Trends and Reviews of Technological Engineering in Education
Vol. 2, No. 3 (2026) | ISSN 3071-2211 | Open Access

RESEARCH ARTICLE

A Knowledge-Graph and Constraint-Solving Engine for Explainable Engineering Learning Pathways

Ziyu Wang[1], Yisong Chen[2, *]
[1] School of Business, Wake Forest University, Winston-Salem, USA 27109
[2] College of Computing, Georgia Institute of Technology, Atlanta, USA 30332
[*] Correspondence: ychen841@gatech.edu

Received 8 May 2026 · Accepted 16 August 2026 · Published 30 September 2026

Abstract

Engineering learners move through pathways that connect standards, local course outcomes, prerequisites, practical tasks, assessment evidence, equipment, and safety requirements. A graph can represent these relationships, but graph reachability alone does not guarantee that a proposed sequence is feasible, accessible, current, or safe. A recommender can rank activities, yet a score does not show which rules made a path admissible or why an alternative was excluded. This article specifies a knowledge-graph and constraint-solving engine for generating explainable engineering learning pathways. The design separates a versioned curriculum graph from a privacy-minimized learner-state view, compiles approved competency goals and graph relationships into a finite-domain constraint model, and uses a constraint programming solver to find sequences that satisfy hard prerequisites, safety, availability, and learner access requirements while optimizing declared preferences. Each path is accompanied by an evidence-linked explanation that identifies the target outcome, prerequisite chain, constraints applied, trade-offs, and unresolved conditions. When infeasibility is proven, the system reports a conflict set if available and offers only policy-approved alternatives; it never relaxes a safety rule automatically. A synthetic microgrid-design scenario illustrates the model, and a conformance and evaluation plan defines tests for semantic integrity, solver correctness, explanation fidelity, accessibility, privacy, and curriculum review. No empirical deployment or learning-effect result is claimed. The engine is intended as decision support that leaves curriculum authority with educators and pathway choice with learners.

Keywords: Knowledge Graphs; Constraint Programming; Engineering Education; Learning Pathways; Explainable AI; Competency Standards

1. Introduction

1.1. Why engineering pathways require both semantics and feasibility

Engineering learning is organized around connected capabilities rather than a simple list of courses. A student may need to understand circuit behavior before instrumenting a prototype, use a calibrated measurement before interpreting a control-system response, and demonstrate safe practice before accessing a physical laboratory. These relationships can cross courses, software environments, equipment, and academic terms. Accreditation criteria also expect programmes to define and assess student outcomes in relation to a curriculum, but they do not prescribe a single course sequence or a universal interpretation of every local outcome [1]. Competency frameworks such as 1EdTech CASE provide structured means to exchange standards, competencies, associations, and rubric information [2]. The local programme must still interpret each statement and decide what evidence shows readiness.

A knowledge graph can connect standards, course outcomes, concepts, activities, assessment criteria, resources, and evidence with explicit relation types. A learner pathway can then be grounded in the same graph as the programme map. However, a graph path only establishes that links exist. It does not automatically verify that a learner meets a prerequisite, that a laboratory resource is available at the required time, or that a safety condition holds. Conversely, a constraint solver can enforce formal conditions but needs a well-defined model whose educational meanings and versions are inspectable. The proposed engine uses the graph to preserve semantics and provenance, and constraint programming to determine whether a sequence satisfies declared rules.

Existing learning-path work has used knowledge graphs to connect topics, learners, resources, and context [3]. Constraint-based planning has also been applied to sequence tasks and schedules in e-learning [4]. These approaches motivate a system that goes beyond similarity ranking: the pathway should be feasible under explicit conditions, and the user should be able to inspect why it is recommended. This article presents a technical design, not an efficacy study. It specifies an ontology profile, compilation rules, a constraint model, explanation records, governance controls, and a future validation plan.

1.2. Aim and contributions

The aim is to specify a standards-grounded engine that generates engineering learning pathways which are both constraint-feasible and explainable. The main contributions are: a layered graph model for programme standards, educational tasks, resources, assessments, learner evidence, and provenance; a graph-to-constraint compiler that preserves versions and rule identifiers; a separation between mandatory conditions and learner or educator preferences; a path-level explanation trace tied to the same constraints used by the solver; an explicit infeasibility protocol that does not weaken safety or access requirements; and a synthetic engineering example with a testable implementation profile. The output is a candidate pathway and rationale, not a grade, admission decision, or automatic declaration of competence.

1.3. Scope and design principles

The design covers pathway planning for an individual learner within a governed

© 2026 TRTEE. Open Access under CC BY 4.0

programme or course cluster. It assumes that authorized educators have supplied the competency definitions, prerequisite relations, valid assessment rules, and operational constraints. It does not infer competence from clicks, attendance, or resource consumption. It does not decide whether two institutions teach equivalent outcomes, replace formal assessment, or establish professional licensure. The learner-state model stores only the claims needed for the present planning purpose, with provenance and uncertainty. A pathway is an editable proposal: learners can inspect, decline, or request a different option; educators can correct graph content or review exceptions; and changes produce a new versioned plan.

Four principles guide the design. First, semantic claims and operational rules are explicit and attributable to a source. Second, hard requirements are distinguished from preferences so that optimization cannot trade away safety, accessibility, or a required outcome. Third, evidence status is conservative: missing or stale evidence is unknown until reviewed, not a negative judgement. Fourth, explanations are generated from the actual graph facts and solver trace, rather than from a separately trained model that might invent a reason.

2. Technical and educational foundations

2.1. Knowledge graphs for curriculum semantics

A knowledge graph represents entities and typed relations in a form that can be queried, validated, and linked across sources [5]. For curriculum use, this representation makes distinctions that are often lost in flat course catalogs. A competency is not identical to a course; a prerequisite is not merely a topic similarity; a rubric criterion is not an assessment artifact; and an evidence record is not a claim that the evidence is current. The graph therefore stores these as different node classes and relations, each with an identifier, issuer or steward, version, effective interval, and provenance.

