arXiv:submit/8331222 [cs.AI] 5 Nov 2026

# ChemE-as-Code: An Open-Source Platform for Agentic AI in Chemical Engineering

Artur M. Schweidtmann

Process Intelligence Research Group, Department of Chemical Engineering,
Delft University of Technology, Van der Maasweg 9, 2629 HZ Delft, The Netherlands
a.schweidtmann@tudelft.nl, ORCID 0000-0001-8885-6847

Preprint submitted to AIChE Journal, November 2026

**Abstract**

ChemE-as-Code provides an open-source environment in which AI agents built on large language models (LLMs) can carry out chemical engineering work and engineers can check it. Its calculation tools range from catalysis through process simulation to plant layout, HAZOP and life cycle assessment, and skills tell a general-purpose agent how to use them. Engineering files are linked, machine-readable and versioned in Git; calculations run outside the LLM, each value records its origin, and automated checks report inconsistencies and out-of-date results by file and line. In an illustrative ammonia case, agents developed a written task into a preliminary basic-engineering package (simulation files, P&IDs, layout, HAZOP, datasheets, relief) for USD 224 of API cost in the design sessions. Planted faults and engineering changes show that the checks find what their rules cover and miss the rest, so they guide human review rather than replace it. No independent engineer has reviewed the design.

**Keywords:** agentic AI, large language models, Plant-as-Code, process design, P&ID

## 1 Introduction

Generative artificial intelligence (AI) is expected to reshape chemical engineering [1, 2]. AI agents, its newest form, can go beyond answering questions: they can use tools, edit files and iterate on their own results [3, 4]. Engineering work can thus increasingly be proposed, executed and revised by computational agents. In the past years, software development has shown what this opens up. AI assistants have sped up well-defined programming tasks [5], and the length of tasks that agents complete on their own keeps growing [6]. Software teams can use agents in this way because their work already lives in versioned text files that automated tests can check. We expect chemical process engineering to benefit in the same way once its work takes the same form.

That form is harder to reach in chemical engineering than in software. A plant is described by many coupled documents: process models, specifications, process flow diagrams (PFDs), piping and instrumentation diagrams (P&IDs), equipment data, layouts, safety studies and environmental inventories. P&IDs are the central reference documents of a plant over its life cycle [7–9] and are still created largely by hand [7, 10]. The documents depend on each other, but the dependencies are rarely recorded. This means that a change in one place, such as a higher design capacity, silently invalidates numbers elsewhere: in the simulation, the drawings, the equipment data, the layout, the safety analysis and the inventory [11]. Nothing tells the engineer which of them. Much process information also still sits in documents that must be digitized before a change can be

assessed [12]. Integrated information models, exchange standards and digital threads address parts of this problem [13–16]. They let tools exchange data. They do not record, for a whole project and at every revision, which results are still consistent with each other, which are out of date, and who has reviewed what.

AI agents make this problem larger. They produce more proposals and revisions than people can read, and their output can look plausible but still be invalid. Earlier generators of flowsheets and P&IDs, from rule-based systems [10, 17, 18] to learned models [19–21], have a related limit: they produce part of a design without the decisions behind it and outside the engineering workflow. As agents propose faster, verification increasingly sets the pace of engineering [22].

A further question is how agents should be organized. Our earlier review proposed teams of specialized agents, one per discipline, mirroring an engineering organization [4]. Here we take a different route. Domain knowledge sits in skills and tools, and one general-purpose agent loads what a task needs. A skill is a short instruction file that tells the agent how to perform an engineering task; a tool is a program that performs a calculation or a check. This keeps the engineering knowledge in files that engineers can read, test and version, independently of a particular agent or model, and it reduces the need for coordination between discipline-specific agents, a reported source of failure in multi-agent systems [23]. Separate sessions are still used where independence helps, as in review. Which option works better is an open question that needs a controlled comparison.

The acceleration that AI agents promise depends on the engineering environment in which they work. We present ChemE-as-Code, an open-source platform that provides such an environment. It implements the Plant-as-Code perspective of Schweidtmann and Demeyer [22] and extends it from process and plant design to process chemistry, simulation, techno-economic assessment (TEA) and life cycle assessment (LCA); the name stands for treating chemical engineering work as code. Early industrial pilots on individual projects have pointed to substantially faster design iterations [22], and ChemE-as-Code makes the approach open, broad and testable. The platform rests on two foundations. First, a plant is one Git repository of linked, machine-readable files, from the design basis (the requirements the plant must meet: product, capacity, feed, site and utilities) to the life cycle inventory. Every revision is checked by programs, and the checks and calculations behind each value are kept with it; we call this record the evidence of a revision. Second, an open set of calculation tools and agent skills spans the scales of chemical engineering, from catalysis and reaction kinetics to plant layout and LCA (Figure 1). Agents and engineers work through the same files and command line, and in the proposed workflow a person reviews the evidence before a revision is accepted. We argue that the innovation lies in this combination of capable agents with an engineering environment designed for them. Models will change quickly, but we expect the way agents interact with engineering files, tools, evidence and reviewers to remain relevant.

We ask three questions. RQ1 (representation): can a process project be represented as linked, machine-readable files across disciplines that agents and people edit? RQ2 (verification): do automated checks find inconsistencies, out-of-date results and affected files when the design changes, where do they fall short, and can generated files be regenerated reproducibly? RQ3 (agent-enabled engineering): can a general-purpose agent carry out interconnected engineering tasks with these skills and tools, at what cost and with which limitations? The paper contributes a paradigm for AI-accelerated, human-directed Chemical Engineering-as-Code, described as five shifts in how engineering work is organized (Section 3); the open-source platform that implements it (Section 4); and an end-to-end demonstration on a natural-gas ammonia plant with an evaluation of what the checks find and miss (Section 5). The case shows that the workflow runs. It does not establish the quality of the design or a productivity gain, which remains our hypothesis.

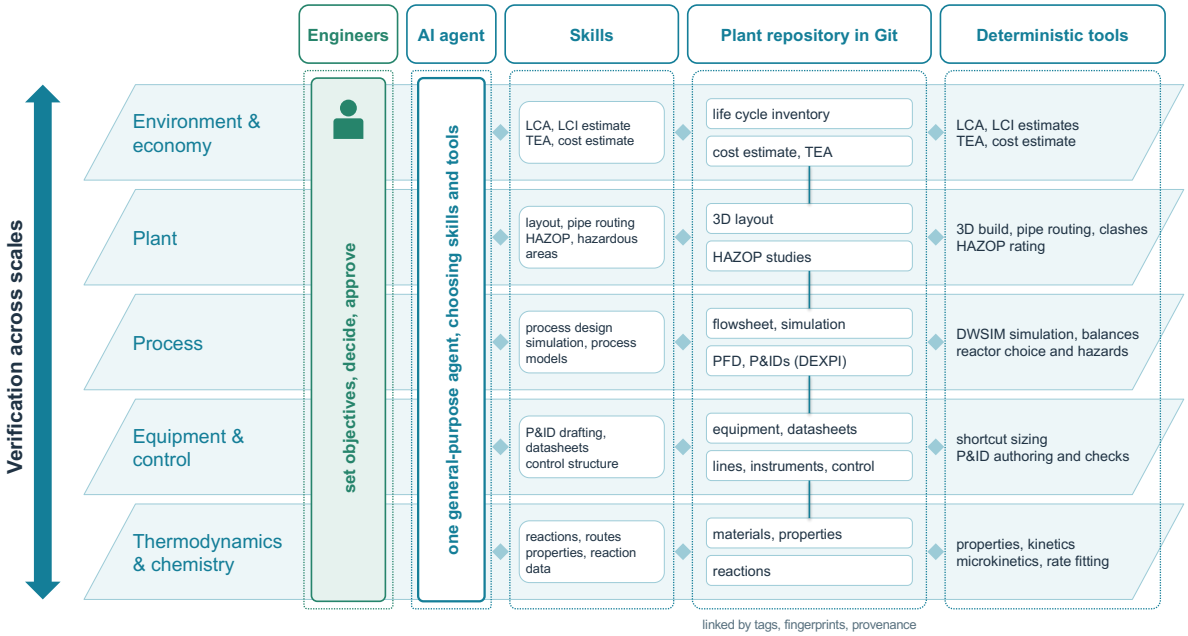



Figure 1: ChemE-as-Code across engineering scales. One general-purpose AI agent selects, by task, the skills (instruction files) and the calculation tools of a scale, from chemistry and thermodynamics through process and equipment design to plant layout, safety, cost and environmental assessment. Engineers (green) set objectives, decide and, in the proposed workflow, approve revisions. All work goes into one shared, versioned plant repository in Git. Its files refer to each other by equipment and line tags, each calculated value records its inputs, and each value records where it came from, so that a change at one scale can be traced to the files it affects at other scales. The domain knowledge lives in the skills and tools, not in several specialized agents. Concept adapted from Rupprecht et al. [4].

## 2 Related Work

ChemE-as-Code builds on decades of process systems engineering (PSE), engineering information management and software practice. What it adds is the combination: one versioned project in which agents make changes and programs check them for people. This section places that combination against four lines of prior work.

### 2.1 Process Engineering Information and Its Management

Simulation, flowsheeting, optimization and synthesis are the foundations of PSE [24], and connecting the information they produce is an old goal of the field [25]. Integrated models for data and documents [13] and the OntoCAPE ontology [14, 26] make process engineering knowledge machine-readable and reusable. Knowledge graphs now link process knowledge, models and the origin of data in digital twins [27–29]. Information technology for integrated process and plant design [30], the digitization of legacy drawings [12] and the digital thread in manufacturing [16] pursue the same aim. Model-based systems engineering and life cycle standards make traceability and configuration management explicit practice [31, 32], and design research has studied how a change to one component

propagates through a product's structure [11].

In industrial practice, engineering document management systems (EDMS) control the revision, approval status and distribution of documents over a plant's life cycle [33]. Data-centric engineering databases keep P&IDs, equipment and instrument lists in one store, so that the tags and connections on a drawing agree with the lists as long as both are edited in that tool [30, 34]. These systems solve part of the version and dependency problem. However, an EDMS tracks documents rather than their content: it records which revision of a P&ID is current, not whether that revision agrees with the instrument list or the simulation. Consistency in integrated tools ends at the tool boundary, and much of the information gathered in design does not reach operation in electronic and interoperable form [34].

### 2.2 Exchange Standards

P&ID data usually sit in proprietary formats of computer-aided engineering tools. DEXPI, based on ISO 15926 [35], defines a vendor-independent information model and XML exchange format for P&IDs and process models [15]. Version 2.0.1 covers block flow diagrams (BFDs), PFDs and P&IDs [36] and is the format ChemE-as-Code stores. pyDEXPI makes DEXPI data available to Python and to machine learning as graphs [37], so that DEXPI can act as the interface between generative models and industrial tools; graph learning on DEXPI plant topologies [38] and large language model (LLM) interaction with P&IDs [29, 39] build on it. CAPE-OPEN makes unit operations and property packages interoperable between process simulators [40], AutomationML exchanges engineering data in automation engineering [41], and building engineering has open, version-controlled common data environments such as Speckle [42]. However, standards solve exchange, not revision consistency. A valid DEXPI file says nothing about whether the equipment list, the simulation or the HAZOP study still agrees with it. This means that a standard can carry a design between tools, but it cannot tell an engineer what a change has invalidated.

### 2.3 AI in Process Engineering

AI in process engineering is starting to move from generators toward LLM agents, although these are still first works, and most methods address one document or one task at a time. The field reaches from expert systems, including knowledge-based P&ID configuration [10, 17, 18], through machine learning in PSE [43–45], to generative AI and LLMs [1, 2], which need to be grounded in domain knowledge [46]. For flowsheets and P&IDs, prior work has produced text representations [47–49], generative models that complete flowsheets and propose control structures [19, 20, 50, 51], and rule-based and learned methods that detect and correct errors [21, 52–54]. Open engineering data are scarce [55, 56], and generative models produce topologies without the attributes and decisions behind them, are scored against a single reference design although several designs can be valid, and are not integrated into engineering workflows. More recently, LLM agents carry out engineering tasks with tools, from process improvement, optimization and control [57–59] to P&ID generation [60, 61], simulation and model generation [62–66], conceptual separation design [4] and question answering on P&IDs [29, 39]. Most use task-specific tools with little human oversight [4]. Some recent work connects several tasks. Laub et al. digitize process descriptions into flowsheets and build converged Aspen Plus simulations, benchmarked against flowsheets from 50 experts [65]. Tian et al. go from textual specifications to executable process simulations with a multi-agent workflow [66], and Zhang et al. combine a knowledge graph with agents to assemble and calibrate process models for digital twins [28]. These works stop at flowsheets, simulations and process models; they do not manage and verify revisions across the further disciplines of a plant project. Beyond chemical engineering,

tool-augmented agents plan and run chemistry experiments [3, 67], software-engineering agents work through agent–computer interfaces [68] and are evaluated on real repository issues [69], and multi-agent frameworks coordinate several LLM agents on one task [70, 71].

### 2.4 Engineering-as-Code and Reproducibility

Software engineering manages change through version control, continuous integration and delivery [72], and Infrastructure-as-Code applies the same practice to computing infrastructure [73]. Engineering-as-Code transfers it to engineering design [74]. In plant and automation engineering, industrial practice came first: Davy Demeyer's Design-Ops work applied Engineering-as-Code and agentic practices before this work [22, 75], and early industrial pilots on individual projects point to substantially faster design iterations [22]. These pilots use closed software and their details are confidential, so their scope and results cannot be examined or reproduced. Plant-as-Code, proposed by Schweidtmann and Demeyer [22], provides a perspective on Engineering-as-Code for process plants.

Engineering results can only be trusted if a reader can see where a value came from and can regenerate it. Computational science has practices for this. Rules for reproducible research [76] and the FAIR principles [77] describe how to keep data and code findable and reusable, containers fix the software environment [78], and reproducible builds require that the same inputs yield outputs that are identical to the byte [79]. W3C PROV-O provides a vocabulary for recording which activity and which agent produced a result [80]. In engineering projects, by contrast, whether a value was given by a supplier, assumed by an engineer or calculated by a tool is often recorded only in documents, and a result often cannot be regenerated after a change.

Taken together, prior work provides rigorous models, shared semantics, portable formats, fast generation and the software practice for managing change. ChemE-as-Code brings them into one open project in which every revision is checked across disciplines and kept with its evidence, with tools and skills that let agents work across the scales of chemical engineering.

## 3 The Paradigm of AI-Accelerated Chemical Engineering-as-Code

Chemical Engineering-as-Code changes how engineering work is organized, not what engineers calculate. It extends Plant-as-Code beyond the plant to the whole discipline, including process chemistry, simulation and LCA. We describe this change as five shifts. Each has a counterpart in software development [68, 72], which we use as an analogy: what transfers is the way work is organized, not the nature of the product. Figure 2 shows one iteration of the resulting workflow for a capacity change, the example we use throughout: the plant is to make 20% more product, so the feed, the balances, the equipment, the drawings, the safety study and the inventory all have to follow.

**(i) From documents to connected engineering knowledge.** Today, a project is a set of deliverables, each made in its own tool and passed on as a document. In the new paradigm, the design basis, models, drawings, lists, layouts, safety studies and assessments form one connected body of machine-readable knowledge, together with the decisions and evidence behind them. The software analogy is source code and configuration kept as text in one repository, as in Infrastructure-as-Code [73]. The difference is that drawings and documents do not disappear: they remain as views of the knowledge, issued when a stage requires them. Not every model has to become executable code, but the representations, decisions and dependencies that engineering work needs must be readable, changeable and checkable by people and programs alike. For the engineer, this means

that a pump can be followed from the flowsheet to the P&ID, the equipment list and the HAZOP study, which an assistant bound to one tool cannot do [22].

**(ii) From manual execution to human-directed agentic engineering.** Engineers have so far carried out most engineering activities themselves, with tools for individual calculations. Now they have started to direct some workflows instead. They state objectives, constraints, specifications and acceptance criteria, and agents draft, calculate, revise and document, using deterministic engineering tools for every calculated number. The analogy is a developer who directs coding agents that open pull requests for review. The difference is where responsibility sits: in engineering, the engineer keeps the responsibility for the design, so an agent's draft is a proposal until a person accepts it. Engineers can then spend more of their time on alternatives, critical decisions and review. If a first draft becomes cheap, a broader exploration of design alternatives may become affordable.

**(iii) From isolated deliverables to continuously evolving projects.** A deliverable is issued at the end of a phase and revised in the next. In Chemical Engineering-as-Code, people and agents change a shared project in small revisions. A revision is one recorded change with its reason, and it can span process, equipment, piping, control, safety and assessment at once. The history of the design, and why it took its present form, stays with the design. The analogy is small, reviewed commits with a recorded history instead of a release at the end of a development phase. The difference is that the engineering phases from concept to detailed engineering remain, with their formal releases and approvals; what can become much shorter are the iterations within them [22].

**(iv) From periodic checking to continuous verification.** Checks are traditionally tied to milestones such as design reviews and HAZOP studies. Agents, however, can propose revisions faster than anyone can check them [22], so milestone checks cannot keep up, and verification has to move into every iteration. A check is a program that tests one rule over the files, for example that every instrument on a P&ID is in the instrument list, and reports a finding with the file and line where the rule fails. Agents repair findings before a person looks at the revision, and every revision carries its evidence: the results of the checks and the calculations behind its values. The analogy is automated tests and continuous integration, which run on every commit. The speed of this loop depends on how cheap, local and trustworthy the checks are. Three states of work are kept apart. A *proposed* revision is what a person or agent has drafted. A *checked* revision has no error-level findings; its remaining warnings and the requirements that cannot yet be checked are stated with it. An *approved* revision has been accepted by a responsible engineer. The difference from software is firm: checks never replace approval, which is given by people, and a HAZOP study still needs a team.

**(v) From discipline-specific automation to integrated engineering.** Instead of automating a flowsheet, a P&ID or a HAZOP study one at a time, the aim is to connect these tasks, so that a change and its consequences are followed across disciplines and project stages. The capacity increase of Figure 2 then reaches the balances, the simulation, the drawings, the safety studies and the inventory in one revision, and the checks list which files the change affected, as far as the platform records the links between files. The analogy is one change made across several services of a software system in one repository and tested as a whole. The difference is that an engineering change crosses disciplines with their own tools, standards and responsible people, so the links between files have to be recorded explicitly.

When such projects are open and interoperable, engineering knowledge can accumulate. Designs, models, decisions and evidence that are today rebuilt within each organization could be inspected, reused and extended by others, and work at scales that were impractical, such as broad searches over design alternatives or the reassessment of many designs when a shared assumption changes, could

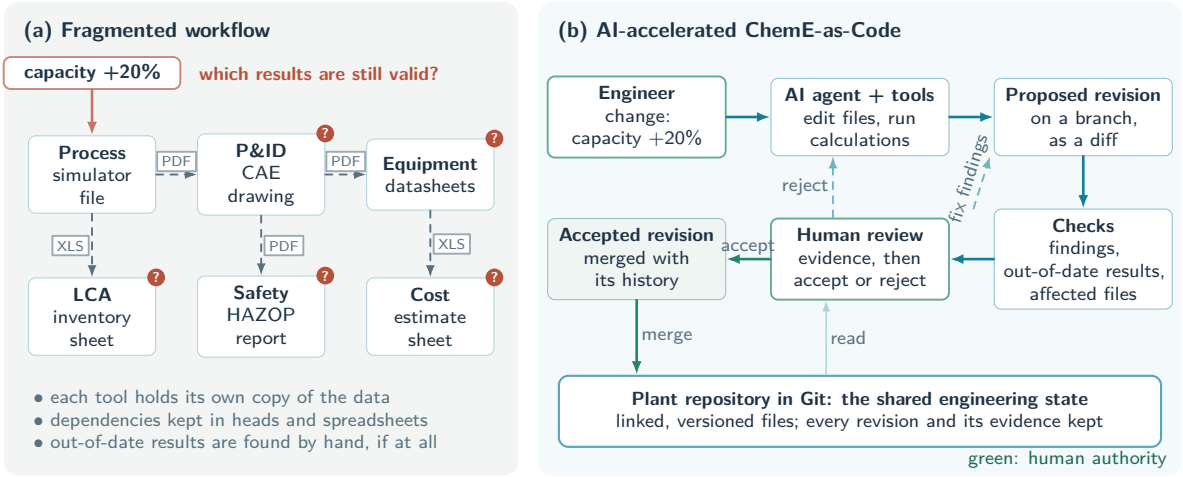



Figure 2: The same engineering change in two workflows. (a) Fragmented workflow: a capacity change enters one tool, and which downstream results are still valid has to be worked out by hand. Many industrial tools are integrated, but the consistency of a whole revision across disciplines is difficult to check. (b) AI-accelerated ChemE-as-Code: an engineer states the change, an agent with deterministic tools proposes a revision on a branch, the checks report findings, out-of-date results and affected files, the agent fixes what it can, and a person accepts or rejects the revision. The plant repository holds the shared engineering state, and accepted revisions are merged with their history and evidence.

become possible (Section 6). These shifts do not depend on a particular technology. Version control, plain-text formats, exchange standards such as DEXPI and today's agents are implementation choices. The paradigm is that engineering knowledge is accessible to computational agents in a shared environment where every action is traceable, every result can be checked, and people decide.

# 4 The ChemE-as-Code Platform

ChemE-as-Code combines a shared plant repository with calculation tools, skills for agents and automated checks. Table 1 defines the six terms that the rest of the paper relies on.

The platform consists of three open repositories. The *core* is the toolkit: its `cheme` command validates, formats, renders, compares and generates engineering files, and it includes the skills that instruct agents and a browser *viewer* (`cheme serve`). The *project template* is the starting layout for a new plant repository. The *process library* holds complete plants, including the case study of this paper. A plant is one Git repository of plain files for every discipline, from the design basis through process, equipment, piping, control, safety and layout to cost and the life cycle inventory (LCI), which engineers and agents edit and the core reads, checks and extends (Figure 3). This means that an agent sees the whole plant at once, as a coding agent sees a whole software repository rather than one file [68, 69]. A change across disciplines is one revision, reviewed as one diff and reverted as a whole, and the checks compare files of the same revision with each other. Large software organizations keep most of their code in a single repository for related reasons, namely unified versioning and atomic changes across projects [81]. At version 1.1.0 the core comprises about 143,000 lines of Python with about 3,000 test functions. SI Section S2 gives its detailed architecture, and SI Section S4 lists its tools and skills by scale.

Table 1: Six terms used in this paper, with the software practice each one resembles. The analogy transfers the way work is organized, not the engineering content. A full glossary is given in SI Section S4.

| Term | Meaning in this paper | Software analogy |
|---|---|---|
| Check, finding | A program tests one rule over the files; where the rule fails, it reports a finding with the file, the line, a severity and a hint | Automated test; failing test |
| Skill | A short instruction file that tells an agent when and how to perform an engineering task, which tools to use, and what is left to people | Runbook, contributor guide |
| Tool | A deterministic program that calculates or checks; it never calls an LLM | Library, compiler |
| Provenance | The record of where a value came from: given (with a source), assumed (with a reason) or calculated (by a named tool) | Lock file that records the origin of each dependency |
| Fingerprint; out of date | A hash of a calculated value's inputs; when the inputs change, the value is out of date until it is recalculated | Build cache, invalidated when a source changes |
| Gate, sign-off | A gate is a set of checks a project stage must pass; a sign-off is the recorded acceptance of a stage by a reviewer | Required status check; merge approval |

### 4.1 Engineering Files in Git

Git keeps versions of files and records who changed what, when and why. It does not know what the files mean. Git does not distinguish a pump from a pipe, and it merges text without regard to engineering meaning, so two revisions that are each consistent can merge into one that is not. Suppose one revision renames pump P-301A to P-303A in the equipment list, and another, made in parallel, adds a line to P-301A on the P&ID. Git merges both without complaint, and the merged state names a pump on the P&ID that the list no longer has. ChemE-as-Code adds the engineering meaning that Git lacks. Each file declares its kind and the schema it follows, items refer to each other by tag (the number of a piece of equipment or a line, such as H-101 or NG-1003), and the checks run on every proposed revision in continuous integration. In the example, `cheme validate` reports that the P&ID refers to a tag that is not in the equipment list, with the file and the line, before the revision can be accepted.

