



Cite this: DOI: 10.1039/d6dd00418k

A unified AI-agent-assisted platform for coupled experimental and simulation-based workflows in self-driving laboratories

Sebastian Putz, Ahmed Khalil Mama and Matthias Franzreb *

Self-driving laboratories (SDLs) promise to accelerate experimental research, yet translating scientific intent into executable automated workflows remains a bottleneck. We present a template-based platform that separates workflow design from execution. No-code interfaces combined with artificial intelligence (AI) agent-assisted code generation enable domain scientists to create reusable workflow templates, that is, parameterized specifications of a complete experiment or simulation, from device orchestration or model definition through analysis and reporting to data storage. Templates are generated once and thereafter executed deterministically, without large language model (LLM) involvement at run time. The platform integrates multi-device laboratory automation, mechanistic simulation and standardized data pipelines with electronic lab notebook (ELN) documentation. We demonstrate the approach on a chromatographic case study comprising automated isotherm determination for two proteins on a mixed-mode resin, mechanistic simulation of two elution strategies and experimental column validation, implemented entirely through wizard-created templates without manual scripting. Regenerating every artifact 20 times, one-shot and without manual correction, showed that reliability depends on the task and on the LLM. With the LLM deployed in the case study (GPT-5.1), workflow-structural scripts and device protocols without substantial logic were correct in 87 to 100% of generations, whereas the most complex protocol reached 50%. In contrast, the more capable GPT-5.5 reached 100% on the same protocol. Generated templates therefore require verification, supported by explicit review and, where necessary, version-controlled revision. By making workflow design accessible without removing determinism from execution, the approach offers a route to laboratory automation that is reusable, auditable and extensible beyond the domain demonstrated here.

Received 26th June 2026
Accepted 7th October 2026

DOI: 10.1039/d6dd00418k

rsc.li/digitaldiscovery

1. Introduction

SDLs promise to transform experimental research by integrating robotic automation with AI for decision-making, enabling autonomous planning, execution and analysis of experiments.^{1–4} In (bio-)chemical process development, SDLs have been successfully integrated in diverse applications such as microbial cultivations,^{5,6} protein- and genome engineering,^{7–9} organic synthesis,^{10–13} materials discovery^{14–18} and downstream processing.^{19–21} Despite rapid advances in laboratory automation hardware, a critical bottleneck persists in translating a scientist's experimental intent into executable, automated workflows. This still demands substantial programming expertise, creating a barrier that limits SDL adoption to groups with dedicated software engineering resources.^{4,22} In addition, the flexibility of existing SDLs is limited as adapting them to new processes requires substantial re-engineering, limiting the

return on investment and discouraging exploration of new experimental workflows.^{3,23}

Recent work has begun addressing this gap through the integration of LLM-based agents that can autonomously plan and execute laboratory experiments on SDLs. Systems such as Coscientist,²⁴ ChemCrow,²⁵ CRISPR-GPT²⁶ and ORGANA²⁷ use LLMs to interpret high-level natural-language instructions into executable low-level code for robotic application programming interfaces (APIs), and orchestrate multi-step experimental procedures in real time. These agent-driven approaches are pioneering demonstrations of autonomous scientific experimentation and reasoning. The mentioned systems operate through direct, on-the-fly control, *i.e.*, the agent reasons from scratch for each execution, generating code and decision chains for an experiment or a set of experiments. While this approach is powerful for exploratory tasks and requires minimal human effort and expertise, this mode of operation may be less suited for systematic, repeated workflows such as screening campaigns, standard operating procedures, and multi-parameter optimization studies where reproducibility, auditability, and iterative refinement of a defined protocol are

Karlsruhe Institute of Technology (KIT), Institute of Functional Interfaces (IFG), Department for Bioengineering and Biosystems, 76344 Eggenstein-Leopoldshafen, Germany. E-mail: matthias.franzreb@kit.edu

essential. For these use cases, the on-the-fly workflow generation through an LLM can be inefficient due to higher costs and latency as well as unwanted variability introduced into the generated workflows arising from the non-deterministic output of LLMs.^{28–31}

Reproducibility and reusability are, by contrast, longstanding concerns in laboratory automation, and machine-executable, parameterized protocol representations are not themselves new. Formal languages and repositories for encoding an experimental procedure unambiguously and re-executing it with different parameter values are well established in laboratory automation and the life sciences.^{32–36} What such approaches have historically required is that the workflow be implemented by experts able to program the instruments, the analysis and the data handling, which is precisely the barrier described above. In this study we retain the determinism and auditability of that established approach while removing its authoring burden. Rather than delegating both planning and execution to an autonomous agent, our platform employs AI agents during an interactive creation phase, in which they assist domain scientists in translating a high-level experimental specification into a workflow template, *i.e.*, a parameterized, executable specification of a complete experiment, from device orchestration through data analysis and reporting to structured storage. The individual components a template for our platform consists of, namely the device protocols, the orchestration script, the analysis script and the report script, are referred to as artifacts. Values that differ between individual runs, such as concentrations, volumes or gradient lengths, are left as placeholders within these artifacts, and supplying a set of values for them instantiates the template into the executable scripts of one specific run.

Templates are created through wizards, guided no-code interfaces in which the scientist defines experimental steps, selects devices and specifies parameters by drag-and-drop and natural-language prompts, while the AI agents generate the underlying code. The scientist therefore retains full control over workflow structure and methodology, while the AI handles the technical translation layer. Reporting to an ELN and structured storage in the application database are themselves artifacts of the template rather than separate manual steps, so that every execution produces documented and queryable results by construction. The resulting template is persistent and fully specified. It can be executed repeatedly with different parameter settings, shared across users, versioned and systematically varied through optimization campaigns, without requiring an LLM in the execution loop.

We demonstrate the system on the chromatographic purification of proteins, a domain central to downstream bioprocessing in which systematic evaluation of binding isotherms, dynamic binding capacities, and elution conditions across multiple stationary phases is essential for process development.³⁷ Using our platform, we show end-to-end template creation and execution of experimental workflows conducted on two multi-device SDLs seamlessly connected to simulative workflows using mechanistic chromatography models *via* the CADET-Process framework.³⁸ Critically, we

demonstrate how experimental and simulative templates can be linked, enabling automated experimental parameter determination from batch experiments, transfer to computational models to predict elution behavior and final validation in column experiments within an integrated campaign. We further quantify the reliability of agent-assisted template generation by repeating the generation of every agent-generated artifact 20 times, showing that reliability depends strongly on the logical complexity of the artifact and on the capability of the underlying LLM.

The demonstration presented in this work addresses the automated workflow layer, on which autonomous operation depends. While not demonstrated here, the platform also implements closed-loop operation for autonomous campaigns, and the validated templates reported in this work are directly reusable as the building blocks of such campaigns.

2. Results

First, the platform itself is described, namely its architecture (Section 2.1), the structure of workflow templates and their execution (Section 2.2), and the wizard interfaces through which templates are created (Section 2.3). Section 2.1 first outlines the architecture in general terms, independent of the software and hardware used, and then describes its implementation in the authors' laboratories. Implementation details that are not required to follow the paper are given in the SI. Section 2.4 then presents the chromatographic case study, comprising the domain-specific configuration and the three linked workflows for isotherm determination, elution behavior simulation and experimental validation. Section 2.5 quantifies the reliability of agent-assisted template generation.

2.1 Platform architecture

2.1.1 General architecture. The platform comprises a central software application and the laboratory infrastructure it controls. The software application is organized in four layers, namely a user interface layer, an orchestration and logic layer, a data and documentation layer, and a laboratory devices and simulation layer (Fig. 1). The user interface layer provides the wizards for template creation and the interfaces for planning and monitoring experiments and simulations. The orchestration and logic layer manages templates and their execution and hosts the AI agents, which are invoked during template creation. The data and documentation layer holds documentation, structured results and all generated files, and the laboratory devices and simulation layer connects the application to the laboratory instruments and to the simulation engine.

Two characteristics of this architecture are relevant for the following sections. First, instruments are addressed exclusively through a uniform device abstraction, which exposes the operations of each instrument independently of how it is controlled. Orchestration scripts, and the agents that generate them, therefore work only with the operations an instrument offers and never with its native control interface. Second, documentation and data storage are integral to every template

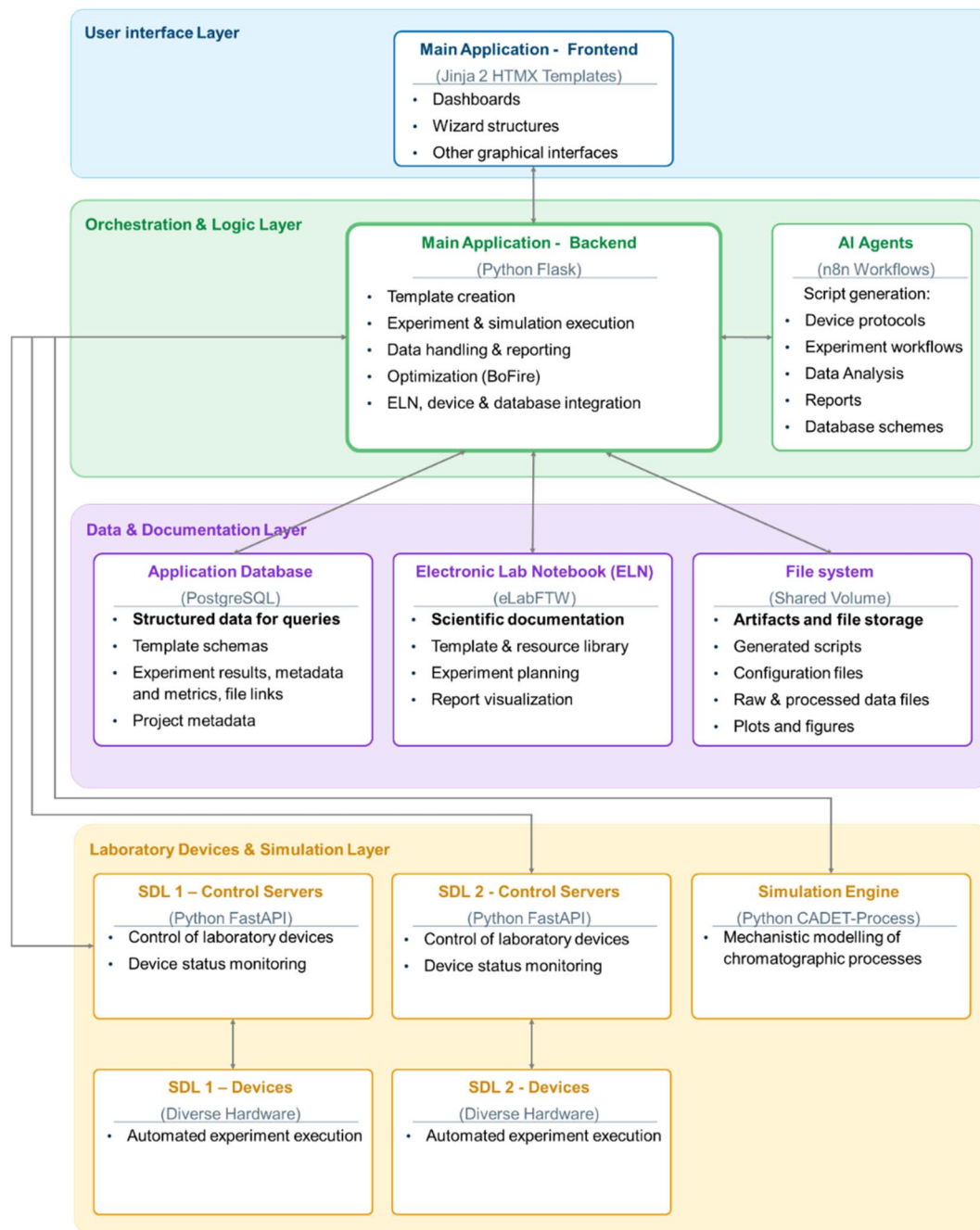


Fig. 1 Platform software architecture. Four-layer architecture separating user interaction (web frontend with wizard interfaces), orchestration logic (Flask backend with AI agent delegation via n8n), data persistence (PostgreSQL database, eLabFTW ELN, shared file system), and hardware/simulation control (FastAPI device control servers on each SDL, CADET-Process simulation engine). All services are deployed as Docker containers on a centralized Linux server.

rather than separate manual steps, so that each run produces a human-readable report in an ELN and a structured, queryable record in a database.