RDF provides a graph data model and named resources that can be shared across systems [6]. OWL 2 can express class relationships and logical axioms [7], while SHACL can validate that graph data meet declared shapes, cardinalities, datatypes, and value constraints [8]. These technologies serve different purposes. OWL-style entailment can classify or infer relationships under an ontology’s semantics; SHACL validation can report that a required field or permitted value is missing. Neither alone is a pathway planner. The proposed profile uses graph semantics to identify candidates and validate records, then translates the relevant subset into finite-domain constraints.

2.2. Competency standards and local interpretation

Engineering programmes map student outcomes to courses and use direct and indirect measures as appropriate for assessment [1]. Such standards support systematic programme design, but the engine must not treat an accreditation outcome as a complete executable lesson specification. A programme-level outcome such as designing a solution to meet specified needs must be decomposed locally into assessable capabilities, prerequisite knowledge, tasks, contexts, and evidence rules. CASE can carry competency frameworks and associations in an exchangeable form [2], but it cannot determine that a local rubric is pedagogically adequate or that a specific student has met the outcome. Mapping approval remains an academic governance activity.

© 2026 TRTEE. Open Access under CC BY 4.0

Every mapping record therefore states its source standard and version, local outcome, relation type, rationale, approving role, effective period, and evidence expectations. Relation types distinguish exact adoption, local decomposition, prerequisite support, contextual refinement, and proposed alignment. The system does not interpret lexical similarity as equivalence. A changed standard or rubric marks dependent mappings for review, and a pathway compiled against a withdrawn version is not silently presented as current.

2.3. Constraint programming for sequence generation

Constraint programming represents decisions by variables, domains, and rules for admissible assignments. A solver searches for feasible or optimal configurations. Google OR-Tools CP-SAT uses integer decision variables and constraints, but permits floating-point objective coefficients through internal scaling [9]. Scaling precision and optimality bounds require review. This profile uses declared integer time units and preference coefficients to support auditable priorities. The solver is replaceable only if model semantics are preserved. Record the model and solver status and link explanations to stable constraint identifiers rather than implementation indices.
Pathway construction is a scheduling and planning problem. Candidate activities have selection decisions, positions, durations, resources, outcomes, and prerequisites. Existing verified evidence can discharge a prerequisite; a selected earlier activity alone cannot. An approved earlier assessment may support a conditional plan, but dependent access requires a successful, valid claim at execution time. Safety, resource availability, and access requirements remain hard conditions. Preference optimization ranks admissible plans, for example by reducing delay or respecting modality preferences. A preference score cannot compensate for a violated hard condition.

2.4. Explanation in constraint solving

A useful explanation should expose the derivation that matters to a user. Research on step-wise explanations for constraint satisfaction problems formalizes how simple inference steps can be selected to make a solution trace understandable [10]. Work on unsatisfiable subset optimization explores how a concise conflict can explain why a constraint problem has no solution [11]. In an educational pathway, this suggests two complementary outputs: a positive explanation that links a selected activity to the target outcome and its prerequisites, and a diagnostic explanation that identifies a validated conflict set, when available, after infeasibility is established. Explanations should be faithful to the compiled model and should not reveal learner information that the current planning task does not require.

3. Requirements and assurance model

The system has six actors: curriculum stewards who manage standards and mappings; instructors who author activities and evidence rules; learners who review and select a pathway; an evidence service that supplies purpose-limited learner claims; a graph and policy registry that provides signed versions; and a pathway service that compiles and solves each request (Figure 1). The protected assets include learner identity, assessment

© 2026 TRTEE. Open Access under CC BY 4.0

details, accommodation information, learning history, institution-only curriculum material, safety rules, and mapping approval records. The service should receive a pseudonymous learner key where possible and return a learner-specific plan only to authorized interfaces. The assurance model assumes that a source record has been reviewed under the institution’s assessment process, that graph maintainers have authorization to publish changes, and that the solver implementation is tested against the stated model semantics. These assumptions do not remove the need for local review. A wrong prerequisite, incomplete safety gate, or biased resource map can produce a formally valid but educationally inappropriate plan. The service therefore logs versions and rule identifiers, provides correction routes, and separates automated feasibility from educator judgment (Table 1).

Table 1. Functional Requirements and Acceptance Conditions for the Reference Engine

ID	Requirement	Acceptance condition
R1	Preserve the meaning and version of standards, mappings, and local outcomes.	Every generated plan references resolvable graph and policy versions.
R2	Use evidence claims with provenance and validity.	No mastery state is created from exposure, attendance, or unreviewed inference.
R3	Separate hard constraints from preferences.	No objective weight can relax a hard prerequisite, safety, or accessibility rule.
R4	Return a feasible ordered pathway or an explicit failure state.	Solver status, candidate set, and unmet conditions are retained.
R5	Explain selections and exclusions using model facts.	Each explanation item maps to graph facts or a named constraint.
R6	Support learner and educator correction.	Users can report incorrect evidence or mapping and request a new plan.
R7	Minimize learner data use.	Only evidence necessary for the declared goal is loaded for compilation.

3.1. Threats and failure states

The principal technical threat is semantic or operational corruption. A stale standard version can make a pathway appear aligned when it is not; an incorrect prerequisite can omit needed preparation; a malicious or mistaken resource record can hide a safety condition; and incomplete evidence can be misread as failure. Other threats include unauthorized access to learner-state data, reconstruction of sensitive attributes from detailed explanations, accidental disclosure of accommodation status, solver timeout presented as proof of infeasibility, and preference manipulation that steers learners toward a provider or modality. The engine mitigates these risks through provenance, role-based publication, data minimization, explicit solver statuses, safe failure behavior, and human review (Table 2).