The following standards and formats are used. P&IDs are stored in DEXPI 2.0.1 [15], based on ISO 15926, and handled with pyDEXPI [37]; the BFD and PFD are generated as DEXPI process models from the flowsheet. All other engineering information is held in YAML files, each validated against a versioned JSON Schema that the file declares; the declaration is added to older files by `cheme migrate`, the mechanism through which files will move to later schema versions. Decisions are kept in a decision log (`DESIGN.md`), reviews as Markdown records, and stage sign-offs in a generated status file. Three properties make these files suitable for agents and for review. They are *canonical*: `cheme fmt` writes DEXPI XML with a readable identifier on every object, so that a P&ID sheet of tens of thousands of lines still yields a small, reviewable diff. They are *linked*: list items refer to P&ID objects by their tags, flowsheet nodes link to equipment and sheets, and HAZOP studies record the revision of the sheets they analyzed. Relations are recorded only where a check needs them, not in a universal object model, so an ontology, a knowledge graph or an EDMS can be built from a plant repository. They are *typed*: every YAML kind has a schema, so a wrong unit or a missing field is a finding, not a surprise downstream.

The files at one commit are the engineering state of the plant and its source of truth. A person or an agent changes it by proposing a revision, normally on a branch, which becomes the next state

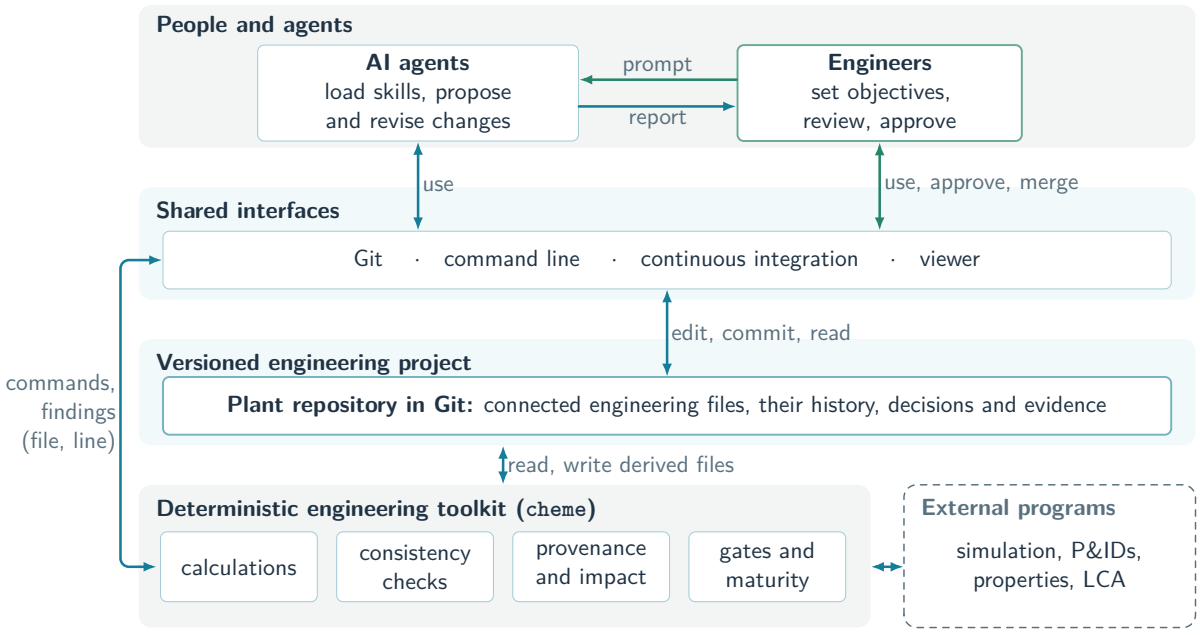



Figure 3: Architecture of ChemE-as-Code in four layers. Engineers prompt AI agents, which load skills and report back, and both work on a versioned plant repository in Git through the same shared interfaces (Git, command line, continuous integration, viewer). Engineers accept and approve changes (green: human authority, a policy that the sign-off rules support and that branch protection can enforce). The deterministic toolkit calculates and checks; it prints every finding with file and line on the command line and in continuous integration, where the agent and the engineer read it. For simulation, P&IDs, properties and LCA it calls established external programs (DWSIM, pyDEXPI, thermo and CoolProp, Brightway). It contains no LLM calls. SI Section S2 gives the detailed architecture.

once accepted in review and merged. In the proposed workflow, acceptance is a human decision, and the change is merged by the reviewer, not by its author; the platform does not enforce this yet, but branch protection on the Git host can. In the case study (Section 5), a reviewer agent accepted the changes, so the design is agent-generated, with the author's recorded interventions and no independent human review. Such an acceptance lets a change merge; it is not the approval of a responsible engineer.

Every file is either *authoritative* or *derived*. Authoritative files are written by people or agents: the design basis, given and assumed values, P&IDs, lists, layout inputs, HAZOP records, decisions and reviews. Derived files are generated by a tool: the BFD and PFD from the flowsheet, simulation runs and calculated values, the 3D plant model, the material take-off (MTO), estimates and LCI datasets. Derived files are never edited by hand. Where implemented, they record a fingerprint of their inputs and the version of the tool that generated them, and `cheme validate` reports a file or value whose inputs no longer match as out of date, as it reports a hand edit of a generated sheet. A calculated value never overwrites a given or assumed one: a tool writes only the values it calculated itself, so if a given flow no longer closes a balance, a check reports the imbalance instead of a tool correcting the input. Nothing is regenerated, rewritten or declared invalid automatically. A derived file stays as it is, marked out of date, until a person or agent runs the command that regenerates it, and regenerating it with the same inputs and the same tool version gives output that is identical to the byte (SI Section S4).

### 4.2 Tools and Skills for Agents

The core separates what agents propose from what tools compute. Its engineering domains (chemistry, process, equipment, piping, control, safety, spatial and assessment) contain no calls to LLMs. Calculated values enter the files only through deterministic commands or recorded scripts; in the case study, the agents wrote some of these scripts themselves, which Section 6 discusses. Established programs do the heavy calculations: DWSIM solves the flowsheet, thermo, chemicals and CoolProp supply physical properties, pyDEXPI reads and writes P&IDs, and Brightway computes the LCA [82]. The core's own tools cover what sits between them: balances and propagation, shortcut sizing of exchangers, pumps, compressors and columns, the 3D model with pipe routing and clash detection, HAZOP rating, cost estimation from published correlations, and TEA. SI Section S4 lists them by scale.

The agent's interface is the command line and the files themselves, the same interface a software engineer uses. The loop is simple. The agent edits a file, runs `cheme fmt` and `cheme validate`, reads the findings, fixes what they report, and validates again. Each finding is one line with a stable identifier, the rule, a severity, the file and, where one applies, the line, and a hint. An agent can therefore act on a finding as it would on a failing test. The engineer sees the same lines in continuous integration and in the viewer.

Skills carry the engineering procedure. A skill is a short instruction file in the plant repository that tells the agent which inputs a task needs, how to carry it out, which tools to call, and which decisions are left to people. The 109 skills of version 1.1.0 span the scales of a project, from component properties and reaction kinetics through thermodynamic-model selection, flowsheet simulation and reactor screening to equipment sizing, P&ID authoring, control, 3D layout, HAZOP, cost estimation, TEA and LCA. About half of them are process knowledge: unit operations, utilities, heat integration, and the technologies of gas processing, refining, petrochemicals and heavy chemicals.

The skill for selecting a thermodynamic model shows the division of labor between skill, tool and person. It is used before any phase calculation and before a property method is recorded in the materials file. The skill tells the agent to establish the facts first: the pressure range, whether the mixture is polar, whether electrolytes are present, whether interaction parameters are available, and whether a second liquid phase or vapor association is expected. The agent fills in only what the given information or well-established chemistry supports and leaves the rest unset. The tool behind the skill is a rule engine that implements Carlson's published decision tree. If a fact it needs is missing, it refuses and names the missing fact as a question, which the agent answers or escalates. The result is a recommendation with its reasoning, recorded with provenance, and the skill states that it is not a validated choice. In the ammonia case, this skill recommended a predictive equation of state or a cubic equation of state with Huron–Vidal mixing rules for the polar loop mixture at 155 bar. The simulation route could not run either, so the agents used Peng–Robinson as a substitute and recorded the decision, and the calculation maturity later reported these mixtures as outside the benchmarked range of the method (SI Section S8). The HAZOP skill works in the same way: it drafts nodes, deviations and safeguards, the rating tool looks up the risk band in the matrix, and every entry stays marked as proposed until a HAZOP team accepts it.

This division means that engineering expertise becomes accessible to agents without being embedded in an LLM. The numbers come from tools that can be tested and versioned. The procedures come from skills that engineers can read, review and improve like any other file. The decisions stay with people.

### 4.3 Checks, Gates and Human Approval

Agents produce more revisions than an engineer can read, so the checks tell the engineer where to look and what not to trust yet. Each check tests one rule that a program can decide, in four layers [22]: every file matches its schema (a design temperature is a number with a unit); files agree with each other (a line's diameter in the line list matches the P&ID that draws it); balances and simulations hold (every mass balance closes within its tolerance); and selected safety requirements are met (a trip does not share its transmitter with the control loop it protects against). The core defines 466 rules (SI Section S3), and every finding names the file and, where one applies, the line.

A design change shows how the checks work together. Suppose an agent raises the design capacity by 20% and the feed flows with it. The balance rules report that the flowsheet no longer closes; every calculated result whose input dependencies are recorded, from stream properties to the simulation run, is reported as out of date through its fingerprint; and `cheme impact` lists the files and reviews that need attention. Nothing is recalculated on its own. The agent or the engineer reruns the relevant tools and checks whether the affected results are now current, and the reviews that the change made out of date are redone. This sequence (check, identify the affected work, recalculate, review) follows only the dependencies the platform records; a duty copied by hand from the flowsheet into a datasheet is not among them (Section 5.5).

The checks support the decision whether work can proceed to the next project stage; they do not take it. Each stage has gates (Table 1), for example that every item on the equipment list has a datasheet marked final, and a stage is signed off only when its gates pass and no serious review finding is open (SI Section S3). The platform also reports calculation maturity, a program-evaluated summary of which calculation requirements have been met (SI Section S3). Passing these requirements establishes only that the conditions encoded in them have been met, not that the design is right or that an engineer has approved it. Approval is reserved for people in the proposed workflow; the core records reviews but does not enforce this, and in the case study it accepted sign-offs on agent reviews.

## 5 Ammonia Case Study

We demonstrate the platform on a natural-gas ammonia plant, following the design from its initial requirements through simulation, engineering documents and safety studies. We chose ammonia because its design couples many disciplines, from reaction equilibria and recycle convergence through equipment, piping, instrumentation and layout to cost estimation. We ask whether agents can use the platform to develop a connected engineering project (RQ1, RQ3), how far the checks identify errors and incomplete changes, whether derived files can be regenerated (RQ2), and whether an agent can revise the plant through the same workflow (RQ2, RQ3). Three kinds of success have to be kept apart: execution, meaning that agents generated and modified a multidisciplinary project; formal verification, meaning that the implemented rules and gates checked the conditions they cover; and engineering correctness, which this case does not establish, because no independent engineer has reviewed the design.

### 5.1 Setup

Agents developed the plant in two stages (Table 2): a concept stage of three sessions that produced the design basis, a balanced flowsheet and the equipment list, and an engineering stage that started from a written brief by the author (SI Section S5). In the engineering stage, an orchestrating session ran one implementation job per section, each followed by a reviewer job in a separate session that

returned the work for correction or merged it once continuous integration had passed. These are workflow roles of one general-purpose agent, not discipline-specific agents. Unlike in the proposed workflow, reviewer agents accepted the changes, so no engineer's acceptance enters the record. The author's interventions were the brief, the budget and stop rules, and the decisions the agents escalated (SI Section S9). The evaluations of Section 5.5 ran on the reported state, process-library tag `nh3-run-final`, where every file of the case is public.

Table 2: The ammonia case study at a glance. Sources: the run records and the engineering files at process-library tag `nh3-run-final`; details in SI Sections S5 to S10.

| | |
|---|---|
| Plant | 1,500 t/d of refrigerated liquid ammonia from natural gas in the Rotterdam area: desulfurization, prereforming, fired primary and air-blown secondary reforming, water-gas shift, $CO_2$ removal, methanation, drying, 150 bar synthesis loop with refrigeration and purge-gas recovery; electric drives, steam export |
| Input from the author | Concept stage: a task and a literature selection in a process-library issue. Engineering stage: a written brief ("take ammonia-1 to a complete design: simulation, basic engineering, HAZOP, datasheets, relief"), budget caps and stop rules |
| Stages | Concept stage, 27 October 2026: 3 sessions, 40 min. Engineering stage, 29 to 30 October 2026: 60 sessions, about 27 h |
| Agent and roles | Claude Opus 5.5 in every session; the roles implement, review, fix, sign off and merge differ only by instructions, section task and loaded skills |
| Human interventions | Eight in the engineering stage, counting the brief: budget and stop rules, two platform bug fixes outside the design, and the decisions the agents escalated |
| Output | Process: flowsheet and converged DWSIM simulation. Engineering: DEXPI P&ID sheets, equipment, line and instrument lists, control loops, datasheets, routed 3D layout. Safety: HAZOP study, relief cases. Assessment: AACE class 4 estimate, TEA, LCI. Records: decisions and reviews (counts in SI Section S9) |
| Checks at the reported state | 0 errors, 268 warnings; all seven concept and 13 basic gates pass, three of five detail gates; calculation maturity *balanced*, the second of four levels (SI Section S3) |
| Cost | USD 255 of recorded API cost for 63 sessions, of which USD 224 in the 56 design sessions (Table 3) |
| Reviewed by | Reviewer agents of the same model; no independent engineer |

## 5.2 The Engineering Package

The engineering stage turned the concept-stage flowsheet into a connected package: a converged simulation, P&IDs with their lists, control loops and datasheets, a routed 3D layout, a HAZOP study, relief cases, a class 4 estimate with TEA, and an LCI, in one repository whose files refer to each other by tag (Table 2). It passed every concept- and basic-stage gate and three of the five detail gates with no error-level finding. Figure 4 follows one piece of equipment, the reformer furnace H-101, through these files: the 3D model, the P&ID, the HAZOP study and the links between them. Rules check that its tag refers to the same item in the flowsheet, the equipment list, the P&ID, the datasheet, the layout and the HAZOP study, and the HAZOP study records the revision of the sheet it analyzed, so a later change to the sheet marks the study as needing review. The figure also shows a limit of these links. The datasheet and the cost basis state a duty of 68,616 kW, whereas the flowsheet changed later and now gives 69,829 kW; no rule compares the two, so no finding reports the difference.

The techno-economic assessment gives a levelized cost of 688 EUR/t against 439 EUR/t assumed for imported ammonia. Under its assumptions, 2025 Dutch industrial energy prices and a US Gulf Coast import price without freight (SI Section S7), the plant would not compete with imports. The

LCI records every exchange of the flowsheet per kg of ammonia and gives a gate-to-gate carbon footprint of 1.46 kg $CO_2$-eq per kg, with 6% of the burden allocated to the exported steam by exergy. This counts only the plant's own emissions: the background datasets for natural gas supply, electricity and utilities are deliberately left unlinked as open gaps, so the figure is a screening result, not a cradle-to-gate footprint.

The package is preliminary, not an engineering-approved basic design. The basic gates passed only after the author accepted two layout exceptions that the agents had escalated rather than decided (SI Section S9), the stage sign-offs rest on agent review records, and review findings remain open. The calculation maturity reaches *balanced*, not *predictive*: several reactors and separators use specified conversions and splits, the furnace, refrigeration and steam system are evaluated by agent-written scripts on simulated duties, and the property method has not been benchmarked for most of the mixtures (SI Sections S7 and S8). Three of the eight relief valves still lack a required capacity, and every HAZOP entry is an agent's proposal, because no HAZOP team has met.

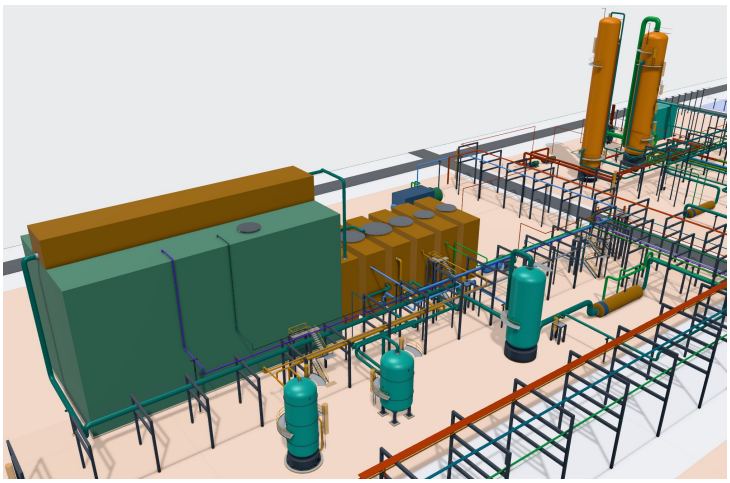
Reviewer agents recorded 2 high, 39 medium and 52 low findings in 26 review sessions, and 13 times over 7 review rounds they sent work back for fixing. The findings included engineering defects as well as formal ones: exchangers with a large pressure step between their sides had no tube-rupture case, and heat-transfer coefficients for gases near atmospheric pressure had been used for the 150 bar loop. As the brief required, eight jobs stopped and reported a blocker instead of relaxing a gate. One job, however, marked datasheets final so that a gate would count them, and a reviewer caught this. The run also found eight defects in the platform software, of which three were fixed by agent jobs outside the design (SI Section S9). These reviewers use the same model as the designers and are therefore not independent [22].

### 5.3 Cost and Effort

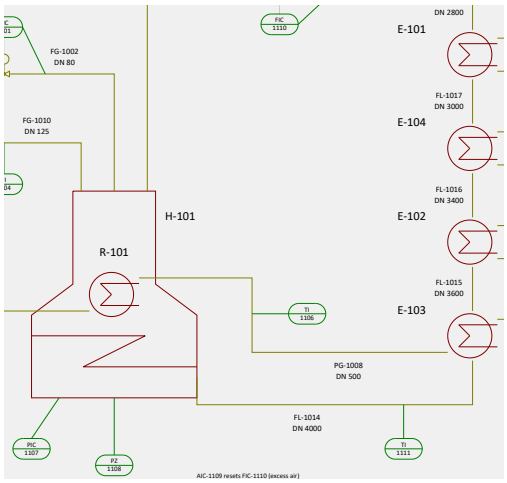
The recorded run used 63 agent sessions at an API cost of USD 255.31 (Table 3): USD 12.02 for the concept stage (3 sessions, 40 minutes) and USD 243.30 for the engineering stage (60 sessions, about 27 hours). Layout and its downstream sections took 40% of the design cost and 26 of the 56 design sessions, because they needed the most iteration. Review and fix jobs took about a quarter of the design cost, and nearly half when the close-out jobs (USD 44), which are booked as implementation, are included. These figures describe one execution with a particular model and heavy caching (565 million cached input tokens), on a first-generation toolchain and with a workflow nobody had practiced before. They bound neither the best nor the typical case, and a more mature platform and more practiced briefs will change them. An order-of-magnitude estimate of the inference energy from the recorded tokens and published per-token figures [83, 84] is given in SI Section S9. The cost figures exclude the orchestrating session, whose own API cost was not recorded, as well as human expertise, infrastructure and independent review. They are not evidence of productivity compared with an engineering team.

### 5.4 Plausibility Against Open References

To obtain an outside reference, we compared eight of ten planned indicators with the EU reference document on best available techniques (BAT) for ammonia (BREF) [85] and the International Energy Agency (IEA) ammonia roadmap [86] (Table 4; the two NOx indicators are not modeled, and the net energy appears twice, as delivered and with electricity at an assumed 40% conversion). The reference values were written before the engineering stage and withheld from the design jobs. Overall, the design appears plausible: process $CO_2$, loop pressure and conversion per pass lie inside the published ranges, and the energy indicators lie close to them. The comparison is only an indication,



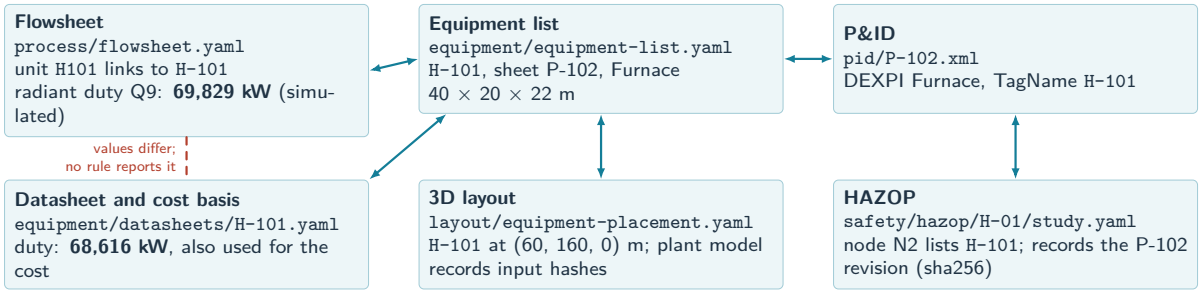
(a) 3D plant model: H-101 (green block), its convection section (brown blocks) and the surrounding plant




(b) P&ID P-102, excerpt around H-101, R-101 and the exchangers

| Deviation | Cause | Consequence | Safeguards | Risk: initial, residual | Actions |
|---|---|---|---|---|---|
| S5: More temperature (**R-101** tubes) | TIC-1103 fails low or reads low (or PIC-1101 fails, PV-1101 sticks open); **H-101** is over-fired | **R-101** tube walls exceed their design temperature; tube rupture; process gas burns in the firebox | SG7 TIC-1103 high alarm (not credited, part of the causing loop); SG8 TIC-1103 resetting PIC-1101 | people 3/C high, asset 4/C high; residual not rated | A6 burner management; A7 independent trip, tube skin temperature |
| S7: less combustion air, draft | ID fan K-103 trips or its damper closes; **PIC-1107** holds the last speed; furnace at positive pressure | Hot flue gas and flames at peepholes and openings; burns to people on the burner platforms; incomplete combustion | SG9 **PIC-1107** high pressure alarm; SG10 interlock I-102: **PZ-1108** high-high closes XV-1102 (fuel gas) and XV-1105 (tail gas) | people 3/C high; residual people 3/B medium | none |

(c) HAZOP node N2, two scenarios shortened from the worksheet; deviations are HAZOP guideword phrases; proposed by agents, not reviewed by a team; tags in blue appear in (b)




(d) H-101 across the linked files; solid arrows are references checked by rules

Figure 4: The reformer furnace H-101, one piece of equipment, followed through the files that describe it: the 3D model (a), the DEXPI P&ID (b), the HAZOP study (c) and the links between the files (d). Panel (d) also shows a limit of the checks: rules check that a tag refers to the same item across files, while a duty that differs between two files is not reported. All drawings, the full HAZOP worksheet source and further 3D views are given in SI Sections S6 and S10 or the repository. The design has not been reviewed by independent engineers.

Table 3: Recorded API cost of the ammonia run per section in US dollars, split into implementation jobs and the review, fix, sign-off and merge jobs that followed them. Values are rounded. Data: the run ledger at `nh3-run-final`; per-session values in SI Section S9.

| Section | Sessions | Implementation | Review | Total |
|---|---|---|---|---|
| Concept stage: design basis, balanced flowsheet, preliminary LCI | 3 | $8.71 | $3.30 | $12.02 |
| S1 Concept close-out and sign-off | 5 | $6.50 | $2.61 | $9.11 |
| S2 Process simulation (DWSIM) | 3 | $28.03 | $2.41 | $30.44 |
| S3 P&IDs, 14 sheets | 6 | $28.04 | $6.59 | $34.63 |
| S4 Control and lists | 2 | $5.07 | $1.95 | $7.01 |
| S5 Layout, MTO, estimate, TEA, basic sign-off | 26 | $57.26 | $32.23 | $89.50 |
| S6 HAZOP | 4 | $6.12 | $5.71 | $11.83 |
| S7 Datasheets | 2 | $6.36 | $1.73 | $8.09 |
| S8 Relief | 2 | $5.06 | $1.70 | $6.76 |
| S9 Detail gates | 3 | $12.43 | $1.68 | $14.11 |
| Design, subtotal | 56 | $163.58 | $59.92 | $223.51 |
| Platform software fixes (outside the design) | 6 | $20.13 | $9.55 | $29.68 |
| Key performance indicators | 1 | $2.13 | – | $2.13 |
| Total | 63 | $185.84 | $69.47 | $255.31 |

for three reasons. First, the references leave their boundaries and conversion factors partly open. The net specific energy is 26.97 GJ/t with electricity counted as delivered and 30.1 GJ/t at an assumed conversion efficiency of 40%, against a BAT range of 27.6–31.8 GJ/t (±1.5 GJ), and the comparison cannot establish whether the design is more or less efficient than a BAT plant. Second, the design imports more electricity than the IEA's reference plant (2.1 against 0.3 GJ/t), because all its machines are electric, which lowers its natural-gas figure, and its lower steam export is not a merit, since its furnace, steam and refrigeration models are simplified. Third, the comparison was not fully blind, because a design agent opened the BREF during the run. We found no design value that moved toward the reference figures, but we cannot exclude influence (SI Section S11).

