

MCP Design Strategies for AI Agents in the Chemical Engineering Industry

Byunghyun Ban¹²

¹Department of Music, Gyungkook National University

²NaNa Lab

bhban@nanalab.kr

Abstract

Large language model (LLM) agents are increasingly being applied to chemistry and chemical engineering tasks, including synthesis planning, autonomous experimentation, process simulation, process design, optimization, and carbon accounting. Existing studies demonstrate that agents become powerful when connected to external tools, simulators, databases, and laboratory or process interfaces. However, many of these studies focus on what agents can do once appropriate tools are available, while providing less guidance on how chemical engineering experts should design, implement, and validate those tools for industrial environments. This paper proposes design strategies for Model Context Protocol (MCP)-based AI agents in the chemical engineering industry, including a taxonomy of industrial MCP servers, including Process Data MCP, SOP and Work Instruction MCP, Safety and Hazard MCP, Process Simulation MCP, Maintenance MCP, Quality and Laboratory Data MCP, Environmental and Carbon Accounting MCP, and Operator Knowledge Capture MCP. For each MCP type, the paper identifies required data, protocols, worker collaboration methods, and evaluation criteria. The paper concludes that future chemical engineering agents should be developed through field-validated MCP ecosystems rather than through agent frameworks alone.

Keywords: Model Context Protocol, MCP, AI agents, chemical engineering, process engineering, industrial AI, operator knowledge, field validation, process simulation, safety, human-AI collaboration

1. Introduction

Can AI agents understand chemical engineering processes? In industrial settings, the answer depends less on the agent framework itself than on whether expert know-how is encoded into executable, auditable, and field-validated tools.

Recent work in chemistry and chemical engineering has shown that LLM-based agents can perform increasingly complex scientific and engineering tasks when connected to external tools. ChemCrow integrated GPT-4 with eighteen expert-designed chemistry tools and demonstrated applications in organic synthesis, drug discovery, and materials design [1]. Coscientist showed that LLM agents can design, plan, and perform chemical experiments by using tools such as internet search, documentation search, code execution, and experimental automation [2]. These studies demonstrate that the practical capability of scientific agents comes not from language generation alone, but from reliable tool use.

A similar transition is now occurring in chemical engineering. Recent work has integrated LLM agents with process simulators, allowing natural-language interaction with rigorous simulation software through tool interfaces. For example, one process-simulation agent connects an LLM to AVEVA Process Simulation through a Model Context Protocol (MCP) server, where the agent interprets user requests and the MCP server executes deterministic simulator operations [3]. Other recent studies propose multi-agent workflows that transform textual process specifications into executable simulation configurations, or reasoning agents that automate process simulation, optimization, carbon accounting, and

decarbonization analysis [4,5]. These works show that LLM agents can support chemical process design, simulation, and optimization when provided with domain-specific tools.

However, these studies also reveal a missing layer. Existing chemistry and chemical-engineering agent studies primarily demonstrate what agents can do when connected to tools, simulators, databases, or robotic systems. They do not generally provide a field-oriented design strategy for constructing MCP tools in industrial chemical engineering environments. Even when MCP is used, the scope is often limited to a specific technical interface, such as a process simulator server, rather than a broader methodology for designing plant-data MCPs, SOP MCPs, safety MCPs, maintenance MCPs, quality MCPs, carbon-accounting MCPs, and operator-knowledge MCPs.

This gap is important because industrial chemical engineering agents cannot be deployed by agent frameworks alone. A chemical plant does not need an agent that can merely discuss distillation, reaction engineering, maintenance, safety, or quality control in natural language. It needs an agent that can retrieve live process data, check approved operating procedures, compare current values against operating limits, run validated simulators, identify unsafe recommendations, search maintenance history, summarize quality deviations, and escalate uncertain cases to human experts.

The Model Context Protocol provides a useful foundation for this transition. MCP is an open protocol for connecting LLM applications with external data sources and tools [6,7]. MCP servers can expose executable tools, contextual resources, and reusable prompts, allowing agents to invoke external functions, retrieve relevant information, and operate within structured interaction templates [8-10]. In chemical engineering, this means MCP can serve as the boundary between natural-language reasoning and executable industrial action.

Yet the value of MCP depends on its design. A generic MCP server that exposes arbitrary APIs is not enough for chemical engineering. Industrial MCP tools must encode process constraints, safety limits, SOPs, equipment hierarchies, data schemas, validation rules, and the tacit knowledge of experienced operators, technicians, and engineers. Without such field-specific design, chemical engineering experts may know that agents are promising, but still lack guidance on which MCP servers to build, what data and protocols are required, how field workers

should participate, and how these tools should be evaluated.

This paper addresses that gap by proposing MCP design strategies for AI agents in the chemical engineering industry. The central claim is:

The bottleneck of industrial chemical engineering agents is not the absence of another agent framework, but the absence of field-validated MCP tools that encode plant data, safety constraints, operating procedures, and tacit worker expertise into executable and auditable interfaces.

The paper makes four contributions.

1. It reframes MCP as the executable boundary between LLM agents and chemical engineering operations.
2. It proposes a taxonomy of MCP servers required for chemical engineering industrial agents.
3. It describes the data, protocols, worker collaboration methods, and evaluation criteria required for each MCP type.
4. It proposes a field-validation framework for developing MCP tools with plant workers, engineers, safety officers, quality personnel, and maintenance technicians.

2. Background: From Tool-Augmented Agents to Field-Validated MCPs

2.1 Tool-augmented agents in chemistry

Tool-augmented agents have become a major direction in chemistry-oriented LLM research. ChemCrow showed that integrating expert-designed tools into an LLM can produce capabilities beyond text-only generation. The system used GPT-4 as the language model and connected it to eighteen chemistry tools, enabling tasks in synthesis planning, materials discovery, and chemical reasoning [1].

Coscientist similarly demonstrated that LLM agents can go beyond literature summarization when they are connected to tools for search, code execution, documentation retrieval, and experimental automation. Its value lies not only in language generation, but in the orchestration of external tools for experimental design and execution [2].