2.1.2 Implementation in this work. In its current deployment, the platform controls two SDLs. SDL 1 integrates, among other instruments, a liquid handling station, a collaborative robot arm for labware transfer, a vacuum filtration unit and a microplate reader, and is used in this work for batch isotherm determination. SDL 2 integrates over 20 devices, including

a laboratory-scale chromatography system (ÄKTA pure 25, Cytiva), and is used in this work for column validation experiments. The instruments and their primary control interfaces are listed in Tables S1 and S2, and the laboratories are described in Section S1. Both laboratories were previously used for autonomous, closed-loop optimization campaigns on solid-phase extraction³⁹ and enzymatic reaction workflows⁴⁰ with hand-written, purpose-built templates. The platform presented here replaces the manual implementation of workflows with

a wizard-driven template layer, so that the same instruments are addressed through workflows created without programming. Beyond the physical laboratories, the application also drives mechanistic process simulations, so that experimental and simulative workflows are created, executed and documented through the same interfaces.

The software is deployed as Docker containers on a central Linux server (Fig. 1). The user interface layer is a web frontend built with Jinja2 (ref. 41) and HTMX.⁴² The orchestration and logic layer is a backend implemented in Python with Flask,⁴³ which delegates code generation during template creation to AI agents implemented as n8n workflows⁴⁴ (Section S2). In the data and documentation layer, reports, parameter masks and reusable resources are held in the ELN eLabFTW,⁴⁵ structured results in a PostgreSQL database, and scripts and raw data in a shared file system. In the laboratory devices and simulation layer, simulations run in a dedicated container using CADET-Process³⁸ and the CADET-Core solver,⁴⁶ and instruments are accessed through device control servers as described below. Each layer is described in detail in Section S1.

The device abstraction is realized as one Python wrapper class per instrument, whose methods issue authenticated Hypertext Transfer Protocol (HTTP) requests to device control servers implemented with FastAPI.⁴⁷ These device control servers run on the computers connected to the instruments of each laboratory and translate every request into a call to the native control interface of the instrument, be it a vendor library, an API, an automation interface or a network protocol. Orchestration scripts invoke device operations sequentially, report every step to the backend for a complete execution trace and, if an operation fails, record the failing step and the partial results before terminating. Workflows may also contain defined manual interventions. In that case, the orchestration script registers a pause with the application backend and waits until the user confirms completion in the web interface, allowing steps that cannot be automated to be embedded in an otherwise automated sequence without leaving the platform.

2.2 Workflow templates

All processes on the platform, whether experimental or simulative, are organized around reusable workflow templates that separate workflow design from execution (Fig. 2) and encapsulate the complete logic required to conduct, analyze, document and store an experiment or simulation.

A workflow template consists of several interdependent components that are generated together during an interactive creation phase in dedicated wizard interfaces (Section 2.3). For experimental workflows, these include device-specific protocols for involved instruments, a main orchestration script that coordinates the sequential execution of device operations, a data analysis script, a report script that compiles results into a structured document and uploads them to the ELN and a database commit script that writes processed results and corresponding metadata to the application database with a template-specific schema. For simulative workflows, the template contains a CADET-Process simulation script defining

the system topology, process sequence, and parameterized model equations in place of the experimental orchestration script but otherwise follows the same structure. All scripts contain placeholders for parameters that vary between individual runs, while the workflow structure, device orchestration logic, and data pipeline remain fixed.

To conduct an experiment or simulation, the user instantiates the template by providing values for the parameterized fields through the ELN mask. These values replace the placeholders in the script templates and execution then proceeds deterministically. For experimental workflows, the orchestration script coordinates the involved devices as described in Section 2.1 and for simulative workflows, the CADET-Process script is executed directly. Upon completion, the integrated data pipeline processes the results and produces the database entry and the ELN report described above, so that every run yields a complete, standardized record without manual intervention. Videos S1–S3 show the planning and execution of experiments and simulations in the application. It has to be noted that Video S2 shows an experimental workflow on SDL 1 for a magnetic particle assay, not the batch isotherm determination reported in this work.

For optimization campaigns, a Bayesian optimization engine based on BoFire⁴⁸ closes this cycle by suggesting the next parameter set for template instantiation after each run. This closed-loop mode is implemented but was not used in the present work, which focuses on no-code AI-agent supported workflow template generation and its reproducibility. All workflows reported here were executed as automated processes with parameters defined *a priori* by the user.

2.3 Template creation

Template creation is performed through dedicated wizard interfaces within the web application. Both the experimental and the simulative wizard produce a complete, reusable workflow template with all associated scripts, and in both the scientist designs the workflow while the platform handles the technical translation into executable code. They differ in how that code is produced. The simulation wizard assembles it deterministically from pre-validated modules, whereas the experimental wizard delegates the generation of device protocols and pipeline scripts to specialized AI agents. Only the latter therefore carries the non-determinism inherent in LLM output, which is quantified in Section 2.5.

2.3.1 Simulation templates. The simulation template wizard follows a deterministic, form-based approach that does not rely on LLM-generated code as a CADET-Process configuration has a well-defined modular structure (Fig. 3a). The system topology is graphically assembled as a flow sheet from a palette of available unit operations as well as the connections between them (Fig. 3b). This allows topologies ranging from single columns to multi-column configurations with recycle loops. The user then defines the process sequence, for example equilibration, loading, washing, elution and regeneration, with parameters set either as a fixed value or as a placeholder for later instantiation (Fig. 3c), and selects the analyses and plots to

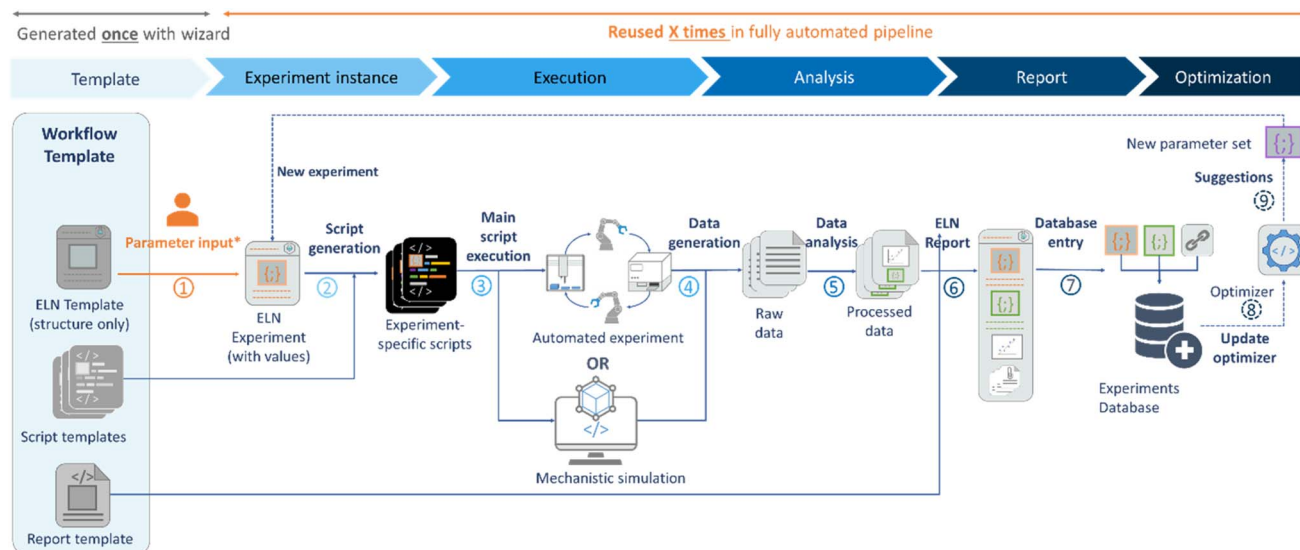


Fig. 2 Template-based SDL workflow sequence. A workflow template (left) is created once through an interactive wizard and comprises an ELN template defining the parameter structure, script templates for device orchestration or simulation, data analysis, reporting and database storage. To conduct an experiment or simulation, the user instantiates the template by providing parameter values (1), creating an ELN experiment entry. The platform then generates experiment-specific scripts by substituting placeholders with the provided values (2) and executes the main orchestration script (3), which coordinates either an automated laboratory experiment or a mechanistic simulation. Upon completion, raw data is collected (4), processed by the analysis script (5), and compiled into a report that is uploaded to the ELN (6). Processed results are committed to the application database (7). For autonomous optimization campaigns, the optimizer is updated with the observed objective values (8) and suggests the next parameter set (9), which is used to instantiate a new run from the same template, closing the optimization loop.

be generated. From this configuration, the wizard assembles the parameterized simulation script from pre-validated code modules for each unit operation and process step and generates the ELN template automatically, so that the user does not need to know in advance which variables a complete simulation requires. Flow sheets and binding model parameters, such as the isotherm coefficients of a compound–resin pair, can be stored as reusable ELN resources, allowing experimentally determined parameters to be reused across simulation campaigns. A detailed walkthrough is provided in Video S4 and Section S3.

2.3.2 Experimental templates. The experimental template wizard combines structured graphical input with AI-assisted code generation (Fig. 4a). The user selects the target SDL, specifies whether the process is automated or autonomous, creates the ELN template defining experimental parameters, and assembles the workflow by dragging device operations from an interactive palette into a sequential execution order (Fig. 4b). Additionally, the wizard provides higher-level compound steps that encapsulate frequently used multi-device sequences into single reusable blocks. This graphical workflow definition captures the full device orchestration logic, including multi-device coordination such as robotic arm transfers between instruments, without requiring the user to write any code.

Subsequently, the workflow is parsed for steps requiring individual subprotocols, which include scripts for the liquid handling devices and analysis devices such as a plate-reader or mass spectrometer. For subprotocols with complex degrees of freedom, such as liquid handling protocols, the wizard delegates code generation to specialized AI agents. The user

provides natural-language descriptions of the protocol and specifies which parameters should remain variable across runs (Fig. 4c). Specialized AI agents with access to the device API documentation and to tools to write, simulate and verify their code then generate the device protocols, the main orchestration script, and the data analysis, report and database commit scripts (Section S2). The user can review and, if necessary, manually edit any generated script before finalizing the template. A detailed walkthrough is provided in Video S5. Both simulative and experimental templates are maintained in a template library within the web application, where users can version, publish, share, copy and edit templates.

Templates can be revised after creation through an agent-assisted interface. Each template is version-controlled *via* Git.⁴⁹ The user describes the required change, the agent proposes it as a set of edits against the current version, and no change reaches a template without the user's explicit confirmation. Revision applies to all artifacts of a template, including device protocols, the orchestration script and the data pipeline scripts. Since a template is executed unchanged across arbitrarily many runs, revision is a deliberate, recorded event rather than a modification during execution, and the version used for any given run remains identifiable. A walkthrough of the revision interface is provided in Section S3.