### 5.5 What the Checks Find and Miss

Three experiments test the checks and the regeneration of derived files on fresh copies of the reported state (`cheme` 1.1.0, DWSIM 9.0.5), and a fourth asks an agent to change the plant. Unmodified, the state gives 0 errors and 268 warnings and passes 23 of 25 gates. Table 5 summarizes the results; SI Sections S12 and S13 list every fault and change.

**Planted faults.** To see what the checks find, we injected one fault at a time into a copy of the files and ran the checks. We first wrote 19 faults knowing the rules, aimed at schema, consistency, physical, out-of-date and review rules, such as a line enlarged in the line list but not on the P&ID, or a feed temperature changed without rerunning the simulation. All 19 triggered their intended rule, with a finding at the edited file in 12 cases and at the edited line in 10. This confirms that the rules fire on the faults they target, but it says nothing about faults nobody wrote a rule for. We therefore had a separate agent session write 12 further faults on a copy of the engineering files from which the checks, skills and reviews had been removed, so that it could not see the rules. We count a fault as identified when a new finding names the wrong condition itself; a finding that only says that a file changed or that a result is out of date does not count. By this definition, the checks identified 4 of the 12: a mistyped relief-valve tag, a vessel placed on top of another, a hand-edited HAZOP risk band and a trip sharing its transmitter with the control loop. Five raised only a generic change

Table 4: Key performance indicators of the ammonia design against open references, for plausibility only. Per t of ammonia; lower heating values; electricity as delivered (3.6 GJ/MWh) unless stated; exported steam at its enthalpy above make-up water. The references leave boundaries and conversion factors partly open, so comparability is limited (SI Section S11). BAT: best available techniques.

| **Indicator** | **Design** | **Reference** |
|---|---|---|
| Net energy, GJ/t | 26.97 | 27.6–31.8 ±1.5 (BREF BAT); 27.6 (IEA BAT) |
| Net energy, electricity at 40%, GJ/t | 30.14 | 27.6–31.8 (BREF BAT); 28.05 (IEA, if delivered; our inference) |
| Gross energy, GJ/t | 30.13 | 32.4 (IEA BAT) |
| Natural gas, GJ/t | 28.01 | 29.3–31.1 (BREF reported, feed plus fuel, not BAT); 32.1 (IEA BAT) |
| Process $CO_2$, t/t | 1.164 | 1.15–1.40 (BREF) |
| Direct $CO_2$, t/t | 1.558 (1.84 with purchased power at 475 g/kWh) | 1.8 (IEA BAT) |
| Net steam export, GJ/t | 3.16 | 4.8 (IEA BAT) |
| Loop pressure, bar(a) | 154 | 100–250 (BREF) |
| Conversion per pass, % | 25.5 on all gas (30.2 on $H_2$+$N_2$) | 20–30 (BREF, basis undefined) |
| NOx in flue gas, mg/$Nm^3$ | not modeled | 90–230 (BREF BAT) |
| NOx factor, kg/t | not modeled | 0.29–0.32 (BREF benchmark) |

signal, among them a relief valve set above its vessel's design pressure and plain carbon steel chosen for a −33 °C drum. Three raised nothing, including a let-down valve changed to fail open. Four further planted values that were implausible but self-consistent, such as a datasheet pressure off by a factor of ten, also raised only change signals. This means that the checks find what contradicts an implemented rule, but not an error that no rule covers, even when another file states the conflicting value.

**Engineering changes.** To see how far the checks reach, we made five engineering changes on separate copies, from a re-rated pump to a 20% capacity increase, and ran the checks once after each. Beforehand, a separate agent session had listed, for each change, the files it expected to be affected and whether each needed an edit, a regeneration or a review; no engineer has checked these lists, so the numbers are indicative. The checks flagged or named 13 of 22 expected files for a re-tagged pump and 5 of 161 for the capacity increase, and fewer than half for the other three changes. After the capacity increase, the checks raised balance errors, marked the simulation and 105 stream values as out of date, lowered the calculation maturity and refused the PFD rebuild, but they did not flag most downstream disciplines, because the checks stop at the first broken layer until it is repaired. This means that one pass of the checks shows the engineer where to start, not everything a change will touch.

**Reproducibility.** We regenerated derived files with fixed software versions. The six generators tested (BFD and PFD, plant model, one P&ID rendering, two datasheet drawings, LCA, MTO), two DWSIM runs and library builds from one commit gave output identical to the byte [79]. After a feed-temperature change, a 72 s simulation rerun cleared all 106 out-of-date findings. On another operating system, the BFD, PFD, plant model and the two datasheet drawings regenerated identically, whereas the LCA and MTO files differed in their line endings, a defect in the commands that write them. The derived files are therefore reproducible with fixed software versions, apart from these line endings.

**Agent-driven change.** Finally, one unattended agent session was asked to raise the capacity to

1,800 t/d and propagate the change as far as it could on a budget described as tight, resolving what the checks report and listing the equipment it judged must be resized, without relaxing a gate or signing off (SI Section S13). In 49 minutes and for USD 8.55 of a USD 19.90 cap, it raised the design capacity and the feed flows, re-simulated the plant to 1,799 t/d with closed balances, and updated control setpoints, datasheet duties, relief loads, the estimate and the LCA; it changed 128 of the 161 files on the expected-impact list. A separate rerun of the checks found 0 errors and 252 warnings, a calculation maturity of *balanced*, and four failing gates: the two detail gates that had failed before and two reviews that the change had made out of date. Yet the plant was not resized. The flows and duties are 20% larger on paper, but no P&ID or line changed, the equipment dimensions and the layout still describe the 1,500 t/d plant, and 36 lines exceed their design velocities by the agent's own screening, not by any check, because no rule compares a flow with a line size. The agent reported this and stated that it would not trust its costs. The tight budget may have contributed, so this result does not measure the most that agents can do. The outcome is mixed: an agent carried out substantial cross-disciplinary work through the shared repository, but the checks follow only the dependencies the platform records, so zero errors did not mean that the change was complete, and a reviewer was still needed.

Table 5: Summary of the experiments on the ammonia plant (process-library `nh3-run-final`, `cheme` 1.1.0); every fault and change is listed in SI Sections S12 and S13. Fault counts are coverage on these sets. "Reached" counts files of an expected-impact list, itself written by an agent, that a finding flagged or named when the checks were first run after the change. These are internal checks and do not establish physical or industrial validity.

| **Experiment** | **Result** | **Description** |
|---|---|---|
| *Planted faults* | | |
| Faults written knowing the rules | 19/19 | intended rule fired; these rules work as designed |
| Faults written without seeing the rules | 4/12 | 5 more raised only a change signal, 3 raised nothing |
| Implausible but self-consistent values | 0/4 identified | e.g. a datasheet pressure off by a factor of ten; only change signals |
| *Engineering changes (files reached when first checked)* | | |
| Pump re-rated; re-tagged | 4/19; 13/22 | re-tagging reaches the most; spare, cost and electrical not reached on a re-rating |
| Line enlarged | 6/14 | HAZOP input and 3D model flagged; list and P&IDs named |
| Feed gas changed; +20% capacity | 4/115; 5/161 | balance errors end the first pass before most later disciplines are checked |
| *Reproducibility* | | |
| Same commit, same environment | identical | six generators tested, library builds, simulation |
| Other operating system (subset) | identical | BFD, PFD, model, two datasheet drawings; LCA and MTO differ in line endings |
| *Agent-driven change* | | |
| +20% capacity, one session | 128/161 changed | 0 errors, but no P&ID or line resized and 36 lines above design velocity |

# 6 Discussion and Future Perspectives

We return to the problem of Section 1: agents produce engineering work faster than people can check it. The case shows one way to organize that work and where it still falls short.

**AI acceleration and the practice of engineering.** In the proposed workflow, agents act on findings, people review a diff with its evidence, and progress is judged by the evidence instead of the number of drawings. We argue that this is the precondition for acceleration: without it, faster drafting only produces more unchecked documents. How much of the acceleration survives review remains unclear. The length of software tasks that agents complete on their own has doubled about every seven months since 2019 [6], and most technology professionals report that AI has increased their productivity [87], but such gains depend on the task, and this study does not establish a net time saving once engineering review is counted. In the ammonia case, a package that passed the basic-stage gates cost USD 224 in API calls for its design sessions, but its reviewers sent work back 13 times, the last gates needed decisions from a person, and an independent engineer would still have to review the result. We expect the cost of first drafts to fall much faster than the cost of checking and review, which would then set the pace and change what teams must build and education must teach [22]. Controlled studies against engineering baselines, comparable to SWE-bench for software agents [69], are in preparation for process design, HAZOP and other tasks.

**Known and novel processes.** Ammonia from natural gas is among the best-documented processes in chemical engineering, and the route was set in the first task. The case therefore tests whether agents can carry a known process through engineering work consistently, not whether they can conceive a new one. We believe that particular care is needed with novel processes, for three reasons. LLMs draw on published knowledge and may fall back on familiar flowsheets where a different one would be better. New processes lack the kinetic, property and cost data on which the deterministic tools depend, so more of the design rests on assumptions. And the rules and gates encode established practice: they can show that an unconventional design is inconsistent, not that it is good, and they may flag a sound new idea as an error. Cheap and traceable iterations may still help creative work by letting many more alternatives be generated and screened, with people judging which are worth pursuing; testing this needs design tasks on processes published after the model's knowledge cutoff, or hypothetical routes judged by experts.

**Consistent is not correct.** Keeping calculations outside the LLM makes them repeatable and testable, but a deterministic calculation with a wrong assumption, an unsuitable property method or an incomplete model gives the same wrong result every time. The ammonia case illustrated this in part: the energy indicators looked favorable until their boundaries were examined, a wrong design value that no rule covers raised no finding that named it, and a capacity change raised no errors although the plant had not been resized. The agents also wrote some of the calculation scripts themselves. This means that the checks establish consistency within the dependencies the platform records, and an engineer must still judge the assumptions, the models and what the rules do not cover. The missed faults point to rules that compare values that today only change together, such as a datasheet duty against the flowsheet; a rule is a small piece of deterministic code, and benchmarks with planted faults on open plants would show how far the coverage has come.

**An open ecosystem.** Much engineering knowledge sits in proprietary project files, vendor-specific tools and internal databases. ChemE-as-Code keeps it in open, documented formats, so a project remains readable without any single application, tools and skills from different organizations can work on the same files, and commercial systems, whose scope and certification open tools do not replace, can read and write them through DEXPI and CAPE-OPEN [15, 40]. Open projects matter

most. Complete, machine-readable plants with their models, drawings, decisions, safety studies and inventories have rarely been openly available [55, 56]. They could serve as benchmarks, teaching material and shared expertise [22], and because their assumptions and open points are public, engineers can review them by raising issues and proposing changes. Life cycle inventories derived from such projects can be recalculated from their models, which established databases such as ecoinvent [88] do not expose. The Process Atlas, an open database of process models and LCIs generated from the process library, is a forthcoming application [89]; its breadth and data quality are not results of this paper.

**Risks.** Three risks follow directly from this work [22]. The first is over-trust. In the ammonia case, a capacity change produced zero errors although the plant had not been resized; an engineer who reads a clean check as a complete design is misled. The platform reports calculation maturity and open warnings next to the errors and records who reviewed what, but the habit of reading the evidence rather than the status has to be built. The second is liability and certification. When an agent drafts a design and an agent accepts it, no person has taken responsibility for it, and current codes assume that one has. Rules are needed on who approves an agent-made design, which record of the agent's work an approval must cite, and which checks a certifying body accepts as evidence; the run records, review records and sign-off rules of this platform are a starting point, not a substitute. The third is the training of engineers. Junior engineers learn much of their craft through the routine checking and data transfer that agents now take over, and the profession could lose the judgment that review needs unless this experience is built in new ways, for example by having junior engineers review and challenge agent work together with its evidence. An error in a shared skill, rule or template could also spread to many designs at once, and machine-readable plant models raise questions of cyber security and intellectual property, which we note without treating them here.

**Limitations.** The case study is illustrative and agent-generated, with the author's recorded interventions. Its reviewers are agents of the same model as its designers, so their findings show what the model catches in its own work, not what an engineer would catch. The checks establish consistency within the rules that exist; engineering judgment and plausibility, mechanical design, relief sizing, dynamics, the correctness of a HAZOP rating and dependencies not yet fingerprinted, such as datasheet duties, lie outside them, which is why the held-out faults and the capacity change found what they did. The case uses one general-purpose agent in workflow roles and offers no controlled comparison with other agent architectures or with the same agent without skills, the comparison that would show what the skills contribute. Proprietary tools hold most industrial data, DEXPI coverage is incomplete, and agents are nondeterministic, so two runs of the same brief give different designs. ChemE-as-Code is a first open research prototype, and these limits should be read in that context. The same pattern could extend to the operation of existing plants, to laboratory work where experiments and synthesis procedures are already written as code [90, 91], and to molecular and product design; the core is layered by domain so that such domains can be added.

# 7 Conclusions

AI agents offer the prospect of substantially accelerating chemical process engineering and reducing its cost, but realizing this requires more than capable models. We argue that engineering knowledge must be connected, machine-readable and checkable, so that agents can work across disciplines while engineers keep responsibility for decisions and approval.

ChemE-as-Code implements this through two foundations: a versioned repository of linked engineering files, and an open library of deterministic tools and agent skills that spans chemical engineering from catalysis and reaction kinetics to plant layout and LCA. A general-purpose agent uses these across disciplines, while calculations, checks and approval stay outside the LLM, with approval reserved for people in the proposed workflow. In an illustrative natural-gas ammonia case, agent sessions turned a written brief into a preliminary package that passed the platform's basic-stage gates, with HAZOP, datasheets and relief cases, for USD 224 of API cost in the design sessions, and a comparison with open references indicates that the design is plausible. The checks caught every fault written with knowledge of the rules but only some of the faults written without them, and a capacity change raised no errors although the plant had not been resized. An error-free result therefore shows consistency, not a complete or correct design. The case shows that the workflow runs. It does not establish the quality of the design or a productivity gain, and no independent engineer has reviewed the result.

Beyond automation, ChemE-as-Code lays a foundation for an open chemical engineering ecosystem in which process models, engineering data, decisions and the evidence behind them are reusable assets rather than remaining inside single tools and projects. The longer-term opportunity is to change how engineering knowledge is created, checked, shared and reused, first in process and plant design and then across chemical engineering.

## Data and Code Availability

ChemE-as-Code is developed openly in the ChemE-as-Code organization on GitHub (`https://github.com/ChemE-as-Code`); all repositories listed below are public. The core library, which validates, formats, renders and compares DEXPI P&IDs and engineering YAML files kept in Git and includes the web viewer for plant repositories, is available at `https://github.com/ChemE-as-Code/core` (GNU AGPL-3.0-or-later). It builds on pyDEXPI, available at `https://github.com/process-intelligence-research/pyDEXPI`. A template for starting new plant repositories is available at `https://github.com/ChemE-as-Code/project-template`; the template carries no license, so that the content of each plant repository remains its owner's to license. The process library, including the ammonia case study (`processes/ammonia-1`) and its run records at tag `nh3-run-final`, is available at `https://github.com/ChemE-as-Code/process-library` (code under GNU AGPL-3.0-or-later, except the LCI database installer, which is under the MIT license; process data and designs under CC BY 4.0 unless a process states otherwise). The project website is `https://cheme-as-code.org`, and the Process Atlas generated from the process library is available at `https://atlas.cheme-as-code.org`. The scripts and raw results of the evaluation in Section 5.5 are archived on Zenodo [92].

## Declaration of AI Use

The author used GPT-6 (OpenAI) and Claude Opus 5.5 and Claude Fable 5.1 (Anthropic) to assist with drafting, language editing, literature searches and figures. The author reviewed and revised all content, verified the sources and takes full responsibility for the manuscript.

## Acknowledgments

This research is supported by the Netherlands Organisation for Scientific Research (NWO) under the Veni grant (ID 20196), for which we express sincere gratitude. The author thanks the current

and former members of the Process Intelligence Research Group, whose work over the past five years contributed to this paper. In particular, the author thanks Sophia Rupprecht and Qinghe Gao for their prior contributions on multi-agent systems for separation process design.

## Conflict of Interest

The author declares no potential conflict of interests.

## References

[1] Artur M. Schweidtmann. Generative artificial intelligence in chemical engineering. *Nature Chemical Engineering*, 1(3):193, 2024. doi: 10.1038/s44286-024-00041-5.

[2] Benjamin Decardi-Nelson, Abdulelah S. Alshehri, Akshay Ajagekar, and Fengqi You. Generative AI and process systems engineering: The next frontier. *Computers & Chemical Engineering*, 187:108723, 2024. doi: 10.1016/j.compchemeng.2024.108723.

[3] Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6:525–535, 2024. doi: 10.1038/s42256-024-00832-8.

[4] Sophia Rupprecht, Qinghe Gao, Tanuj Karia, and Artur M. Schweidtmann. Multi-agent systems for chemical engineering: a review and perspective. *Current Opinion in Chemical Engineering*, 51:101209, 2026. doi: 10.1016/j.coche.2025.101209.

[5] Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. The impact of AI on developer productivity: Evidence from GitHub Copilot. Preprint at `https://arxiv.org/abs/2302.06590`, 2023.

[6] Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, Ryan Bloom, Thomas Broadley, Haoxing Du, Brian Goodrich, Nikola Jurkovic, Luke Harold Miles, Seraphina Nix, Tao Lin, Chris Painter, Neev Parikh, David Rein, Lucas Jun Koba Sato, Hjalmar Wijk, Daniel M. Ziegler, Elizabeth Barnes, and Lawrence Chan. Measuring AI ability to complete long software tasks. In *Advances in Neural Information Processing Systems*, volume 38, pages 102382–102435, 2025. doi: 10.52202/085713-3086.

[7] Moe Toghraei. *Piping and Instrumentation Diagram Development*. John Wiley & Sons and American Institute of Chemical Engineers, Hoboken, NJ, 2019. ISBN 9781119329343.

[8] Gavin Towler and R. K. Sinnott. *Chemical Engineering Design: Principles, Practice and Economics of Plant and Process Design*. Butterworth-Heinemann, Amsterdam, 2008. ISBN 9780750684231.

[9] International Organization for Standardization. ISO 10628-1:2014 diagrams for the chemical and petrochemical industry – part 1: Specification of diagrams. ISO, Geneva, 2014.

[10] Hülya Uzuner and Gerhard Schembecker. Knowledge based design of piping and instrumentation diagrams. *Chemie Ingenieur Technik*, 84(5):747–761, 2012. doi: 10.1002/cite.201100230.

[11] P. John Clarkson, Caroline Simons, and Claudia Eckert. Predicting change propagation in complex design. *Journal of Mechanical Design*, 126(5):788–797, 2004. doi: 10.1115/1.1765117.

[12] Maximilian F. Theisen, Kenji Nishizaki Flores, Lukas Schulze Balhorn, and Artur M. Schweidtmann. Digitization of chemical process flow diagrams using deep convolutional neural networks. *Digital Chemical Engineering*, 6:100072, 2023. doi: 10.1016/j.dche.2022.100072.

[13] Birgit Bayer and Wolfgang Marquardt. Towards integrated information models for data and documents. *Computers & Chemical Engineering*, 28(8):1249–1266, 2004. doi: 10.1016/j.compchemeng.2003.08.008.

[14] Jan Morbach, Andreas Wiesner, and Wolfgang Marquardt. OntoCAPE—a (re)usable ontology for computer-aided process engineering. *Computers & Chemical Engineering*, 33(10):1546–1556, 2009. doi: 10.1016/j.compchemeng.2009.01.019.

[15] David B. Cameron, Wilhelm Otten, Heiner Temmen, Monica Hole, and Gregor Tolksdorf. DEXPI process: Standardizing interoperable information for process design and analysis. *Computers & Chemical Engineering*, 182:108564, 2024. doi: 10.1016/j.compchemeng.2023.108564.

[16] Thomas D. Hedberg, Joshua Lubell, Lyle Fischer, Larry Maggiano, and Allison Barnard Feeney. Testing the digital thread in support of model-based manufacturing and inspection. *Journal of Computing and Information Science in Engineering*, 16(2):021001, 2016. doi: 10.1115/1.4032697.

[17] Helmut Blitz, Jochen Engelke, Raymund Sonnenschein, and Henner Schmidt-Traub. Rechnergestützte konfigurierung von RI-fließbildern am beispiel von pumpen. *Chemie Ingenieur Technik*, 66(4):470–475, 1994. doi: 10.1002/cite.330660404.

[18] Michael Obst, Falk Doherr, and Leon Urbas. Wissensbasiertes assistenzsystem für modulares engineering. *at – Automatisierungstechnik*, 61(2):103–108, 2013. doi: 10.1524/auto.2013.0011.

[19] Gabriel Vogel, Lukas Schulze Balhorn, and Artur M. Schweidtmann. Learning from flowsheets: A generative transformer model for autocompletion of flowsheets. *Computers & Chemical Engineering*, 171:108162, 2023. doi: 10.1016/j.compchemeng.2023.108162.

[20] Edwin Hirtreiter, Lukas Schulze Balhorn, and Artur M. Schweidtmann. Toward automatic generation of control structures for process flow diagrams with large language models. *AIChE Journal*, 70(1):e18259, 2024. doi: 10.1002/aic.18259.

[21] Lukas Schulze Balhorn, Marc Caballero, and Artur M. Schweidtmann. Toward autocorrection of chemical process flowsheets using large language models. In *34th European Symposium on Computer Aided Process Engineering / 15th International Symposium on Process Systems Engineering*, volume 53 of *Computer Aided Chemical Engineering*, pages 3109–3114, 2024. doi: 10.1016/b978-0-443-28824-1.50519-6.

[22] Artur M. Schweidtmann and Davy Demeyer. Engineering process plants as code in the age of agentic AI. Preprint at `https://arxiv.org/abs/2610.40239`, 2026.

[23] Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *Advances in Neural Information Processing Systems*, volume 38, pages 135676–135729, 2025. doi: 10.52202/085713-4082.

[24] Ignacio E. Grossmann and Arthur W. Westerberg. Research challenges in process systems engineering. *AIChE Journal*, 46(9):1700–1703, 2000. doi: 10.1002/aic.690460902.

[25] Wolfgang Marquardt and Manfred Nagl. Workflow and information centered support of design processes—the IMPROVE perspective. *Computers & Chemical Engineering*, 29(1):65–82, 2004. doi: 10.1016/j.compchemeng.2004.07.018.

[26] Andreas Wiesner, Jan Morbach, and Wolfgang Marquardt. Information integration in chemical process engineering based on semantic technologies. *Computers & Chemical Engineering*, 35 (4):692–708, 2011. doi: 10.1016/j.compchemeng.2010.12.003.

[27] Jethro Akroyd, Sebastian Mosbach, Amit Bhave, and Markus Kraft. Universal digital twin – a dynamic knowledge graph. *Data-Centric Engineering*, 2:e14, 2021. doi: 10.1017/dce.2021.10.

[28] Shuyuan Zhang, Jiyizhe Zhang, and Alexei A. Lapkin. A knowledge graph framework for digital twins of chemical processes. *Nature Chemical Engineering*, 3(5):286–297, 2026. doi: 10.1038/s44286-026-00392-1.

[29] Achmad Anggawirya Alimin and Artur M. Schweidtmann. GraphRAG for engineering diagrams: ChatP&ID enables LLM interaction with P&IDs. *AIChE Journal*, 72(10):e70540, 2026. doi: 10.1002/aic.70540.

[30] Christian Bramsiepe, Nicolai Krasberg, Christoph Fleischer, Lukas Hohmann, Norbert Kockmann, and Gerhard Schembecker. Information technologies for innovative process and plant design. *Chemie Ingenieur Technik*, 86(7):966–981, 2014. doi: 10.1002/cite.201400029.

[31] Azad M. Madni and Michael Sievers. Model-based systems engineering: Motivation, current status, and research opportunities. *Systems Engineering*, 21(3):172–190, 2018. doi: 10.1002/sys.21438.

