.
Complex systems run as broken systems.	Twins validated on "steady-state" data may miss chronic instability.
Human operators are adaptive problem-solvers.	Their interventions form part of the feedback structure, not external noise.
All successful safety work is invisible.	Model success is hard to measure precisely because stability hides risk.
Change introduces new failure modes.	Updating or re-scoping Twins without understanding interactions can re-introduce latent risk.

Table 1: Adapted from Richard I. Cook, "How Complex Systems Fail," 2000.

Cook's principles remind us that **complex systems cannot be controlled in a deterministic sense; they can only be managed.**

The practical value of a Digital Twin in such an environment is not to dominate complexity but to make it *visible and navigable*—a tool for understanding system dynamics, supporting operator awareness, and enabling adaptive management rather than rigid prediction.

In such environments, cause-and-effect relationships exist, but they are **non-stationary**: they evolve over time. That is why a model calibrated to yesterday's data may lose predictive power tomorrow. Complexity does not invalidate modeling; it simply means that **behavior is relational, not compositional**. Understanding those relations—rather than perfecting component fidelity—is what makes a Digital Twin useful.

5. Complicated vs. Complex Systems

The distinction between complicated and complex systems is critical—not in the sense of the Nelson index, but in the sense of system behavior.

Attribute	Complicated System (e.g., CDU)	Complex System (e.g., FCCU)
Dominant logic	Deterministic	Nonlinear / feedback-driven
Behavior	Predictable, repeatable	Emergent, context-dependent
Modeling approach	First-principles equations	Hybrid (physics + data + heuristics)
Control tuning	Parameter optimization	Continuous adaptation
Failure mode	Single cause	Multi-factor combination
Validation	Steady-state comparison	Behavior envelope tracking

The difference lies not in system size but in **interaction density and feedback diversity**.

A small system can be complex (e.g., biological fermenter), and a large one can be merely complicated (e.g., pipeline network).

Recognizing where a process lies on this spectrum is essential before defining what a "Digital Twin" should achieve. Many project failures originate from treating complex systems as complicated ones—expecting deterministic control where only probabilistic or adaptive behavior is possible.

This distinction echoes *Richard Cook's* insight in *"How Complex Systems Fail" (2000)*: failures are rarely traceable to a single cause but arise from multiple small factors interacting in ways that only become obvious in hindsight. The implication for DT projects is clear—**predictive accuracy is not guaranteed by completeness of data alone**.

The next section examines how this hidden complexity shapes Digital Twin project outcomes.

6. The Hidden Impact of Complexity in Digital Twin Projects

Complexity exerts its influence long before a Digital Twin reaches deployment.

When the interactions that define system behavior are poorly understood, the project may appear on track — until late-stage integration exposes inconsistencies that trace back not to coding or calibration errors, but to **missing recognition of system complexity**.

Most Digital Twin shortfalls stem not from modeling tools, but from *the assumptions built into their requirements.*

a) Ambiguous Requirements

Many DT charters use language such as *"develop a real-time Digital Twin for optimization."*

The goal sounds concrete but hides an unspoken premise: that the process is deterministic and predictable.

If, instead, the process exhibits strong feedback, adaptation, or emergent modes, "optimization" may have no single solution.

Without an explicit complexity assessment, requirements stay vague, validation criteria become subjective, and project success can only be declared — never demonstrated.

> **Example:** Two FCCUs built to identical design produced diverging yield patterns.
>
> The project requirement called for "5 % yield improvement through Digital Twin optimization."
>
> In one unit the model performed flawlessly; in the other it diverged within weeks.
>
> The problem wasn't the model—it was that *the two systems belonged to different complexity regimes.*

b) Overconfidence in Data and Models

Machine-learning and hybrid approaches are often promoted as universal solutions.

They are powerful, but they cannot extract information that is **absent from the data**.

Rare events — defluidization, feed transitions, or catalyst behavior under stress — leave few examples in historical datasets.

As a result, models trained on past behavior **hallucinate stability** when confronted with unobserved dynamics.