© 2026 TRTEE. Open Access under CC BY 4.0

Table 2. Failure States and Required System Responses

Condition	Meaning	Required response
Unknown evidence	The required claim is absent or not current.	Ask for review or a defined assessment; do not label the learner unqualified.
Ambiguous mapping	The target outcome has more than one unresolved interpretation.	Pause compilation and refer the mapping to its steward.
Infeasible model	No assignment satisfies the active hard constraints.	Return a conflict set if available; preserve hard gates.
Search limit reached	Search stopped without a final optimality or infeasibility proof.	Keep a feasible incumbent; otherwise report UNKNOWN.
Stale graph version	A dependency was withdrawn or changed after compilation.	Invalidate the plan and recompile from an approved version.
Missing access data	Required access constraints are unavailable or unconfirmed.	Offer a planning conversation; never infer an accommodation or remove access support.

4. Reference architecture and graph profile

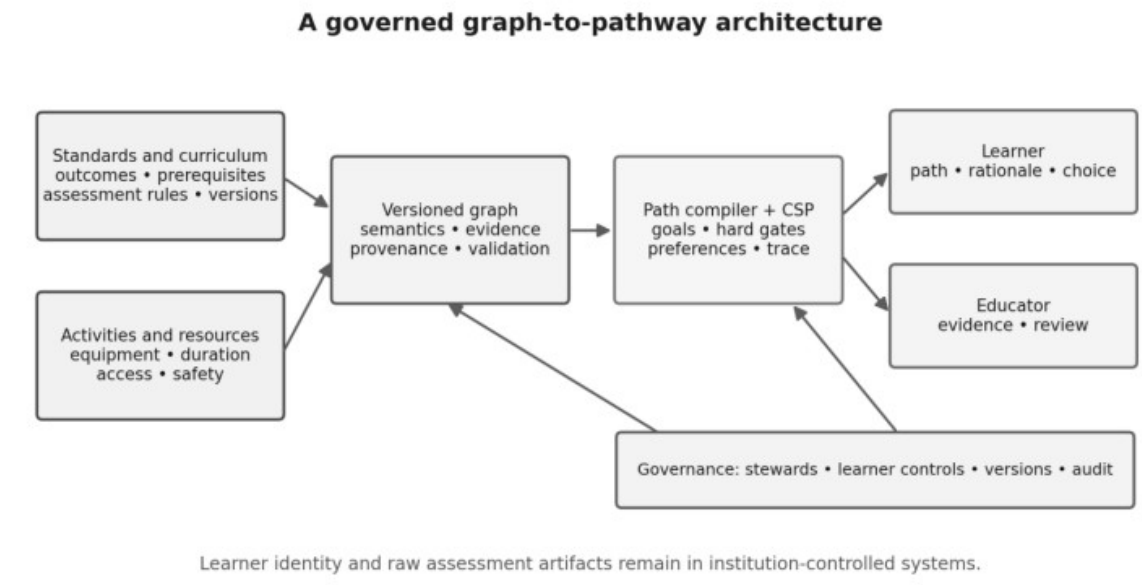


Figure 1. System Boundaries and Trust Roles

4.1. Graph layers and typed relations

The reference graph has six linked layers. The standards layer stores external framework statements and their stable identifiers. The programme layer stores local outcomes, course

© 2026 TRTEE. Open Access under CC BY 4.0

units, prerequisites, and approved mappings. The activity layer describes learning tasks, resources, modalities, durations, equipment, locations, and scheduling windows. The assessment layer links criteria to evidence rules and assessment events. The learner-evidence layer contains minimal claims such as “criterion met,” “assessment pending,” or “valid through date,” linked to an evidence source that may remain outside the planning service. The governance layer records provenance, version, approvals, effective periods, access policy, and deprecation links. These layers can be represented as named graphs or equivalent versioned partitions (Table 3).

Table 3. Core Graph Entities and Relations in the Proposed Profile

Entity or relation	Required attributes	Planning use
Competency outcome	Identifier, statement, framework/version, steward.	Defines the goal and its approved decomposition.
Prerequisite relation	Source, target, type, strength, rationale, validity.	Creates ordering or evidence constraints.
Learning activity	Outcome contribution, duration, modality, context, version.	Provides candidate decisions for the path.
Assessment criterion	Criterion ID, rubric level, method, validity rule.	Defines what counts as evidence for a claim.
Resource and session	Capacity, time, location, equipment, access features.	Constrains scheduling and feasible delivery.
Learner evidence claim	Target, status, provenance, date, expiry, confidence class.	Satisfies a prerequisite when policy permits.
Mapping and provenance	Source, target, relation, agent, approval, timestamp.	Explains alignment and supports change impact.

A learner evidence claim is deliberately narrower than a learner profile. It states which approved assessment criterion was met, by what source and under which valid assessment rule. The planning service does not need a video, raw sensor trace, full rubric comments, or the student’s name to test whether an accepted criterion is current. Evidence can carry a status such as verified, provisional, expired, disputed, or unknown. Only verified claims satisfying the policy may discharge a hard prerequisite; a provisional claim can trigger a review state but not a safety-critical authorization.

4.2. Validation, provenance, and versioning

Graph records are checked before publication. SHACL shapes can require that each activity have a stable identifier, owner, duration, modality, outcome link, and accessibility declaration; that each prerequisite reference existing outcome versions; and that every safety-critical task name its approved gate and assessor role [8]. Validation reports identify the focus node and violated shape. A data-valid graph is not necessarily pedagogically valid, so a curriculum steward separately reviews whether the relationships and thresholds are appropriate.

The engine records provenance for graph statements, mappings, assessment rules, and derived plan elements. PROV-O provides a standard vocabulary for describing entities, activities, agents, derivation, and attribution [12]. The plan record stores the source graph digest, framework versions, learner-state snapshot identifier, compiler version, solver

© 2026 TRTEE. Open Access under CC BY 4.0

version, constraint-set digest, objective configuration, creation time, and output digest. Corrections link new versions to prior records for audit and invalidation. Exact replay also needs retained input artifacts and the solver settings and conditions in Section 5.5.
Versioning is dependency-aware. If a framework statement changes, the registry identifies local mappings that depend on it. If a rubric or safety gate changes, affected evidence claims and pathway templates are marked for reassessment by the responsible steward. The engine distinguishes a backward-compatible metadata correction from a semantic change. A semantic change invalidates compiled paths that relied on the old meaning; the user interface shows the reason for re-planning rather than silently replacing the path.