2.4 Chromatographic purification process development with templated workflows

To demonstrate the integrated experimental–simulative capabilities of the platform, we performed an end-to-end

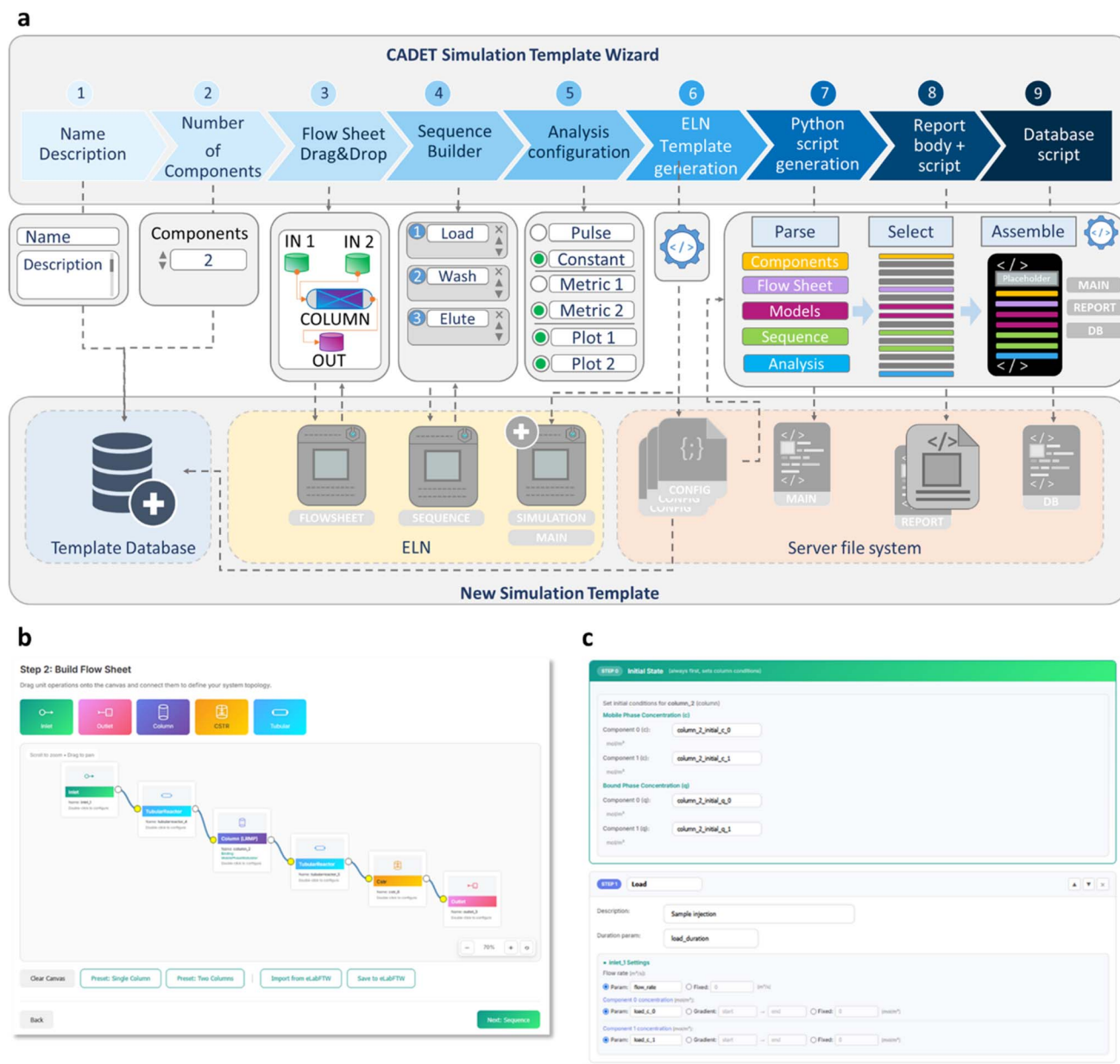


Fig. 3 No-code implementation of simulation workflow templates for CADET-Process. (a) Schematic overview of the simulation template wizard. The nine-step process guides the user from naming and component definition (1 and 2) through graphical flow sheet assembly (3) and process sequence definition (4) to analysis configuration (5). In steps (6–9), the platform deterministically generates all template artifacts: an ELN template is created (6), the wizard parses the user's configuration into its constituent elements (components, flow sheet, models, sequence, analysis) and selects the corresponding pre-validated code modules to assemble the complete simulation script (7), and report and database scripts are generated (8 and 9). The completed template is stored across three persistence layers: the application database, the ELN (containing flow sheet, sequence, and simulation resources), and the server file system (containing configuration, main, report, and database scripts). (b) Flow sheet builder interface showing the drag-and-drop assembly of a chromatographic system topology with unit operations (Inlet, TubularReactor, Column, CSTR, Outlet) connected on an interactive canvas. Preset configurations and ELN import/export functions (bottom) support reuse of previously validated system designs. (c) Process sequence definition interface showing the specification of initial column conditions (top, green) with per-component mobile and bound phase concentrations using parameterized placeholder fields, and a load step (bottom) with parameterized duration, flow rate, and component-specific concentration gradients. Fields marked with placeholder names (e.g., column_2_initial_c_0, load_duration) remain variable across template instantiations.

chromatographic case study comprising automated isotherm determination in batch format, mechanistic simulation of two elution strategies, and experimental validation on a column. The study targeted the separation of ovalbumin and transferrin

on a mixed-mode resin (Toyopearl MX-Trp-650M, Tosoh) with salt-modulated elution.

2.4.1 Implementation for the chromatographic case study. Three workflow templates were created for the study, each

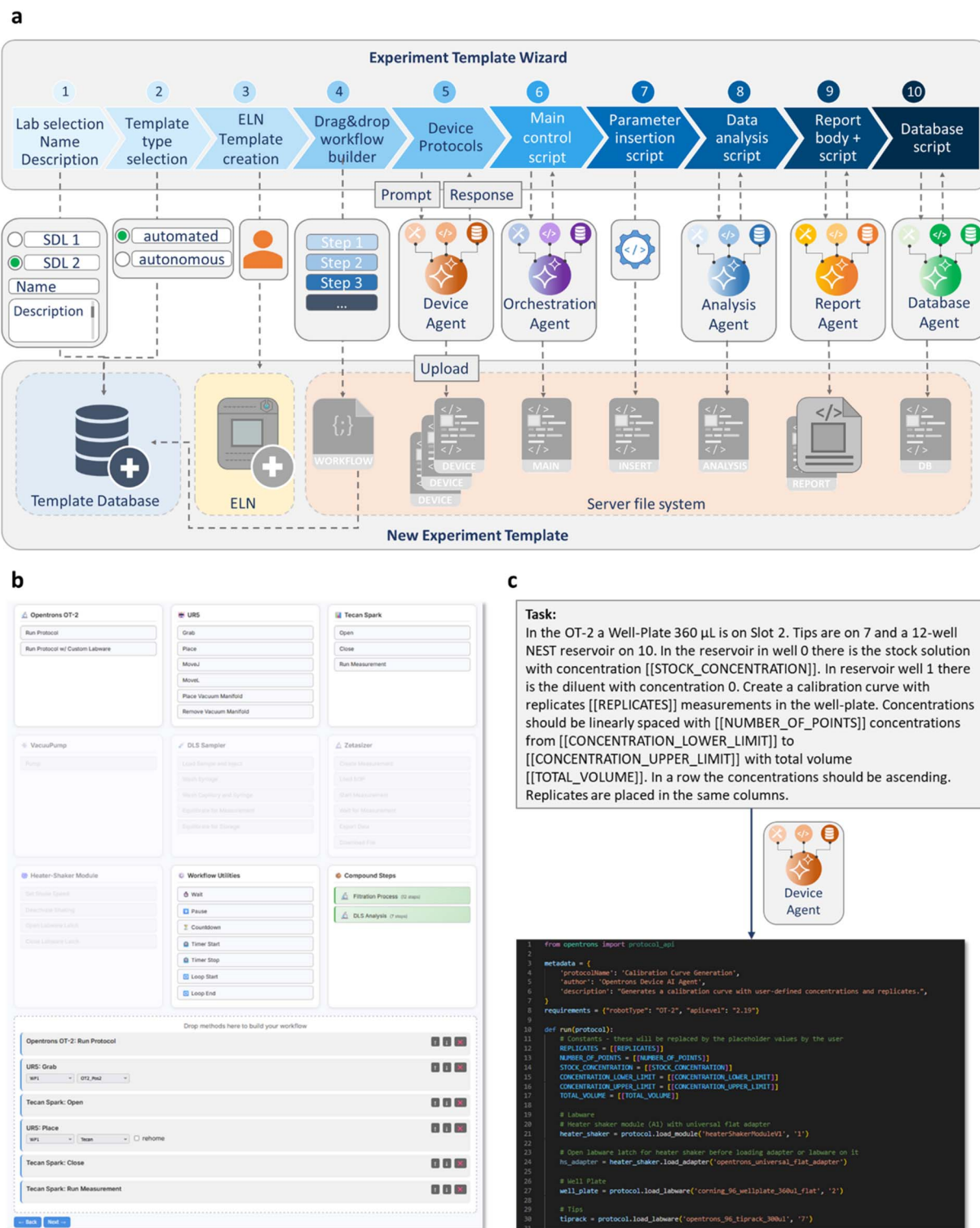


Fig. 4 No code implementation of experimental workflows using AI-agents. (a) Schematic overview of the experimental template wizard. The user selects the target laboratory and specifies the process type (automated or autonomous) (steps 1 and 2), and an ELN template is created (step 3). In (step 4), the user assembles the experimental workflow by dragging device operations into the desired execution sequence. In (steps 5–10), specialized AI agents sequentially generate the template scripts: the device agent produces instrument-specific protocols from parameterized prompts containing the user's natural-language protocol descriptions (step 5), followed by the orchestration agent generating the main control script (step 6), and the analysis, report, and database agents producing the corresponding data processing, documentation, and storage scripts (steps 7–10). A parameter insertion script handles placeholder-to-value substitution at instantiation time (step 7). All generated artifacts are persisted across the template database, the ELN, and the server file system (workflow definition, device protocols, main orchestration, parameter insertion, analysis, report, and database scripts). (b) Drag-and-drop workflow builder showing the device palette (top) with available instruments and their callable methods and the assembled multi-device workflow (bottom) defining a sequential experimental procedure including pipetting,

robotic plate transfer, and absorbance measurement. (c) AI-assisted device protocol generation. The user prompt (top) provides the device agent with the experimental context, labware configuration, and parameterized instructions using placeholder syntax (e.g., [[STOCK_CONCENTRATION]], [[REPLICATES]]). The device agent produces an executable Opentrons OT-2 protocol (bottom) in which the placeholders are assigned as variables, ready for substitution with run-specific values during template instantiation.

through the wizard interfaces described in Section 2.3 and without manual scripting. The batch isotherm template was created with the experimental wizard for SDL 1 and coordinates a liquid handling robot, a six-axis robotic arm for plate transfers, a heater-shaker module, a vacuum manifold with a pump for filtration and a plate reader for absorbance measurements, in a 71-step workflow comprising buffer mixing, protein dilution, resin equilibration, protein incubation, multiple filtration steps and absorbance readout (see Experimental section and Fig. S5). The simulation template was created with the simulation wizard and defines the ÄKTA pure 25 system topology, the process sequence and the parameterized model equations. The fast protein liquid chromatography (FPLC) templates for validation were created with the experimental wizard for SDL 2 and control method execution on the chromatography system together with the subsequent data pipeline. A detailed documentation of the implementation process including utilized prompts is given in Section S3.

Of the three, the batch isotherm template posed the greatest demands on code generation. One of its liquid handling protocols required programmatic volume calculations, dynamic volume tracking across sequential pipetting operations, and conditional selection of source and target wells, logic that would typically require considerable manual programming effort. All templates nevertheless executed successfully on the first attempt without refinement, and in each case the pipeline from execution through data analysis, parameter fitting and ELN report generation proceeded without further intervention. No use was made of the revision interface described in Section 2.3, so that the templates reported here contain the original scripts as generated in a single pass through the wizard. Whether this is characteristic of agent-assisted template generation is examined in Section 2.5.