[32] International Organization for Standardization. ISO/IEC/IEEE 15288:2023 systems and software engineering – system life cycle processes. ISO, Geneva, 2023.

[33] International Electrotechnical Commission. IEC 82045-1:2001 document management – part 1: Principles and methods. IEC, Geneva, 2001.

[34] Michael Wiedau, Lars von Wedel, Heiner Temmen, Richard Welke, and Nikolaos Papakonstantinou. ENPRO data integration: Extending DEXPI towards the asset lifecycle. *Chemie Ingenieur Technik*, 91(3):240–255, 2019. doi: 10.1002/cite.201800112.

[35] International Organization for Standardization. ISO 15926-1:2004 industrial automation systems and integration – integration of life-cycle data for process plants including oil and gas production facilities – part 1: Overview and fundamental principles. ISO, Geneva, 2004.

[36] DEXPI e.V. DEXPI specification 2.0.1. `https://dexpi.org/specifications/`, accessed 25 October 2026, 2026. Fix release replacing DEXPI 2.0.

[37] Dominik P. Goldstein, Lukas Schulze Balhorn, Achmad Anggawirya Alimin, and Artur M. Schweidtmann. pyDEXPI: A Python framework for piping and instrumentation diagrams (P&IDs) using the DEXPI information model. *Systems and Control Transactions*, 4:1365–1370, 2025. doi: 10.69997/sct.139043.

[38] Jonas Oeing, Kevin Brandt, Michael Wiedau, Gregor Tolksdorf, Wolfgang Welscher, and Norbert Kockmann. Graph learning in machine-readable plant topology data. *Chemie Ingenieur Technik*, 95(7):1049–1060, 2023. doi: 10.1002/cite.202200223.

[39] Achmad Anggawirya Alimin, Dominik P. Goldstein, Lukas Schulze Balhorn, and Artur M. Schweidtmann. Talking like piping and instrumentation diagrams (P&IDs). *Systems and Control Transactions*, 4:1676–1681, 2025. doi: 10.69997/sct.159477.

[40] Jean-Pierre Belaud and Michel Pons. Open software architecture for process simulation: The current status of CAPE-OPEN standard. *Computer Aided Chemical Engineering*, 10:847–852, 2002. doi: 10.1016/S1570-7946(02)80169-9.

[41] International Electrotechnical Commission. IEC 62714-1:2018 engineering data exchange format for use in industrial automation systems engineering – automation markup language – part 1: Architecture and general requirements. IEC, Geneva, 2018.

[42] Paul Poinet, Dimitrie Stefanescu, and Eleni Papadonikolaki. Collaborative workflows and version control through open-source and distributed common data environment. In *Proceedings of the 18th International Conference on Computing in Civil and Building Engineering*, volume 98 of *Lecture Notes in Civil Engineering*, pages 228–247, 2021. doi: 10.1007/978-3-030-51295-8_18.

[43] Venkat Venkatasubramanian. The promise of artificial intelligence in chemical engineering: Is it here, finally? *AIChE Journal*, 65(2):466–478, 2019. doi: 10.1002/aic.16489.

[44] Artur M. Schweidtmann, Erik Esche, Asja Fischer, Marius Kloft, Jens-Uwe Repke, Sebastian Sager, and Alexander Mitsos. Machine learning in chemical engineering: A perspective. *Chemie Ingenieur Technik*, 93(12):2029–2039, 2021. doi: 10.1002/cite.202100083.

[45] Prodromos Daoutidis, Jay H. Lee, Srinivas Rangarajan, Leo Chiang, Bhushan Gopaluni, Artur M. Schweidtmann, Iiro Harjunkoski, Mehmet Mercangöz, Ali Mesbah, Fani Boukouvala, Fernando V. Lima, Antonio del Rio Chanona, and Christos Georgakis. Machine learning in process systems engineering: Challenges and opportunities. *Computers & Chemical Engineering*, 181:108523, 2024. doi: 10.1016/j.compchemeng.2023.108523.

[46] Venkat Venkatasubramanian and Arijit Chakraborty. Quo vadis ChatGPT? From large language models to large knowledge models. *Computers & Chemical Engineering*, 192:108895, 2025. doi: 10.1016/j.compchemeng.2024.108895.

[47] Loïc d'Anterroches. Process flow sheet generation & design through a group contribution approach. PhD thesis, Technical University of Denmark, 2006.

[48] Gabriel Vogel, Edwin Hirtreiter, Lukas Schulze Balhorn, and Artur M. Schweidtmann. SFILES 2.0: an extended text-based flowsheet representation. *Optimization and Engineering*, 24(4): 2911–2933, 2023. doi: 10.1007/s11081-023-09798-9.

[49] Vipul Mann, Mauricio Sales-Cruz, Rafiqul Gani, and Venkat Venkatasubramanian. eSFILES: Intelligent process flowsheet synthesis using process knowledge, symbolic AI, and machine learning. *Computers & Chemical Engineering*, 181:108505, 2024. doi: 10.1016/j.compchemeng.2023.108505.

[50] Lukas Schulze Balhorn, Kevin Degens, and Artur M. Schweidtmann. Graph-to-SFILES: Control structure prediction from process topologies using generative artificial intelligence. *Computers & Chemical Engineering*, 199:109121, 2025. doi: 10.1016/j.compchemeng.2025.109121.

[51] Laura Stops, Roel Leenhouts, Qinghe Gao, and Artur M. Schweidtmann. Flowsheet generation through hierarchical reinforcement learning and graph neural networks. *AIChE Journal*, 69(1): e17938, 2023. doi: 10.1002/aic.17938.

[52] Rimma Dzhusupova, Richa Banotra, Jan Bosch, and Helena Holmström Olsson. Pattern recognition method for detecting engineering errors on technical drawings. In *2022 IEEE World AI IoT Congress (AIIoT)*, pages 642–648, 2022. doi: 10.1109/aiiot54504.2022.9817294.

[53] Lukas Schulze Balhorn, Niels Seijsener, Kevin Dao, Minji Kim, Dominik P. Goldstein, Ge H. M. Driessen, and Artur M. Schweidtmann. Rule-based autocorrection of piping and instrumentation diagrams (P&IDs) on graphs. In *Proceedings of the 35th European Symposium on Computer Aided Process Engineering (ESCAPE 35)*, volume 4 of *Systems and Control Transactions*, pages 1656–1661, 2025. doi: 10.69997/sct.150968.

[54] Lukas Schulze Balhorn, Dominik P. Goldstein, Niels Seijsener, Kevin Dao, Ge H. M. Driessen, and Artur M. Schweidtmann. A graph-based generative artificial intelligence methodology for autocorrection of utility-system P&IDs. *Chemie Ingenieur Technik*, page e70135, 2026. doi: 10.1002/cite.70135.

[55] Lukas Schulze Balhorn, Edwin Hirtreiter, Lynn Luderer, and Artur M. Schweidtmann. Data augmentation for machine learning of chemical process flowsheets. In *33rd European Symposium on Computer Aided Process Engineering*, Computer Aided Chemical Engineering, pages 2011–2016, 2023. doi: 10.1016/b978-0-443-15274-0.50320-6.

[56] Artur M. Schweidtmann. Mining chemical process information from literature for generative process design: A perspective. In *Proceedings of the 10th International Conference on Foundations of Computer-Aided Process Design FOCAPD 2024*, volume 3 of *Systems and Control Transactions*, pages 84–91, 2024. doi: 10.69997/sct.184704.

[57] Donghyeon Lee, Jaewook Lee, and Dongil Shin. GPT prompt engineering for a large language model-based process improvement generation system. *Korean Journal of Chemical Engineering*, 41(12):3263–3286, 2024. doi: 10.1007/s11814-024-00276-1.

[58] Tong Zeng, Srivathsan Badrinarayanan, Janghoon Ock, Cheng-Kai Lai, and Amir Barati Farimani. LLM-guided chemical process optimization with a multi-agent approach. *Machine Learning: Science and Technology*, 6(4):045067, 2025. doi: 10.1088/2632-2153/ae2382.

[59] Javal Vyas and Mehmet Mercangöz. Autonomous industrial control using an agentic framework with large language models. *IFAC-PapersOnLine*, 59(6):349–354, 2025. doi: 10.1016/j.ifacol.2025.07.170.

[60] Shreeyash Gowaikar, Srinivasan Iyengar, Sameer Segal, and Shivkumar Kalyanaraman. An agentic approach to automatic creation of P&ID diagrams from natural language descriptions. Preprint at `https://arxiv.org/abs/2412.12898`, 2024.

[61] Sakhinana Sagar Srinivas, Shivam Gupta, and Venkataramana Runkana. AutoChemSchematic AI: Agentic physics-aware automation for chemical manufacturing scale-up. Preprint at `https://arxiv.org/abs/2505.24584`, 2025.

[62] Emma Pajak, Abdullah Bahamdan, Klaus Hellgardt, and Antonio del Rio Chanona. Multi-agent LLMs for automating sustainable operational decision-making. *Systems and Control Transactions*, 4:1824–1829, 2025. doi: 10.69997/sct.156776.

[63] Wenli Du and Shaoyi Yang. The potential and challenges of large language model agent systems in chemical process simulation: from automated modeling to intelligent design. *Frontiers of Chemical Science and Engineering*, 19(10):99, 2025. doi: 10.1007/s11705-025-2587-5.

[64] Sophia Rupprecht, Yassine Hounat, Monisha Kumar, Giacomo Lastrucci, and Artur M. Schweidtmann. Text2Model: Generating dynamic chemical reactor models using large language models (LLMs). *Systems and Control Transactions*, 4:1706–1711, 2025. doi: 10.69997/sct.165009.

[65] Jan-Frederic Laub, Luca Bosetti, and André Bardow. Text-to-flowsheet: an LLM-assisted pipeline for expert-level digitization and automated simulation of chemical processes. *Digital Discovery*, 5(7):2937–2951, 2026. doi: 10.1039/D6DD00060F.

[66] Xufei Tian, Wenli Du, Shaoyi Yang, Han Hu, Hui Xin, Shifeng Qu, and Ke Ye. From text to simulation: A multi-agent LLM workflow for automated chemical process design. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(35):29705–29713, 2026. doi: 10.1609/aaai.v40i35.40215.

[67] Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023. doi: 10.1038/s41586-023-06792-0.

[68] John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R. Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, pages 50528–50652, 2024. doi: 10.52202/079017-1601.

[69] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

[70] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In *Conference on Language Modeling (COLM)*, 2024.

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

[72] Jez Humble and David Farley. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley Professional, 2010. ISBN 9780321601919.

[73] Kief Morris. *Infrastructure as Code: Managing Servers in the Cloud*. O'Reilly Media, 2016. ISBN 9781491924358.

[74] Difei Song. Engineering as code: Bringing software engineering methodology to engineering design. Preprint at `https://arxiv.org/abs/2609.04216`, 2026.

[75] Mike Bacidore. Agentic AI needs SDA to bring a faster pace and more innovation to the world of automation. Control Design, `https://www.controldesign.com/design/development-platforms/article/55270321/agentic-ai-needs-sda-to-bring-a-faster-pace-and-more-innovation-to-the-world-of-automatio` 2025. Interview with D. Demeyer, 24 February 2025.

[76] Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, and Eivind Hovig. Ten simple rules for reproducible computational research. *PLOS Computational Biology*, 9(10):e1003285, 2013. doi: 10.1371/journal.pcbi.1003285.

[77] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, et al. The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, 3:160018, 2016. doi: 10.1038/sdata.2016.18.

[78] Carl Boettiger. An introduction to Docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79, 2015. doi: 10.1145/2723872.2723882.

[79] Chris Lamb and Stefano Zacchiroli. Reproducible builds: Increasing the integrity of software supply chains. *IEEE Software*, 39(2):62–70, 2022. doi: 10.1109/MS.2021.3073045.

[80] Timothy Lebo, Satya Sahoo, and Deborah McGuinness. PROV-O: The PROV ontology. W3C Recommendation, 30 April 2013, `https://www.w3.org/TR/prov-o/`, 2013.

[81] Rachel Potvin and Josh Levenberg. Why Google stores billions of lines of code in a single repository. *Communications of the ACM*, 59(7):78–87, 2016. doi: 10.1145/2854146.

[82] Chris Mutel. Brightway: An open source framework for life cycle assessment. *Journal of Open Source Software*, 2(12):236, 2017. doi: 10.21105/joss.00236.

[83] Felipe Oviedo, Fiodar Kazhamiaka, Esha Choukse, Allen Kim, Amy Luers, Melanie Nakagawa, Ricardo Bianchini, and Juan M. Lavista Ferres. Energy use of AI inference, efficiency pathways, and test-time scaling. *Joule*, 10(8):102430, 2026. doi: 10.1016/j.joule.2026.102430.

[84] Tina Vartziotis, Rodopi Kosteli, Elli Vartziotis, George Dasoulas, Michael Keckeisen, Konstantinos Skianis, Sotirios Kotsopoulos, and Francesca Dominici. From tokens to watt-hours: Analytical energy estimation for LLM inference on modern GPUs. Preprint at `https://arxiv.org/abs/2607.26571`, 2026.

[85] European Commission. Reference document on best available techniques for the manufacture of large volume inorganic chemicals – ammonia, acids and fertilisers. European IPPC Bureau, Seville, `https://eippcb.jrc.ec.europa.eu/reference/large-volume-inorganic-chemicals-ammonia-acids-and-fertilisers`, 2007.

[86] International Energy Agency. Ammonia technology roadmap: Towards more sustainable nitrogen fertiliser production. IEA, Paris, `https://www.iea.org/reports/ammonia-technology-roadmap`, 2021.

[87] Derek DeBellis, Kevin Storer, Nathen Harvey, Matt Beane, Rob Edwards, Edward Fraser, Ben Good, Eirini Kalliamvakou, Gene Kim, Eric Maxwell, Sarah D'Angelo, Sarah Inman, Ambar Murillo, and Daniella Villalba. DORA 2025 state of AI-assisted software development report. DORA, Google, `https://dora.dev/dora-report-2025/`, 2025.

[88] Gregor Wernet, Christian Bauer, Bernhard Steubing, Jürgen Reinhard, Emilia Moreno-Ruiz, and Bo Weidema. The ecoinvent database version 3 (part I): overview and methodology. *The International Journal of Life Cycle Assessment*, 21(9):1218–1230, 2016. doi: 10.1007/s11367-016-1087-8.

[89] Artur M. Schweidtmann and Jana M. Weber. Process atlas: an open database of chemical process engineering models and life cycle inventories. In preparation, 2026.

[90] Leonel Aguilar, Michal Gath-Morad, Jascha Grübel, Jasper Ermatinger, Hantao Zhao, Stefan Wehrli, Robert W. Sumner, Ce Zhang, Dirk Helbing, and Christoph Hölscher. Experiments as code and its application to VR studies in human-building interaction. *Scientific Reports*, 14(1): 9883, 2024. doi: 10.1038/s41598-024-60791-3.

[91] S. Hessam M. Mehr, Matthew Craven, Artem I. Leonov, Graham Keenan, and Leroy Cronin. A universal system for digitization and automatic execution of the chemical synthesis literature. *Science*, 370(6512):101–108, 2020. doi: 10.1126/science.abc2986.

[92] Artur M. Schweidtmann. ChemE-as-Code evaluation data: ammonia case study (fault injection, change propagation and reproducibility). Zenodo, `https://doi.org/10.5281/zenodo.23913145`, 2026.

# Supporting Information for ChemE-as-Code: An Open-Source Platform for Agentic AI in Chemical Engineering

Artur M. Schweidtmann
Process Intelligence Research Group, Department of Chemical Engineering,
Delft University of Technology, Van der Maasweg 9, 2629 HZ Delft, The Netherlands
a.schweidtmann@tudelft.nl

## S1 Reproducibility Instructions

The engineering files of the case study are in `https://github.com/ChemE-as-Code/process-library`, folder `processes/ammonia-1`, at tag `nh3-run-final` (commit `3c02baa`). The engineering stage started at tag `nh3-run-start`, and the tags `nh3-run-S1` to `nh3-run-S9` mark the end of each section. The run records are in `processes/ammonia-1/runs/`, with the ledger, interventions, plan, prompts and one record per session.

The platform is at `https://github.com/ChemE-as-Code/core`, including the viewer and the agent runner. The engineering stage started on release `v1.0.0` (`9eff826`). On the author's decision it moved to `v1.0.1` (`e871601`) for the simulation continuation and to `v1.1.0` (`c2a4513`) for the last layout section, and every session records its version. The other versions are pyDEXPI 2.1.0, DWSIM 9.0.5 in Docker (image built from core's `docker/dwsim/Dockerfile`) and the model `claude-opus-5-5`. The checks of the reported state are rerun with `cheme check` and `cheme maturity` in the plant folder.

## S2 Platform: Detailed Architecture

Figure S1 shows the detailed architecture of ChemE-as-Code. The core's layers (foundation, domains, project and interfaces), the import rules between them and the 109 skills are documented in the core repository, with Architecture Decision Records in `docs/adr/`.

## S3 Platform: Rules, Calculation Maturity, Gates and Sign-offs

**Rules.** The checks follow the four rule layers of Schweidtmann and Demeyer (schema, consistency, physical, safety). At core 1.1.0 the code defines 466 rule identifiers: 390 written out in the source and 76 built from a prefix and a suffix (for example `cheme.layout-<rule>`); there is no central rule registry, so the count is by identifier. Every finding carries a stable identifier, a severity (error, warning, info), the file, the line where one applies, a message and a hint. The checks stop at the first broken layer until it is repaired, for example at a balance that does not close, which is why one pass of the checks after a large change reaches only part of the affected files, as the engineering-change experiments of the main text show.

**Calculation maturity.** `cheme maturity` is a checklist evaluated by a program, not a judgment. It rates how far the numbers of a design are backed by evidence in the repository on four

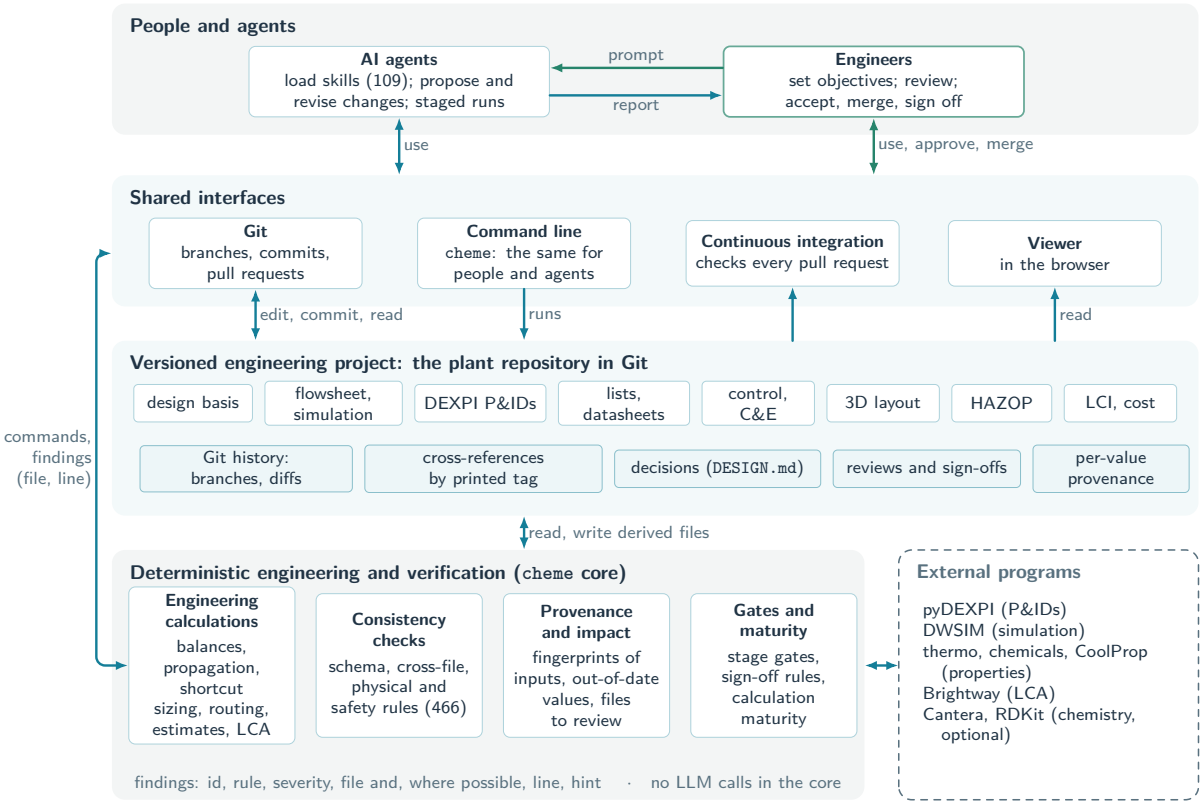



Figure S1: Detailed architecture of ChemE-as-Code (core 1.1.0): actors and interfaces, the engineering files and project records in a plant repository, the four groups of deterministic capabilities, and the external programs the core calls (Cantera and RDKit as optional extras). The architecture figure of the main text shows the simplified version.

cumulative levels, *sourced*, *balanced*, *predictive* and *engineering*, whose requirements Table S1 lists. A requirement the command cannot check blocks the level, so a level is claimed only when its requirements have been checked.

**Gates and sign-offs.** A gate is a set of checks that a project stage must pass, for example that every item on the equipment list has a datasheet marked final; core 1.1.0 defines 7 concept, 13 basic and 5 detail gates. A stage is signed off only on a clean, committed state with all its gates passing and no open critical or high review finding. Review records name their reviewer kind: `human`, `agent` or `self`, where self marks a review by the same session that made the change. The sign-off command refuses a sign-off on a self review unless someone else has approved it, and a sign-off counts only once it is merged, so an agent's review is never mistaken for an engineer's. A revision counts as checked when it is valid (every file matches its schema), consistent (no cross-file or physical rule reports an error) and current (no derived result is out of date); warnings, completeness and calculation maturity are reported with it, so this is not an overall pass or fail. Passing gates does not constitute a review. Approval and independent review are reserved for people in the proposed workflow; the core does not enforce this, and in the case study it accepted sign-offs on agent reviews.

Table S1: Calculation maturity levels evaluated by `cheme maturity`. Levels are cumulative. Maturity reflects the evidence in the repository, not independent validation of the plant.

| **Level** | **Required evidence (selection)** | **Explicitly not covered** |
|---|---|---|
| Sourced | schema-valid flowsheet; design basis with capacity, feeds and products; every feed flow and composition given with a source | – |
| Balanced | no error-level process findings; every block and step balance evaluable and closed within the declared tolerance in every case; reactors and separators rest on a sourced model basis | energy closure reported but not required |
| Predictive | simulation translates and has a current, converged run per case with recorded solver and versions; no excluded steps; separations and reactions predicted inside the simulation; property methods used within their benchmarked range; energy audit closed; no crossed exchangers | mixture properties outside the benchmarked claim |
| Engineering | valid equipment list, final datasheets for every item, valid P&IDs, all flowsheet links resolved | datasheet duties against the simulation (unsupported); mechanical design, relief sizing, dynamics |

## S4 Platform: Terms, Tools and Skills by Scale

**Glossary.** Table S2 defines the platform terms used in the main text. Rule and command names are those of core 1.1.0; "stale" in a rule name means out of date.