The failure occurs not at runtime, but at design time, when model scope is defined without testing for system complexity.

c) Human and Organizational Interactions

Humans are an inseparable part of industrial systems.

Operators, planners, and maintenance teams constantly adjust boundaries and priorities, effectively changing the system configuration in real time. A Digital Twin that excludes these interventions is modeling a *different* system.

Each manual adjustment—an override, a production schedule change, a maintenance deferral —adds a feedback path that no control diagram shows.

Complexity increases silently, until the model's assumptions no longer match the plant's reality.

d) Why Complexity Remains Hidden

Traditional project scoping frameworks — phase-gate reviews, deterministic milestones, linear budgets — are designed for *complicated* systems with stable cause-effect relationships.

They work well when uncertainty is reducible by more detail.

In complex systems, uncertainty is **structural**, not informational: it persists no matter how much data are collected.

The linear project model therefore misrepresents effort, cost, and risk.

By the time instability surfaces, budgets and expectations are fixed, and re-scoping appears as failure rather than learning.

e) Visible Symptoms

Typical warning signs that complexity has been ignored include:

- Model performance deteriorates under disturbance even though calibration is correct.
- Operators distrust or disable model recommendations.
- Root-cause analysis cycles endlessly between “bad data” and “bad model.”
- Incremental fixes accumulate, yet systemic mismatch persists.

These are not anomalies—they are **symptoms of an unacknowledged complexity gap**.

f) Turning the Gap into Insight

Recognizing complexity early does not add bureaucracy; it replaces rework with foresight.

A short **complexity-profiling exercise** at concept stage can reveal which aspects of a unit are predictable and which are interaction-driven.

That insight informs model architecture, validation design, and user expectations before development begins.

The cost is measured in weeks; the savings, in months of re-engineering.

7. Why Traditional Project Scoping Fails

Traditional project frameworks evolved to manage complicated systems, where uncertainty decreases with more detail. In complex systems, this assumption breaks down.

Most industrial projects are managed through frameworks designed for deterministic outcomes. They define inputs, outputs, milestones, and deliverables. Such methods are effective for **complicated** systems but fragile for **complex** ones.

Three systemic gaps dominate:

1. Deductive Bias

Traditional engineering starts with known principles and deduces outcomes.

Complex systems often require the inverse: inductive reasoning from observed behavior.

Project teams trained in deduction assume that more detail equals more control, adding variables and constraints without recognizing that **model completeness does not imply predictive validity**.

2. Linear Planning

Gantt charts and waterfall milestones assume sequential progress—each step based on a stable foundation. But in complex environments, discoveries at later stages invalidate earlier assumptions. By the time instability surfaces, budgets and expectations are locked. Corrective iteration appears as failure, not learning.

3. Undefined Success Criteria

When system behavior is uncertain, defining "model accuracy" becomes ambiguous. Is success defined by matching outputs within 1% of reality, or by correctly predicting qualitative trends?

Without clear criteria, the project drifts toward over-engineering or premature declaration of victory.

These shortcomings reflect an **organizational blind spot**: complexity is treated as noise rather than a property of the system.

Consequently, budgets inflate, schedules slip, and results underwhelm.

Yet, these failures are avoidable—not by abandoning Digital Twins, but by **re-engineering how projects are scoped**.

Recognizing these limitations leads naturally to the need for a structured approach to diagnose and quantify system complexity before scoping begins.

To manage this, projects need a structured way to expose and quantify complexity before scope and validation targets are frozen. The next section introduces such an approach.

8. Introducing the Complexity Assessment Framework

Complexity cannot be reduced—it must be **characterized**.

Understanding where and how a system's complexity manifests is the foundation for setting realistic expectations.

The **aprocesr Complexity Assessment Framework** provides a structured, repeatable, and collaborative method for identifying the interaction patterns that determine Digital Twin feasibility.

The framework is not an audit or a scorecard; it is a **conversation structured by evidence**.

It translates the abstract notion of "complexity" into measurable dimensions that link directly to modeling strategy and resource allocation.

Each dimension falls into one of three categories—**Structural/Physical**, **Dynamic/Operational**, or **Informational/Organizational**—as described earlier.