2.4.2 Automated isotherm determination. Seven initial protein concentrations (0.2–7.0 mg mL^{−1}) were tested in triplicate at five NaCl concentrations (0, 100, 200, 300 and 500 mM) for each protein with a fixed amount of resin (7.7 µL) at a fixed pH of 4.0 (Fig. 5a, b and Table 1).

Both proteins exhibited Langmuir-type adsorption behavior across all conditions where binding was resolvable, but displayed qualitatively different salt responses, reflecting their different charge states at the working pH. Ovalbumin (pI ≈ 4.5–5.0^{50,51}), carrying only a weak net positive charge at pH 4.0 showed predominantly electrostatic binding that was readily disrupted by salt. The effective Langmuir constant K_L^{eff} dropped by 68% between 0 and 100 mM NaCl and no detectable binding remained at 500 mM. Transferrin (pI ≈ 5.2–6.1 depending on iron saturation state^{52,53}) with a higher net charge, showed an overall higher binding affinity and a more gradual decline with ionic strength, retaining $K_L^{\text{eff}} = 6.88 \text{ mL mg}^{-1}$ even at 500 mM NaCl, implying hydrophobic contribution to its interaction with the resin. Its maximum binding capacity did not decrease

monotonically but passed through a maximum at 200 mM NaCl, 1.6 times the value without added salt, indicating a salt-promoted contribution to binding characteristic of a mixed-mode interaction.

To model salt-dependent binding behavior for mechanistic chromatography simulation, the effective Langmuir constants across all salt concentrations were fitted to the mobile phase modulator (MPM) model^{54,55} (eqn (3), see Experimental section, ‘Batch incubation chromatography for isotherm parameter determination’), yielding the hydrophobicity parameter γ and the ion-exchange characteristic β for each protein (Table 2). The R^2 values of 0.826 and 0.908 are bounded by the precision with which the effective Langmuir constants themselves are determined rather than by a deficiency of the fit, since these constants enter the regression as estimates but are treated as exact observations.

2.4.3 Prediction of elution behavior. Using the MPM parameters, mechanistic simulations were performed for two elution strategies: a linear gradient from 0 to 2000 mM NaCl over 20 CV, and a step elution at 650 mM for 3.5 CV and 1600 mM NaCl for 6 CV, each followed by a 2000 mM NaCl clean-in-place (CIP) step (Fig. 5c and d). Both simulation templates were created using the simulation wizard and modelled the ÄKTA pure 25 system (see Experimental section and Section S4 for details).

Under both strategies the simulations predicted that ovalbumin elutes as a single peak while transferrin does not elute at all. Ovalbumin was predicted at 35.5 min in the gradient, corresponding to 1312 mM NaCl at the peak maximum, and at 18.1 min in the two-step elution, on the 1600 mM step.

2.4.4 Experimental validation. Experimental validation was performed on SDL 2 in triplicate for both strategies (Fig. 5e and f). Two peaks were observed within the elution window of each method. Under the gradient, they appeared at 21.28 ± 0.04 min and 29.77 ± 0.04 min, corresponding to 660 ± 4 mM and 1182 ± 3 mM NaCl. Under the two-step elution, the peaks appeared at 12.20 ± 0.00 min and 17.33 ± 0.00 min, corresponding to 632 ± 0 mM and 1237 ± 3 mM NaCl. The second peak of the two-step method thus appeared on the rising edge of the second step rather than on its 1600 mM plateau, so that the retention times of the simulated and measured peaks nearly coincide although the corresponding salt concentrations differ. In both methods the second peak dominates, and another peak appears during the subsequent alkaline CIP step (Fig. S8).

Since detection at 280 nm is not protein-specific, the peaks cannot be assigned experimentally. The observations are, however, consistent with the simulated behavior if the second and larger peak of each method is ovalbumin and the smaller earlier peak an impurity of the protein preparations, with transferrin released only under the alkaline conditions of the NaOH CIP step. On this assignment, the predicted retention of ovalbumin agrees closely under the two-step elution, 18.1 min

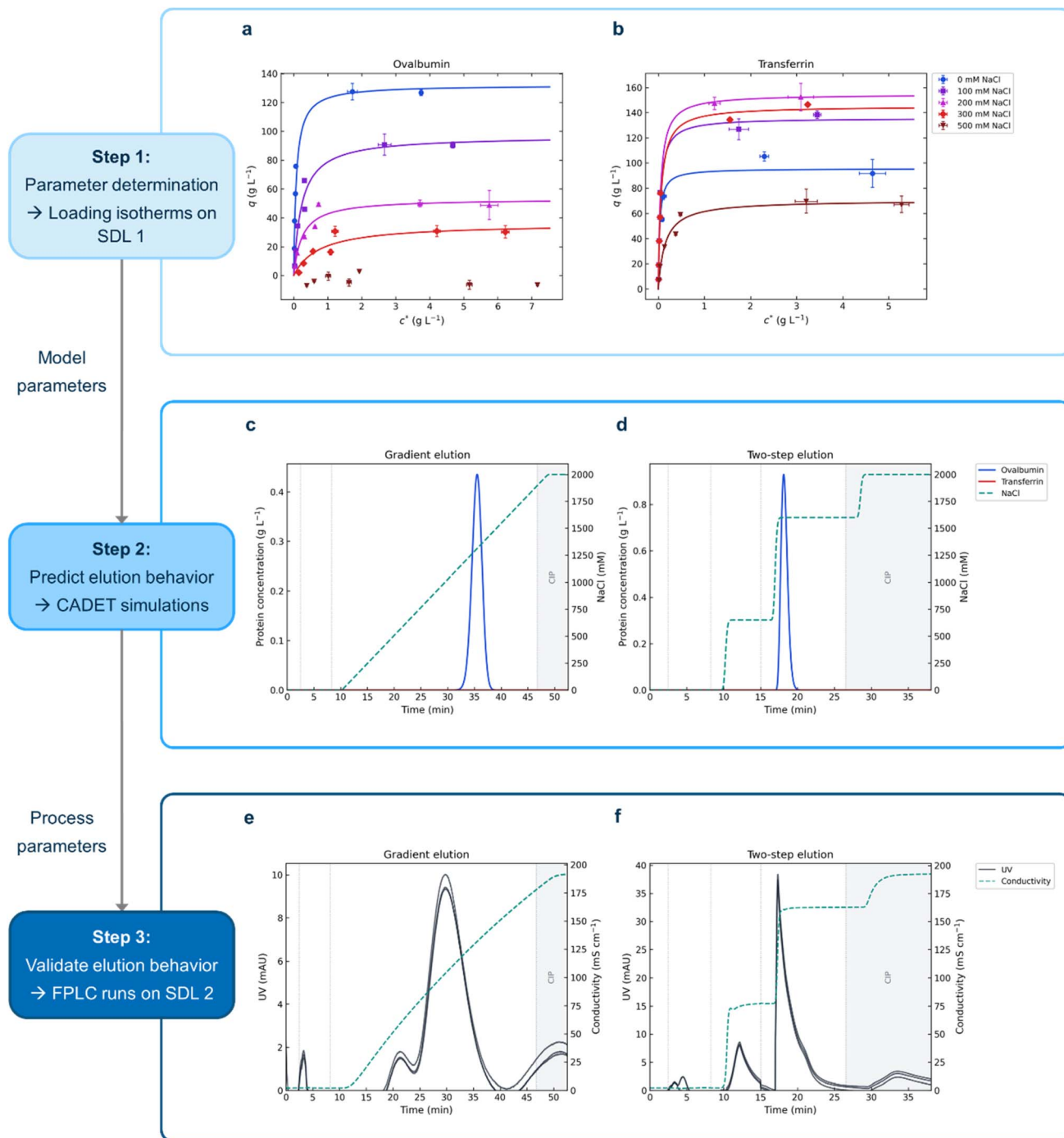


Fig. 5 End-to-end chromatographic case study with templated workflows. Automated determination of loading isotherms for ovalbumin and transferrin on a mixed-mode resin, mechanistic simulation of two elution strategies and experimental column validation. The case study proceeds in three stages, each implemented as a templated workflow *via* the wizard interfaces of the application. (a and b), step 1: parameter determination. Salt-dependent loading isotherms for ovalbumin (a) and transferrin (b) determined by automated batch incubation on SDL 1 at five NaCl concentrations (0, 100, 200, 300, 500 mM). Solid lines represent Langmuir model fits (eqn (2) and Table 1). The fitted effective Langmuir constants were used to parameterize the MPM model (eqn (3) and Table 2). All data represent the mean and standard deviation of replicated experiments ($n = 3$). (c and d), step 2: prediction of the elution behaviour. Simulated chromatograms from CADET-Process using the experimentally determined MPM parameters for a linear gradient elution over 20 column volumes (CV) (c) and a two-step elution at 650 and 1600 mM NaCl (d), each followed by a clean-in-place (CIP) step at 2000 mM NaCl. Protein concentrations are given on the left axis and the NaCl concentration on the right. (e and f), step 3: experimental validation. Corresponding FPLC experiments performed on SDL 2 for the linear gradient (e) and the two-step elution (f), shown as separate solid lines for each replicate ($n = 3$). UV absorbance at 280 nm is given on the left axis and conductivity, the experimental analogue of the NaCl concentration, on the right. In all panels, dotted vertical lines mark the process phase boundaries and the shaded region marks the CIP step. The time axis of (e) and (f) is offset by the pump washes and equilibration preceding the load, so that all panels start at the beginning of the load; for (f) the two brief pump-wash transitions between elution phases are excised, so that the measured and simulated timelines coincide. The subsequent alkaline CIP is outside the domain of the MPM model, which carries no pH dependence, and is shown in full in Fig. S8.

predicted against 17.33 min observed, and falls late under the gradient, 35.5 min against 29.77 min.

2.4.5 Deviations between simulation and experiment. The simulations therefore reproduced the qualitative outcome of both elution strategies, correctly predicting that one protein elutes within the applied salt range while the other does not, and placing the eluting peak close to its observed position under the two-step strategy.

The quantitative deviations reflect for the most part limitations of the MPM model as formulated. In the MPM model the entire salt dependence is carried by the effective Langmuir constant, while the maximum binding capacity enters as a single, salt-independent value, here as $q_{\max,0}$ obtained from the batch isotherm without added NaCl. The measured capacities vary considerably across the calibration range, decreasing by a factor of 3.7 for ovalbumin between 0 and 300 mM NaCl and passing through a maximum for transferrin at 200 mM. Retaining the zero-salt capacity therefore overestimates the capacity available to ovalbumin at every salt concentration tested, and hence its retention at the concentrations at which it elutes, which is the direction of the deviation observed under the gradient. For transferrin the effect is not unidirectional, since its capacity exceeds the zero-salt value across 100 to 300 mM NaCl before falling below it at 500 mM.

The functional form is moreover monotonic under the fitted parameters. The ion-exchange characteristic β is non-negative, as required by the model implementation, and the fitted hydrophobicity parameter γ is negative for both proteins, so that both terms of eqn (3) decrease with increasing modulator concentration and a binding maximum at intermediate salt cannot be represented. The salt-promoted contribution evident in the transferrin capacities resides in $q_{\max}$ rather than in K_L^{eff} and is therefore not accessible to a regression performed on the effective Langmuir constants alone.

Two further consequences of the model structure should be noted. Since the desorption term scales with the modulator concentration raised to β , it vanishes in the phases containing no added NaCl, so that binding during load and wash is irreversible by construction and quantitative capture is predicted rather than computed. The single-component isotherm parameters likewise do not account for competitive binding

during co-adsorption. Finally, the model carries a single modulator and no pH dependence, so the release of the remaining protein during the alkaline CIP step lies outside its domain entirely.