Scientific agents become useful when they can act through verified tools.

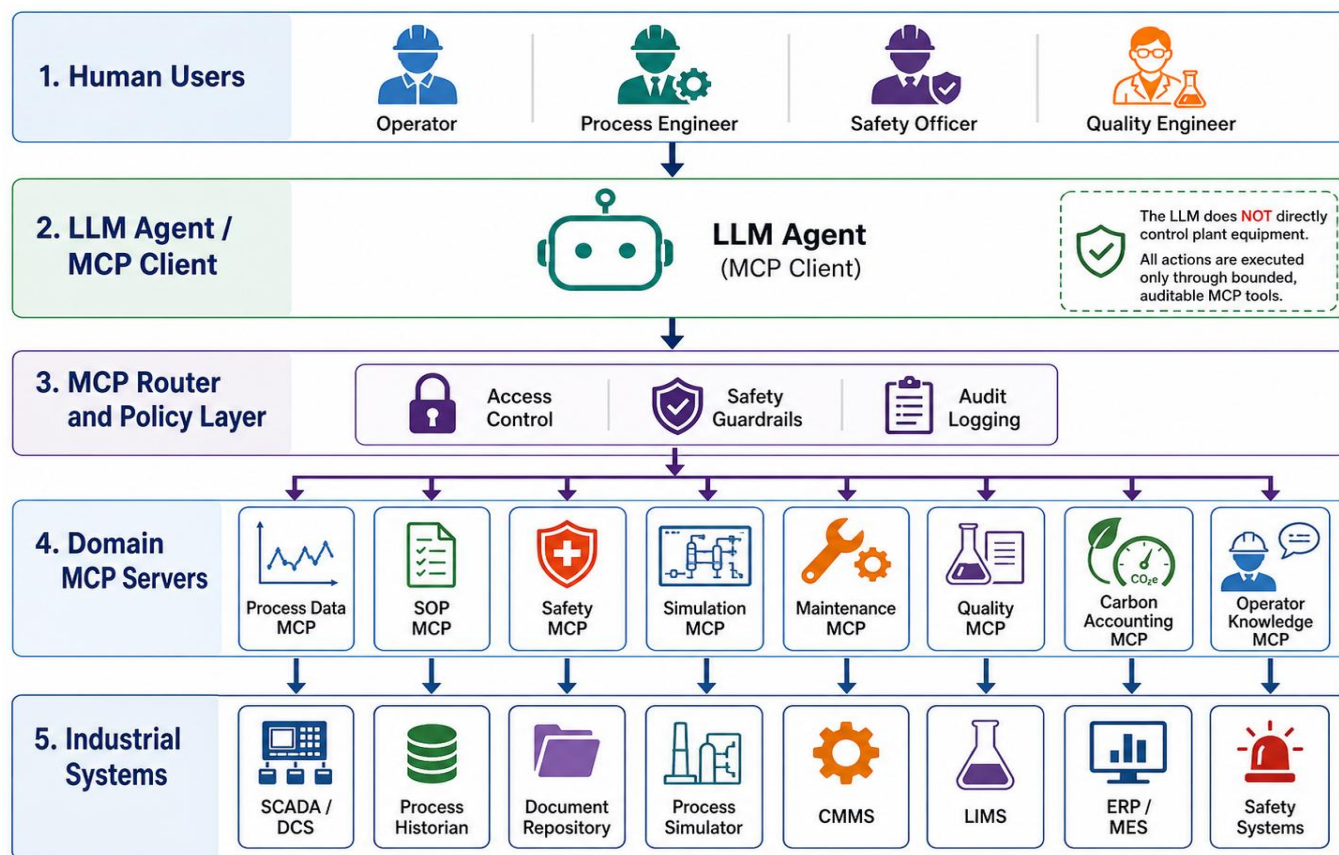


Figure 1. MCP-based AI agent architecture for chemical engineering. The LLM agent does not directly control plant systems; bounded, auditable MCP tools mediate all actions

For chemistry, this may mean querying molecule databases, planning reactions, checking safety, controlling robotic equipment, or executing code. For chemical engineering, it means something more industrial: reading process historians, running simulators, checking operating constraints, retrieving SOPs, supporting maintenance decisions, and producing auditable recommendations under safety and reliability constraints.

2.2 Chemical engineering agents and process simulation

Chemical process simulators are central to process design, optimization, and troubleshooting. However, constructing and interpreting simulations requires expert knowledge. Recent work has integrated an LLM agent with AVEVA Process Simulation through MCP. In that architecture, the LLM interprets natural-language requests and invokes a curated toolset exposed by a dedicated MCP server. The MCP server wraps the simulator's Python API into high-level tools with defined inputs, outputs, and usage

descriptions, while deterministic engineering operations are executed by the simulator through the server [3].

This architecture is important because it separates natural-language reasoning from deterministic engineering execution. The LLM plans and selects tools; the MCP server performs structured operations; the simulator executes engineering calculations.

Other work on automated chemical process design has explored multi-agent LLM workflows that transform textual process specifications into executable simulation configurations [4]. Another recent study on reasoning-agent-driven distillation decarbonization combined LLM-based reasoning with toolsets for process simulation, optimization, carbon accounting, and decarbonization intervention evaluation [5]. These studies demonstrate the feasibility of tool-augmented chemical engineering agents, but they do not fully answer the field-design question: which MCP tools should industrial chemical engineers build, how should those tools encode plant know-how, and how should they be validated with field workers?

2.3 MCP as an executable boundary

MCP should not be treated as just another API connector. In industrial AI, MCP can function as the boundary between:

- natural language and typed action,
- reasoning and execution,
- generic models and domain systems,
- agent autonomy and industrial control,
- plant data and auditable recommendations.

MCP tools allow language models to invoke external functions such as database queries, API calls, and computations [8]. MCP resources allow servers to expose contextual data such as files, database schemas, or application-specific information [9]. MCP prompts allow servers to provide structured interaction templates to clients [10]. These three primitives—tools, resources, and prompts—are well suited to chemical engineering environments, where agents must combine plant data, documents, computation, and task-specific instructions.