The Ten Assessment Dimensions

Structural / Physical — Physical & Control Interconnection
How physical coupling and control interactions shape predictability.

1. **Structural Interdependence** — Density of material/energy/information links across subsystems.
2. **Feedback Loop Density** — Number and strength of interacting control loops and dynamic couplings.
3. **Nonlinearity / State-Dependent Behavior** — Gain shifts, thresholds, or regions where small input changes produce large effects.

Dynamic / Operational — Behavior Over Time
How dynamics evolve, adapt, or produce surprises during operation.

4. **Temporal Variability** — Variation across operating time scales; stability vs. regime shifts.
5. **Human Decision Influence** — Extent to which manual interventions change feedback structure.
6. **System Boundary Change / Integration** — Moving boundaries, integration with external systems, live reconfiguration.
7. **Emergent Behavior Potential** — Risk of outcomes arising from interactions (e.g., self-generated oscillations).

Informational / Organizational — Information & People
How data quality and cross-functional coupling influence behavior.

8. **Data / Information Quality & Completeness** — Ambiguity, missing signals, latency, or low signal-to-noise.
9. **Rate of Change / Evolution** — Frequency of changes: feed, equipment, control logic, or constraints.
10. **Organizational Complexity / Cross-Functional Coupling** — Hand-offs, conflicting objectives, and coordination overhead across teams.

Comparing CDU and FCC profiles illustrates the framework's diagnostic power:

A **Crude Distillation Unit** typically scores:

- Low on dimensions 2, 3, 5, 7 (minimal feedback interaction, linear response, stable operation)
- Moderate on dimensions 1, 4, 6 (some coupling, predictable variation)
- Variable on dimensions 8, 9, 10 (depends on site-specific factors)
- **Result**: Complicated system — high-fidelity first-principles modeling feasible

A **Fluid Catalytic Cracking Unit** typically scores:

- High on dimensions 1, 2, 3, 7 (strong coupling, multiple reinforcing loops, nonlinear behavior, emergent oscillations)
- Moderate to High on dimensions 4, 5, 6 (regime shifts, operator intervention dependency)
- Variable on dimensions 8, 9, 10

- **Result**: Complex system — hybrid physics-data modeling required, behavioral validation essential

During a guided workshop, participants rate each dimension (Low / Moderate / High) based on observed behavior and system evidence.

The result is a **complexity profile**, often shown as a radar chart, that highlights where project risk and analytical effort should concentrate.

- A profile dominated by *high structural* scores implies strong physical coupling and the need for hybrid or reduced-order physics modeling.
- *High dynamic* scores indicate sensitivity to control interactions and require co-simulation or real-time validation strategies.
- *High informational or organizational* scores suggest that human and data factors will limit adoption more than physics will.

The objective is not to label a unit as "prohibitively complex," but to **set a realistic boundary for what fidelity is achievable**—and where investment will return the greatest value.