2.5 Reliability of agent-assisted template generation

The workflows of the preceding case study were generated once and used productively without refinement. Since a template is created once and thereafter executed unchanged, non-determinism is confined to the creation phase, but it raises the question of how reliably the agents produce a correct workflow. We therefore repeated the generation of every agent-generated artifact of the two experimental templates 20 times independently, using identical prompts and pinned LLM versions, and scored each replicate automatically against a rubric fixed before the runs.

The reference for each artifact is the version created during the original template creation and used to produce the results reported in Section 2.4, that is, the script that actually ran on the instruments or processed the experimental data. A replicate was counted as correct only if it would have the same effect on the experiment as that reference. The criteria were for device protocols, delivering the same volumes to the same wells and issuing the same module commands with the same speeds, temperatures and durations; for orchestration scripts, driving every device of the workflow in the same sequence; and for the data pipeline scripts, reproducing the published analysis outputs, report structure and database records. Choices that do not affect the outcome, such as pipetting order, tip handling, report section headings and database column names, were recorded but not counted as failures, and executability alone was never sufficient. The full scoring criteria and additional details on this reproducibility study are given in Section S6. The assessment concerns the initial generation only. The agent-assisted revision described in Section 2.3 was not used, since the question is how reliably the agents produce a correct artifact without user involvement.

All artifacts were generated with GPT-5.1, the LLM deployed in the platform's agents at the time the case-study templates were created. Reliability was high for the workflow-structural artifacts and task-dependent for the device protocols for the Opentrons OT-2 of the batch isotherm template (Fig. 6a). For both the batch isotherm and the FPLC gradient template, the orchestration script was generated correctly in 20 of 20 replicates, while the analysis, report and database scripts ranged from 16 to 19 of 20. Among the five device protocols of the

Table 1 Langmuir isotherm parameters. For Ovalbumin at 500 mM NaCl no binding was detectable and thus K_L^{eff} and $q_{\max}$ were set to 0 and R^2 was not available

Protein	c_{NaCl} (mM)	K_L^{eff} (mL mg ⁻¹)	$q_{\max}$ (mg mL ⁻¹)	R^2 (—)
Ovalbumin	0	13.44	132.18	0.891
	100	4.34	96.77	0.966
	200	4.06	53.34	0.933
	300	1.33	36.08	0.855
	500	0	0	—
Transferrin	0	33.4	95.59	0.936
	100	23.05	135.73	0.983
	200	20.22	154.62	0.985
	300	17.17	145.16	0.985
	500	6.88	70.51	0.971

Table 2 MPM model parameters. γ and β were regressed from the effective Langmuir constants of Table 1 at 100, 200, 300 and 500 mM NaCl, with K_L^0 fixed to the value measured without added NaCl. For ovalbumin the non-binding condition at 500 mM NaCl was included as $K_L^{\text{eff}} = 0$

Protein	K_L^0 (mL mg ⁻¹)	β (—)	Γ (mM ⁻¹)	R^2 (—)
Ovalbumin	13.44	0.1253	−0.0044	0.826
Transferrin	33.4	0.0145	−0.0024	0.908

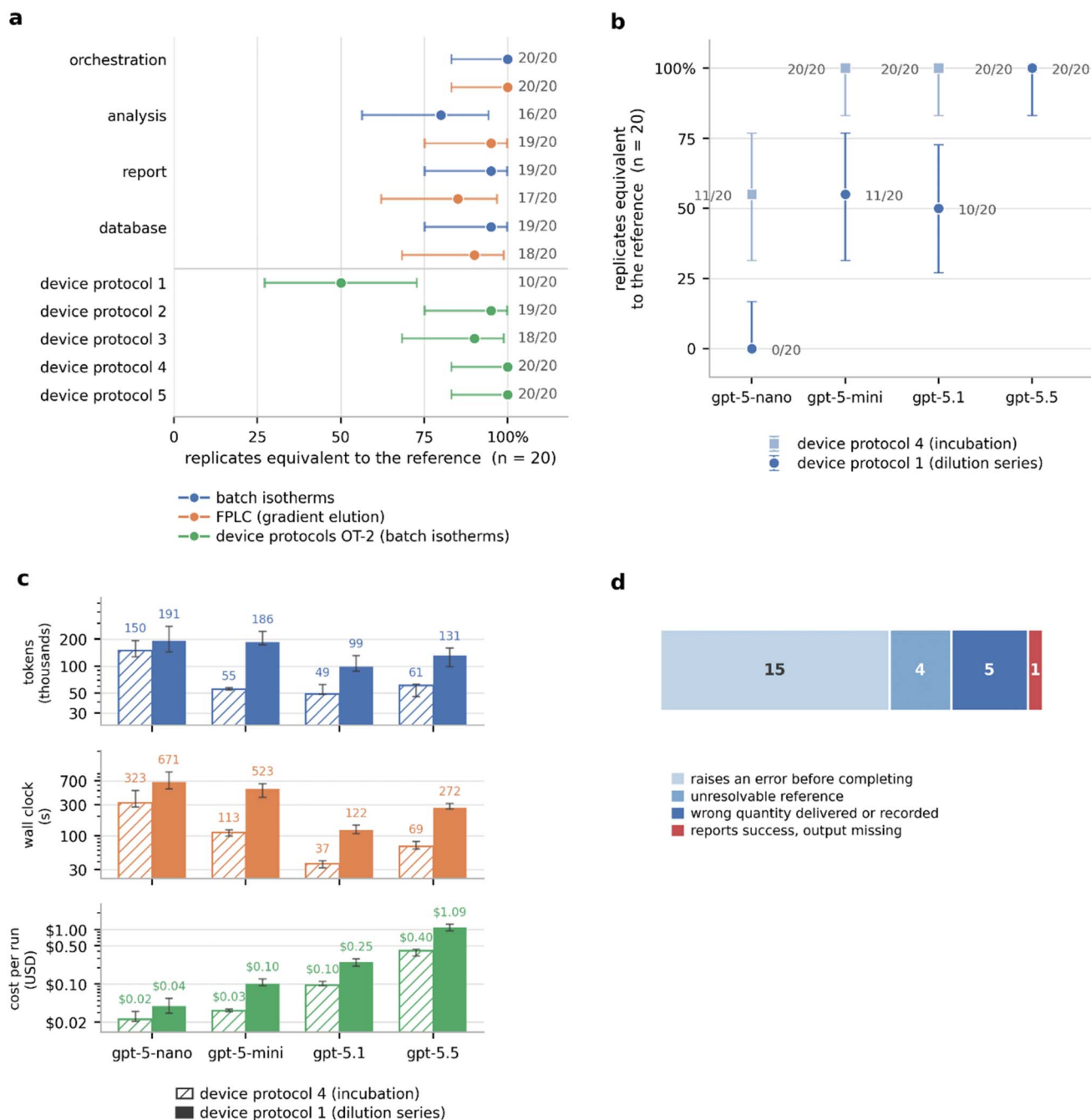


Fig. 6 Reliability and cost of agent-assisted template generation. All panels report 20 independent generations per artifact, each an independent invocation of the corresponding agent with an identical prompt and a pinned model version. (a) Fraction of replicates judged semantically equivalent to the reference artifact, that is, to the version created during the original template creation and used to produce the results in Section 2.4. Points are the observed fraction and whiskers the Clopper–Pearson 95% confidence interval. All artifacts were generated with GPT-5.1. The orchestration, analysis, report and database scripts are shown for both the batch isotherm and the FPLC gradient template; device protocols 1 to 5 belong to the batch isotherm workflow only. The underlying prompts correspond to the deployed specification (Section S6). (b) The most complex device protocol, which prepares the dilution series (protocol 1), and one of the simplest, which incubates a plate (protocol 4), each generated 20 times with four models under otherwise identical conditions. Points are the observed fraction and whiskers the Clopper–Pearson 95% confidence interval. (c) Cost of one complete generation run for the same two protocols and four models, where a run comprises the agent's full loop including its internal validation and correction cycles. Bars are medians over the 20 replicates and whiskers the interquartile range. Monetary cost is computed per replicate from its own input and output token counts at the list prices given in Table S16, without prompt caching, and includes every replicate irrespective of outcome. (d) Classification of all 25 failures observed with GPT-5.1 across both templates, by the effect the artifact would have on an experiment. Failures of the smaller models in the model comparison of panels (b) and (c) are reported separately in Section S6.

isotherm workflow, four reached 18 to 20 of 20, while the protocol preparing the dilution series reached only 10 of 20 (95% confidence interval 27 to 73%). This protocol carries the volume arithmetic of the entire experiment, computing stock and buffer volumes for every combination of salt and protein concentration, and must additionally track the liquid remaining in each reservoir well and select source wells accordingly as they are depleted. As shown in the following section, this rate is specific to the LLM used and a more capable LLM generates the same protocol correctly in every replicate.

To investigate how model capability influences the template generation, the most complex protocol, which prepares the dilution series (Opentrons OT-2 protocol 1) and the simplest protocol, which only incubates a plate (Opentrons OT-2 protocol 4) from the batch isotherm workflow were generated 20 times each with GPT-5-nano, GPT-5-mini, GPT-5.1 and GPT-5.5 (Fig. 6b). On the simple protocol, GPT-5-mini, GPT-5.1 and GPT-5.5 succeeded in all 20 replicates and are statistically indistinguishable. Only GPT-5-nano fell to 11 of 20. On the dilution series the same three models separate sharply. GPT-5-mini and GPT-5.1 reached 11 and 10 of 20 against 20 of 20 for GPT-5.5, and GPT-5-nano produced no correct protocol at all. Model capability therefore binds only where the task carries substantive logic. Cost per run behaves differently in money and in time (Fig. 6c). Monetary cost follows model pricing, rising monotonically from \$0.04 for GPT-5-nano to \$1.09 for GPT-5.5 on the dilution series at list prices without prompt caching (Table S16). Token consumption and wall-clock time, in contrast, are highest for the smallest model. GPT-5-nano required 191 thousand tokens and 671 s per run on the dilution series, nearly twice the tokens and five times the time of GPT-5.1, while never producing a correct protocol. On the dilution series the median tool-call count is four for GPT-5-nano and GPT-5-mini against two for GPT-5.5, and on the simple protocol GPT-5-nano required five where GPT-5.5 needed three. GPT-5-mini also consumed 157 thousand input tokens against 115 thousand for GPT-5.5 and took twice as long per run, without reaching a comparable success rate (Table S15). The distribution of consumption differs in kind for the smallest model, whose completion tokens account for 44% of its total against 13 to 16% for the others, consistent with generating extensively without converging. Choosing a smaller model therefore trades monetary cost against latency, and below a threshold of capability it buys neither, since the additional token and time consumption does not yield a correct protocol.

Relating cost to the observed success rate shows that the appropriate model depends on the task. On the simple protocol, where GPT-5-mini and GPT-5.5 are indistinguishable at 20 of 20, GPT-5-mini produces an equivalent artifact for roughly one thirteenth of the cost, \$0.03 against \$0.40. On the dilution series only GPT-5.5 produces a correct artifact with high reliability at all. Extending the agentic workflows of the platform by adding a mechanism for scoring task difficulty and routing generation to the smallest model adequate for a given artifact would therefore reduce cost substantially without affecting the outcome.

The failures were classified by what the artifact would do to an experiment rather than by which check detected them, since a single underlying mistake can trigger several checks (Fig. 6d, Section S6). The relevant distinction is whether a defect announces itself. The 25 failures considered here are those of the deployed configuration, that is, all artifacts of both templates generated with GPT-5.1. The failures of the different models in the model comparison are reported separately in the (Table S13).