However, MCP introduces design and security challenges. Industrial MCP tools must be permissioned, logged, typed, validated, and constrained. MCP security guidance emphasizes risks and best practices for MCP implementations, including authorization, threat modeling, and secure handling of protocol-specific risks [11]. In chemical plants, these concerns are amplified because tool calls may influence safety-critical decisions.

3. Problem Statement

3.1 Existing agent studies demonstrate capability, not deployment strategy

Recent agent studies in chemistry and chemical engineering show that LLM-based agents can perform meaningful tasks when connected to external tools. ChemCrow and Coscientist demonstrate chemistry-oriented tool use and experimental automation [1,2]. Process-simulation agents demonstrate that LLMs can interact with simulation software through structured tools or MCP servers [3]. Multi-agent process design studies demonstrate that LLM workflows can translate natural-language process specifications into simulation configurations [4]. Reasoning-agent studies show that LLMs combined with toolsets can support simulation,

optimization, carbon accounting, and decarbonization analysis [5].

However, these works mainly answer the question:

What can an agent do when appropriate tools are already provided?

They do not fully answer the more practical industrial question:

How should chemical engineering experts design, validate, and govern the MCP tools that agents will actually use in the field?

This distinction matters. Demonstrating an agent workflow is not the same as specifying an industrial MCP ecosystem.

3.2 Agent frameworks are not enough

Many AI agent papers emphasize frameworks: planner-executor loops, reflection, multi-agent collaboration, memory modules, tool-use policies, or prompt harnesses. These are useful, but they are not sufficient for chemical engineering industry deployment.

A plant engineer will not ask only whether an agent can reason. The engineer will ask:

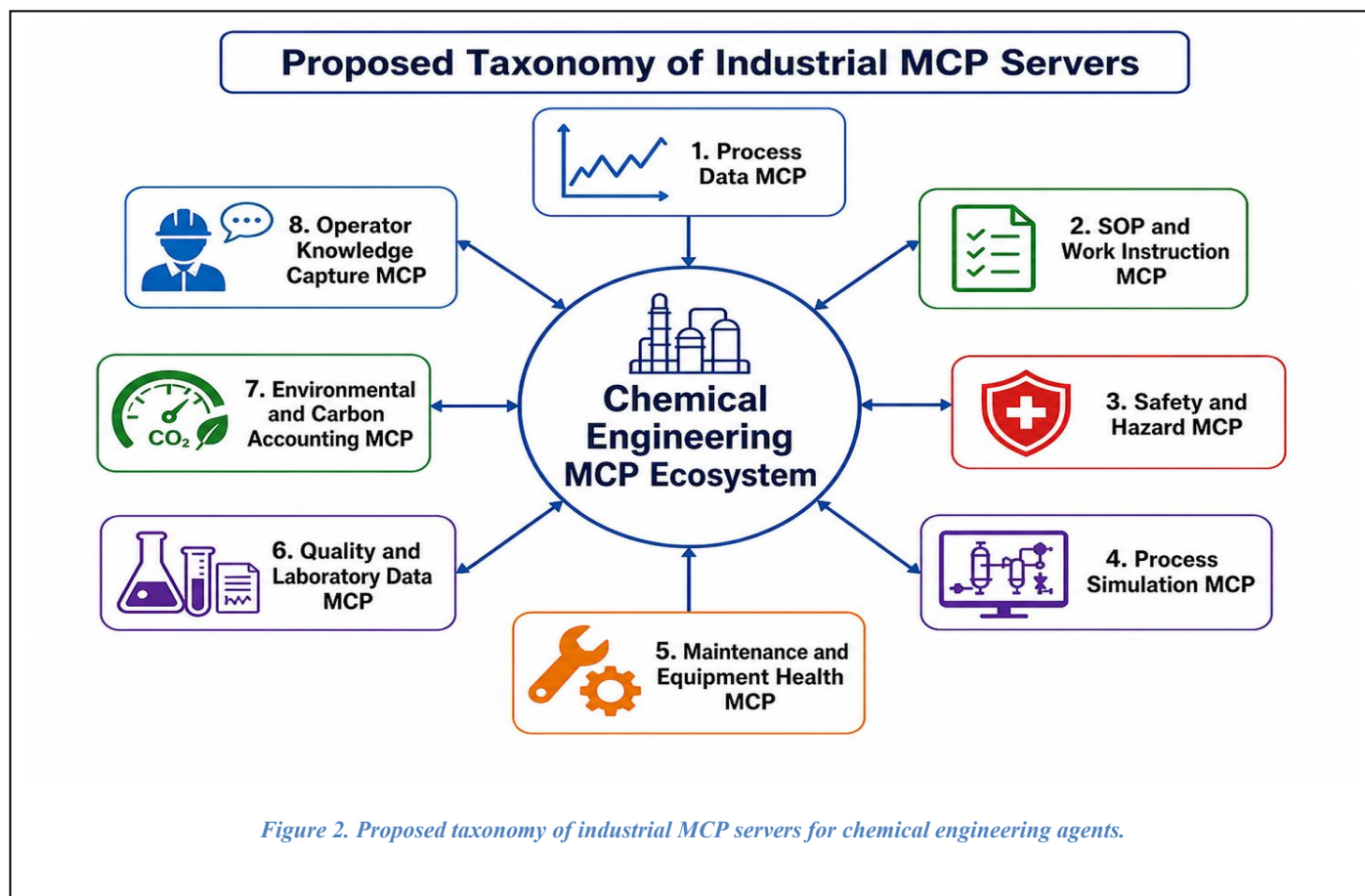
- Can it retrieve the correct process tags?
- Can it distinguish current values from historical trends?
- Can it find the latest approved SOP?
- Can it check design limits?
- Can it run a validated simulator?
- Can it avoid unsafe operating recommendations?
- Can it explain which data and procedure it used?
- Can it incorporate field experience from operators?
- Can its recommendations be audited after the fact?

These questions are not primarily about agent framework design. They are about MCP tool design, data governance, safety constraints, and field validation.