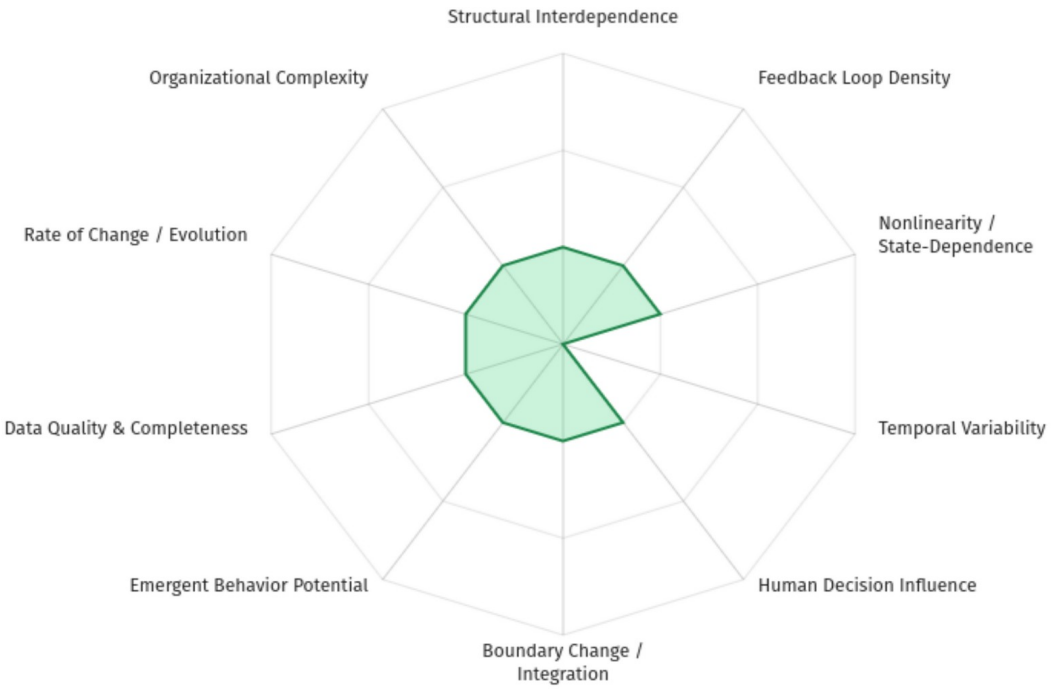


Figure 2: Typical CDU complexity profile

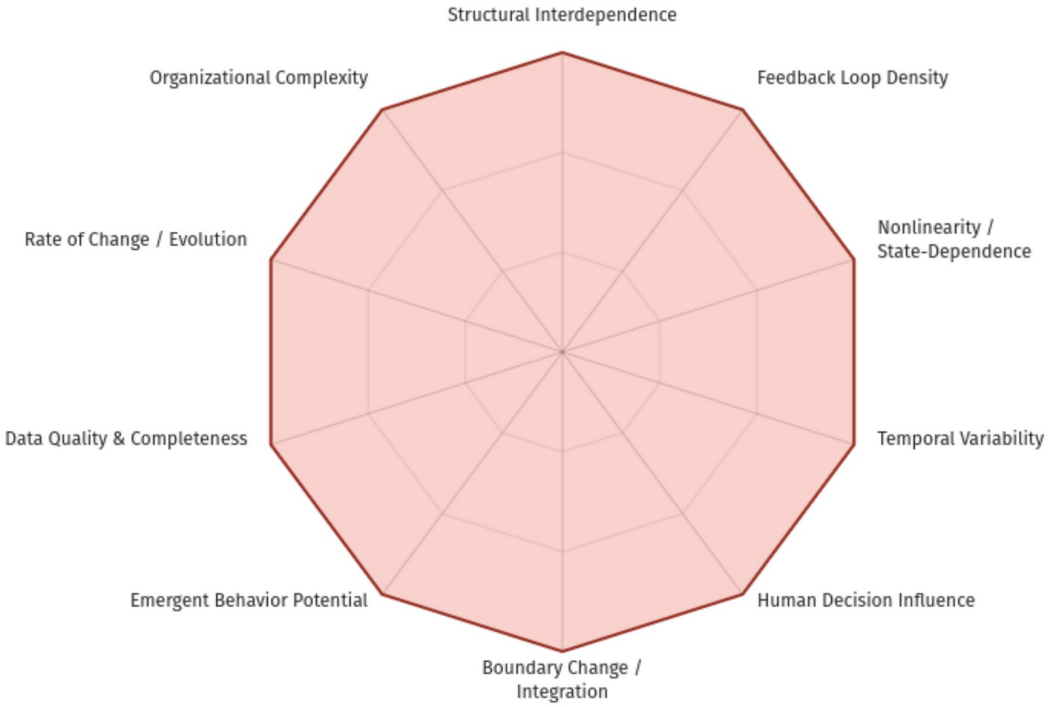


Figure 3: Typical FCCU complexity profile

9. From Profile to Requirements

Assessment alone has little value unless it informs action.

The next step is to **translate complexity insights into system requirements**—the formal bridge between plant reality and model design.