Table S2: Platform terms used in the main text.

| Term | Meaning |
|---|---|
| Skill | A short instruction file (`SKILL.md`) that tells an agent when and how to perform an engineering task, which inputs it needs and what to do when they are missing, which tools to call, how to validate the result, how outputs and their provenance are recorded, and what is left to people. |
| Tool | A deterministic program of the core, or an external program it calls (DWSIM, pyDEXPI, thermo, chemicals, CoolProp, Brightway; Cantera and RDKit as optional extras), that calculates or checks. No tool calls an LLM. |
| Check, rule, finding | A check runs one rule over the files. Where the rule fails, it reports a finding with a stable identifier (for example `cheme.line-dn-mismatch`), a severity (error, warning, info), the file, the line where one applies, a message and a hint. |
| Design basis | The requirements the plant must meet (product, capacity, feed, site, utilities), kept in `process/design-basis.yaml`; the starting point of every calculation. |
| Plant repository | One Git repository that holds every engineering file of a plant, from the design basis to the life cycle inventory, with its history, decisions and reviews. |
| Engineering file | A file of the plant repository: DEXPI XML for P&IDs, YAML for the flowsheet, lists, layout, HAZOP and assessment data, Markdown for decisions and reviews. |
| Evidence | The record kept with a revision: the results of the checks, the calculations behind each value with their provenance, and the review records. |
| Revision | One recorded change to the plant repository with its reason (a commit, normally on a branch and proposed as a pull request). A revision is proposed, checked or approved. |
| Tag | The identifier of an item of equipment, a line or an instrument (H-101, NG-1003, TIC-1103) by which files refer to each other. |
| Authoritative, derived | An authoritative file is written by people or agents (design basis, lists, P&IDs, HAZOP records, decisions, reviews). A derived file is generated by a tool (BFD, PFD, simulation run, 3D model, MTO, estimate, LCI dataset) and is never edited by hand. When two authoritative files disagree, for example a line whose nominal diameter differs between the line list and the P&ID, neither takes precedence: a finding names both, and a person or an agent resolves it. |
| Provenance | The recorded origin of a value: `given` (with a source), `assumed` (with a reason) or `calculated` (with the name of the tool and the fingerprint of its inputs). |
| Fingerprint; out of date | The hash of the inputs of a calculated value or generated file. When the inputs no longer match it, `cheme validate` reports the value or file as out of date: the rule `process-calculated-stale` names a stream value and the input that changed, and `generated-outdated` names a generated file. Nothing is regenerated automatically. |
| Impact | The list of files and reviews that need attention after a change, collected by `cheme impact` from out-of-date values, outdated generated files and reviews whose analyzed inputs changed. |
| Calculation maturity | The level to which the numbers of a design are backed by evidence in the repository, evaluated by `cheme maturity` on four cumulative levels: sourced, balanced, predictive, engineering (Section S3). |
| Gate | A set of checks that a project stage (concept, basic, detail) must pass, for example that every listed item has a datasheet marked final. |
| Sign-off | The recorded acceptance of a stage, allowed only on a clean, committed state with all gates passing and no open critical or high review finding; it counts once merged. |
| Reviewer kind | The kind of reviewer named in a review record: `human`, `agent` or `self` (a review by the session that made the change). A self review cannot carry a sign-off unless someone else has approved it. |

**Tools and skills by scale.** Table S3 lists, per engineering scale, the files, the deterministic tools and the skills of core 1.1.0 (88 commands, 109 skills), and how the ammonia case used them. The skills are `SKILL.md` files under `src/chemeascode/domains/<domain>/skills/<name>/` in the core repository (the review skill under `src/chemeascode/project/skills/review/`); `cheme init` and `cheme skills update` copy them into a plant repository, where agents read them. The tools are the modules of the same `domains/<domain>/` packages, called through the `cheme` command. The records of the ammonia run are incomplete for skills (19 of 57 run records have no published transcript), and none of this support has been independently validated.

Table S3: Engineering scales covered by ChemE-as-Code (core 1.1.0): files and formats, deterministic tools, agent skills, and how the ammonia case used them. Only implemented support is listed. The tools range from validation to screening calculations and preliminary estimates; none replaces a detailed engineering calculation.

| **Scale** | **Files (format)** | **Deterministic tools** | **Agent skills** | **How the ammonia case used them** |
|---|---|---|---|---|
| Molecules, reactions, kinetics | materials, reactions, reaction evidence, kinetic data (YAML); reaction library bundled with the core (YAML) | component properties with their sources (thermo, chemicals, CoolProp); ideal-gas formation data; elementary kinetics from NIST; microkinetics (coverages, turnover frequency, degree of rate control); rate-law fitting with parameter uncertainty; element balances and equilibrium (screening) | materials, property retrieval, reactions, reaction-evidence search, route selection | materials, reaction library and thermochemistry only; no kinetics, microkinetics or rate-law fitting |
| Thermodynamics and process | design basis, flowsheet with cases and per-value provenance, simulation specification and run record (YAML, JSON); BFD and PFD generated as DEXPI | balances and propagation; stream properties within validated ranges; interaction-parameter coverage and flash; DWSIM simulation; energy audit; reactor choice, thermal hazard screening and Monte Carlo reactor uncertainty (screening) | process design and simulation, flowsheets, phase calculation, thermodynamic-model and separation-method selection, heat integration, unit operations, utilities; process knowledge for gas processing, refining, petrochemicals and heavy chemicals | simulation, balances, properties and energy audit; no reactor screening or Monte Carlo uncertainty |
| Equipment, piping, control | DEXPI 2.0.1 P&IDs; equipment, line and instrument lists and datasheets (YAML); control loops, sequences, cause and effect (YAML, Markdown) | P&ID validation, rendering, diff and deterministic edits (pyDEXPI); list–drawing cross-checks; shortcut sizing (exchanger area, pump head and net positive suction head, compressor power, column stages); loop and sequence reports | datasheets, P&ID drafting and editing, P&ID design guidance by equipment type and P&ID review, DEXPI XML, list synchronization, control loops, sequences and narrative, interlocks and trips | P&IDs, lists, datasheets and control; weights and nozzles open |
| Plant layout (3D) | 3D layout inputs (site, buildings, structures, placement) and generated plant model; electrical list (YAML) | 3D model build, pipe routing, clash, headroom and access rules, steel and heating, ventilation and air conditioning (HVAC) ducts; cable trays and cable schedule | layout, pipe routing, structures and access, HVAC ducts, electrical cabling | the most iterated scale; electrical gate open |

| Scale | Files (format) | Deterministic tools | Agent skills | How the ammonia case used them |
|---|---|---|---|---|
| Safety | hazardous-area zones; HAZOP studies with their analyzed inputs (YAML) | HAZOP checks, risk-matrix rating and reports | area classification, HAZOP study, equipment causes, assessment and review, relief scenarios, safeguard crediting | HAZOP and relief as agent drafts; screening zones, no area classification |
| Environment and economy | take-off, cost basis and rates, TEA inputs; LCA scope, factors and unallocated LCI datasets (YAML, CSV) | MTO; equipment cost from published correlations; factor estimate with a declared AACE class; TEA (net present value, internal rate of return, levelized cost); LCA with Brightway, regional scenarios and case comparison; LCI estimates without a flowsheet (Tier 0/1) | take-off and estimate, TEA, LCA, LCI estimate | take-off, estimate, TEA, screening LCA without Brightway; no scenarios |
| Project (all scales) | decisions (`DESIGN.md`), reviews, gates and sign-offs (Markdown, YAML, JSON) | validation, impact, calculation maturity, gates and sign-off | review | all, with agent reviews only |

**Skills by domain.** The 109 skills of core 1.1.0 divide as follows: chemistry 5; process 64, of which 9 are on flowsheets and simulation, 7 on heat integration, 10 on unit operations, 6 on utilities and 32 are technology references for gas processing, refining, petrochemicals and heavy chemicals; equipment 1, piping 17 and control 5; plant layout 4; safety 8; assessment 4; and one project-level review skill. Each skill is documented in its own versioned `SKILL.md` at `https://github.com/ChemE-as-Code/core/tree/v1.1.0/src/chemeascode/domains`, and the package's skills manifest (`src/chemeascode/templates/skills-manifest.json`) records a hash of every bundled version of each skill.

## S5 Ammonia: Orchestration Prompt

The prompt below was given to the Claude Code session that orchestrated the agent jobs of the engineering stage (intervention I-000). It is the main human input to this stage: it sets the scope, the order of work, the budget caps, the review and merge rule, the run accounting and the conditions under which the session had to stop and ask. It is reproduced verbatim below; further human interventions during this stage are listed in `processes/ammonia-1/runs/interventions.md`.

```
# Task: take ammonia-1 to a complete design (simulation, basic engineering, HAZOP, datasheets, relief) as the
↪  main case study of the ChemE-as-Code platform paper

You orchestrate agent jobs; you do not do the engineering yourself. Jobs run through core's agent runner
(`core/agent-cli`, `jobs.py`, Claude Agent SDK, Opus 5.5) in `ChemE-as-Code/process-library`, folder
`processes/ammonia-1`. The design is AI-generated end to end, including reviews and merges; that is
intended and the paper says so.

## Why this run matters
This plant is the only case study of the platform paper (AIChE Journal). The paper reports what agents
produced, what the checks found, what it cost and what humans did. The record of the run matters as much
as the design: record everything as it happens; nothing may be reconstructed afterwards.

## Starting point (read first, do not redo)
- `processes/ammonia-1`: concept stage, 1,500 t/d natural-gas SMR, air-fed secondary reformer, HT/LT
  shift, aMDEA (CO2 vented), methanation, 150 bar loop, purge recovery. Balanced flowsheet (7 blocks /
  51 steps), BFD/PFD generated, 44 equipment items, DESIGN.md D-001..D-021, open inputs I-01..I-10,
  preliminary LCI, review `reviews/2026-10-27-pfd-lci-review.md` (finding #3 still open).
- No process simulation yet: `process/simulation.yaml` only marks adiabatic reactors. Reactor outlets
  are calculated by cheme; separators, CO2 removal, purge recovery, steam system and duties are
  specified values from an unqualified balance script (DESIGN.md D-006).
- Earlier sessions `runs/20261027-*` (design USD 8.71, review, CI fix) on pre-release core 44680fa:
  ledger rows marked "basis (pre-existing)", kept separate from this run.
- Issues: process-library#86 (brief, benchmark rules, run accounting); core#315 (property qualification
  for syngas/NH3: may cap calculation maturity below predictive; report it, do not work around it).

## Step 0 (no API spend)
1. Core release: verify that tag `v1.0.0` exists on ChemE-as-Code/core. If it does not, stop and ask me.
   Install `cheme` (and run `agent-cli`) from that tag for every job; record tag and commit.
   If `cheme validate` or `cheme migrate --check` asks for a migration, run it as its own recorded job.
2. Held-back benchmark: write `benchmarks/ammonia.yaml` (specific energy GJ/t LHV, natural gas feed+fuel,
   process and total direct CO2 t/t, steam export/import, loop pressure and conversion per pass, NOx)
   from cited open sources (EU BREF LVIC-AAF, IEA Ammonia Technology Roadmap 2021, Appl/Ullmann). Commit
   it on branch `benchmark/ammonia`, which no job checks out; post its commit hash and time on #86. No
   job prompt may contain these values.
3. DWSIM: check that Docker runs and the image `cheme-dwsim:<version required by the release>` exists
   (build it once from core's `docker/dwsim/Dockerfile`, as core's README describes). A simulation job
   that cannot start DWSIM stops and asks me.
4. Write `processes/ammonia-1/runs/plan.md` (sections, jobs, caps, the gates each section must pass,
   stop rules), commit, tag `nh3-run-start`. Show me plan and caps, then start.

## Target, in this order (never skip or relax a gate to get further)
1. **Concept closed**: review finding #3 and I-01..I-10 resolved or assumed with a reason; plot plan;
   class 5 estimate and screening TEA; concept gates pass; sign-off.
2. **Process simulation** (before the P&IDs): a DWSIM model of the whole plant through core
   (`process/simulation.yaml`, skill `process-simulation`, `cheme process simulate`), every case of the
   design basis, recycles (synthesis loop, purge recovery, steam system) converged, with recorded
   solver, property package and versions. Thermodynamic model chosen with the skill
   `thermodynamic-model-selection` (cubic EOS for syngas and the loop; state what the aMDEA CO2 removal
   needs and, if DWSIM or core cannot represent it, keep it as a specified block and say so).
   Simulated values replace the script's specified ones where the simulation covers them, as
   `calculated`, never overwriting given or assumed values; deviations are findings, not edits.
   Energy audit closed; `cheme maturity` after the simulation reported. If core#315 keeps the plant
   below `predictive`, record exactly which requirement is unsupported; do not work around it.
3. **Basic engineering**: complete P&IDs for every section (DEXPI 2.0.1, `cheme validate` clean at
   error level), equipment/line/instrument lists complete and linked, PFD current, control loops,
   sequences, interlocks and cause and effect, 3D layout with every DN25+ process line routed, zones,
   MTO, class 4 estimate and TEA, design and layout reviews; basic gates pass; sign-off.
4. **HAZOP** of all P&IDs (nodes, deviations, causes, safeguards, actions), rated against
```

```
   `safety/risk-matrix.yaml`; HAZOP gates pass; actions closed or carried with an owner and reason.
5. **Datasheets** for every equipment-list item (gate `datasheets`), sized from the simulation with
   core's tools where they exist, provenance on every value.
6. **Relief**: relief scenarios for every PSV and relief device on the P&IDs (skill
   `relief-scenarios`), governing case and required capacity per device; what core cannot size is said.
7. Detail-stage gates as far as they go (instrument list complete, utilities and small bore routed,
   cable trays and schedule); report which detail gates pass and which do not.

## How to run
- One implementation job per section, each followed by an **independent reviewer job** (separate
  session, review record with `reviewer_kind: agent`, findings by severity, `VERDICT: approve|fix`).
  `fix` → a fix job (critical/high first) → review again (max two rounds, then ask me).
- Branches `stage/<stage>-<topic>`; the implementing job pushes its branch and opens a PR. **The
  reviewer job merges the PR** after `VERDICT: approve` and green CI, never the author job
  (STAGES.md rule). Use `--approve auto` for reviewer jobs (merge allowed; push to main and force
  push denied) and `--approve deny` otherwise, except for pushing the job's own branch.
- Sign-offs with `cheme check --stage <s> --write --review <record>` as the last commit of the stage
  branch; a sign-off counts once merged.
- Caps (put them in plan.md): concept close-out 10, process simulation 25, P&IDs 45, control and lists
  15, 3D layout 25, HAZOP 20, datasheets 20, relief 15, detail gates 15, reviews and fixes 40, reserve
  15 (about 245 in total). A job that hits its cap stops; do not raise a cap without asking me.

## Budget and API key switch
- The current key has **USD 150**. Run `jobs.py --total 150` for the first batch.
- Switch keys only at a clean point: between jobs, all branches pushed, nothing running, ledger up to
  date. When the next job's cap would take the batch over 150, stop at that point and ask me to switch
  the key. I set the new key in the environment; you never see, print or store it. Continue with a new
  batch (`--total` as I tell you) and resume the plan.
- In the ledger, record which key batch each session ran on (batch 1, batch 2), never the key.

## Run accounting (the paper needs every item)
`processes/ammonia-1/runs/ledger.csv`, one row per session: run id, job, section, role
(implement/review/fix), key batch, model, cost USD, input/output/cache tokens, turns, wall time,
start/end commit, exit status (DONE/BLOCKED/cap/crash), PR number, merged by (session id), gates
passed/total and `cheme maturity` level after the job, review findings by severity.
Plus:
- `runs/interventions.md`: every human action (key switch, answer to a question, prompt change, manual
  fix, decision), with time, why and commit. Your own orchestration edits count too. Merges by
  reviewer agents are not interventions but are listed in the ledger.
- Tags `nh3-run-S<n>` at the end of each target section.
- `cheme runs record` writes `runs/<run-id>/` for every job; check after each job that it exists and is
  merged with the job's branch.
- Platform problems (crashes, wrong or missing findings): one issue each in ChemE-as-Code/core, linked
  in the ledger. Do not fix core during the run; if core blocks a section, stop it and ask me.

## Stop and ask me when
- the key switch point is reached;
- DWSIM cannot be started, core needs a fix, the design brief would change, or a gate would have to be
  relaxed or accepted;
- a job ends BLOCKED twice on the same problem, or a review still says `fix` after two rounds.

## Report at the end (comment on process-library#86, and to me)
Sections reached and signed off; gates passed per stage; calculation maturity reached and what blocks
the next level; simulation: cases converged, property package, what is simulated and what stays
specified; cost and tokens per section and total, per key batch, and "first complete
basic-engineering draft" vs "reported state"; sessions, PRs, agent merges and human interventions;
review findings found and fixed; core issues filed. Only after the design is final: evaluate
`benchmarks/ammonia.yaml` against the design, in range / out of range with reasons. State plainly what
was assumed, what was not reached and what the checks cannot see.
```

# S6 Ammonia: Engineering Drawings

All drawings in this section are generated from the plant repository at process-library tag `nh3-run-final` (commit `3c02baa`) by the platform's tools, `cheme render` for the BFD and the P&IDs and `cheme layout render` for the plot plan of the committed 3D plant model (`cheme` 1.1.0, pyDEXPI 2.1.0), and printed to vector PDF; the 3D views are captured from the platform's viewer opened on a static

export of the repository (`cheme export-static`). The design has not been reviewed by independent engineers.

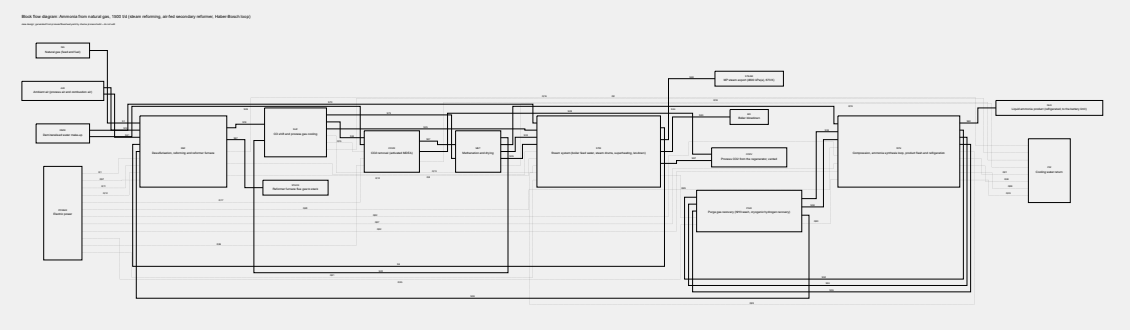


Figure S2: Ammonia BFD, generated from `process/flowsheet.yaml` by `cheme process build` (7 blocks; the PFD, with 62 steps and 121 streams, is in the repository).

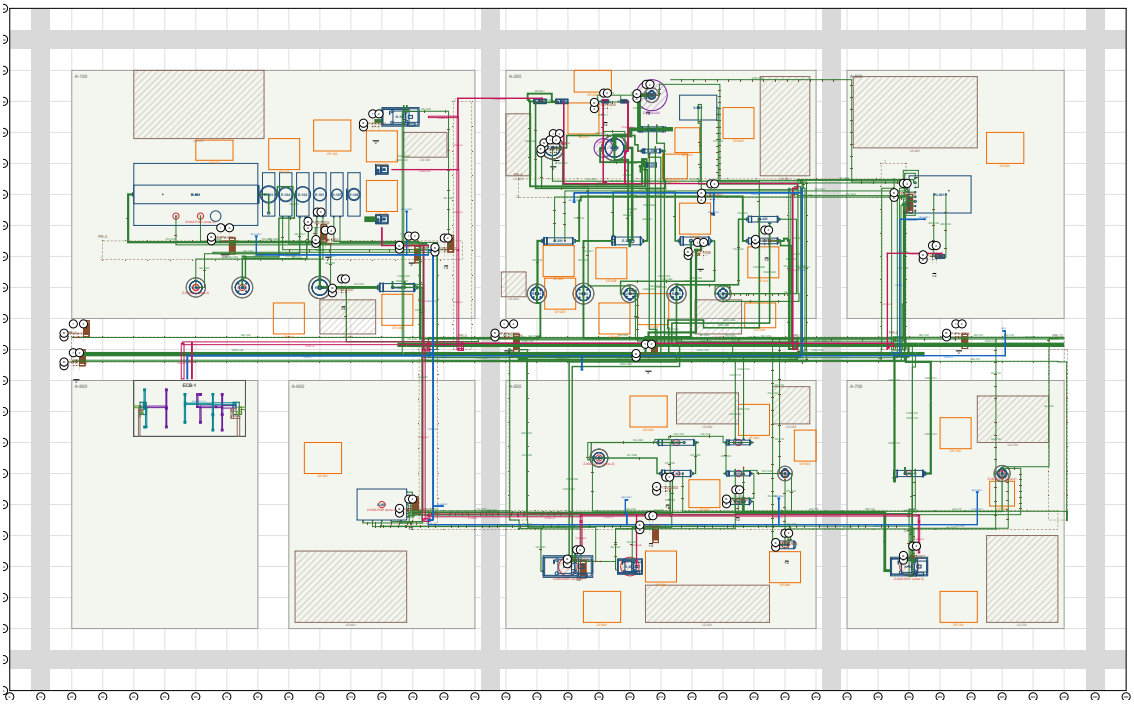


Figure S3: Ammonia plot plan, drawn from the generated 3D plant model (`layout/plant-model.yaml`) with equipment, pipe racks, routed pipes, hazardous-area zones, crane positions and laydown areas.

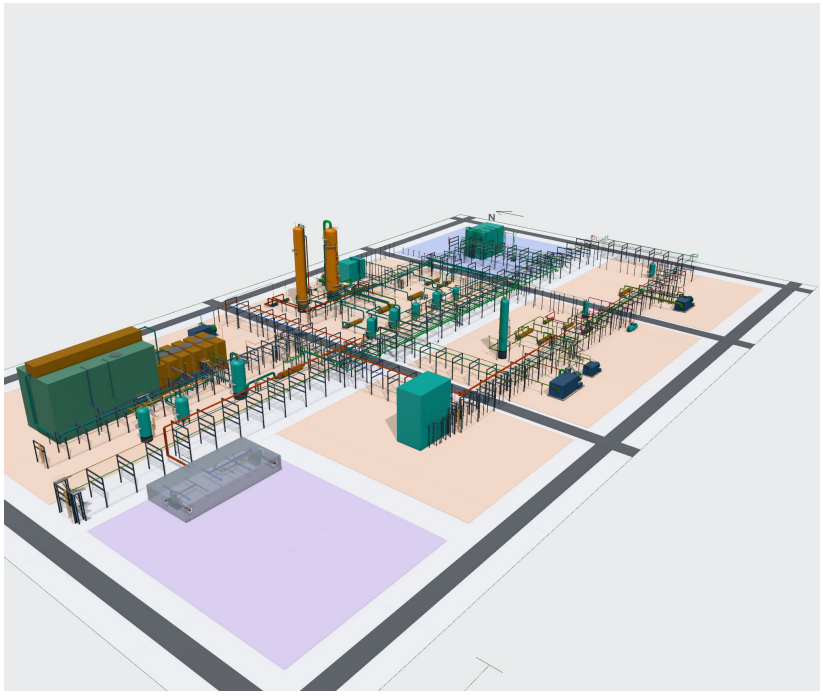


Figure S4: Ammonia 3D plant model in the viewer, view from the south-west: equipment, structures, pipe racks, routed pipes, cable trays, buildings and areas, generated from the layout inputs, the lists and the P&IDs.



Figure S5: Ammonia 3D plant model, reforming area: reformer furnace H-101 with its convection section, vessels with platforms and ladders, the $CO_2$ absorber and regenerator in the background.

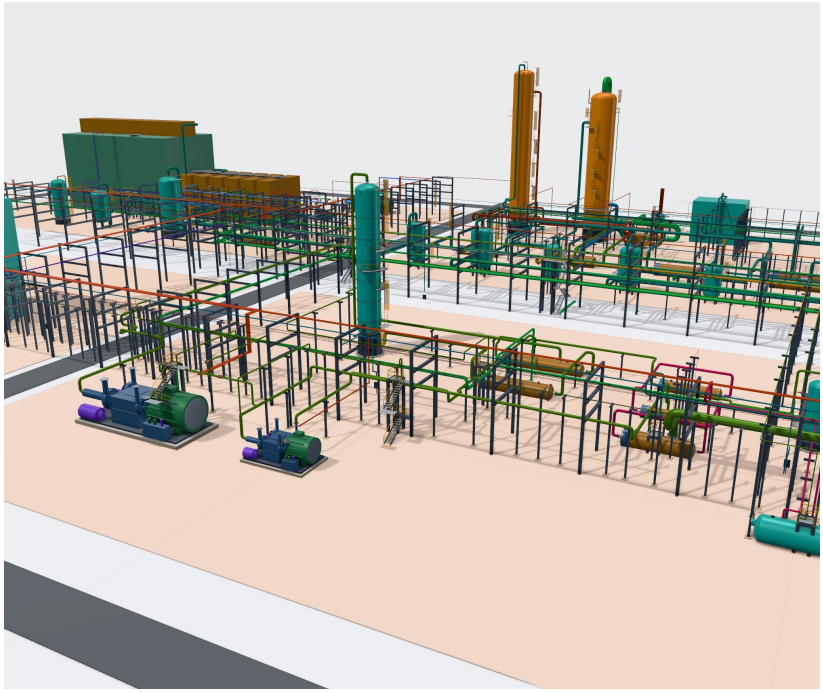


Figure S6: Ammonia 3D plant model, synthesis-gas compression and loop area with the compressor trains on their skids and the pipe racks.