4.3. Standards-to-goal compilation

A request begins with a readable educational goal, such as “prepare for supervised microgrid commissioning,” and a machine-readable goal identifier approved by the programme. The goal resolver follows only active mappings, prerequisite edges, and outcome decompositions for the selected version. It rejects unresolved equivalence links, cycles that have no approved interpretation, and targets without assessable evidence rules. A goal can resolve to several alternative sub-outcomes, but the alternatives and their semantics must be stated by the curriculum steward. The compiler records every graph edge traversed so a reviewer can inspect why an outcome entered the target set.
The mapping compiler does not convert every ontology axiom into a solver rule. It selects a controlled rule subset: prerequisite relations, required outcomes, accepted evidence states, task coverage, resource constraints, safety conditions, and explicitly approved exceptions. Richer semantic inferences remain visible as graph facts until a steward approves their operational meaning. This prevents an entailment such as broad subclass membership from becoming an unintended prerequisite or qualification decision.

4.4. Learner state and evidence handling

Learner state is loaded for the planning purpose and bounded to the required outcomes. Each outcome has a state, valid interval, source, and evidence rule version. The state categories are satisfied, not yet evidenced, expired, disputed, and unknown. “Not yet evidenced” is not equivalent to “unable.” The pathway compiler can recommend an assessment or practice activity when evidence is missing, but it cannot assert mastery until the published assessor rule is met. If a learner disputes a record, the service marks it disputed and routes the case according to institutional policy.
An optional confidence value is allowed only when its interpretation and calibration are defined by the originating assessment process. It cannot be invented from engagement telemetry or treated as a substitute for a rubric decision. The reference profile therefore uses categorical evidence states by default. Fine-grained assessment evidence stays at source and is retrieved only by an authorized reviewer. The pathway interface explains which evidence claim was used without exposing unrelated course history. This data-minimization approach follows established privacy and ethical principles for learning analytics and should be assessed against a documented institutional privacy-risk process [15–17].

5. Pathway compilation and constraint model

© 2026 TRTEE. Open Access under CC BY 4.0

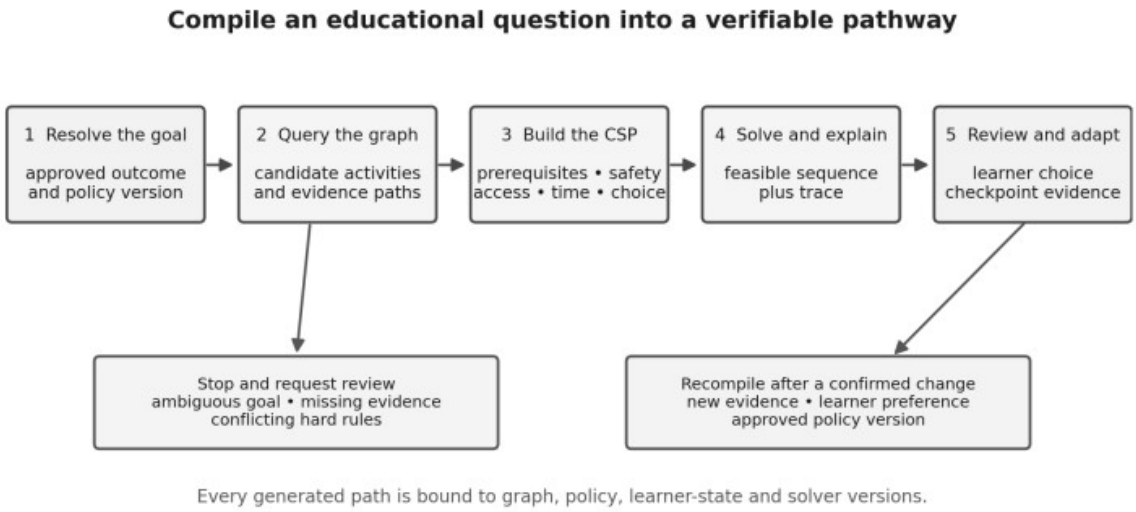


Figure 2. Compiler Pipeline

5.1. Planning variables and domains

Let G = (V, E) be the versioned graph. For request r, derive target outcomes H, candidate activities A, and evidence snapshot L (Figure 2). Binary x(a) selects activity a; p(a) assigns distinct sequence positions to selected activities, and s(a) may identify a session. Distinct positions do not prevent time overlap: selected intervals requiring learner attendance must not overlap and must allow required travel or transition time. Session times also respect durations and precedence. Domains use active records and the planning horizon. Variables y(h) denote planned outcome coverage and q(c) select assessment checkpoints. Unselected activities impose no precedence or capacity use. Coverage remains conditional on successful assessment, not a verified learner claim.

A path is an ordered sequence P = [a1, a2, …, an] of active activities supporting H. Existing verified evidence or approved earlier checkpoints can support planned prerequisite coverage. Selecting an activity alone does not establish mastery: each unmet outcome needs an assessment rule and checkpoint. A plan with pending assessments is conditional. Before dependent activity access, the required evidence must be verified and valid for that activity; failure or expiry triggers review and re-planning.