The process described here is based on **Specifications Engineering**—an applied adaptation of INCOSE's *Requirements Engineering* principles for industrial environments.

Whereas formal Requirements Engineering defines a full life-cycle discipline, Specifications Engineering focuses on producing clear, testable, and implementation-ready deliverables that translate system needs into actionable model and project specifications.

This translation follows three conceptual layers:

1. **Profile → System Understanding**

   Identify which complexity categories dominate.

   Example: FCCUs show high dynamic and organizational complexity; CDUs show mainly structural.

2. **System Understanding → Specification Depth**

   Determine the level of effort required for data fidelity, control coupling, and validation scope.

Example: High dynamic complexity demands closed-loop testing rather than static validation.

3. **Specification Depth → Project Scope and Resources**

   Define modeling approach, computational needs, data governance, and stakeholder engagement.

In the broader **aprocesr methodology**, these three conceptual layers align with a four-step *Specifications Engineering* workflow: **Understand → Conceptualize → Prepare → Specify.**

The first two steps correspond to system understanding, the third ensures organizational and data readiness, and the final step formalizes specifications and validation criteria.

This approach brings the rigor of Requirements Engineering to a level of practicality suited for industrial Digital Twin projects—retaining the logic of systems engineering while discarding the bureaucracy that often makes it inaccessible outside aerospace or defense.

This is, in essence, a **requirements-engineering process** adapted from systems-engineering principles (INCOSE 2023) but scaled for industrial practicality.

Rather than hundreds of low-value requirements, the focus is on a concise set of **high-impact functional and validation specifications** that directly address identified complexity drivers.

Typical outputs include:

- A functional hierarchy describing what the Digital Twin must *represent*
- An interface map showing what it must *connect* to
- A validation matrix defining how success will be *measured*

Complexity thus becomes a **design constraint**, not a post-hoc discovery.

10. Selecting the Right Modeling Strategy

Different systems warrant different modeling philosophies.

Choosing the wrong one at the start nearly guarantees cost overruns or failure.

System Type	Recommended Approach	Key Risks if Misapplied
Complicated / Predictable	First-principles deterministic models.	Excessive detail may waste effort, but results are reliable.
Complex / Adaptive	Hybrid models combining physics with data-driven learning.	Purely physics-based models miss emergent behavior; purely data-driven models hallucinate outside training range.
Data-Rich but Physically Opaque	Machine-learning surrogate constrained by physics.	Loss of interpretability; spurious correlation.
Sparse-Data Legacy Systems	Reduced-order physics models calibrated with expert heuristics.	Unvalidated extrapolation; narrow envelope of reliability.

For systems dominated by uncertainty and continual adaptation, **Bayesian or other probabilistic frameworks** can complement these approaches by updating model confidence as new evidence becomes available.

Their value lies not in replacing physics or data models, but in **managing uncertainty**—a defining property of complex systems where prediction must remain probabilistic rather than absolute.

The framework acts as a **decision gate** before committing resources.

It answers the practical question:

> *What level of fidelity is realistically achievable, and what will it cost to sustain?*

---

11. ROI and Implementation Implications

a) Prevention vs Discovery

Early assessment costs weeks; late discovery costs months.

Experience shows that investing **1–2 % of project effort** in complexity assessment and scoping can avoid **20–30 %** re-engineering later.

b) The Value of Realism

In complex systems, *realism*—not precision—drives ROI.

A model that reproduces the *direction* and *qualitative response* of system behavior often provides more value than one that matches numeric outputs under narrow conditions.

Complexity-aware specifications focus on behavioral envelopes rather than single-point accuracy.

c) Sustainability and Lifecycle

Every Digital Twin is a living entity that evolves with the plant.

Complexity assessment informs lifecycle planning—showing which elements require continuous calibration and which can remain static.

Clear expectations about this balance reduce sustainment cost and ensure long-term credibility of the model.