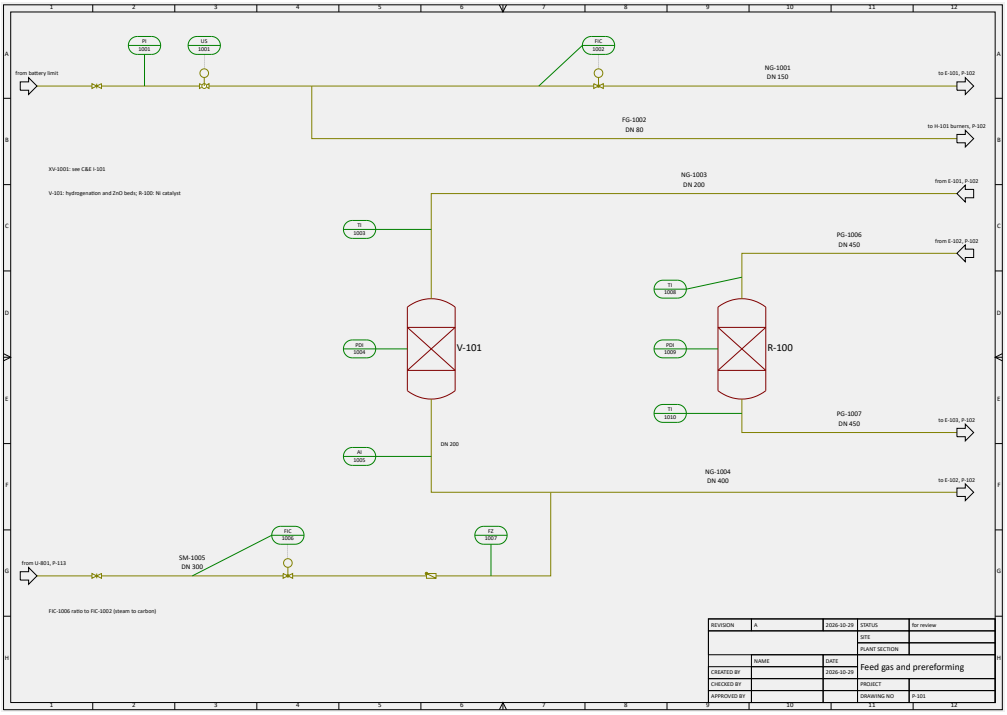



Figure S7: Ammonia DEXPI P&ID P-101, feed gas and prereforming (reported state).

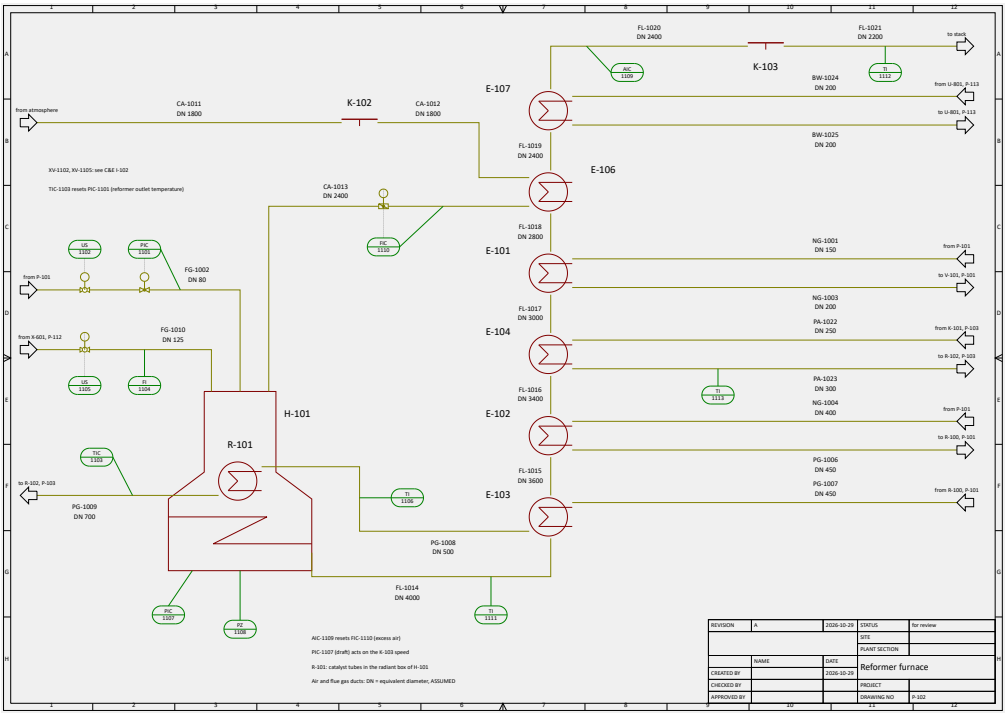



Figure S8: Ammonia DEXPI P&ID P-102, reformer furnace (reported state).

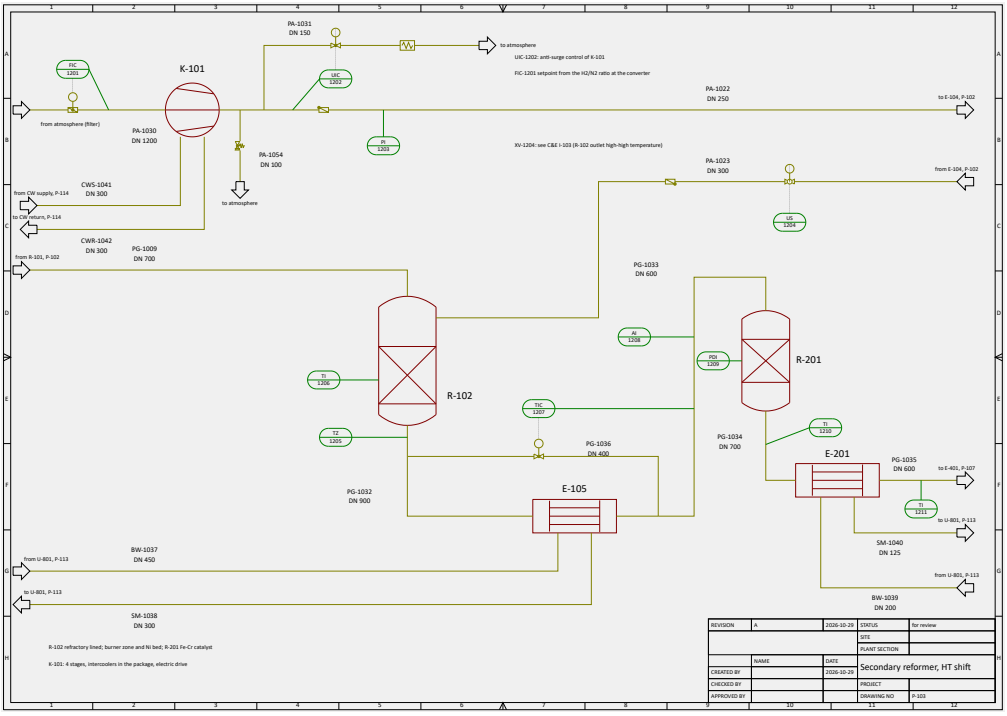



Figure S9: Ammonia DEXPI P&ID P-103, secondary reformer, HT shift (reported state).

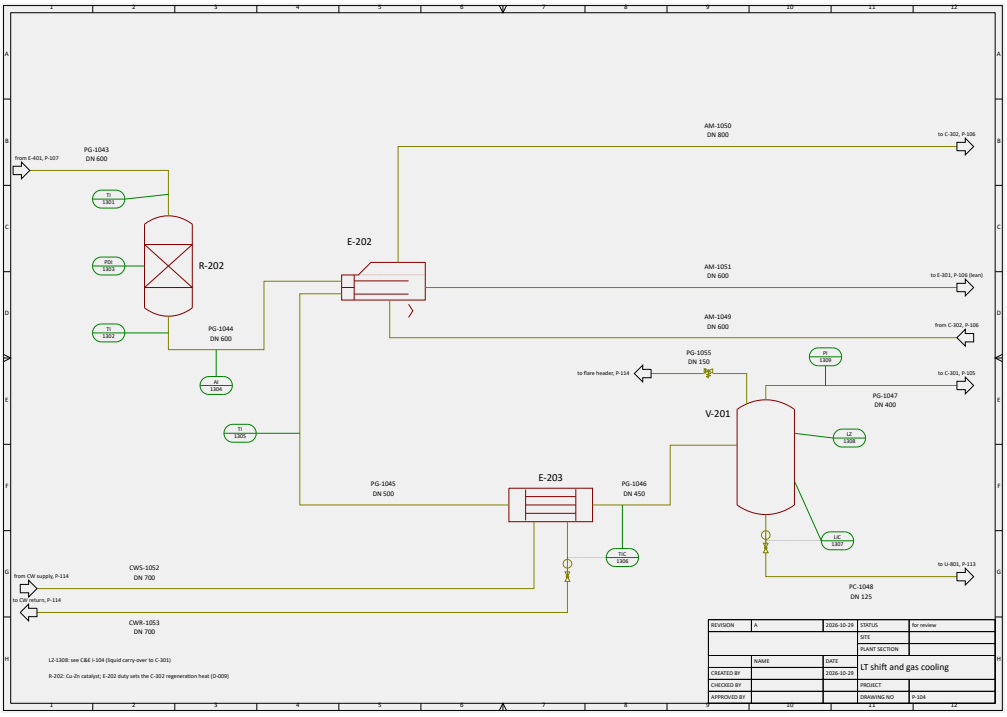



Figure S10: Ammonia DEXPI P&ID P-104, LT shift and gas cooling (reported state).

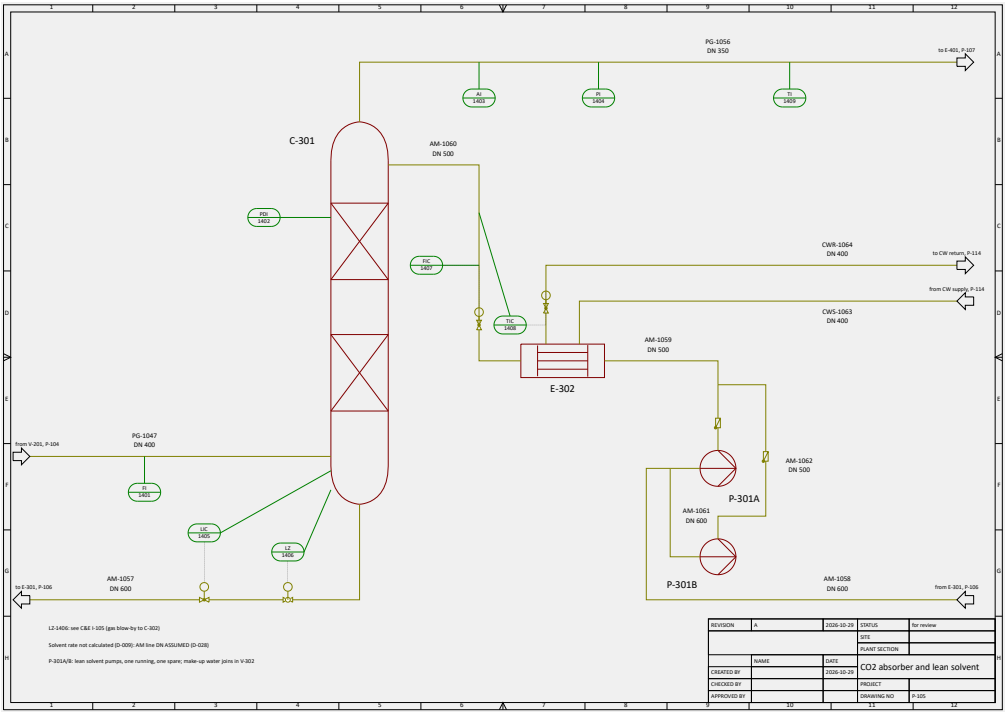



Figure S11: Ammonia DEXPI P&ID P-105, $CO_2$ absorber and lean solvent (reported state).

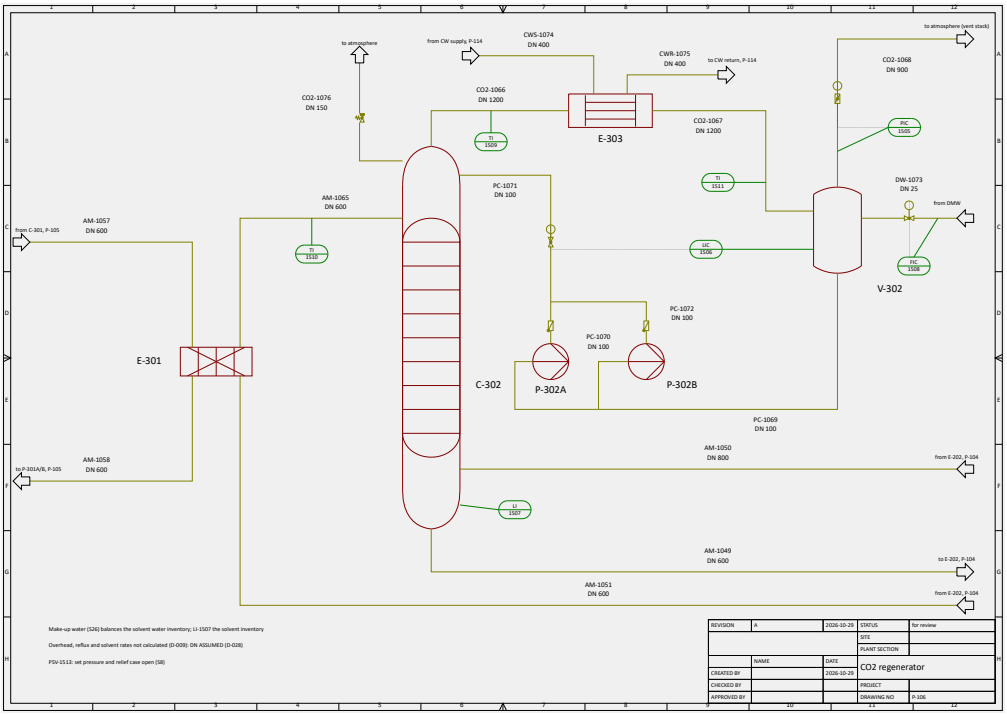



Figure S12: Ammonia DEXPI P&ID P-106, $CO_2$ regenerator (reported state).

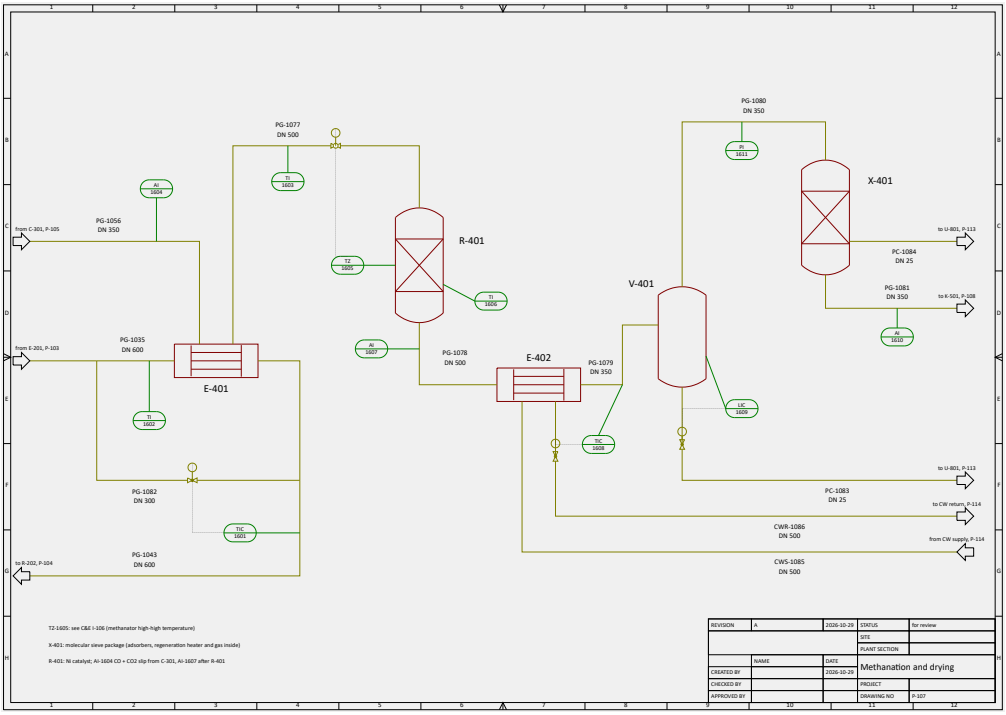



Figure S13: Ammonia DEXPI P&ID P-107, methanation and drying (reported state).

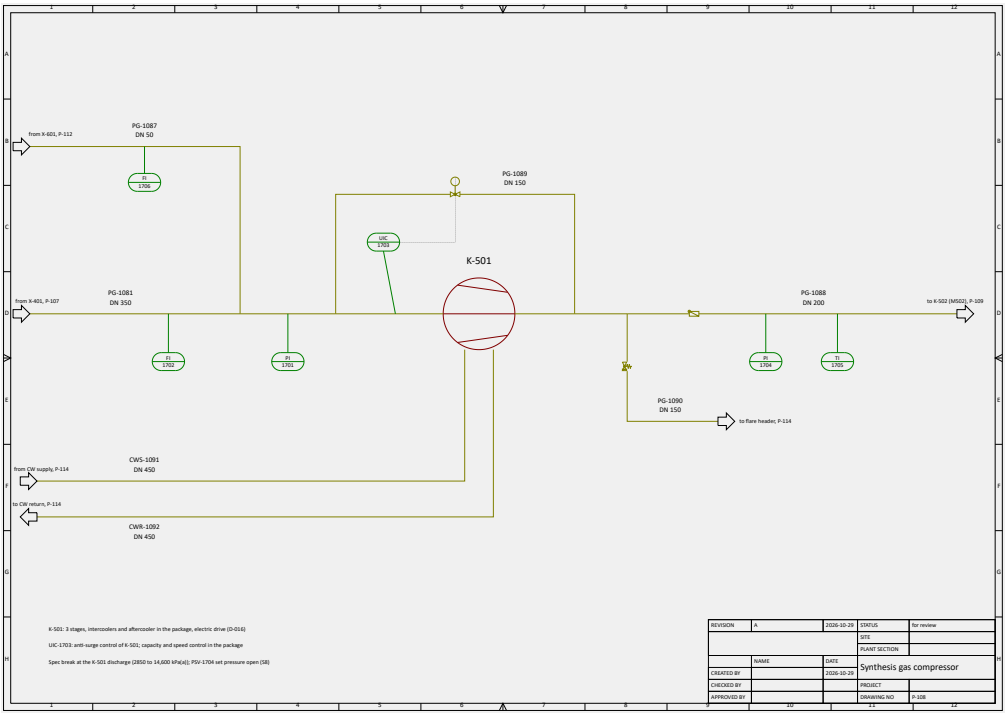



Figure S14: Ammonia DEXPI P&ID P-108, synthesis gas compressor (reported state).

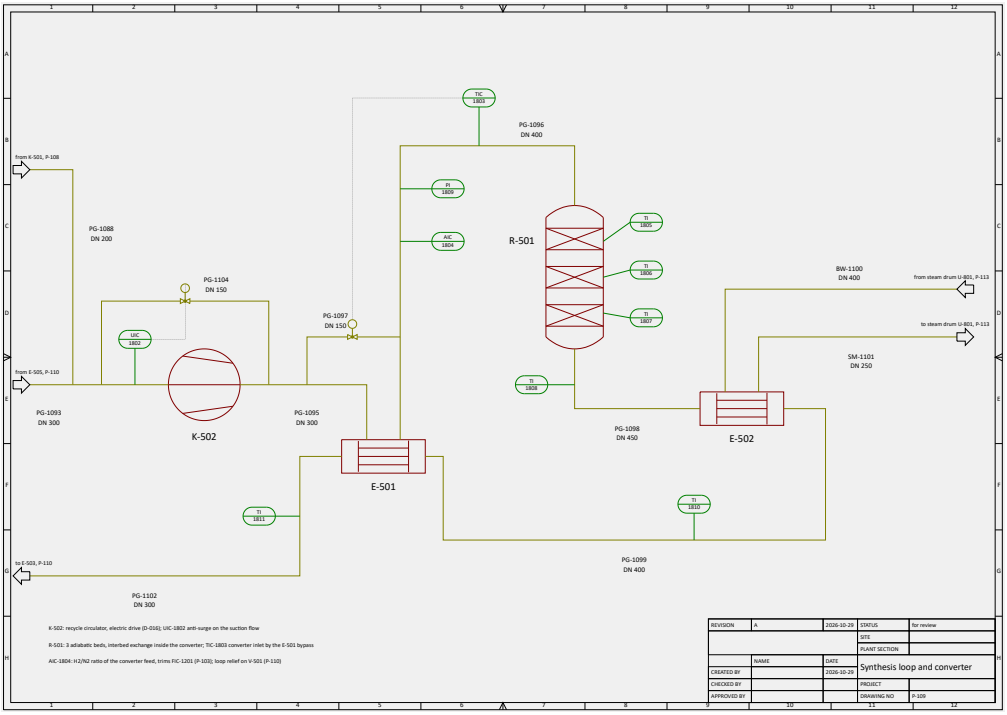



Figure S15: Ammonia DEXPI P&ID P-109, synthesis loop and converter (reported state).

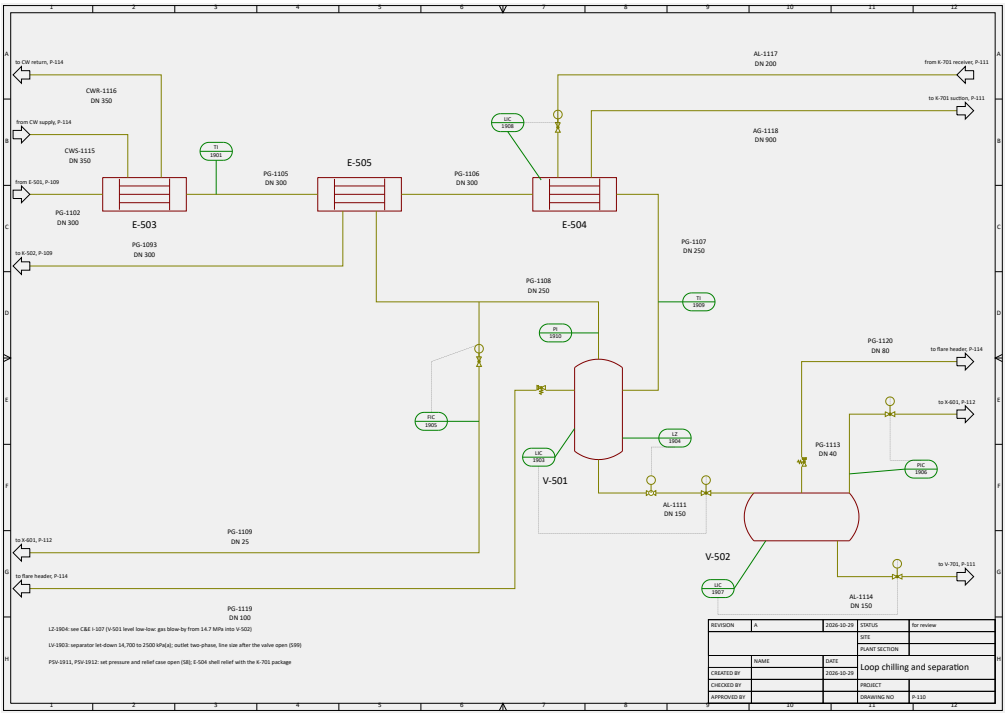



Figure S16: Ammonia DEXPI P&ID P-110, loop chilling and separation (reported state).