5.2. Hard constraints

Hard constraints define admissible plans. Required outcomes need verified evidence or planned assessment checkpoints. Strict prerequisites require accepted prior evidence or an earlier assessment whose success remains a runtime gate. Safety-critical tasks require the authorized safety claim to cover the task time and scope, with required assessor and equipment conditions rechecked before access. Sessions must respect availability, existing bookings, capacity, location, and equipment limits. A single-learner request uses remaining capacity, and a reservation must be confirmed against concurrent bookings. Required accessibility conditions exclude incompatible options before optimization. Every plan must respect its time horizon and non-negotiable programme rules. These constraints

© 2026 TRTEE. Open Access under CC BY 4.0

cannot be assigned soft penalties.
Hard conditions derive from approved graph statements and request-scoped operational facts. Each safety rule records its identifier, version, scope, and accountable approver. Prerequisite types retain their approved meaning: strict prior evidence imposes precedence; mandatory concurrent support imposes co-selection and timing constraints; recommended preparation remains optional. A concurrent requirement is not discarded merely because it is not a strict predecessor. If a required record is missing, compilation stops for the affected activity and reports what must be confirmed rather than guessing an interpretation.

5.3. Soft preferences and objective design

Hard goals and operational conditions define the feasible set F; they are never objective terms. Within F, declared preferences, such as modality, continuity, and preparation order, precede avoidable time or resource cost. Sequential lexicographic solving may fix a higher-priority value only after exact optimality is proved. A gap-qualified or interrupted status alone does not establish exact priority optimality; retain the bounds. An equivalent weighted objective requires published integer objective bounds and weights whose dominance over all lower priorities is proved, with overflow checked. The user can inspect the priorities and request alternatives, but a different feasible path is not guaranteed. Preferences cannot make an inadmissible plan acceptable.
The system does not optimize a proxy such as predicted grade, engagement, or probability of completion as a default. Such targets can reproduce access differences or reward platform behavior rather than learning. If a programme later proposes a predictive objective, it requires separate approval, evidence that the target is educationally defensible, subgroup evaluation, and a non-predictive alternative. In the reference design, path quality is assessed through feasibility, transparent preference satisfaction, and educator review rather than by a learned score.

5.4. Formal model and solver status

Let M = (X, D, C_h, C_s, J), with variables X, joint domain D (the product of variable domains), hard constraints C_h, soft predicates C_s, and ordered objective J [13, 14]. Then F = {z in D : c(z) is true for every c in C_h}. For optimization, FEASIBLE means an incumbent, not proven optimality; OPTIMAL is interpreted with recorded solver tolerances [9]. INFEASIBLE requires proof; UNKNOWN means neither a solution nor infeasibility was established; MODEL_INVALID denotes an invalid model. A time limit may leave a feasible incumbent. Store status, objective value and bound when available, solver tolerances, and termination reason.
CP-SAT uses integer and Boolean variables [9]. Time units and preference scales must be declared; use exact integer scaling where possible. Any conservative rounding must not shorten required durations, advance release times, or extend deadlines. It may discard feasible continuous schedules, so report this restriction and check returned plans against the original times. Boundary tests must cover sessions and evidence expiry. A replacement solver must preserve these constraints and the explanation trace (Table 4), without redefining competency or assessment rules.

Table 4. Illustrative Rule Translation from Graph Facts into Constraint Predicates

© 2026 TRTEE. Open Access under CC BY 4.0

Graph fact	Compiled predicate	Classification
Activity A requires prior evidence for outcome H.	x(A) implies valid prior H or a gated earlier H assessment.	Hard
Activity B requires current electrical-safety claim.	x(B) implies valid safety at B's time and scope; recheck access.	Hard
Session S has capacity k and resource R.	Bookings plus this request ≤ k; reserve R exclusively per slot.	Hard
Learner prefers virtual preparation first.	Prefer virtual-before-lab among feasible paths, as a soft objective.	Soft preference
Two activities provide an approved equivalent practice route.	If required and not evidenced, select an approved route.	Hard choice rule
Learner prefers fewer campus visits.	Minimize visit count after all higher-ranked goals are met.	Soft preference

5.5. Compilation algorithm

Compilation should be deterministic for a fixed request and version bundle. Record candidate bounds, exclusions, and the horizon (Table 5). INFEASIBLE applies to this compiled model; it cannot rule out routes omitted by candidate generation. Exact path replay requires retained inputs, fixed solver settings, stopping rules, and tie-breaking; a seed alone is insufficient. Record seed, workers, limits, version, and status. Nondeterministic or wall-clock-limited runs may return different paths and objective values. Validate each path and report its objective and bound where available. Do not label alternatives equivalent without checking the relevant criteria.

Table 5. Reference Compilation Algorithm

Step	Operation	Output or stop condition
1	Resolve goal, framework version, purpose, and policy.	Stop if mapping is missing, inactive, or ambiguous.
2	Load minimum learner evidence and approved access constraints.	Stop for confirmation if a required fact is unavailable.
3	Query bounded graph neighborhood for candidate activities and assessments.	Record candidates, exclusions, and graph provenance.
4	Validate entities, evidence rules, safety gates, and resource records.	Stop on shape error or expired rule.
5	Compile hard constraints and ordered preference objectives.	Store model digest and stable constraint IDs.
6	Solve under configured limit and capture exact solver status.	Record status and termination reason separately.
7	Generate explanation from graph facts and solver trace.	Validate each claim against its evidence and model scope.

© 2026 TRTEE. Open Access under CC BY 4.0

Step	Operation	Output or stop condition
8	Present plan and alternatives for learner or educator review.	Recompile only after a declared input or version change.

6. Explanation and failure diagnosis

6.1. A pathway explanation tied to evidence

Each selected activity needs a record of its target, prerequisites, accepted evidence, pending assessments, and placement constraints. Distinguish verified readiness from a conditional plan. An expired safety claim explains a gate; a workshop is mandatory only if the approved rule requires that route. A remote-first objective may favor virtual preparation but does not itself prove why an order was selected. Report objective values and ties. Changing only a soft preference preserves F and may return the same path; a different path requires a verified alternative. Counterfactual necessity claims need a checked derivation or re-solve under stated assumptions.

Each explanation sentence is assembled from a controlled template and a trace element with a graph node or constraint identifier. A language model may be added later as a presentation layer only if it is restricted to paraphrasing validated trace records, the source facts remain accessible, and an automated checker rejects unsupported statements. The language model must not create prerequisites, infer mastery, change constraint severity, or choose a pathway. The reference implementation can use template-based text, which is easier to validate and compare against the formal model.