3.3 Harness engineering versus field-tool engineering

Prompt harness engineering improves how agents think, speak, and interact. Field-tool engineering determines what agents can safely know and execute.

For chemical engineering applications, field-tool engineering is more important than general prompt orchestration. A well-prompted agent without plant data,



SOP access, simulation capability, safety constraints, and worker-validated rules remains a conversational assistant. A field-validated MCP ecosystem can turn the same agent into an auditable decision-support interface.

3.4 Missing layer: field-validated MCP design

The missing layer in industrial chemical engineering agents is not another general-purpose agent framework. The missing layer is a methodology for designing MCP servers that encode plant data, process constraints, SOPs, safety rules, maintenance history, quality data, environmental calculations, and worker expertise into executable and auditable tools.

This paper addresses that gap by proposing MCP design strategies for the chemical engineering industry.

4. Proposed MCP Taxonomy for the Chemical Engineering Industry

This section proposes eight MCP server categories required for chemical engineering industrial agents.

4.1 Process Data MCP

Purpose

The Process Data MCP provides structured access to real-time and historical process data.

Required data

- sensor tags
- historian data
- DCS/SCADA logs
- temperature, pressure, flow, level, and composition values
- alarm logs
- batch records
- maintenance events
- operating ranges
- process-unit metadata

Required protocols and interfaces

- OPC UA
- PI System or historian API
- MQTT
- SQL or time-series database access
- REST API
- tag dictionary

- equipment hierarchy schema
- timestamp normalization

Example tools.

Tool	Function
get_current_process_values	Retrieve current process values for selected tags.
query_historian	Query time-series data over a specified period.
detect_trend_shift	Detect abnormal trend changes.
compare_with_normal_range	Compare values with approved operating ranges.
summarize_batch_profile	Summarize a batch or campaign profile.
retrieve_alarm_sequence	Retrieve and summarize recent alarm sequences.

Worker collaboration

Operators and process engineers should help define critical tags, normal operating ranges, meaningful trend deviations, alarm patterns that matter in practice, false alarm patterns, and unit-specific interpretation rules.

Evaluation criteria.

Evaluation item	Metric
Data retrieval accuracy	Correct tag, unit, timestamp, and value retrieval.
Tag mapping accuracy	Agreement between requested variable and actual tag.
Trend detection	Precision and recall against historical abnormal events.
Operator agreement	Operator judgment on whether trend summaries are useful.
Latency	Time to retrieve and summarize data.
Auditability	Completeness of tool-call logs.

4.2 SOP and Work Instruction MCP

Purpose

The SOP and Work Instruction MCP retrieves approved procedures and converts them into actionable, cited, and version-controlled guidance.

Required data

- SOP documents
- work instructions
- startup and shutdown procedures
- emergency response manuals
- cleaning and changeover procedures
- inspection checklists
- permit-to-work forms
- document approval metadata
- document effective dates

Required protocols and interfaces

- document repository API
- vector index with citation support
- version-control metadata
- role-based document access
- approval state tracking
- document retirement rules

Example tools.

Tool	Function
retrieve_sop	Retrieve SOPs relevant to equipment, unit, or situation.
get_latest_approved_procedure	Confirm latest approved document version.
check_step_compliance	Compare a described action with required SOP steps.
generate_operator_checklist	Create checklist from approved procedure.
compare_current_action_with_sop	Identify deviations from SOP.

Worker collaboration

Field workers should help identify procedural steps that are written but not followed, practical steps that are followed but not documented, ambiguous SOP sections, situations where strict SOP execution requires judgment, and unsafe shortcuts that should never be recommended.

Evaluation criteria.

Evaluation item	Metric
SOP retrieval accuracy	Whether the correct procedure is retrieved.
Citation correctness	Whether each instruction is traceable to source text.
Version accuracy	Whether obsolete procedures are avoided.
Missing critical step rate	Frequency of omitted safety-critical steps.
Operator usefulness	Worker rating of checklist clarity and usefulness.
Unsafe recommendation rate	Must approach zero.

4.3 Safety and Hazard MCP

Purpose

The Safety and Hazard MCP ensures that agent recommendations respect safety constraints, hazard analyses, and escalation rules.

Required data

- HAZOP worksheets
- LOPA tables
- P&IDs
- SDS/MSDS data

- equipment design limits
- interlock lists
- alarm setpoints
- safety-critical operating windows
- incident and near-miss records
- emergency response procedures

Required protocols and interfaces

- hazard ontology
- equipment-to-tag mapping
- safety-critical action blacklist
- permit-to-work interface
- approval workflow
- emergency escalation routing
- interlock status interface

Example tools.

Tool	Function
check_operating_limit	Check whether a condition exceeds approved operating limits.
retrieve_hazard_scenario	Retrieve relevant HAZOP or LOPA scenarios.
check_interlock_status	Check safety interlock status.
summarize_sds_hazard	Summarize material hazards from SDS.
recommend_safe_escalation	Suggest safe escalation path rather than direct action.
block_unsafe_action	Reject tool calls that violate safety constraints.

Worker collaboration

Safety officers, operators, and engineers should identify high-risk operating regions, known near-miss patterns, actions that require dual approval, unsafe recommendations produced during testing, and cases where the safest response is escalation rather than optimization.

Evaluation criteria.

Evaluation item	Metric
Unsafe action rate	Frequency of unsafe recommendations.
Hazard retrieval accuracy	Whether relevant HAZOP scenarios are retrieved.
SDS correctness	Accuracy of material hazard summaries.
Escalation appropriateness	Whether the tool recommends proper human escalation.
Safety officer approval	Expert assessment of outputs.
Fail-safe behavior	Whether the tool refuses uncertain or dangerous requests.

4.4 Process Simulation MCP

Purpose

The Process Simulation MCP connects agents to process simulation software for what-if analysis, sensitivity studies, flowsheet inspection, and optimization.

Required data

- simulator project files
- flowsheets
- thermodynamic packages
- feed compositions
- unit operation parameters
- convergence settings
- operating constraints
- design cases
- simulation result files

Required protocols and interfaces

- Aspen Plus, Aspen HYSYS, AVEVA Process Simulation, gPROMS, DWSIM, or internal simulator APIs
- Python API wrapper
- simulation sandbox
- parameter boundary schema
- convergence error handling
- result export schema
- version-controlled model files

Example tools.