15 of the 25 failures raised an error before completing, either because the protocol was rejected by the simulator or because the script terminated during execution, and would therefore be apparent without inspecting the output. The remaining ten failures produce a result that appears complete. Yet, five delivered an incorrect quantity, in that three device protocols dispensed the wrong volume into a well, and two analysis scripts returned fitted isotherm parameters and a peak height outside the tolerance of the reference. Four report scripts introduced a placeholder for which no corresponding parameter exists, so the field would remain unsubstituted in the generated report. One analysis script reported success while producing none of the required output files. These differ in how readily a user would notice them. The missing output files and the unsubstituted placeholders are conspicuous, since the run produces no analysis results and the report renders with an empty field. An incorrect dispensing volume is not visible in the generated code but would become apparent on execution. The two quantitative deviations are the least detectable. The pipeline completes, a report is generated and a value is written to the database, and nothing distinguishes the result from a correct one without an independent reference. Failures that could propagate unnoticed into the experimental record are therefore a small minority, 2 of 25, but they are the ones that motivate verifying a generated analysis against known data before productive use.

These results place the template execution reported in Section 2.4 in context. All protocols were generated once and executed successfully at the first attempt, but for the dilution-series protocol the model deployed at the time produces a correct artifact in about half of independent generations, so that outcome reflects a favorable draw as much as a dependable process. Agent-assisted template generation is therefore reliable for workflow-structural scripts and for device protocols without substantial arithmetic, but not certain for the most complex artifacts. For this reason, the platform provides explicit review of every generated artifact and version-controlled, agent-assisted revision rather than treating generation as a single irreversible step. The limitation tracks model capability rather than the approach itself, as the same protocol was generated correctly in all 20 replicates by the more capable GPT-5.5. Once a template has been verified by the user, all subsequent executions are deterministic and no further generation occurs, so the uncertainty is confined to the creation phase and cannot propagate into the experimental record.

3. Conclusion

In this work, we presented a template-based workflow approach for SDLs that separates workflow design from workflow execution, combining interactive no-code wizard interfaces with AI-agent-assisted code generation to enable domain scientists to create, execute and document complex experimental and simulative workflows without manual programming. The approach was demonstrated through a chromatographic case study involving automated isotherm determination, mechanistic simulation of two elution strategies and experimental column validation. All steps were executed entirely through wizard-created templates integrating two multi-device SDLs. Regenerating every artifact of the experimental templates 20 times with GPT-5.1, the LLM deployed in the platform's agents at the time of the case study, showed that reliability is task-dependent. The orchestration scripts, which encode the sequence of device operations, were correct in all 20 replicates for both templates, while the analysis, report and database scripts reached 16 to 19 of 20. Among the five device protocols, four reached 18 to 20 of 20 and the protocol carrying the volume arithmetic of the experiment reached 10 of 20, giving 87 of 100 pooled. Reliability further depends on the capability of the underlying LLM, especially for complex tasks. On a simple protocol without volume arithmetic three of four LLMs tested succeeded in all replicates, whereas on the dilution series only the most capable LLM, GPT-5.5, reached 20 of 20 against 10 of 20 for GPT-5.1. A generated template therefore requires verification before productive use, after which it executes deterministically, and the limiting factor is one that successive LLM generations can be expected to relax without any change to the platform itself.

While parameterized, reusable workflows are long established in laboratory automation, the advance presented in this study is that they are authored conversationally and graphically rather than programmed, and that reporting and structured data storage are artifacts of the template rather than steps added afterwards. The contribution lies in agent-supported end-to-end authoring combined with the deliberate restriction of the agents to the design phase. Our approach occupies the middle ground between fully autonomous agent-driven systems, which generate and execute experimental logic on the fly, and conventional laboratory automation, which is reproducible but requires programming expertise. AI agents assist during the one-time template design phase, while execution proceeds deterministically from the fixed templates. This separation is particularly well-suited for systematic workflows such as screening campaigns, standard operating procedures and optimization studies, where reproducibility and consistent data handling are essential.

A key design principle is modularity at both the hardware and software level. A new instrument is integrated by implementing one device wrapper class and the corresponding endpoint on the device control server. The instrument is then registered in the device inventory so that it becomes selectable in the wizards, and its operations and calling conventions are

added to the documentation available to the device protocol agent. The orchestration mechanism, execution logic and data pipeline are unaffected, since they operate on the wrapper interface rather than on instrument-specific code. On the software side, both wizards support reusable building blocks, ELN resources for simulation configurations and compound steps for experimental sequences, so that the platform's capabilities grow incrementally with each new integration.

The effort required to extend the platform differs between the experimental and the simulative side. On the experimental side the platform encodes no domain-specific knowledge as it holds no representation of the principles of a given field and does not check a planned experiment for plausibility against such knowledge. A domain is therefore represented entirely by its devices and their methods, so that the operation palette of the wizard is populated from the registered devices. Extending the platform to a new domain reduces to integrating the devices that domain requires, together with corresponding additions to the knowledge provided to the AI agents, whose system prompts currently carry information about the integrated devices and analysis of their data. Extension on the experimental side is thus bounded by the number of devices rather than by the number of domains, which is consistent with the platform being in routine use for processes well outside chromatography. A wizard for device integration and laboratory configuration is under development, which will parse this device knowledge automatically and supply it to the agents, further reducing the effort required. On the simulative side the template wizard is structured around the unit operations, flow sheets and process sequences of CADET-Process, so extending it to a different modelling framework would require a corresponding wizard for that framework, and the effort scales with the number of modelling frameworks. In both cases the generic layer transfers unchanged, though extension currently remains a software development effort.

Several limitations of the proposed approach and its current implementation should be acknowledged. While the wizard interfaces eliminate the need for programming, creating a workflow still requires domain expertise regarding the experimental methodology and available devices. The AI-assisted code generation operates as a single interaction rather than a continuous dialogue during authoring, so a generated protocol has to be verified by the user before productive use, in practice by executing it once on the target instrument. Most failures observed in the replication study surfaced as an error rather than as a plausible-looking result, so verification does not require exhaustive inspection of the generated code. The exceptions were quantitative deviations in generated analysis code, which motivates checking a generated analysis against known data. Subsequent corrections are supported through the agent-assisted, version-controlled revision (see Section 2.3 and S2). Once created, a template has a fixed sequential structure. Conditional branching based on intermediate results, *e.g.*, skipping a wash step if a measured concentration falls below a threshold, or dynamically adjusting volumes mid-protocol, cannot be expressed in the current template format. Templates are currently tied to the specific

device configuration of the SDL they were created for, limiting cross-laboratory portability. The simulation wizard currently exposes only selected features of the CADET-Process framework, and capabilities such as process optimization, fractionation constraints, and reaction modeling are not yet accessible. CADET itself already covers further domains including fermentation, ultra-/diafiltration, and enzymatic reactions which could be integrated by exposing these existing features. The platform further relies on the user prompt to specify any data quality control, so that results are processed, documented and stored irrespective of data quality unless such checks are requested. Finally, the case study showed that automating a workflow does not by itself guarantee that its parameters transfer quantitatively to a different experimental scale, as the batch-derived isotherm parameters predicted the observed elution behavior qualitatively but placed the eluting peak late under the gradient strategy. Identifying and addressing such gaps remains a scientific task for the user, which the platform supports but does not perform.

Future development will pursue several directions to address these limitations and extend the platform's capabilities. Workflow design could be elevated from the current device-method level to a higher-level, device-agnostic process representation, for example "equilibrate resin, incubate with protein, separate phases, measure concentration", which is then translated into concrete device operations based on the available instruments and their exposed methods in a given laboratory. Such an abstraction would make workflows portable between laboratories with different instrumentation, and would allow a campaign to be specified in terms of the operations it requires rather than of the templates that exist for a particular setup. Independently of this, AI agents could operate at the campaign planning level, selecting, parameterizing and sequencing validated templates from a growing library rather than generating experimental logic anew. This maintains the core architectural principle as the agent would propose the composition of a campaign while each execution remains deterministic, and the scientist would retain the decision, with the agent advising, supervising and suggesting next steps rather than directing the campaign itself. Expanding the simulative capabilities by integrating further modelling frameworks, will broaden the scope of accessible simulative workflows and can reduce experimental time and cost. Finally, support for alternative ELN systems beyond eLabFTW would make the system more broadly applicable.

In summary, restricting the AI agents to the design phase is what allows conversational, no-code authoring and fully deterministic execution to coexist in a single workflow. For laboratory automation more generally, this separation lowers the barrier to encoding an experimental procedure once and reusing it, since the authoring step no longer requires programming while the resulting workflow retains the reproducibility and auditability of conventional automation. Where such workflows are to be run in closed loop, the same separation provides the layer on which autonomous operation rests. With validated templates in place, an autonomous campaign requires only an optimizer that instantiates them with successive parameter sets, which the

platform already implements. As self-driving laboratories mature from bespoke, single-purpose systems toward flexible, multi-domain research platforms, architectures that separate creative design from reliable execution will be essential for scaling laboratory automation to the breadth of different workflows that modern experimental science demands.

4. Experimental

4.1 Batch incubation chromatography for isotherm parameter determination

To determine the loading isotherm parameters of diverse proteins to chromatography resins at varying salt concentrations an automated experimental workflow was implemented for SDL 1 using the wizard of the web application. The volumes, durations, buffers, salt concentrations, ligands, resins and measurement wavelength can be freely adjusted when creating an experiment from the template. The values used for the experiments presented here are given below. A schematic of the complete workflow is shown in Fig. S5.

The workflow is built around a 96-deep-well filter plate (AcroPrep Advance, 1 mL, 0.2 μ m Supor, Pall), which serves as the reaction vessel throughout. The resin is retained by the filter membrane at the bottom of each well, while all liquid phases are added from above by a liquid handling robot (OT-2, OpenTrons) and removed from below by vacuum filtration. Two dedicated devices support this arrangement, each used at a distinct point in the procedure. A resin loader (ResiQuot, MediaScout) is used once, before the automated workflow begins, to charge the wells with a defined resin volume. A well-plate vacuum manifold (PerkinElmer) is used repeatedly during the automated workflow, for every liquid removal step.

Resin was loaded in a manual preparatory step. The resin (Toyopearl MX-Trp-650M, Tosoh) was suspended in ethanol, and the suspension was distributed over the cavities of the resin loader and drawn through by vacuum, forming a dense packed plug of defined volume in each cavity. Excess resin above the cavities was scraped off level, after which the plugs were transferred into the wells of the filter plate. This procedure dispenses 7.7 μ L of packed resin per well with high volumetric precision.⁵⁶ The charged filter plate was stored in ethanol until use. All quantities of bound protein are therefore expressed per volume of packed resin.

The automated workflow then proceeds with two defined manual interventions, in which the filter plate is sealed before and unsealed after the incubation step to prevent evaporation and cross-contamination during agitation. At each of these points the orchestration script registers a pause in the web application and waits until the user confirms completion, as described in Section 2.1. All other steps proceed without manual intervention. Labware transfers between the devices are handled by a six-axis robotic arm (UR-5e, Universal Robots), and agitation during the incubation steps by a shaker module (Heater Shaker Module V1, OpenTrons) mounted on the deck of the liquid handling robot. Each liquid removal step is performed as a filtration cycle: the robotic arm places a 96-deep-well collection plate beneath the manifold, positions the

manifold and places the filter plate on top, the vacuum pump (MD 4 VARIO select, Vacuubrand) is operated at 25% of its maximum speed for 30 s, the assembly is vented for 5 min, and the filter plate is returned to the liquid handling robot.

The workflow begins with buffer preparation. Binding and equilibration buffers were obtained by mixing low- and high-salt stock solutions in 12-well reservoirs. For the experiments presented here, 20 mM sodium acetate at pH 4.0 containing 0, 100, 200, 300 or 500 mM NaCl was prepared for equilibration (1× concentrated) and containing 0, 200, 400, 600 or 1000 mM NaCl for the binding step (2× concentrated), mixed from 20 mM sodium acetate stock solutions at pH 4.0 with 0 and 1000 mM NaCl. Protein solutions of ovalbumin and transferrin were prepared at twice the desired final concentration by dilution of protein stock solutions (1.4 and 14.0 mg mL⁻¹) in 20 mM sodium acetate at pH 4.0 without NaCl, in a 96-deep-well plate, giving initial protein concentrations of 0.2, 0.5, 1.0, 1.5, 2.0, 5.0 and 7.0 mg mL⁻¹ for isotherm determination (Opentrons OT-2 protocol 1).