6.2. Explanations for excluded alternatives

Not selected does not mean infeasible. Distinguish candidates filtered by an approved rule, alternatives proven infeasible, feasible paths ranked lower or tied, and options not assessed within the search bounds. State a factual exclusion only when its rule and inputs establish it. A claim that an activity cannot occur in any feasible path needs a valid derivation or proof with that activity required; a timeout is not proof. Compare objectives only for verified feasible alternatives. Provide role-appropriate evidence and correction routes without exposing peers' schedules, hidden attributes, or private assessor notes.

6.3. Infeasible plans and conflict sets

After infeasibility is proven, a diagnostic procedure may return a conflict set with source identifiers. A sufficient core establishes a conflict relative to the stated background constraints; it need not be minimal. A subset-minimal unsatisfiable set has no unsatisfiable proper subset. A minimum-cardinality set has the fewest members; a cost-optimal set minimizes a declared cost [11]. These are distinct guarantees. Label only the property actually verified, and state when diagnosis is unavailable or incomplete. For example, a required supervised assessment and no available eligible session may conflict within a fixed horizon. A timeout is never proof of infeasibility.

Remedies may change a horizon, session, assessment record, or approved mapping; recompile after review. Changing only a soft preference leaves F unchanged and cannot cure infeasibility. Removing a safety gate, lowering an assessment threshold, or treating unknown evidence as satisfied is

© 2026 TRTEE. Open Access under CC BY 4.0

never an automatic remedy. Any policy change requires its normal approval, a new version, and a fresh compilation. This keeps explanation and recourse connected to accountable educational decisions.

7. Synthetic engineering pathway example

7.1. Goal and graph instance

Consider a fictional microgrid-commissioning goal requiring circuit analysis, sensor calibration, controller programming, integration testing, and current electrical-safety practice. The graph offers circuit and controller modules, virtual calibration practice, a supervised bench, an integration lab, and a capstone checkpoint. The learner has current accepted circuit and controller evidence, expired safety evidence, and no calibration claim. For this example, assume the approved safety route requires a refresher and a passed assessment before bench access. The learner prefers remote preparation and has one accessible bench session in two weeks (Table 6). The proposed sequence remains conditional on successful assessments and a compatible schedule; the table alone does not establish timed feasibility.

Table 6. Synthetic Graph Facts and Learner-State Inputs Used in the Example

Item	Versioned fact or state	Planning consequence
Target	Supervised microgrid commissioning, programme map v3.2.	Requires five approved outcome claims.
Circuit analysis	Verified assessment claim, valid through next year.	No repeat activity required.
Controller programming	Verified lab-simulation assessment.	Satisfies the controller prerequisite.
Electrical safety	Claim expired; supervised bench requires current claim.	Safety refresher and assessment must precede the bench.
Sensor calibration	No current evidence; virtual exercise and bench assessment available.	Practice and an assessment checkpoint are included.
Availability	One accessible bench session in the next two weeks.	The path must reserve its capacity and meet the time window.

7.2. Feasible path and explanation

The compiler first resolves the target into the five required outcome predicates and checks which are already satisfied. It removes circuit-analysis and controller-programming activities from the required set because the relevant evidence is current and accepted by policy. It then includes the electrical-safety refresher and assessment, which can produce a claim only after a pass; the virtual sensor-calibration practice; the supervised sensor bench and its assessment; and the microgrid integration lab with its commissioning checkpoint. Precedence rules place the safety assessment before any physical bench access and place

© 2026 TRTEE. Open Access under CC BY 4.0

calibration practice before the supervised measurement task. The available lab session is reserved only after those conditions are satisfied.
The learner-facing rationale identifies the verified claims that shortened the path, explains why the safety refresher is required, and shows the remaining assessment checkpoints. It states that virtual calibration practice honors the learner’s preference, while the bench remains mandatory because the target includes a supervised physical measurement criterion. It does not state that the learner has mastered calibration before the assessment. An alternative route can be shown if an additional accessible bench session or an approved equivalent assessment becomes available; that route is recompiled from new operational facts rather than inferred from the current model.

7.3. Infeasibility example

If the accessible bench session is cancelled and the planning horizon cannot be extended, the hard model becomes infeasible. The service reports that the target cannot be completed within the selected two-week window because the required supervised measurement assessment has no available session after the safety gate. It may offer a later horizon, a different accessible session, or human review of an approved equivalent assessment. It cannot replace the bench with the virtual exercise because the programme’s current mapping requires supervised physical evidence for this goal. The diagnostic is a planning explanation, not a judgement about learner ability.
This scenario contains no measured performance, student record, or implementation result. It illustrates how graph semantics, evidence state, hard constraints, and a learner preference interact under the design. The example is deliberately small so that a reviewer can manually inspect each generated rule and proposed action.

8. Implementation and governance profile

8.1. Service contract and interfaces

A pathway request contains a goal identifier and version, purpose, planning horizon, learner-approved preferences, accessibility constraints needed for the request, and a reference to the minimal evidence snapshot. The response contains a solver status, ordered activities, required checkpoints, a learner-readable explanation, alternative feasible paths where requested, graph and policy digests, and a correction route. Internal diagnostic fields include stable constraint identifiers, solver version and status, candidate exclusions, and provenance. The interface should not expose technical constraint syntax to learners by default, but each explanation must have a route to its source and a human contact for disputed records.
The graph service and planning service should use least-privilege access. Learner identifiers are removed or replaced by a request-scoped token before compilation when identity is not needed. Accommodation constraints are treated as sensitive input; the output states that a selected option meets the submitted access requirement without revealing a diagnosis. Logs retain only the fields needed to reproduce or audit a plan under the institution’s retention policy. Exported plans include a human-readable version and a machine-readable record for the learning platform or advising system.