Tool	Function
load_flowsheet	Load a validated flowsheet.
inspect_unit_operation	Retrieve unit operation configuration.
set_stream_condition	Modify stream condition within allowed bounds.
run_simulation	Execute simulation.
get_simulation_results	Retrieve simulation results.
run_sensitivity_analysis	Perform parameter sensitivity analysis.
optimize_operating_condition	Search for feasible operating conditions.
diagnose_convergence_failure	Analyze convergence failure and suggest safe next steps.

Worker collaboration

Process engineers should define allowed parameter ranges, known simulator limitations, convergence troubleshooting rules, differences between simulator assumptions and plant reality, and conditions that appear optimal in simulation but are infeasible in operation.

Evaluation criteria.

Evaluation item	Metric
Simulation execution success	Percentage of successful simulation runs.
Result reproducibility	Whether repeated runs produce consistent results.
Feasibility of recommendations	Engineer assessment of suggested conditions.
Convergence handling	Ability to diagnose and recover from failures.
Time saved	Reduction in manual simulation workflow time.
Boundary compliance	Whether all tool calls respect process constraints.

Recent process-simulation agents already show the importance of this pattern. In one MCP-based process simulation framework, the MCP server acts as an intermediary between the LLM agent and process simulation software, wrapping simulator APIs into high-level tools with structured inputs and outputs [3]. That work supports the present paper’s claim that MCP design is central to chemical engineering agents.

4.5 Maintenance and Equipment Health MCP

Purpose

The Maintenance MCP supports equipment-health monitoring, fault diagnosis, maintenance-history retrieval, and work-order preparation.

Required data

- CMMS or EAM records
- equipment hierarchy
- maintenance history
- vibration data
- motor current
- valve position
- pump and compressor performance curves
- inspection logs
- spare-parts inventory
- failure mode records

Required protocols and interfaces

- CMMS/EAM API
- historian integration
- equipment ID schema
- failure mode taxonomy
- maintenance work-order protocol
- spare-parts inventory interface

Example tools.

Tool	Function
get_equipment_history	Retrieve maintenance history.
analyze_vibration_trend	Analyze vibration trends.
predict_failure_risk	Estimate equipment failure risk.
retrieve_failure_mode	Retrieve FMEA or known failure modes.
check_spare_parts_availability	Check spare part inventory.
draft_maintenance_recommendation	Draft maintenance recommendation for human approval.

Worker collaboration

Maintenance technicians should contribute symptoms preceding common failures, practical diagnosis heuristics, false-positive maintenance burdens, equipment-specific exceptions, field notes from repeated failures, and priority rules for work orders.

Evaluation criteria.

Evaluation item	Metric
Failure prediction performance	Precision, recall, lead time.
Technician agreement	Expert agreement with diagnosis.
Work-order relevance	Whether suggested work orders are useful.
False maintenance rate	Unnecessary maintenance recommendations.
Downtime impact	Potential or measured downtime reduction.
Explainability	Whether technicians understand the recommendation.

4.6 Quality and Laboratory Data MCP

Purpose

The Quality and Laboratory Data MCP connects agents to quality-control results, laboratory information, product specifications, and batch traceability.

Required data

- LIMS data
- QC test results
- product specifications
- batch records
- lab instrument outputs
- impurity profiles
- chromatogram or spectra metadata
- out-of-specification records
- sample chain-of-custody data

Required protocols and interfaces

- LIMS API

- batch ID linkage
- sample ID linkage
- specification version control
- chain-of-custody metadata
- OOS escalation procedure

Example tools.

Tool	Function
get_batch_quality_results	Retrieve quality results for a batch.
check_spec_compliance	Compare results with product specification.
trace_quality_deviation	Trace possible process causes of quality deviation.
link_process_to_quality	Analyze process-quality relationships.
summarize_oos_case	Summarize out-of-specification case.
retrieve_related_batches	Find similar batches or deviations.

Worker collaboration

QC analysts and production engineers should define how specifications are interpreted, how borderline results are handled, common sampling errors, known process-quality relationships, practical escalation routes, and audit requirements.

Evaluation criteria.

Evaluation item	Metric
Specification-check accuracy	Correct pass/fail classification.
OOS detection	Correct identification of out-of-spec cases.
Batch traceability	Accuracy of batch and sample linkage.
QC staff agreement	Expert assessment of output.
False compliance rate	Incorrectly marking non-compliant material as compliant.
Audit readiness	Traceability of all retrieved evidence.

4.7 Environmental and Carbon Accounting MCP

Purpose

The Environmental and Carbon Accounting MCP supports energy analysis, emissions calculation, carbon accounting, and decarbonization scenario evaluation.

Required data

- utility consumption
- steam and electricity balances
- fuel usage
- emission factors
- process energy data

- carbon intensity scenarios
- waste treatment records
- energy price data
- regulatory reporting requirements

Required protocols and interfaces

- GHG Protocol or ISO-compatible calculation schema
- emission factor version control
- energy balance validation
- scenario definition format
- regulatory reporting templates
- process simulator linkage

Example tools.

Tool	Function
calculate_scope_emissions	Calculate Scope 1, 2, or 3 emissions.
run_energy_balance	Perform process energy balance.
compare_energy_scenarios	Compare energy supply scenarios.
estimate_decarbonization_impact	Estimate emission reduction from process change.
generate_carbon_report	Generate auditable carbon report.
check_emission_factor_version	Verify emission factor source and version.

Worker collaboration

Environmental engineers and operations teams should validate meter reliability, emission factor selection, energy balance assumptions, feasibility of proposed decarbonization actions, operational consequences of energy changes, and reporting requirements.

Evaluation criteria.

Evaluation item	Metric
Calculation accuracy	Agreement with manual carbon accounting.
Scenario reproducibility	Repeatability of scenario results.
Regulatory consistency	Alignment with reporting rules.
Engineer agreement	Environmental engineer review score.
Sensitivity analysis	Effect of emission factor changes.
Auditability	Traceability of all assumptions and data sources.