The ethanol storage solution was then removed from the resin by a filtration cycle. The resin was equilibrated three times, each time by adding 300 µL of equilibration buffer at the salt concentration assigned to the respective well (Opentrons OT-2 protocol 2), incubating for 5 min at 1300 rpm, and removing the supernatant by a filtration cycle. After equilibration, 150 µL of the 2× concentrated protein solution and 150 µL of the 2× concentrated binding buffer were added to the equilibrated resin (Opentrons OT-2 protocol 3), yielding the intended final protein and salt concentrations upon mixing, and incubated for 2 h at 1300 rpm (Opentrons OT-2 protocol 4) after the manual sealing step.

After incubation, the unbound protein solution was removed by a final filtration cycle into a clean collection plate and 200 µL of the filtrate was transferred to a 96-well flat-bottom UV plate (UV-star 96 well F-bottom transparent, Greiner) for UV absorption measurements (Opentrons OT-2 protocol 5) at 280 nm using a multi-mode plate reader (Spark, Tecan). All experiments were performed as triplicates.

The equilibrium protein concentrations were obtained from the raw absorbance data with a calibration curve and the equilibrium load of the resin $q_{e,i,c_{\text{mod}}}$ (mg mL⁻¹) with component i at the modulator concentration c_{mod} (mM) calculated by eqn (1).

$$q_{e,i,c_{\text{mod}}} = \frac{c_{0,i} - c_{e,i,c_{\text{mod}}}}{V_{\text{ads}}} V_0 \quad (1)$$

Here, $c_{e,i,c_{\text{mod}}}$ (mg mL⁻¹) is the equilibrium concentration of component i in the mobile phase the modulator concentration c_{mod} , $c_{0,i}$ (mg mL⁻¹) is the initial concentration of component i , V_0 is the total liquid volume during binding (300 µL) and V_{ads} is the volume of the resin (7.7 µL).

To describe the salt-dependent adsorption behavior of the proteins, the MPM model was used. First, a Langmuir isotherm model⁵⁷ (eqn (2)) was fitted to the experimental data of each modulator concentration separately, to obtain the effective Langmuir constant $K_{L,i,c_{\text{mod}}}^{\text{eff}}$ (mL mg⁻¹) and maximal loading capacity $q_{\text{max},i,c_{\text{mod}}}$ (mg mL⁻¹) for component i at that condition.

$$q_{e,i,c_{\text{mod}}} = q_{\text{max},i,c_{\text{mod}}} \frac{K_{L,i,c_{\text{mod}}}^{\text{eff}} c_{e,i,c_{\text{mod}}}}{1 + K_{L,i,c_{\text{mod}}}^{\text{eff}} c_{e,i,c_{\text{mod}}}} \quad (2)$$

To obtain the hydrophobicity γ_i and ion-exchange interaction β_i factors of the MPM model^{54,58} describing the influence of the modulator concentration c_{mod} on binding, the determined effective Langmuir constants $K_{L,i,c_{\text{mod}}}^{\text{eff}}$ at different NaCl concentrations were fitted to eqn (3).

$$K_{L,i,c_{\text{mod}}}^{\text{eff}} = K_{L,i}^0 e^{\gamma_i c_{\text{mod}}} \left(\frac{c_{\text{mod}}}{c_{\text{ref}}} \right)^{-\beta_i} \quad (3)$$

Here γ_i (mM⁻¹) describes the hydrophobic contribution and $\beta_i(-)$ the ion-exchange characteristic. $K_{L,i}^0$ (mL mg⁻¹) is a pre-exponential constant corresponding to the ratio $k_{a,i}/k_{d,i}$ of the CADET implementation of the MPM model, and $c_{\text{ref}} = 1$ mM is a reference modulator concentration introduced so that the power-law term is dimensionless and $K_{L,i}^0$ carries the units of an effective Langmuir constant. Because the power-law term diverges as $c_{\text{mod}} \rightarrow 0$, eqn (3) is defined only for $c_{\text{mod}} > 0$. The regression was therefore performed over the conditions containing added NaCl (100, 200, 300 and 500 mM), with $K_{L,i}^0$ held fixed at the effective Langmuir constant measured without added NaCl (Table 1) and with the constraint $\beta_i \geq 0$ as required by the model implementation. $K_{L,i}^0$ is thus an anchoring constant and not the $c_{\text{mod}} \rightarrow 0$, limit of eqn (3), which does not exist.

4.2 Column chromatography simulations

For mechanistic modelling of column chromatography experiments CADET-Process (v0.11.1) was used in Python 3.11. An ÄKTA pure 25 (Cytiva) system setup (Fig. S7) was modelled, using a single inlet, a single tubular reactor for the system dead volume, a column with a lumped rate with pores transport model and a single outlet. The simulation workflows were implemented with the simulation template wizard and executed via the web application.

The binding parameters were transferred from the batch experiments to CADET's kinetic MPM formulation as described in Section S4, with $K_{L,i}^0$ entering as the ratio $k_{a,i}/k_{d,i}$. Consistent with the isotherm regression, c_{mod} refers to the added NaCl concentration only. The acetate buffer counter-ions present in all phases were not treated as a modulator component. Because the desorption term of the MPM model scales with $c_{\text{mod}}^{\beta_i}$ is non-negative, it vanishes at zero modulator concentration, so binding is effectively irreversible during the simulated equilibration, load and wash phases, in which no NaCl is present. Quantitative capture during these phases is therefore predicted by construction rather than computed from the fitted affinities. The MPM model contains no pH dependence. All isotherm parameters were determined at pH 4.0 and the simulations describe that condition alone, so the model does not represent the alkaline CIP step or any other change in protein net charge.

Geometric parameters were taken from the manufacturer specification. The bed porosity was determined from blue dextran tracer experiments and the particle porosity calculated

from the total porosity obtained with acetone. The extra-column volume was determined from the system dead volume measured without an attached column. The axial dispersion coefficient was estimated from the correlation of Chung and Wen⁵⁹ and the component-specific film diffusion coefficients from that of Wilson and Geankoplis,⁶⁰ and the pore accessibility factor was set to unity, *i.e.*, no size exclusion was applied. A detailed description of the determination and unit conversion for all simulation parameters is given in SI Section S4 and the utilized parameters are listed in Tables S4–S7. Additionally, the simulation scripts are provided in the accompanying repository.

4.3 FPLC validation runs

To automate the FPLC runs, the device control server leveraged the Orbit controller,⁶¹ a Python-based supervisory controller for chromatography systems developed at Lund University. Orbit supports two connection modes to the UNICORN control software of the ÄKTA system, an OPC connection through the UNICORN OPC server and a direct API connection available from UNICORN version 7.4 onwards. The API mode with UNICORN 7.8 was used in this work, which requires the UNICORN online server to be enabled through the UNICORN Service Tool and the controlling account to be granted access to the system in the ServerAPI access group. Since Orbit and UNICORN cannot hold control of the system simultaneously, control is released from UNICORN before a run. The device control server exposes method upload, run start, run status and result retrieval as HTTP endpoints, so that the orchestration script controls the chromatography system through the same device abstraction as all other instruments (Section 2.1).

Linear gradient elution and step elution experiments were performed using buffer A (20 mM sodium acetate) and buffer B (20 mM sodium acetate, 2.0 M NaCl). Both buffers A and B were adjusted to pH 4.0 with glacial acetic acid. The buffers were mixed in a 0.6 mL static mixer and passed through the injection valve to the chromatography column. The column effluent was monitored sequentially by the UV detector (280 nm) and the conductivity detector, before being directed to waste. The method started with an equilibration step with 5 CV of buffer A to ensure that the stationary and mobile phase were in the desired chemical state regarding pH and ionic strength prior to sample loading. Next, the protein solution (1 mg mL⁻¹ of each ovalbumin and transferrin in 20 mM sodium acetate at pH 4.0) was injected using a 500 µL manually loaded sample loop. During injection, the flow rate was maintained at 0.2 mL min⁻¹ and subsequently increased to 0.5 mL min⁻¹. After a wash step of 5 CV with buffer A, either a linear gradient (0 to 2000 mM NaCl in 20 CV) or stepwise elution (3.5 CV at 650 mM NaCl, then 6 CV at 1600 mM NaCl) was carried out. Finally, a CIP step was conducted between runs using buffer B, 0.5 M NaOH, 0.5 M HCl, and water, applied sequentially to remove residual contaminants and restore column performance.

4.4 Materials

Sodium acetate was purchased from Carl Roth GmbH & Co KG (Karlsruhe, Germany). Acetone, glacial acetic acid, blue dextran,

sodium chloride, ovalbumin, transferrin, sodium hydroxide (NaOH, 1.0 N), hydrochloric acid (HCl, 36.5–38%) were obtained from Merck KGaA (Darmstadt, Germany). The resin Toyopearl MX-Trp-650 M was obtained from Tosoh Bioscience (Tokyo, Japan). All chemicals and reagents were used as received without additional purification. Buffers were prepared in ultra-pure water and pH was adjusted using glacial acetic acid.

Author contributions

Sebastian Putz: conceptualization, methodology, investigation, formal analysis, visualization, writing – original draft. Ahmed Khalil Mama: conceptualization, investigation, writing – review & editing. Matthias Franzreb: conceptualization, supervision, project administration, funding acquisition, writing – review & editing.

Conflicts of interest

There are no conflicts to declare.

Data availability

All data and code supporting this article are openly available in a Zenodo repository at <https://doi.org/10.5281/zenodo.20625765>. The repository contains the AI-agent (n8n) workflows and system prompts, the wizard-generated workflow-template scripts, the experimental scripts and raw data, the chromatography simulation configurations and scripts, and the data analysis scripts to reproduce Fig. 5. The specific version used in this study is version 2.0.0, archived at <https://doi.org/10.5281/zenodo.22016346>. The concept DOI <https://doi.org/10.5281/zenodo.20625765> resolves to the most recent version. Additional supporting data are provided as part of the supplementary information (SI). Supplementary information: details on the laboratories, AI agents, template wizards, simulation configuration, column validation chromatograms, agent reliability study and software architecture. See DOI: <https://doi.org/10.1039/d6dd00418k>.

Acknowledgements

This work was funded by the Karlsruhe Institute of Technology through the subproject BioCAR within the project Auto. MAP and the Helmholtz Programme ‘Materials Systems Engineering’ (Programme No. 43.31.01). During software development, the generative AI coding assistant Claude Code (Claude Opus 4.5) was used to support implementation, debugging, and refactoring of the web application. Generative AI tools were additionally used to assist with language refinement during manuscript preparation. All system design and integration decisions were made by the authors, and all AI-generated content and code was reviewed and verified by the authors, who take full responsibility for the work.