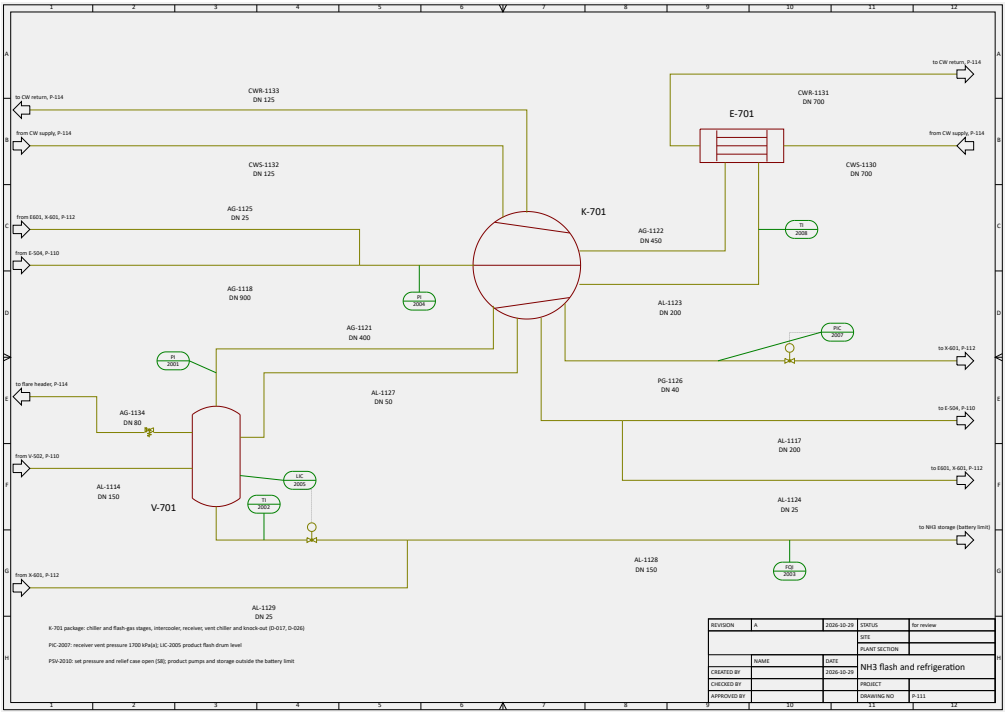



Figure S17: Ammonia DEXPI P&ID P-111, $NH_3$ flash and refrigeration (reported state).

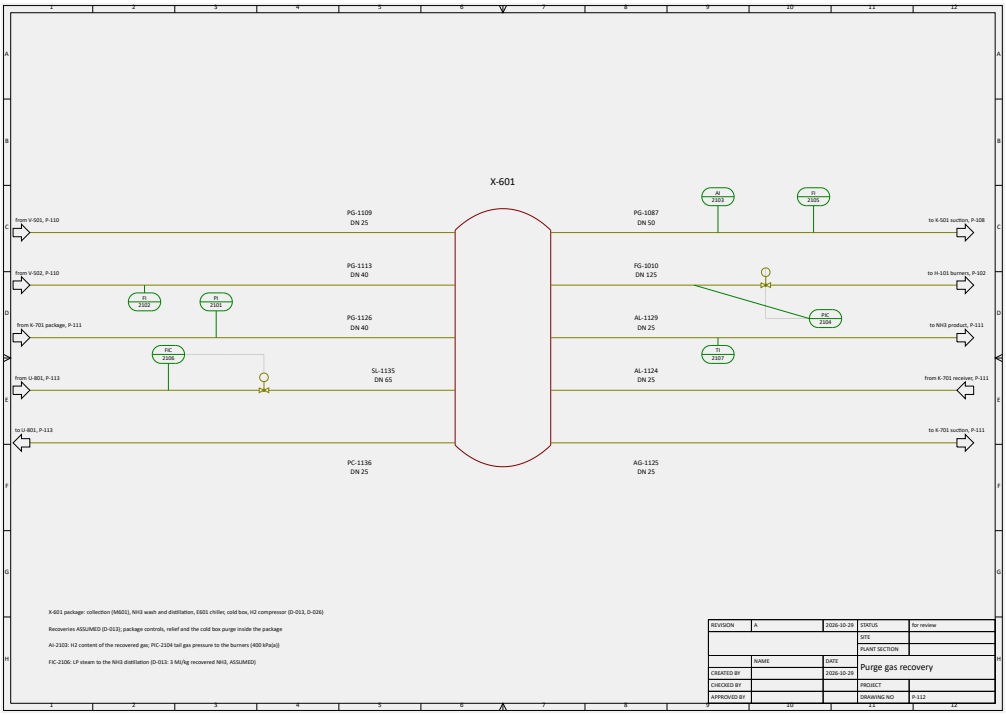



Figure S18: Ammonia DEXPI P&ID P-112, purge gas recovery (reported state).

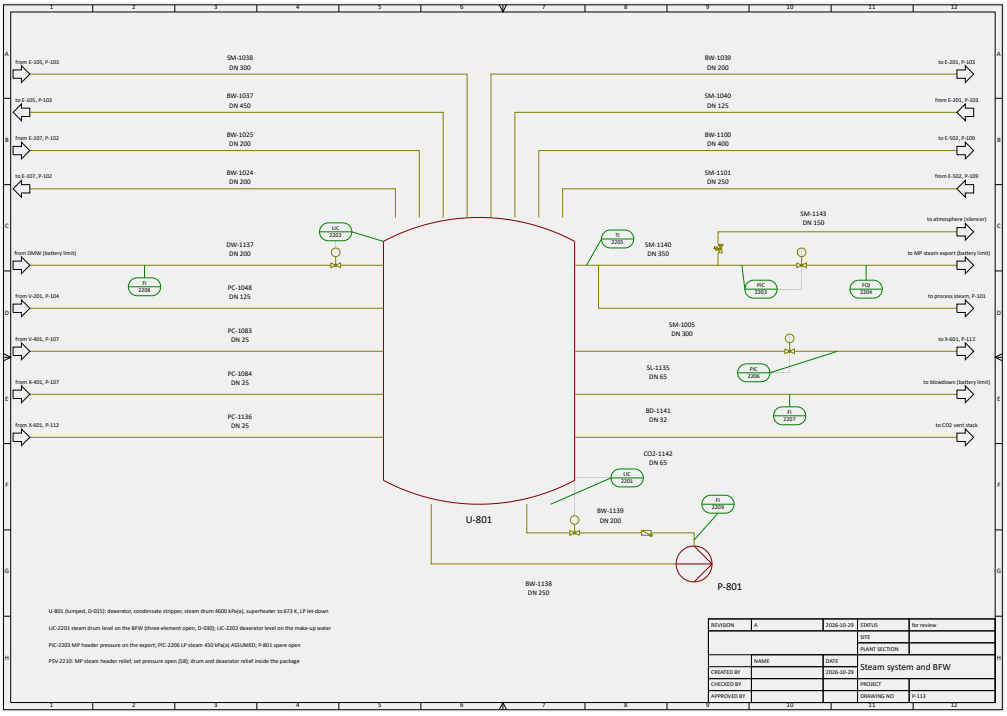



Figure S19: Ammonia DEXPI P&ID P-113, steam system and BFW (reported state).

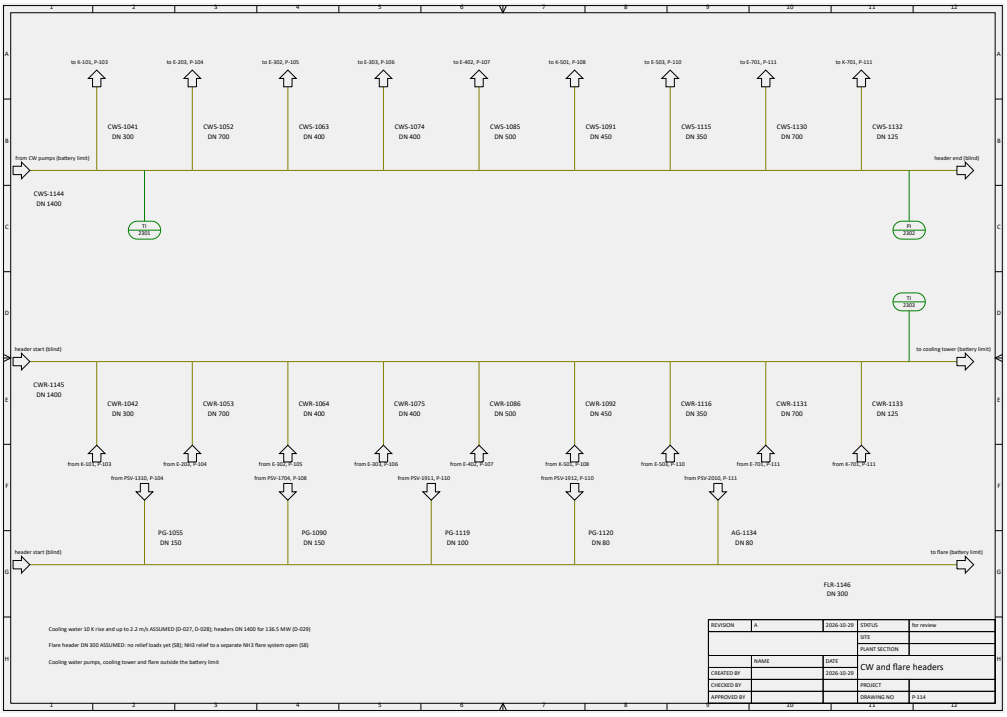



Figure S20: Ammonia DEXPI P&ID P-114, cooling water and flare headers (reported state).

# S7 Ammonia: Simulation and Calculation Inputs

Table S4: Process simulation of the ammonia design (`process/simulation.yaml`, `process/simulation-run.json`, DESIGN.md D-025, D-026).

| | |
|---|---|
| Simulator | DWSIM 9.0.5 in a Docker container (image built from core's `docker/dwsim/Dockerfile`), driven by `cheme process simulate` |
| Property package | Peng–Robinson with DWSIM's own $k_{ij}$, used as a substitute for the methods the selection skill recommended, which the simulation route cannot run; not benchmarked for most of the plant's mixtures (Section S8) |
| Cases | design case (1,500 t/d) |
| Convergence | seven tear streams (synthesis loop, purge recovery and their nested recycles), converged in 121 passes; output 1,499.3 t/d; $H_2/N_2$ at the converter inlet 2.994 after retuning the process air |
| Streams | 121 streams in the flowsheet, 105 with simulated (`calculated`) values |
| Simulated | reformers, shifts, methanation and synthesis converter as equilibrium reactors, compression, heat exchange, flashes and the loop |
| Not simulated | radiant furnace H-101 (radiant balance), refrigeration K-701 and steam system U-801 (evaluated by scripts on simulated duties), aMDEA $CO_2$ removal and purge-gas recovery as specified splits |
| Calculation maturity | *balanced*; the requirements not met are in Table S5 |

Stream tables and all calculation inputs are in `processes/ammonia-1/process/` (`cheme process streams` prints the stream table) and `processes/ammonia-1/DESIGN.md`.

**TEA assumptions.** The techno-economic assessment (`estimate/tea.yaml`) uses 2025 Dutch industrial prices of 11.4 EUR/GJ for natural gas and 101 EUR/MWh for electricity (Eurostat `nrg_pc_203` and `nrg_pc_205`, 2025 means), a $CO_2$ price of 74 EUR/t (EU ETS 2025 average), a real discount rate of 8% over 25 years, and as import price the 2025 US Gulf Coast ammonia price without freight, 439 EUR/t (USGS Mineral Commodity Summaries 2026, 450 USD per short ton at 1.13 USD/EUR). The resulting levelized cost of ammonia is 688 EUR/t.

# S8 Ammonia: Model Basis and Modeling Assumptions

This section lists the main modeling assumptions and known limitations of the design. Agents made these choices and recorded them in `DESIGN.md`, whose decisions the D-numbers refer to; no engineer has reviewed them. The full record, including the reasoning behind each decision, is at `https://github.com/ChemE-as-Code/process-library/tree/nh3-run-final/processes/ammonia-1`. In `process/flowsheet.yaml`, 20 values are given with a source, 135 are assumed with a reason and 268 are calculated.

**Requirements that are not met.** Table S5 lists the requirements of `cheme maturity` that the design does not pass.

**Property method (D-025).** The simulated plant uses Peng–Robinson with DWSIM's own $k_{ij}$. The model-selection skill recommended PSRK or a cubic equation of state with modified Huron–Vidal mixing. The simulation route runs neither, so the agent used Peng–Robinson as a substitute and recorded the substitution for the reviewer. This loses the mixing rule for the polar components, and no $k_{ij}$ is sourced for $NH_3$ with $H_2$, $N_2$, $CH_4$ or Ar, or for $CO_2$–$H_2O$. The main weak spots are therefore the condensation of $NH_3$ with dissolved gases at the loop separator (263 K, 147 bar) and the $CO_2$ solubility in the process condensate. The method has not been benchmarked for these mixtures (Table S5), and no error estimate exists. The aMDEA $CO_2$ removal stays a

Table S5: Requirements of `cheme maturity` not passed by the reported state.

| Requirement | Finding |
|---|---|
| `predictive.coverage` | The furnace radiant section H101, the refrigeration circuit K701 and the steam system U801 are not simulated. |
| `predictive.model-basis` | Eight steps use specified conversions or splits: R100, R102A, H101, R401, C301, X401, X601 and X602. |
| `predictive.property-regime` | Core's benchmarked claim for Peng–Robinson in DWSIM 9.0.5 covers nitrogen, oxygen and argon only. Every feed and stream with $CH_4$, $C_2H_6$, $C_3H_8$, CO, $CO_2$, $H_2$, $H_2O$ or $NH_3$ is therefore outside the claim (266 cases). This states that the method has not been benchmarked for these mixtures in the core, and it does not quantify an error. |
| `predictive.energy-closure` | Unsupported. The plant energy closure is not evaluated with component separators (core issue 60). |
| `balanced.energy-closure` | Reports an imbalance at K601. It is reported but not required for the *balanced* level. Its energy balance differs by 2.2 kW (4%) between two Peng–Robinson implementations with different $k_{ij}$ (D-026). |

specified separator, because it would need an electrolyte model and rate-based absorption, which the simulation route does not support.

Table S6: Main unit models and assumptions. There are no kinetics or catalyst volumes in the design. Pressure drops are assumed per item.

| Unit | Model | Basis |
|---|---|---|
| Primary and secondary reformer bed | Equilibrium (SMR, WGS), approach −10 K. Primary outlet 1083 K, steam to carbon 3.0. Secondary bed at a fixed outlet of 1273 K | Approach and conditions assumed (D-005, D-007) |
| High- and low-temperature shift | Equilibrium (WGS), approach +10 K, outlets 717 K and 507 K | Assumed (D-008) |
| Ammonia converter | One equilibrium step, approach +15 K, outlet 703 K, loop at about 150 bar | Assumed (D-011) |
| Adiabatic beds | DWSIM's adiabatic equilibrium reactor fails its flash, so the beds run at outlet temperatures from the concept balance and their duty is the check (31 kW for the converter) | Workaround (D-024, D-026) |
| Prereformer, burner zone, methanator | Complete conversion | Given, with published sources (D-034) |
| $CO_2$ removal | Specified separator: $CO_2$ fraction to gas 0.002287, co-absorption neglected, regeneration heat 1.0 MJ/kg $CO_2$, pumping 21 kWh/t $CO_2$ | Assumed, no solvent model (D-009) |
| Dryer | All water removed, regeneration not modeled | Assumed (D-010) |
| Purge gas recovery | Specified: 99% $NH_3$ in the wash, then 90% $H_2$, 25% $N_2$, 15% Ar and 5% $CH_4$ to the $H_2$-rich gas | Assumed, vendor data needed (D-013) |
| Flashes | DWSIM Peng–Robinson flash with DWSIM's $k_{ij}$ | $k_{ij}$ unsourced (D-025) |
| Furnace H-101 | Not simulated. Complete combustion, 10% excess $O_2$, air preheat 523 K, bridgewall 1323 K, casing loss 1.5% of fired duty. Target stack 423 K, simulated about 397 K | Combustion basis sourced (D-040), the rest assumed (D-014) |
| Steam system, refrigeration | Agent-written script on simulated duties. One steam level at 4600 kPa(a) and 673 K, 2% blowdown. Refrigeration power at 0.55 of Carnot work | Assumed (D-015, D-017) |
| Machines | All electric. Isentropic efficiency 0.78 per compressor stage, fans 0.70 | Assumed (D-014, D-016) |

**Other assumptions and omissions.** The feeds use a published representative gas and air composition approved by the author (D-040), and the sulfur specification stays assumed (D-004). Ambient conditions are the ISO 13443 reference conditions (288.15 K, 101.325 kPa, dry air), not climate data for the site (D-040), and cooling water runs from 298 to 308 K (D-012). Only the design case is calculated (D-003). The loop is very sensitive to the process air (D-040). With 68,660 kg/h, $H_2/N_2$ at the converter inlet is 2.994; 0.15% more air (68,760 kg/h) lowers it to 2.72, and 0.9% more (69,250 kg/h) to 1.80 with the loop flow up by a third. Nitrogen fed above the stoichiometric ratio can leave the loop only with the purge, about 0.6% of the separator gas, and the purge-gas recovery returns a quarter of it (D-013), so a small excess accumulates in the loop. The hydrodesulfurization hydrogen recycle, dryer regeneration, condensate stripper and by-product methanol are not modeled. Furnace NOx, CO, $CH_4$ and $N_2O$, consumables and fugitive emissions are not estimated (`DESIGN.md`, Section 6). Section 4 of `DESIGN.md` gives the full model table, and `process/simulation.yaml` gives every simulation specification with its provenance.

## S9 Ammonia: Agent Run, Interventions and Change History

**Concept stage.** On 27 October 2026, three sessions (design, review and a fix of continuous integration) produced the starting state of the engineering stage on the branch `case/ammonia-1` (pull request 192), with `cheme` 0.8.0 at core commit `44680fa`. Their task files and reports are in the run records; the session export of the design session is kept out of the repository because it used web tools and may hold third-party text. No intervention log was kept for this stage.

**Engineering stage.** The run followed `runs/plan.md` (tag `nh3-run-start`), written by the orchestrating session with the prompts and section tasks: one implementation job per section, with a reviewer job in a separate session (two late layout close-out jobs had no review of their own; a later review covered the branch); a fix job and a further review on `VERDICT: fix`; sign-off and merge jobs for the stage sign-offs. Jobs ran headless through core's agent runner (`agent-cli`, `jobs.py`) with Claude Opus 5.5 and a budget cap per job. Every session wrote a run record (`runs/<run-id>/`) with commits, times, cost, tokens, the `cheme` commands it ran and the versions of core, pyDEXPI, Claude Code, the Agent SDK and the runner.

Under I-005, the orchestrator did not stop after three review rounds of the layout and estimate section (the plan's stop rule) but started a close-out job on the same pull request; it relaxed no gate. Reviewer and merge agents merged the design pull requests; the orchestrator merged accounting and core-release pull requests and, on the author's instruction, one stacked branch.

**Software changes.** The run found eight defects in core. Three were fixed outside the design as ordinary core changes and released before the run continued (I-002, I-007): issues 57 (multi-recycle convergence) and 59 (crash on pumping steps) in release 1.0.1, and 73 (motor cable entry) in release 1.1.0. Five remain open: 58 (adiabatic equilibrium reactor), 60 (plant energy closure with component separators), 70 (buried pipes against headroom), 71 (one power tray per rack) and 72 (battery-limit lines not routed).

**Cost and what it excludes.** Recorded cost: concept stage USD 12.02, engineering stage USD 243.30, total USD 255.31; the 60 sessions of the engineering stage took 22.4 hours of session time and produced 2.9 million output tokens, with 541 million cached input tokens. The orchestrating session's own API cost was not recorded and is not included. The figures also exclude human expertise, infrastructure and independent review.

**The reported state in numbers.** At `nh3-run-final` the plant repository holds a flowsheet with 62 steps and 121 streams, a converged DWSIM simulation, 14 DEXPI P&ID sheets with 52 equipment items and 122 instruments, 38 control loops, a routed 3D layout, a datasheet for each

Table S7: Human interventions in the engineering stage of the ammonia run (`runs/interventions.md`). A further 50 entries record orchestration steps (starting jobs, pushing branches, opening pull requests, tagging).

| id | time | decision |
|---|---|---|
| I-000 | 29 Oct, 08:40 | Brief for the run given to the orchestrating session (Section S5). |
| I-001 | 29 Oct, 09:25 | Implementation and fix jobs cannot push under the runner's guards in core 1.0.0; the orchestrator pushes their branches and opens the pull requests with the jobs' own descriptions; reviewer jobs merge. |
| I-002 | 29 Oct, 13:10 | The first simulation job is blocked by core (nested recycles do not converge): pause the run, fix core issue 57 (and 59) outside the design as ordinary core changes, release 1.0.1, re-pin the run, continue the simulation. |
| I-003 | 29 Oct, 15:40 | Cap of the simulation continuation: the remaining simulation budget plus the whole reserve (USD 22.94). |
| I-004 | 29 Oct, 16:50 | Information on the remaining credit of the first API key. |
| I-005 | 29 Oct, 17:05 | Second API key for batch 2; continue autonomously overnight; more budget may be used if needed. |
| I-006 | 30 Oct, 08:52 | Decisions escalated by the agents before the basic sign-off: accept eight crane lanes blocked by piping and four buried cooling-water road crossings; approve a representative feed specification; allow an assumed industry-like risk matrix; look up estimate rates and a cost index in published sources; merge the open pull requests. |
| I-007 | 30 Oct, 10:05 | Fix core issue 73 (motor cable entry point) outside the design (release 1.1.0), then clear the compressor tray lanes by design rather than by acceptance; one further attempt at a class 4 estimate (cap USD 10). |

of the 52 items, 8 relief cases, a HAZOP study with 16 nodes and 72 scenarios, an AACE class 4 estimate with TEA, an LCI dataset and 46 recorded decisions.

**Estimate of inference energy.** The ledger at `nh3-run-final` records, for the 63 sessions of both stages, 3.16 million output tokens, 14.7 million cache-write tokens, 565 million cache-read tokens and 7,618 uncached input tokens. The provider publishes no energy data for the model and its size is not public, so this estimate is an order of magnitude only. For output tokens we scale the median of 0.31 Wh for a 300-token query on models above 200 billion parameters, which includes node overhead and data-center overhead (Oviedo et al., Joule, 2026, doi:10.1016/j.joule.2026.102430), giving 1.03 Wh per 1,000 tokens and 3.3 kWh. Their long-query value (3.91 Wh for 5,000 tokens) gives 2.5 kWh. Their model excludes long-context management, which dominates token counts in agent sessions. For input tokens we use 449 mJ per token for a 120-billion-parameter model, the largest reported by Vartziotis et al. (arXiv:2607.26571). This is an analytical, GPU-only value, which we multiply by the data-center overhead of 1.56 they quote. Cache writes then give 2.9 kWh, so the lower case, with cache reads taken as negligible, is about 6 kWh. The upper case charges every cache read as a full input token (110 kWh) and gives about 116 kWh. We consider this pessimistic because a cache read avoids recomputing the prompt, but the true cost of a read is not published, and a larger model would raise both cases. The range of 6 to 116 kWh is wide because the energy of a cached token is unknown; for comparison, even the upper value equals about 11 s of the designed plant's 36.8 MW electricity import. The estimate covers inference only: it excludes the orchestration of both stages, local tools, the simulator, continuous integration and model training.