A recent distillation decarbonization study demonstrates the relevance of such tools: the authors developed a reasoning agent using an LLM and an extensive toolset to execute process simulation, optimization, carbon accounting, and result visualization from natural-language commands [5].

4.8 Operator Knowledge Capture MCP

Purpose

The Operator Knowledge Capture MCP structures tacit field knowledge from operators, technicians, and engineers into validated, retrievable, and updateable knowledge objects. This is the most important MCP for field-oriented industrial agents.

Required data

- operator interviews
- shift handover notes
- abnormal situation narratives
- incident retrospectives
- near-miss explanations
- troubleshooting stories
- maintenance anecdotes
- experienced-worker heuristics
- validation workshop records

Required protocols and interfaces

- semi-structured interview protocol
- anonymization and consent procedure
- knowledge-card schema
- expert validation workflow
- versioning and retirement rules
- confidence score
- disagreement tracking
- retrieval index

Example tools.

Tool	Function
record_operator_case	Record field case in structured form.
structure_tacit_knowledge	Convert narrative into knowledge card.
retrieve_similar_incident	Retrieve similar past cases.
validate_knowledge_card	Check validation state by experts.
update_field_rule	Update field rule after approval.
flag_outdated_knowledge	Identify knowledge requiring review.

Worker collaboration

Field workers should be involved through contextual inquiry, shadowing, semi-structured interviews, shift handover observation, incident reconstruction workshops, think-aloud validation, abnormal scenario workshops, and cross-validation by multiple experienced workers.

Evaluation criteria.

Evaluation item	Metric
Knowledge-card acceptance	Percentage approved by field experts.
Inter-operator agreement	Agreement among experienced workers.
Retrieval usefulness	Operator rating for similar-case retrieval.
Time-to-diagnosis	Reduction in troubleshooting time.
Repeated mistake reduction	Frequency of repeated known errors.
Trust score	Worker trust in agent output.

The Operator Knowledge Capture MCP is the paper's most distinctive proposal. Existing agent studies often emphasize chemistry tools, simulators, robotic APIs, and software workflows. Industrial chemical engineering additionally requires tacit plant expertise: the practical knowledge of what operators notice, which alarms matter, which shortcuts are unsafe, when procedures require judgment, and how abnormal situations are handled.

5. MCP Design Methodology

5.1 Step 1: Select high-value field tasks

MCP development should begin with tasks that are operationally valuable and field-verifiable.

- high frequency
- high cost of error
- strong dependence on plant data
- dependence on expert judgment
- availability of historical cases
- feasibility of safe read-only or recommendation-only deployment
- ability to evaluate with field workers

Examples include alarm interpretation, batch deviation analysis, simulator-assisted troubleshooting, maintenance-history retrieval, and SOP-based checklist generation.

5.2 Step 2: Define tool boundaries

Not all MCP tools should have the same authority. Chemical engineering MCPs should be classified by action level.

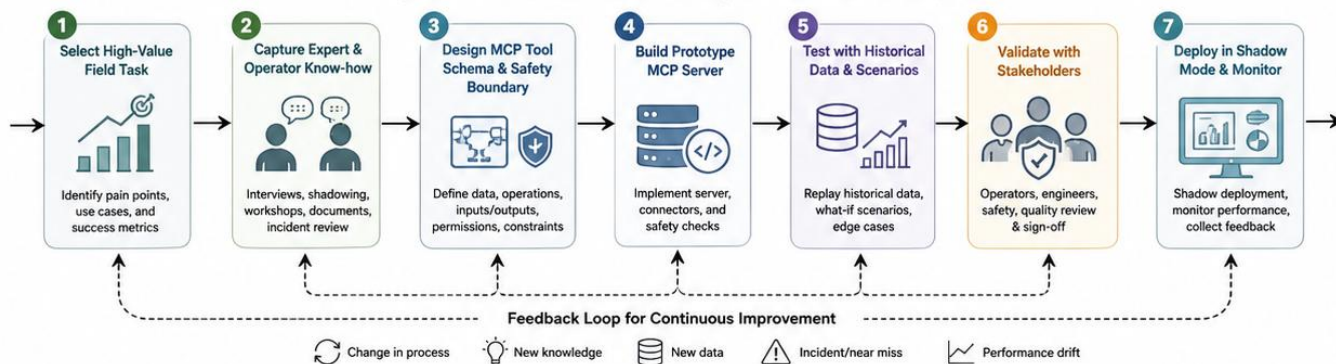


Figure 3. Field-validated MCP design and evaluation loop. MCP tools should be developed through iterative task selection, knowledge capture, schema design, prototyping, scenario testing, expert validation, shadow deployment, and continuous monitoring.

Table 1. MCP tool authority levels.

Level	Description	Example	Initial deployment recommendation
L0 Read-only	Retrieve information only.	SOP search, historian query.	Safe starting point.
L1 Analysis	Perform calculations or summaries.	Trend analysis, mass balance.	Allowed with logging.
L2 Recommendation	Generate recommendation.	Maintenance suggestion, operating advice.	Human approval required.
L3 Controlled execution	Execute action after approval.	Work-order creation.	Limited pilot only.
L4 Autonomous execution	Execute without human approval.	Direct control action.	Generally inappropriate for early deployment.

Chemical engineering MCPs should begin with L0-L2 tools. L3 and L4 tools should require extensive validation, formal hazard review, access control, and audit logging.

5.3 Step 3: Build a data and schema layer

Industrial MCP tools require consistent schema design. Key identifiers include equipment ID, tag ID, unit operation ID, batch ID, material ID, alarm ID, SOP ID, incident ID, maintenance work-order ID, sample ID, recommendation ID, and approval state. Without consistent identifiers, an agent cannot reliably connect process values, maintenance history, quality results, and procedures.