© 2026 TRTEE. Open Access under CC BY 4.0

8.2. Interoperability and standards profile

The interoperability layer can export competency frameworks and associations using CASE where the source and target systems support the relevant specification [2]. RDF provides a general graph interchange model [6]; OWL 2 can support ontology-level class and relation semantics [7]; SHACL can validate local graph profiles [8]; and PROV-O can describe plan and mapping derivation [12]. These standards do not guarantee identical local meaning. Each export therefore includes a profile identifier, framework version, mapping relation, and conformance report. A receiving system must not treat an imported relation as an approved hard prerequisite without local authorization.

8.3. Human authority and change management

Curriculum stewards approve standards mappings and their effective dates. Instructors or assessment committees approve activity-to-outcome relations and evidence rules. Laboratory safety officers approve safety gates, equipment conditions, and exception pathways. Learners control stated preferences and can request correction of evidence or planning inputs. A system administrator manages technical release but cannot unilaterally change educational rules. This separation places authority with the role that owns each decision and gives the plan a clear route for review.

Every semantic change is tested against dependent pathways and assessed for impact before publication. A safety-rule change invalidates paths that used the prior gate. A metadata correction may preserve semantic validity but still updates provenance. A model update includes a migration report that identifies affected activities, goal mappings, evidence claims, and user-facing pathway templates. Rollback restores the prior approved version only when its rules remain valid. Historical plans retain the version that actually produced them and display whether that version is now superseded.

9. Validation and evaluation plan

The first implementation should be evaluated as a technical and educational design before any learner-facing use. Tests are run on synthetic graphs and records initially, followed by expert review of a bounded programme map. This article reports the test plan only and claims no validation result. The test suite must verify the graph profile, compiler transformations, solver behavior, explanation trace, privacy boundary, and user recourse as separate properties (Table 7). Passing software tests does not establish that the curriculum mapping is educationally appropriate; that requires review by authorized educators and affected learners.

9.1. Graph and compiler conformance

Conformance tests check required identifiers, version intervals, provenance links, relation types, cardinality, allowed values, active status, and deprecation behavior. Negative fixtures include an unapproved mapping, an expired safety rule, a missing assessment threshold, a prerequisite cycle without an approved resolution, and two standards versions with identical labels but changed meaning. Compare candidate sets and constraint identifiers with hand-authored cases. Define each edge mutation's expected semantic effect; a changed edge need not change the selected path. Check feasible sets, objectives, and provenance;

© 2026 TRTEE. Open Access under CC BY 4.0

check unaffected model semantics separately from solver tie-breaking.

9.2. Solver and explanation fidelity

Planned tests cover time overlap, travel, rounding, capacity, expiry, failed checkpoints, preference ties, candidate truncation, and interruption with or without an incumbent. Enumerate small cases independently to check F and objective priorities. An irrelevant resource must preserve the projected feasible set and objective; identical paths also need fixed tie-breaking and reproducible search. Removing an assessment causes infeasibility only without valid evidence or an approved alternative. A soft-preference change preserves F and may preserve the selected path. Test that interrupted higher-priority optimization is not labeled exact lexicographic optimality. Validate each explanation's derivation and scope; reject unsupported exclusions and necessity claims.

9.3. Educational validity, access, and privacy

Curriculum review asks whether outcome decompositions, prerequisites, assessments, and equivalent routes match the programme’s intended progression. Learner review asks whether a path is understandable, whether the explanation distinguishes missing evidence from lack of ability, and whether the correction and appeal route is usable. Accessibility review checks that required access preferences are respected in candidate generation and that missing accommodation data prompts a conversation rather than silent exclusion. Interfaces and user-facing explanations should also be checked against applicable accessibility criteria such as WCAG 2.2 [18]. Privacy review examines whether the planner loads or logs more learner information than the stated goal requires and whether role-based views expose sensitive attributes in explanations [15–17].

Table 7. Proposed Pre-Deployment Evaluation Gates (Planned Checks)

Dimension	Evidence to collect	Gate before pilot
Graph integrity	Shape reports, dependency and version tests.	No unresolved semantic or provenance defect.
Compiler correctness	Reference fixtures and mutation tests.	Rules match approved graph semantics.
Solver correctness	Enumerated small cases and status handling.	No false feasibility or false infeasibility in test set.
Explanation fidelity	Trace-to-text audit by technical and curriculum reviewers.	Claims justified by evidence; no unsupported exclusions.
Educational validity	Curriculum steward and learner review.	Targets, evidence, and recourse are accepted for the use.
Accessibility	Accessible task and exception-route scenarios.	No hidden removal of stated access support.
Privacy	Data-flow, log, and role-access review.	Only purpose-required evidence is processed and retained.
Operations	Version withdrawal, rollback, and timeout exercises.	Safe fallback and responsible human contact are available.

9.4. Staged deployment

© 2026 TRTEE. Open Access under CC BY 4.0

A staged process begins with synthetic data and hand-checkable pathway examples. The next stage imports a small, steward-reviewed set of standards and activities without learner evidence and tests version propagation. A shadow mode then compares proposed paths with expert-designed plans without displaying automated recommendations to learners. If the explanations and failure handling pass review, an opt-in low-stakes pilot can present pathways with a visible educator route. Expansion requires fresh review of the curriculum map, accessibility, privacy, operational capacity, and learner feedback. A pilot does not authorize use for progression, grading, or safety clearance beyond the approved claim and assessment process.

10. Discussion

10.1. What the design makes inspectable

The central design choice is to keep curriculum semantics and combinatorial search connected but distinct. The graph says what outcomes, activities, evidence rules, and relationships mean. The constraint model says which combinations satisfy the current pathway request. The explanation trace connects a selected activity to both layers. This separation makes it possible to identify whether a problem comes from a curriculum mapping, a learner evidence state, resource availability, a policy decision, or solver behavior. It also supports replacement of the solver without losing the graph or explanation contract.