Table S8: Run ledger of the ammonia case: one row per agent session (`runs/ledger.csv` at `nh3-run-final`). Cost is the recorded API cost; wall time in minutes; exit as recorded; findings of review sessions as critical/high/medium/low. Basis rows are the three sessions of the concept stage.

| run id | section | role | batch | USD | turns | min | exit | c/h/m/l |
|---|---|---|---|---|---|---|---|---|
| 20261027-212934-design | basis | implement | - | 8.71 | 154 | 29 | DONE | |
| 20261027-215841-review | basis | review | - | 1.08 | 26 | 4 | DONE | |
| 20261027-220854-ci-fix | basis | fix | - | 2.22 | 100 | 7 | DONE | |
| 20261029-095743-S1-implement | S1 | implement | 1 | 6.50 | 143 | 16 | DONE | |
| 20261029-101538-S1-review | S1 | review | 1 | 1.55 | 1 | 5 | no final line | 0/0/1/3 |
| 20261029-102044-S1-review | S1 | review | 1 | 0.33 | 1 | 1 | DONE | |
| 20261029-102348-S1-signoff | S1 | signoff | 1 | 0.29 | 14 | 1 | DONE | |
| 20261029-102543-S1-merge | S1 | merge | 1 | 0.44 | 15 | 11 | DONE | |
| 20261029-103843-S2-implement | S2 | implement | 1 | 17.06 | 281 | 84 | BLOCKED | |
| 20261029-120621-core-59 | core fix | implement | 1 | 1.50 | 38 | 40 | DONE | |
| 20261029-124617-core-59-review | core fix | review | 1 | 1.47 | 36 | 38 | DONE | |
| 20261029-120616-core-57 | core fix | implement | 1 | 10.20 | 96 | 91 | cap | |
| 20261029-133825-core-57-review | core fix | review | 1 | 3.84 | 57 | 51 | DONE | -/1/-/2 |
| 20261029-153041-S2-continue | S2 | implement | 1 | 10.97 | 246 | 55 | DONE | |
| 20261029-162638-S2-review-simulation-r1 | S2 | review | 1 | 2.41 | 57 | 19 | DONE | 0/0/1/3 |
| 20261029-165720-S3a-implement | S3a | implement | 1 | 10.43 | 265 | 28 | DONE | |
| 20261029-172646-S3a-review-pids-front-r1 | S3a | review | 1 | 1.83 | 36 | 11 | DONE | 0/0/3/3 |
| 20261029-174820-S3b-implement | S3b | implement | 1 | 8.01 | 193 | 31 | DONE | |
| 20261029-182100-S3b-review-pids-mid-r1 | S3b | review | 1 | 2.26 | 43 | 18 | DONE | 0/0/2/4 |
| 20261029-185034-S3c-implement | S3c | implement | 1 | 9.60 | 303 | 33 | DONE | |
| 20261029-192515-S3c-review-pids-loop-r1 | S3c | review | 1 | 2.50 | 64 | 19 | DONE | 0/0/2/2 |
| 20261029-195652-S4-implement | S4 | implement | 1 | 5.07 | 107 | 21 | DONE | |
| 20261029-201947-S4-review-control-lists-r1 | S4 | review | 1 | 1.95 | 36 | 16 | DONE | 0/0/2/4 |
| 20261029-204825-S5-implement | S5 | implement | 1 | 13.34 | 224 | 56 | BLOCKED | |
| 20261029-214613-S5-review-design-r1 | S5 | review | 1 | 2.40 | 59 | 12 | verdict fix | 0/1/3/0 |
| 20261029-215947-S5-review-layout-r1 | S5 | review | 1 | 1.79 | 56 | 12 | verdict fix | 0/0/6/2 |
| 20261029-221350-S5-fix-r1 | S5 | fix | 1 | 3.38 | 126 | 16 | BLOCKED | |
| 20261029-223214-S5-review-design-r2 | S5 | review | 1 | 0.98 | 23 | 6 | verdict fix | 0/0/0/0 |
| 20261029-224004-S5-review-layout-r2 | S5 | review | 1 | 1.23 | 33 | 6 | verdict fix | 0/0/0/1 |
| 20261029-224824-S5-fix-r2 | S5 | fix | 1 | 1.37 | 37 | 7 | BLOCKED | |
| 20261029-225748-S5-review-design-r3 | S5 | review | 1 | 0.85 | 22 | 4 | verdict fix | 0/0/0/0 |
| 20261029-230408-S5-review-layout-r3 | S5 | review | 1 | 0.87 | 26 | 6 | verdict fix | 0/0/1/0 |
| 20261029-231511-S5b-implement | S5b | implement | 1 | 0.51 | 16 | 8 | success | |
| 20261029-232403-S5b-implement | S5b | implement | 1 | 16.72 | 245 | 64 | BLOCKED | |
| 20261030-003037-S5b-review-design-r1 | S5b | review | 1 | 1.41 | 46 | 6 | verdict fix | 0/0/2/3 |
| 20261030-003845-S5b-review-layout-r1 | S5b | review | 1 | 1.29 | 40 | 6 | verdict fix | 0/0/1/1 |
| 20261030-004639-S5b-fix-r1 | S5b | fix | 2 | 3.63 | 96 | 17 | BLOCKED | |
| 20261030-010558-S5b-review-design-r2 | S5b | review | 2 | 1.17 | 36 | 8 | verdict fix | 0/0/1/2 |
| 20261030-011656-S5b-review-layout-r2 | S5b | review | 2 | 1.66 | 46 | 9 | verdict fix | 0/0/2/1 |
| 20261030-012910-S5b-fix-r2 | S5b | fix | 2 | 3.28 | 102 | 15 | BLOCKED | |
| 20261030-014659-S5b-review-design-r3 | S5b | review | 2 | 0.93 | 25 | 6 | verdict fix | 0/0/0/3 |
| 20261030-015526-S5b-review-layout-r3 | S5b | review | 2 | 1.44 | 39 | 8 | verdict fix | 0/0/1/2 |
| 20261030-020616-S6-implement | S6 | implement | 2 | 6.12 | 104 | 19 | DONE | |
| 20261030-022721-S6-review-hazop-r1 | S6 | review | 2 | 1.33 | 47 | 6 | verdict fix | 0/1/1/3 |
| 20261030-023452-S6-fix-r1 | S6 | fix | 2 | 2.78 | 84 | 7 | DONE | |
| 20261030-024328-S6-review-hazop-r2 | S6 | review | 2 | 1.60 | 44 | 15 | DONE | 0/0/1/0 |
| 20261030-031037-S7-implement | S7 | implement | 2 | 6.36 | 123 | 30 | DONE | |
| 20261030-034345-S7-review-datasheets-r1 | S7 | review | 2 | 1.73 | 41 | 15 | DONE | 0/0/3/5 |
| 20261030-041117-S8-implement | S8 | implement | 2 | 5.06 | 105 | 25 | DONE | |
| 20261030-043941-S8-review-relief-r1 | S8 | review | 2 | 1.70 | 24 | 16 | DONE | 0/0/1/2 |
| 20261030-050913-S9-sync-basic-layout | S9 | sync | 2 | 2.62 | 96 | 11 | DONE | |
| 20261030-052253-S9-implement | S9 | implement | 2 | 9.81 | 207 | 50 | DONE | |
| 20261030-061623-S9-review-detail-gates-r1 | S9 | review | 2 | 1.68 | 38 | 9 | DONE | 0/0/3/4 |
| 20261030-075249-S5c-implement | S5c | implement | 2 | 15.80 | 283 | 67 | BLOCKED | |
| 20261030-090522-S5d-estimate | S5d | implement | 2 | 6.33 | 85 | 25 | DONE | |
| 20261030-090426-core-73 | core fix | implement | 2 | 8.43 | 111 | 51 | DONE | |
| 20261030-095554-core-73-review | core fix | review | 2 | 4.24 | 73 | 63 | DONE | |
| 20261030-112132-S5e-implement | S5e | implement | 2 | 4.56 | 124 | 22 | DONE | |
| 20261030-114654-S5e-review-design-r1 | S5e | review | 2 | 2.12 | 59 | 11 | DONE | 0/0/1/3 |
| 20261030-120126-S5e-review-layout-r1 | S5e | review | 2 | 1.54 | 55 | 9 | DONE | 0/0/1/1 |
| 20261030-121340-S5e-signoff | S5e | signoff | 2 | 0.32 | 11 | 7 | DONE | |
| 20261030-122336-S5e-merge | S5e | merge | 2 | 0.57 | 18 | 16 | DONE | |

| run id | section | role | batch | USD | turns | min | exit | c/h/m/l |
|---|---|---|---|---|---|---|---|---|
| `20261030-124455-bench-kpis` | benchmark | implement | 2 | 2.13 | 45 | 8 | DONE | |
| engineering stage (60 sessions) | | | | 243.30 | 5106 | 1341 | | |
| total with concept stage (63 sessions) | | | | 255.31 | 5386 | 1381 | | |

## S10 Ammonia: HAZOP, Datasheets and Relief

**HAZOP.** The file `safety/hazop/H-01/study.yaml` holds one study with 16 nodes, 67 deviations, 72 scenarios, 59 safeguards and 46 actions, rated with `cheme safety hazop rate` against an assumed 5×5 risk matrix (I-006). The initial risk of the 104 ratings is 11 low, 34 medium, 44 high and 15 very high. Every entry has `status: proposed` and `by: agent`. No HAZOP team has met, and no rating is accepted.

**Datasheets.** There is one datasheet per equipment-list item (52), sized from the simulation with core's tools where they exist, with provenance on every value. Weights and nozzle locations are still missing, so the detail gate `datasheets` fails. The reviewer found that the datasheets had been marked final to pass the gate, against the skill's rule that a filled datasheet stays a draft until checked.

**Relief.** The relief study (`safety/relief/relief-study.md`, D-033) is an agent's proposal; no relief engineer or HAZOP team has reviewed it. For each of the eight relief valves drawn on the P&IDs it names the credible cases, calculates the relieving rate where the repository allows it, and states the case claimed to govern. Set pressures are assumed equal to the design pressure of the protected item. For five valves it gives a required capacity. For three it does not: the gas blow-by cases of PSV-1513 and PSV-1912 depend on the level valves LV-1405 and LV-1903, which are not sized, and the fire case of PSV-1911 on the ammonia separator V-501 needs a transient supercritical calculation. No orifice, relief line or flare header is sized, two-phase relief is not assessed, and eight further cases for which no device is drawn are listed for the next P&ID revision. The loads come from an agent-written script with published equations, not from a qualified tool. The cases and rates of each device are in the relief study.

## S11 Ammonia: Comparison with Open References

**Method.** The reference values were written after the concept stage and before the engineering stage to `benchmarks/ammonia.yaml` on branch `benchmark/ammonia` (commit `7651243`, 29 October 2026, 09:18 CET), which no design job checked out; no job prompt contained them. After the basic sign-off, a separate job computed the indicators from the design's files (`runs/benchmark-kpis.md`, reproduced by `scripts/benchmark_kpis.py`). Bases: per t of ammonia product (62,472 kg/h); lower heating values from the formation enthalpies of `process/materials.yaml` (natural gas 49.09 MJ/kg); electricity as delivered energy (36.77 MW, 3.6 GJ/MWh); exported MP steam at its enthalpy above the make-up water (3.10 MJ/kg). With the steam credited at its exergy instead, the net specific energy is 28.84 GJ/t. Two indicators (process and total direct $CO_2$) are bound to the LCI dataset and checked in every library build; the energy indicators cannot be expressed over the dataset's mass-based exchanges (process-library issue 272).

**Reference definitions.** BREF Table 2.14 gives 27.6–31.8 GJ(LHV)/t, ±1.5 GJ, as net energy consumption for BAT. It prescribes no formula, electricity factor or steam credit, and Section 2.3.1.1 says that a fair comparison needs the balance with the conversion factors used. Table 2.6 notes that net energy depends on the conversion factor for imported electricity. The range 29.3–31.1 GJ/t is our sum of feedstock 22.1 and fuel 7.2–9.0 in BREF Table 2.6 for conventional reforming, a reported

level and not BAT. The IEA table (Chapter 1) gives BAT values of 21.0 feedstock, 11.1 fuel and 0.3 electricity, with gross 32.4 and net 27.6 after 4.8 of steam; it states no boundary, steam reference or electricity convention, and that its electricity is counted as delivered is our inference. BREF Section 2.2.3.8 states 20–30% per pass without a basis; because the same section describes the synthesis gas as containing inerts, we take the total-gas value of 25.5% as the closer match (30.2% on $H_2$+$N_2$, slightly above the range, is an alternative). The converter is an equilibrium-approach model, so this value is not evidence of kinetic performance.

**Further boundary effects.** BREF Section 2.3.1.1 warns that energy comparisons without knowledge of design and evaluation criteria can be misleading. Delivery as liquid at −33 °C needs an extra 0.3 GJ/t relative to liquid at ambient temperature, and raising the cooling water from 20 to 30 °C adds 0.7 GJ/t. This design delivers refrigerated liquid with cooling water from 25 to 35 °C. We did not correct for these effects. Further, the model's furnace assumes complete combustion and a casing loss of 1.5% of the fired duty, its steam and refrigeration systems are not simulated (Section S8), no check flagged the energy results, and a reviewer agent called the gas use plausible.

**Electricity conversion.** Net energy is $28.008 + 2.119/\eta - 3.161$ GJ/t for the design and $21.0 + 11.1 + 0.3/\eta - 4.8$ for the IEA value: at $\eta = 100\%$, 26.97 and 27.60; at 50%, 29.09 and 27.90; at 40%, 30.14 and 28.05. Neither source states $\eta$, so 40% is our assumption. The design crosses the lower BREF bound at $\eta = 77\%$. The BREF range also carries a stated ±1.5 GJ. Direct $CO_2$ with all purchased power at the IEA's global average of 475 g/kWh is $1.558 + 0.589 \times 0.475 = 1.84$ t/t.

**Limits of the blinding.** One design job (run `20261029-232421-s5b-implement`), whose task told it to find citable data, searched for and downloaded the BREF during the run. Its transcript, which is not published because it may hold third-party text, shows that it saw 27.6–31.8, 22.1 and 7.2–9.0, 100–250 bar and 20–30%. It recorded only 22.1 and 7.2–9.0 in `DESIGN.md` D-034, as a check on the design. The job computing the indicators saw that line when searching the design files and reports not using it; its values come from the script. Gas use was already about 27.8 GJ/t before the search; it later became 28.01 GJ/t through the feed specification approved by the author (D-040). We found no other design value moved toward these figures but cannot exclude influence. The reference values were selected by the orchestrating session, the same model as the designers.

## S12 Evaluation: Faults, Setup and Raw Results

Faults F01 to F19 were written knowing the rules, N1 to N4 are negative controls (implausible but self-consistent values), and the held-out faults H01 to H12 were written by a separate agent session without seeing the rules (the main text calls them faults written without seeing the rules). A held-out fault counts as *identified* when a new finding names the incorrect condition itself; "file changed" means that only a generic change, out-of-date or review signal was raised. H05 was reported at info level, so it would not block a merge.

The evaluation ran on 30 October 2026 at process-library `nh3-run-final` with core v1.1.0 (`c2a4513`), pyDEXPI 2.1.0, DWSIM 9.0.5 in Docker, on Windows x64 (32 logical processors), on scratch clones with one branch per fault. Each fault was followed by `cheme validate`, `maturity`, `check`, `impact` and `fmt -check`, and findings were compared with the baseline with commit hashes masked. Agent sessions used Claude Opus 5.5 through core's runner: held-out faults USD 1.66, reference impact sets USD 2.49, both on a copy of the engineering files without checks, skills, reviews, run records or `cheme` (`DESIGN.md` and some generated files still name rule identifiers), committed before any run; agent-driven change USD 8.55, plus a first attempt of USD 10.94 on an installation without the `props` extra, not reported. The readings of the held-out faults are ours, from their new findings. The scripts, patches and results are archived on Zenodo (`https://doi.org/10.5281/zenodo.23913145`).

Table S9: Injected faults, negative controls and held-out faults. Kind: the kind of rule the fault targets. Rule: the intended rule or maturity requirement that fired; for held-out faults, our reading of the new findings. At: finding at the edited file/line.

| id | kind | fault | rule or reading | at |
|---|---|---|---|---|
| F01 | schema | Equipment list: design temperature of P-301A as a string (wrong type) | yaml.schema | Y/Y |
| F02 | schema | Flowsheet: provenance of the S1 pressure outside its enumeration | process-schema | Y/Y |
| F03 | schema | P-109 DEXPI XML made malformed (one closing tag misspelled) | xml.syntax | Y/Y |
| F04 | schema | P-105: P-301A given a non-existent DEXPI class | dexpi.load | Y/Y |
| F05 | consistency | Instrument PI-1999 added to the instrument list, on no P&ID | instrument-not-on-pid | Y/Y |
| F06 | consistency | Instrument TI-1003 (drawn on P-101) removed from the instrument list | instrument-not-listed | N/N |
| F07 | consistency | Line list: NG-1003 DN 200 -> DN 250 (P-102 draws DN 200) | line-dn-mismatch | Y/Y |
| F08 | consistency | Equipment list: P-301A type CentrifugalPump -> ReciprocatingPump | equipment-type-mismatch | Y/Y |
| F09 | consistency | Flowsheet stream S1 links to non-existent line NG-9999 | process-link-unresolved | Y/Y |
| F10 | consistency | Flowsheet unit E101 links to non-existent equipment E-199 | process-link-unresolved | Y/Y |
| F11 | consistency | Equipment list cites unknown decision D-999 (P-301A notes) | decision-unknown | Y/Y |
| F12 | physical | Stream S3 flow +5 %: balances no longer close | process-balance, balanced.balances-closed | Y/N |
| F13 | physical | Feed S1 flow kept 'given' but its source removed | sourced.boundary | N/N |
| F14 | physical | Dryer X401 component separator: basis_source removed (model basis undeclared) | balanced.model-basis | N/N |
| F15 | out of date | Feed S1 temperature 288.15 -> 293.15 K without re-running the simulation or rebuilding | generated-outdated, process-calculated-stale, process-sim-run-stale, predictive.current | Y/N |
| F16 | out of date | Feed title changed without 'cheme process build' (BFD and PFD drift) | generated-outdated | N/N |
| F17 | out of date | P-301A length 4.0 -> 5.0 m in the equipment list (datasheet and layout model built from 4.0) | datasheet-dimension-mismatch, layout-model-outdated | N/N |
| F18 | review | P-109 title block revision A -> B (cheme sheet meta), committed | hazop-input-changed, review-needed | N/N |
| F19 | review | Layout: P-302A moved 1 m east, model not rebuilt | layout-model-outdated, review-needed | N/N |
| N1 | negative | Datasheet P-301A operating pressure 3100 -> 310 kPa(a) (self-consistent, implausible for the absorber) | not judged; datasheet-drawing-stale | N/N |
| N2 | negative | List and datasheet: V-501 design pressure 161 -> 100 barg (below the 147 bar(a) operating pressure) | not judged; datasheet-drawing-stale, layout-model-outdated, review-needed | N/N |
| N3 | negative | HAZOP H-01: likelihood C -> A on the first scenario (judgment) | not judged; hazop-band-stale, review-needed | Y/Y |
| N4 | negative | DESIGN.md prose: 1,498.5 t/d -> 1,598.5 t/d in text only | not judged; review-needed | N/N |
| H01 | held-out | Product delivery temperature above its boiling point | none | N/N |
| H02 | held-out | V-701 material downgraded to plain carbon steel | file changed | N/N |
| H03 | held-out | V-501 MDMT above its minimum operating temperature | file changed | N/N |
| H04 | held-out | Let-down valve LV-1903 fails open | none | N/N |
| H05 | held-out | Steam low-low trip I-101 uses the control transmitter | identified: control-shared-element, hazop-input-changed | N/N |
| H06 | held-out | PSV-1911 set above the V-501 design pressure | file changed | N/N |
| H07 | held-out | HAZOP risk band understated for front-end overpressure | identified: hazop-band-edited | Y/Y |
| H08 | held-out | Estimate total has transposed digits | file changed | N/N |
| H09 | held-out | Steam co-product exergy replaced by its enthalpy | none | N/N |

| id | kind | fault | rule or reading | at |
|---|---|---|---|---|
| H10 | held-out | PSV-1911 tag mistyped on the P&ID | identified: hazop-input-changed, label-tag, label-tag-unknown, layout-model-outdated, piping-item-not-listed, piping-item-not-on-pid | Y/N |
| H11 | held-out | V-502 placed on top of V-501 | identified: layout-clash, layout-line-length, layout-model-outdated | Y/Y |
| H12 | held-out | Decision log gives E-501 hot outlet below the NH3 dew point | file changed | N/N |

Section S13 gives the engineering change studies.

## S13 Evaluation: Engineering Change Studies

**Scripted changes.** Five engineering changes were made on separate clones of `nh3-run-final`, each committed and followed by one pass of the checks (Table S10). No files were repaired, because this test asks what the checks report first. The checks stop at the first broken layer, so a staged repair would reach more files. A separate agent session wrote the reference impact sets on a stripped copy of the files, and they were committed before the runs. They are an agent's judgment and no engineer has reviewed them. Each file in a set is marked as needing an edit, a regeneration or a review, and Table S11 gives the results.

Table S10: The five scripted changes and the rules that raised new findings.

| Change | Edit | New rules |
|---|---|---|
| C1 pump rerated | P-301A head +25%: discharge pressure 3100 to 3875 kPa(a) in the list note and the datasheet | `datasheet-drawing-stale`, `layout-model-outdated`, `review-needed` |
| C2 pump retagged | P-301A to P-303A with `cheme rename`, and the list entry. Afterward, 22 files still named the old tag | 14 rules, including `control-unknown-tag`, `hazop-unknown-tag`, `electrical-unknown-tag`, `layout-unknown-tag` |
| C3 +20% capacity | Design basis 1500 to 1800 t/d, given feed flows ×1.2, then `propagate`, `props` and `build` | `process-balance`, `process-calculated-stale`, `process-sim-run-stale`, `generated-outdated`, `control-case`, `review-needed` |
| C4 line enlarged | NG-1003 DN 200 to DN 250 in the line list and on P-101 and P-102 | `hazop-input-changed`, `layout-model-outdated`, `review-needed` |
| C5 feed gas | $CH_4$ 94.67 to 90.23, $C_2H_6$ 4.40 to 8.0, $C_3H_8$ 0.23 to 1.07 mol%, then `propagate` and `build` | `process-balance`, `process-calculated-stale`, `process-sim-run-stale`, `generated-outdated`, `review-needed` |

Files of the reference sets that were not reached include the electrical list, the P-301B datasheet and the estimate for C1; the cable schedule and the estimate for C2; and the E-101 and V-101 datasheets, the estimate and the MTO for C4. For C3 and C5, the first pass ends on balance errors in `process/flowsheet.yaml`, and the calculation maturity falls from *balanced* to *sourced*, so most downstream disciplines were not yet flagged. The checks flagged no file outside the reference sets in any of the five changes.

**Agent-driven change: setup.** One unattended session on a fresh clone received a written brief. It asked for 1,800 t/d with the same product, feed and site, propagated from the flowsheet to

Table S11: Change propagation against the reference impact sets: files of the set flagged by a finding or named in its message, in the first pass after the change.

| change | reference | flagged | named | not reached | errors/warnings | maturity |
|---|---|---|---|---|---|---|
| C1 pump re-rated | 19 | 2 | 2 | 15 | 0/2 | balanced → balanced |
| C2 pump re-tagged | 22 | 11 | 2 | 9 | 2/16 | balanced → balanced |
| C3 +20% capacity | 161 | 5 | 0 | 156 | 8/110 | balanced → sourced |
| C4 line enlarged | 14 | 2 | 4 | 8 | 0/3 | balanced → balanced |
| C5 feed gas | 115 | 4 | 0 | 111 | 2/108 | balanced → sourced |

the estimate, TEA and LCA as far as the budget allowed. The agent was told to work from what the checks report, not to relax a gate or sign off a stage, and to log every iteration. It ran with a budget of USD 19.90, used USD 8.55 and 49 minutes, and did not need a human intervention.

**Agent-driven change: what it did.** The session worked in four iterations. It first changed the design basis, feeds and specification duties, re-simulated in DWSIM with the furnace closed by iteration, and rebuilt the BFD and PFD. It then updated the control setpoints, datasheets (duties, flows, powers and exchanger areas), relief loads, motor powers, cable schedule, layout model, MTO, lifting study, estimate, TEA and LCA. The independent recheck of its final commit gave 0 errors, 252 warnings, 21 of 25 gates and a calculation maturity of *balanced*. It changed 128 of the 161 files in the reference set; the 33 it left unchanged include every P&ID, the line list and routing, the layout inputs and the HAZOP study.

**Agent-driven change: what it reported.** These values are from the agent's own log and no engineer has reviewed them: product 1,799.2 t/d; natural gas 0.5706 t/t $NH_3$; electricity 0.589 MWh/t (44.1 MW); steam export 1.02 t/t; carbon footprint 1.465 kg $CO_2$-eq/kg; equipment cost USD 362.0 to 404.4 million; levelized cost 688 to 674 EUR/t. The consumption and emissions per tonne (natural gas, electricity, steam export and carbon footprint) are those of the 1,500 t/d design, because the balance was scaled linearly with the feed. The levelized cost falls because the equipment cost rises by 12%, not 20%: items priced on duty or power were repriced, but vessels and columns kept their 1,500 t/d dimensions, which the agent noted understates the cost.

**Agent-driven change: what stayed open.** The agent judged that every throughput-sized item must be resized or re-rated, and it did not do this. Its list covers machines and motors, the H-101 furnace and R-101 tubes, reactors and catalyst beds at the same space velocity, every exchanger, columns and separators, the steam and cooling water systems, and relief devices. By its own screening, 36 gas lines exceed the design velocities, 20 of them by more than 10%. Equipment dimensions and weights, and with them the layout, MTO and vessel and column costs, remain those of the 1,500 t/d plant. It stated that it would not trust the costs and the levelized cost, the exchanger areas (scaled with duty at unchanged temperature difference), or the relief loads (rated flows without machine curves, with no device sizes calculated). It also noted that higher pressure drops, off-design machines and a different loop pressure are not modeled. The HAZOP study was left for a team to reassess. The brief, log and patch are archived on Zenodo (`https://doi.org/10.5281/zenodo.23913145`).