12. Key Takeaways

1. **Complex ≠ Complicated** – Units like FCCUs display emergent behavior that cannot be decomposed into simple cause-effect chains.
2. **Assess Before You Commit** – Early diagnostic profiling prevents major mis-scoping later.
3. **Specifications Engineering as Bridge** – Translating complexity insights into explicit, testable requirements aligns expectations and resources.
4. **Match Modeling Strategy to Reality** – Choose physics-, data-, or hybrid-based approaches according to dominant complexity drivers.
5. **Realism Delivers ROI** – Digital Twins that mirror behavior, not just structure, sustain their value over time.
6. **Complex Systems Can't Be Controlled — They Can Be Managed** – The role of a Digital Twin in a complex environment is not to impose control but to enhance understanding, visibility, and adaptability. Its success is measured not by perfect prediction but by how effectively it helps people manage what cannot be predicted.

Complexity is not an obstacle to be eliminated; it is a property to be recognized and managed.

13. About aprocesr

aprocesr Ltd. is an independent engineering consultancy specializing in process systems analysis, automation strategy, and Digital Twin feasibility.

Founded and directed by **Sam Matic, P.Eng.**, aprocesr bridges the gap between process reality and modeling ambition through **requirements engineering**, **complexity assessment**, and **hybrid-modeling advisory** services.

The firm collaborates globally with industrial clients and academic partners to translate complexity science into practical project success.

Core Services

- Complexity diagnostics and Digital Twin feasibility studies
- Requirements engineering for automation and modeling projects
- Hybrid model architecture and validation strategy consulting
- Training and workshops on complexity-aware systems thinking for engineers

📧 smatic@aprocesr.com 🌐 aprocesr.com/ADS2025

A. Keywords

Keywords: Digital Twin, system complexity, complexity science, complicated vs. complex, emergent behavior, specifications engineering, requirements engineering, refinery systems, Fluid Catalytic Cracking (FCC), Crude Distillation Unit (CDU), process systems engineering, complexity assessment, hybrid modeling, INCOSE, systems thinking.

B. Glossary

Term	Definition (context of this paper)
Complex System	A system whose behavior arises from interacting components, producing outcomes not predictable from individual parts.
Complicated System	A system with many parts but predictable cause–effect relationships.
Complexity Assessment	Structured evaluation of interdependencies, feedback loops, and variability that affect system behavior.
Specifications Engineering	Applied adaptation of INCOSE's Requirements Engineering focused on generating clear, testable, and implementation-ready specifications.
Requirements Engineering (RE)	Formal discipline for eliciting, analyzing, and managing system requirements throughout a project lifecycle.
Emergent Behavior	Behavior that arises from local interactions without central control, often unexpected and nonlinear.
Hybrid Modeling	Modeling approach that combines first-principles physics with data-driven machine learning elements.
Digital Twin (DT)	A dynamic digital representation of a physical process or system, used for analysis, prediction, and decision support.
Feedback Loop	A causal pathway where system outputs influence its inputs, producing stabilizing (balancing) or amplifying (reinforcing) effects.
Nelson Complexity Index (NCI)	An economic measure of refinery configuration and upgrading capability—not related to system-behavior complexity.
Cynefin Framework	A management model categorizing decision contexts as simple, complicated, complex, or chaotic—distinct from engineering complexity.

References

Matic, S. (2025). "From Pilot to Plant: When Scale Breaks Your Digital Twin", Proceedings of the Asian Downstream Summit 2025, Singapore

Cook, R. I. (2000). *How Complex Systems Fail.* Cognitive Technologies Laboratory, University of Chicago.

INCOSE (2023). *Systems Engineering Handbook, v5.0.* International Council on Systems Engineering.

Snowden, D. (2007). *Cynefin: A Framework for Decision-Making in Complex Contexts.* IBM Systems Journal.

Prigogine, I. (1980). *From Being to Becoming: Time and Complexity in the Physical Sciences.* W. H. Freeman & Co.

Holland, J. H. (1992). *Complex Adaptive Systems.* Daedalus 121(1).

Bar-Yam, Y. (1997). *Dynamics of Complex Systems.* Addison-Wesley.

Mitchell, M. (2009). *Complexity: A Guided Tour.* Oxford University Press.