References

- 1 M. Abolhasani and E. Kumacheva, *Nat. Synth.*, 2023, **2**, 483–492.
- 2 F. Häse, L. M. Roch and A. Aspuru-Guzik, *Trends Chem.*, 2019, **1**, 282–291.
- 3 G. Tom, S. P. Schmid, S. G. Baird, Y. Cao, K. Darvish, H. Hao, S. Lo, S. Pablo-García, E. M. Rajaonson, M. Skreta, N. Yoshikawa, S. Corapi, G. D. Akkoc, F. Strieth-Kalthoff, M. Seifrid and A. Aspuru-Guzik, *Chem. Rev.*, 2024, **124**, 9633–9732.
- 4 H. G. Martin, T. Radivojevic, J. Zucker, K. Bouchard, J. Sustarich, S. Peisert, D. Arnold, N. Hillson, G. Babnigg, J. M. Marti, C. J. Mungall, G. T. Beckham, L. Waldburger, J. Carothers, S. Sundaram, D. Agarwal, B. A. Simmons, T. Backman, D. Banerjee, D. Tanjore, L. Ramakrishnan and A. Singh, *Curr. Opin. Biotechnol.*, 2023, **79**, 102881.
- 5 D. N. Carruthers, P. C. Kinnunen, Y. Li, Y. Chen, J. W. Gin, I. S. Yunus, W. R. Galliard, S. Tan, T. Radivojevic, P. D. Adams, A. K. Singh, J. Sustarich, C. J. Petzold, A. Mukhopadhyay, H. Garcia Martin and T. S. Lee, *Nat. Commun.*, 2025, **16**, 11489.
- 6 K. Fushimi, Y. Nakai, A. Nishi, R. Suzuki, M. Ikegami, R. Nimura, T. Tomono, R. Hidese, H. Yasueda, Y. Tagawa and T. Hasunuma, *Sci. Rep.*, 2025, **15**, 6648.
- 7 J. T. Rapp, B. J. Bremer and P. A. Romero, *Nat. Chem. Eng.*, 2024, **1**, 97–107.
- 8 N. Singh, S. Lane, T. Yu, J. Lu, A. Ramos, H. Cui and H. Zhao, *Nat. Commun.*, 2025, **16**, 5648.
- 9 E. Orsi, L. Schada von Borzyskowski, S. Noack, P. I. Nikel and S. N. Lindner, *Nat. Commun.*, 2024, **15**, 3447.
- 10 C. W. Coley, D. A. Thomas, J. A. M. Lummiss, J. N. Jaworski, C. P. Breen, V. Schultz, T. Hart, J. S. Fishman, L. Rogers, H. Gao, R. W. Hicklin, P. P. Plehiers, J. Byington, J. S. Piotti, W. H. Green, A. J. Hart, T. F. Jamison and K. F. Jensen, *Science*, 2019, **365**, eaax1566.
- 11 J. M. Granda, L. Donina, V. Dragone, D.-L. Long and L. Cronin, *Nature*, 2018, **559**, 377–381.
- 12 S. Steiner, J. Wolf, S. Glatzel, A. Andreou, J. M. Granda, G. Keenan, T. Hinkley, G. Aragon-Camarasa, P. J. Kitson, D. Angelone and L. Cronin, *Science*, 2019, **363**, eaav2211.
- 13 A. M. K. Nambiar, C. P. Breen, T. Hart, T. Kulesza, T. F. Jamison and K. F. Jensen, *ACS Cent. Sci.*, 2022, **8**, 825–836.
- 14 M. B. Rooney, B. P. MacLeod, R. Oldford, Z. J. Thompson, K. L. White, J. Tungjunyatham, B. J. Stankiewicz and C. P. Berlinguette, *Digital Discovery*, 2022, **1**, 382–389.
- 15 R. W. Epps, M. S. Bowen, A. A. Volk, K. Abdel-Latif, S. Han, K. G. Reyes, A. Amassian and M. Abolhasani, *Adv. Mater.*, 2020, **32**, 2001626.
- 16 J. Li, J. Li, R. Liu, Y. Tu, Y. Li, J. Cheng, T. He and X. Zhu, *Nat. Commun.*, 2020, **11**, 2046.
- 17 K. Vaddi, H. T. Chiang and L. D. Pozzo, *Digital Discovery*, 2022, **1**, 502–510.
- 18 B. P. MacLeod, F. G. L. Parlane, T. D. Morrissey, F. Häse, L. M. Roch, K. E. Dettelbach, R. Moreira, L. P. E. Yunker, M. B. Rooney, J. R. Deeth, V. Lai, G. J. Ng, H. Situ, R. H. Zhang, M. S. Elliott, T. H. Haley, D. J. Dvorak, A. Aspuru-Guzik, J. E. Hein and C. P. Berlinguette, *Sci. Adv.*, 2020, **6**, eaaz8867.
- 19 T. Pickles, Y. Leghrib, M. Weisshaar, M. Goncharuk, P. Timperman, T. Doherty, D. D. Ford, J. Moores, A. J. Florence and C. J. Brown, *Digital Discovery*, 2025, **4**, 2025–2032.
- 20 A. D. Clayton, A. M. Schweidtmann, G. Clemens, J. A. Manson, C. J. Taylor, C. G. Niño, T. W. Chamberlain, N. Kapur, A. J. Blacker, A. A. Lapkin and R. A. Bourne, *Chem. Eng. J.*, 2020, **384**, 123340.
- 21 M. Christensen, L. P. E. Yunker, F. Adediji, F. Häse, L. M. Roch, T. Gensch, G. Dos Passos Gomes, T. Zepel, M. S. Sigman, A. Aspuru-Guzik and J. E. Hein, *Commun. Chem.*, 2021, **4**, 112.
- 22 L. M. Helleckes, S. Putz, K. Gupta, M. Franzreb and H. Garcia Martin, *Curr. Opin. Biotechnol.*, 2026, **97**, 103392.
- 23 M. Seifrid, R. Pollice, A. Aguilar-Granda, Z. Morgan Chan, K. Hotta, C. T. Ser, J. Vestfrid, T. C. Wu and A. Aspuru-Guzik, *Acc. Chem. Res.*, 2022, **55**, 2454–2466.
- 24 D. A. Boiko, R. MacKnight, B. Kline and G. Gomes, *Nature*, 2023, **624**, 570–578.
- 25 A. M. Bran, S. Cox, O. Schilter, C. Baldassari, A. D. White and P. Schwaller, *Nat. Mach. Intell.*, 2024, **6**, 525–535.
- 26 Y. Qu, K. Huang, M. Yin, K. Zhan, D. Liu, D. Yin, H. C. Cousins, W. A. Johnson, X. Wang, M. Shah, R. B. Altman, D. Zhou, M. Wang and L. Cong, *Nat. Biomed. Eng.*, 2025, **10**, 245–258.
- 27 K. Darvish, M. Skreta, Y. Zhao, N. Yoshikawa, S. Som, M. Bogdanovic, Y. Cao, H. Hao, H. Xu, A. Aspuru-Guzik, A. Garg and F. Shkurti, *Matter*, 2025, **8**, 101897.
- 28 H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes and A. Mian, *ACM Trans. Intell. Syst. Technol.*, 2025, **16**, 1–72.
- 29 B. Atil, S. Aykent, A. Chittams, L. Fu, R. J. Passonneau, E. Radcliffe, G. R. Rajagopal, A. Sloan, T. Tudrej, F. Ture, Z. Wu, L. Xu and B. Baldwin, *arXiv*, 2024, preprint, arXiv:2408.04667, DOI: [10.48550/arXiv.2408.04667](https://doi.org/10.48550/arXiv.2408.04667).
- 30 S. Sureshkumar, *arXiv*, 2026, preprint, arXiv:2601.09749, DOI: [10.48550/arXiv.2601.09749](https://doi.org/10.48550/arXiv.2601.09749).
- 31 L. Chen, M. Zaharia and J. Zou, *arXiv*, 2023, preprint, arXiv:2305.05176, DOI: [10.48550/arXiv.2305.05176](https://doi.org/10.48550/arXiv.2305.05176).
- 32 A.-M. Anhel, L. Alejaldre and Á. Goñi-Moreno, *ACS Synth. Biol.*, 2023, **12**, 3514–3520.
- 33 B. Bartley, J. Beal, M. Rogers, D. Bryce, R. P. Goldman, B. Keller, P. Lee, V. Biggers, J. Nowak, M. Weston and J. Emerg, *Technol. Comput. Syst.*, 2023, **19**, 1–21.
- 34 B. Miles and P. L. Lee, *SLAS Technol.*, 2018, **23**, 432–439.
- 35 M. I. Sadowski, C. Grant and T. S. Fell, *Trends Biotechnol.*, 2016, **34**, 214–227.
- 36 J. Vrana, O. De Lange, Y. Yang, G. Newman, A. Saleem, A. Miller, C. Cordray, S. Halabiya, M. Parks, E. Lopez, S. Goldberg, B. Keller, D. Strickland and E. Klavins, *Synth. Biol.*, 2021, **6**, ysab006.
- 37 A. T. Hanke and M. Ottens, *Trends Biotechnol.*, 2014, **32**, 210–220.

- 38 J. Schmölder and M. Kaspereit, *Processes*, 2020, **8**, 65.
- 39 S. Putz, J. Döttling, T. Ballweg, A. Tschöpe, V. Biniyaminov and M. Franzreb, *Adv. Intell. Syst.*, 2024, **7**, 2400564.
- 40 S. Putz, N. Teetz, M. Abt, P. Jerono, T. Meurer and M. Franzreb, *Biotechnol. Bioeng.*, 2025, **122**, 3018–3036.
- 41 Jinja2, <https://pypi.org/project/Jinja2/>, accessed 25 February 2026.
- 42 htmx, <https://htmx.org/>, accessed 17 June 2026.
- 43 Flask, <https://flask.palletsprojects.com/en/stable/>, accessed 23 June 2026.
- 44 n8n, <https://n8n.io/>, accessed 25 February 2026.
- 45 N. Carpi, A. Minges and M. Piel, *J. Open Source Softw.*, 2017, **2**, 146.
- 46 S. Leweke and E. von Lieres, *Comput. Chem. Eng.*, 2018, **113**, 274–294.
- 47 FastAPI – FastAPI, <https://fastapi.tiangolo.com/>, accessed 17 June 2026.
- 48 J. P. Dürholt, T. S. Asche, J. Kleinekorte, G. Mancino-Ball, B. Schiller, S. Sung, J. Keupp, A. Osburg, T. Boyne, R. Misener, R. Eldred, C. Kappatou, R. M. Lee, D. Linzner, W. S. Costa, D. Walz, N. Wulkow and B. Shafei, *J. Mach. Learn. Res.*, 2025, **26**, 1–7.
- 49 Git, <https://git-scm.com/>, accessed 13 August 2026.
- 50 S. A. Kidwai, A. A. Ansari and A. Salahuddin, *Biochem. J.*, 1976, **155**, 171–180.
- 51 J. A. Beeley, S. M. Stevenson and J. G. Beeley, *Biochim. Biophys. Acta, Protein Struct.*, 1972, **285**, 293–300.
- 52 A. G. Hovanesian and Z. L. Awdeh, *Eur. J. Biochem.*, 1976, **68**, 333–338.
- 53 F. Kilár, *J. Chromatogr. A*, 1991, **545**, 403–406.
- 54 D. Karlsson, N. Jakobsson, A. Axelsson and B. Nilsson, *J. Chromatogr. A*, 2004, **1055**, 29–39.
- 55 W. R. Melander, Z. El Rassi and C. Horváth, *J. Chromatogr. A*, 1989, **469**, 3–27.
- 56 T. Herrmann, M. Schroder and J. Hubbuch, *Biotechnol. Prog.*, 2006, **22**, 914–918.
- 57 M. A. Islam, M. A. Chowdhury, M. S. I. Mozumder and M. T. Uddin, *ACS Omega*, 2021, **6**, 14481–14492.
- 58 S. Yamamoto, K. Nakanishi, R. Matsuno and T. Kamikubo, *Biotechnol. Bioeng.*, 1983, **25**, 1465–1483.
- 59 S. F. Chung and C. Y. Wen, *AIChE J.*, 1968, **14**, 857–866.
- 60 E. J. Wilson and C. J. Geankoplis, *Ind. Eng. Chem. Fund.*, 1966, **5**, 9–14.
- 61 N. Andersson, A. Löfgren, M. Olofsson, A. Sellberg, B. Nilsson and P. Tiainen, *Biotechnol. Prog.*, 2017, **33**, 923–930.