5.4 Step 4: Encode constraints

MCP tools should encode constraints before allowing analysis or recommendation. Important constraints include design limits, safety limits, operating windows, quality specifications, interlock constraints, environmental regulations, maintenance intervals, human approval requirements, and role-based access limits. A tool should not merely return data. It should also know whether a requested analysis or action is within its allowed operational boundary.

5.5 Step 5: Co-design tools with field workers

MCP tools should be co-designed with operators, technicians, engineers, safety officers, and quality personnel.

1. field observation
2. task decomposition
3. worker interviews
4. abnormal case collection
5. prototype tool walkthrough
6. scenario testing
7. shadow-mode deployment
8. feedback-based revision

Field workers should not be treated only as end-users. They are co-designers of operational knowledge, safety boundaries, and exception-handling logic.

5.6 Step 6: Deploy in safe mode

Initial deployment should be conservative: read-only mode, recommendation-only mode, no direct control, mandatory human approval, access logging, output citation, refusal on uncertainty, and escalation for high-risk cases.

5.7 Step 7: Continuously validate and update

Industrial MCP systems require continuous validation. Monitoring should include tool-call logs, operator override cases, false recommendation reports, near-miss events, SOP updates, tag changes, simulator version changes, worker feedback, and model/tool drift. An MCP tool that was valid last year may become unsafe after equipment changes, SOP revisions, or new operating constraints.

6. Worker Collaboration Protocol

6.1 Workers are not merely users

Field workers are not merely users of AI agents. They are holders of operational knowledge that often does not exist in formal documents.

- what abnormal trends look like
- which alarms matter
- when SOPs require judgment
- how equipment behaves before failure
- which sensor values are unreliable
- which operating changes are risky despite appearing feasible
- how work is actually performed during night shifts, startups, shutdowns, and emergencies

6.2 Collaboration methods

Table 2. Worker collaboration methods for MCP design.

Method	Purpose
Contextual inquiry	Understand real work context.
Shadowing	Observe undocumented procedures.
Semi-structured interview	Capture tacit knowledge.
Incident reconstruction	Analyze abnormal situations.
Think-aloud validation	Test whether agent output matches expert reasoning.
Scenario workshop	Evaluate abnormal-situation recommendations.
Override review	Understand why workers reject AI recommendations.
Cross-validation	Compare knowledge across multiple experts.

6.3 Knowledge capture template

Table 3. Field knowledge card template.

Field	Description
Situation	What situation occurs?
Signals	What signs does the worker observe?
Normal action	What is usually done?
Exception	When is a different action needed?
Risk	What can go wrong?
Data source	Which instrument, document, or experience is used?
Validation	Who validated the knowledge?
MCP mapping	Which tool, resource, or prompt uses this knowledge?
Retirement condition	When should this knowledge be revised or removed?

This template converts tacit knowledge into structured MCP-ready knowledge objects.

7. Evaluation Framework

MCP-based industrial agents should be evaluated at multiple levels.

7.1 Technical evaluation

Metric	Description
Tool-call success rate	Percentage of successful tool executions.
Schema validation rate	Whether inputs and outputs follow schema.
API error rate	Frequency of integration failures.
Latency	Time required for response.
Reproducibility	Whether repeated calls produce consistent results.
Data retrieval accuracy	Correctness of retrieved data.
Simulation convergence rate	Percentage of successful simulation runs.

7.2 Domain correctness

Metric	Description
Engineer agreement	Process engineer approval of outputs.
Operator agreement	Operator approval of recommendation usefulness.
Safety officer approval	Safety review of outputs.
SOP consistency	Alignment with approved procedures.
Regulatory consistency	Alignment with regulatory requirements.
Mass/energy balance consistency	Engineering consistency of calculations.
Unit conversion correctness	Correct conversion of units and dimensions.

7.3 Operational evaluation

Metric	Description
Time-to-diagnosis	Reduction in troubleshooting time.
Manual lookup time	Reduction in document/data search time.
Alarm response time	Faster interpretation of alarm sequences.
Maintenance planning time	Reduction in planning burden.
Quality deviation response time	Faster handling of quality issues.
Simulation workflow time	Reduction in manual simulation effort.
Operator workload	Perceived workload change.

7.4 Safety evaluation

Metric	Description
Unsafe recommendation rate	Frequency of unsafe outputs.
Out-of-bound action rate	Attempts to exceed operating boundaries.
Unauthorized execution rate	Unauthorized tool use.
Failure-to-escalate rate	Cases where escalation was required but not recommended.
Hallucinated procedure rate	Nonexistent or unsupported procedural recommendations.
Prompt-injection resistance	Resistance to malicious or misleading inputs.

7.5 Human factors evaluation

Metric	Description
Trust score	Worker trust in system output.
Perceived usefulness	Worker evaluation of practical value.
Explainability score	Whether workers understand outputs.
Override rate	Frequency of human rejection.
Training time	Time required for workers to use the system.
Adoption intention	Willingness to continue using the tool.

7.6 Field validation stages

Table 4. Field validation stages for MCP tools.

Stage	Description
Offline replay	Test on historical data.
Scenario simulation	Test on designed abnormal scenarios.
Expert review	Engineers and workers review outputs.
Shadow mode	Run during operations without influencing decisions.
Human-approved pilot	Allow recommendations with mandatory approval.
Controlled deployment	Deploy in limited unit, shift, or process area.
Continuous monitoring	Monitor tool performance and safety over time.

8. Security and Governance

8.1 Access control

Chemical engineering MCP systems require strict access control:

- role-based access
- plant-area access
- equipment-level permission
- read/write separation
- approval chains
- restricted high-risk tools
- separation between development and production environments

8.2 Audit logging

Every MCP tool call should be logged:

- user identity
- time
- tool name
- input
- output
- retrieved resources
- recommendation
- human approval
- final action
- error messages

Audit logs are essential for incident analysis, regulatory review, and continuous improvement.

8.3 Safety guardrails

Minimum safety guardrails include:

- no autonomous control by default
- action whitelist
- forbidden action list
- operating-limit checks
- dual approval for high-risk actions
- refusal when data are missing
- emergency stop and escalation rules

8.4 Cybersecurity risks

MCP systems create new attack surfaces because they connect language models to tools, databases, APIs, and potentially industrial systems. Security risks include:

- prompt injection
- malicious document retrieval
- tool poisoning

- unauthorized API calls
- credential leakage
- false sensor data
- unsafe tool invocation
- compromised MCP servers

Official MCP security guidance emphasizes that MCP implementations require security considerations beyond ordinary application development [11]. In chemical engineering, this is especially important because an unsafe tool call may affect process safety, equipment integrity, or regulatory compliance.

8.5 Tool description quality

MCP tool descriptions are not merely documentation. They affect how agents select and invoke tools. Tool descriptions include the tool name, natural-language description, and input schema used by clients and models for tool selection [12]. Therefore, chemical engineering MCP tools should be tested not only for API correctness, but also for description clarity, ambiguity, and misuse risk.

9. Proposed Reference Architecture

9.1 Architecture overview

A chemical engineering MCP-based agent architecture should include the layers shown in Figure 1: human users, an LLM agent/MCP client, a router and policy layer, domain MCP servers, and industrial systems. The key principle is that the LLM does not directly control plant systems; bounded MCP tools mediate access and execution.

9.2 Core design principles

1. The LLM should not directly control plant equipment.
2. MCP tools should expose typed, bounded, auditable functions.
3. Safety checks should occur before recommendations are returned.
4. Read-only tools should be deployed before execution tools.
5. Field workers should validate tool behavior.
6. Human approval should be mandatory for high-risk recommendations.
7. All tool calls should be logged and reviewable.

10. Discussion

10.1 MCP design is more important than agent framework design

In the chemical engineering industry, agent intelligence is bounded by the quality of its executable interfaces. A sophisticated agent cannot be useful if it cannot retrieve the correct process data, interpret SOPs, check safety limits, or run simulations. Conversely, even a relatively simple agent can be valuable if it is connected to reliable, field-validated MCP tools. The industrial question is therefore not: Which agent framework is best? The more important question is: Which plant-specific MCP tools are safe, useful, validated, and auditable?

10.2 Field knowledge as executable infrastructure

Chemical plants contain large amounts of tacit knowledge. Some of it is documented in SOPs, but much of it lives in operator experience, maintenance practice, shift handover habits, and informal troubleshooting routines. Industrial AI agents will fail if this knowledge remains outside the system. The purpose of Operator Knowledge Capture MCP is to convert tacit knowledge into structured, validated, and updateable knowledge cards that agents can retrieve and use.

10.3 Human-in-the-loop is not temporary

In many AI systems, human-in-the-loop is treated as a temporary phase before automation. In chemical engineering, human involvement should be treated as a permanent governance layer. Humans must remain responsible for safety-critical decisions, abnormal situation management, emergency response, operational trade-offs, regulatory compliance, and final approval of high-risk recommendations.

10.4 From digital documents to executable expertise

Many digital transformation projects digitize documents but do not make knowledge executable. MCP offers a different direction. The goal is not merely to store SOPs, process data, or maintenance records. The goal is to operationalize them as tools that agents can invoke, verify, cite, and log. The core transformation is from passive documents to executable expertise.

11. Limitations

- This paper is a conceptual design paper. It proposes MCP design strategies but does not implement all proposed MCP servers.
- Chemical engineering industries differ greatly. Oil refining, petrochemicals, pharmaceuticals, batteries, water treatment, food processing, and specialty chemicals have different data systems, regulations, safety requirements, and operating cultures.
- Field knowledge capture is difficult. Operator expertise may be inconsistent, outdated, context-dependent, or difficult to formalize.
- MCP tools may introduce cybersecurity and safety risks if poorly implemented. The proposed strategy therefore emphasizes read-only deployment, access control, audit logging, and human approval.
- Direct plant control is outside the scope of this paper. The proposed framework is intended primarily for decision support, analysis, simulation, retrieval, and recommendation.

12. Conclusion

AI agents in chemical engineering should not be designed only around agent frameworks, prompt harnesses, or multi-agent orchestration. In industrial environments, the practical bottleneck is the availability of field-validated MCP tools.

This paper proposed MCP design strategies for AI agents in the chemical engineering industry. It argued that MCP servers should encode plant data, SOPs, safety constraints, simulation tools, maintenance records, quality data, carbon-accounting logic, and operator know-how into executable and auditable interfaces. It also proposed eight MCP categories, a worker collaboration protocol, an evaluation framework, and a reference architecture.

Chemical engineering agents do not become useful because they can talk about processes. They become useful when they can safely act through validated tools that reflect real plant data, real procedures, real constraints, and real field expertise.

The next stage of industrial chemical engineering AI should therefore focus less on building another general-purpose agent framework and more on building field-validated MCP ecosystems with operators, engineers, safety officers, quality personnel, and maintenance technicians.

References

- [1] A. M. Bran et al., “Augmenting large language models with chemistry tools,” *Nature Machine Intelligence*, 2024.
- [2] D. A. Boiko, R. MacKnight, B. Kline, and G. Gomes, “Autonomous chemical research with large language models,” *Nature*, 2023.
- [3] J. Liang, N. Groll, and G. Sin, “Large Language Model Agent for User-friendly Chemical Process Simulations,” *arXiv*, 2026.
- [4] X. Tian et al., “From Text to Simulation: A Multi-Agent LLM Workflow for Automated Chemical Process Design,” *arXiv*, 2026.
- [5] S. Tan et al., “Reasoning-agent-driven process simulation, optimization, carbon accounting and decarbonization of distillation,” *Communications Engineering*, 2026.
- [6] Model Context Protocol, “Specification,” 2025.
- [7] Model Context Protocol, “Introduction,” 2025.
- [8] Model Context Protocol, “Tools,” *Specification*, 2025.
- [9] Model Context Protocol, “Resources,” *Specification*, 2025.
- [10] Model Context Protocol, “Prompts,” *Specification*, 2025.
- [11] Model Context Protocol, “Security Best Practices,” *Documentation*, 2026.
- [12] “Model Context Protocol Tool Descriptions Are a Runtime Interface,” *arXiv*, 2026.