The design makes standards alignment reviewable rather than automatic. A standards identifier can establish where an outcome originated; it cannot establish that the local decomposition is complete. Provenance and version information show who approved a mapping and what it depended on. A correction can then be propagated to affected plans. This is particularly important in engineering education, where physical equipment, safety conditions, and accreditation evidence can change on different timelines.

10.2. Limits of optimization and personalization

Optimization can make a pathway internally consistent, but it cannot determine the best educational experience from a mathematical objective alone. The solver respects the objective that programme authors provide. If that objective rewards short duration, it may underweight useful practice; if it rewards predicted engagement, it may encode platform or access bias. The reference design therefore limits optimization to declared scheduling and learner preferences, keeps learning outcomes and safety as hard constraints, and makes objective priorities visible. A learner can choose another feasible path, and a qualified educator can review unusual cases.

Personalization also depends on the quality and scope of learner evidence. A sparse record does not imply low ability, and prior learning may be absent from the current institution’s system. The engine provides assessment and review routes instead of inferring a deficit. It also avoids global learner rankings and does not expose peer trajectories as a normative path. Any future use of predictive learner models would require a distinct technical and governance design with evidence that the prediction is valid for the decision and population.

© 2026 TRTEE. Open Access under CC BY 4.0

10.3. Limitations and open questions

This paper is a design specification. It has not been implemented, benchmarked, tested with a curriculum committee, or evaluated with learners. The proposed graph profile and compiler rules need refinement in a selected engineering programme. Constraint explanation can become expensive for large models, and minimal conflict extraction may not be supported uniformly by solvers. Large activity graphs require candidate-generation bounds and measured runtime. Interoperability standards can exchange structured competencies but cannot guarantee local pedagogical equivalence. These limitations define the next engineering tasks rather than evidence of readiness for deployment.

Future work should test whether explanations remain understandable when a pathway spans several terms or institutions; how accessible options are represented without exposing sensitive diagnoses; how uncertainty and disputed evidence should affect planning; how competency mappings should be retired when standards evolve; and whether solver traces can support an instructor’s curriculum audit without overwhelming users. An implementation should compare pathway outputs with independently authored plans and examine whether the system reduces avoidable scheduling conflicts while preserving learner choice. Such an evaluation should report methods and results before making claims about efficiency or learning improvement.

11. Conclusion

This article specifies a knowledge-graph and constraint-solving engine for explainable engineering learning pathways. The graph preserves standards, local outcomes, activities, assessments, evidence, resources, and provenance as distinct versioned entities. A compiler translates a reviewed subset of those semantics into a finite constraint model. The solver enforces prerequisites, safety, access, availability, and outcome coverage before optimizing declared preferences. The resulting pathway includes an evidence-linked rationale, and infeasible plans produce a bounded diagnostic rather than an unexplained denial.

The design treats formal feasibility as necessary but insufficient for educational validity. Curriculum stewards approve mappings, assessors own evidence rules, learners control preferences and can challenge inputs, and safety authorities retain control of required gates. The proposed conformance and evaluation plan provides a basis for implementation with synthetic and expert-reviewed cases. Until those steps are completed, the engine remains a technical design proposal, not a validated learning intervention.

© 2026 TRTEE. Open Access under CC BY 4.0

Acknowledgements

The authors thank curriculum and accreditation advisers for comments on the competency mapping and acknowledge institutional guidance on governance requirements; this work received no external financial support.

Author Contributions

Z.W.: Conceptualization, Ontology and graph schema design, Constraint model formulation, Software implementation, Evaluation, Visualization, Writing — Original Draft. Y.C.: Methodology, Formal verification of the constraint model, Validation, Supervision, Writing — Review and Editing. Both authors contributed to the design iterations and approved the submitted version.

Conflicts of Interest

The authors report no conflicts of interest.

Generative AI Statement

During the preparation of this article, ChatGPT (OpenAI) was used to refine the wording of the technical sections and to help prepare the pathway diagrams and tables. The authors independently specified the ontology, encoded the constraints, and checked every result; the tool produced no research data or conclusions. The authors are solely responsible for the final content.

© 2026 TRTEE. Open Access under CC BY 4.0

References

[1] ABET, “Criteria for Accrediting Engineering Programs, 2025–2026,” Engineering Accreditation Commission, 2025. https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-engineering-programs-2025-2026/
[2] 1EdTech Consortium, “Competencies and Academic Standards Exchange (CASE) Service, Version 1.1 Specification,” Final Release, 24 January 2025. https://www.imsglobal.org/spec/case/v1p1/
[3] S. Zhang, N. Hui, P. Zhai, J. Xu, L. Cao, and Q. Wang, “A Fine-Grained and Multi-Context-Aware Learning Path Recommendation Model over Knowledge Graphs for Online Learning Communities,” Information Processing & Management, vol. 60, no. 5, art. 103464, 2023. https://doi.org/10.1016/j.ipm.2023.103464
[4] A. Garrido, E. Onaindia, and O. Sapena, “Planning and Scheduling in an E-Learning Environment: A Constraint-Programming-Based Approach,” Engineering Applications of Artificial Intelligence, vol. 21, no. 5, pp. 733–743, 2008. https://doi.org/10.1016/j.engappai.2008.03.009
[5] A. Hogan et al., “Knowledge Graphs,” ACM Computing Surveys, vol. 54, no. 4, art. 71, pp. 1–37, 2021. https://doi.org/10.1145/3447772
[6] R. Cyganiak, D. Wood, and M. Lanthaler, eds., “RDF 1.1 Concepts and Abstract Syntax,” W3C Recommendation, 25 February 2014. https://www.w3.org/TR/rdf11-concepts/
[7] W3C OWL Working Group, “OWL 2 Web Ontology Language Document Overview, Second Edition,” W3C Recommendation, 11 December 2012. https://www.w3.org/TR/owl2-overview/
[8] H. Knublauch and D. Kontokostas, eds., “Shapes Constraint Language (SHACL),” W3C Recommendation, 20 July 2017. https://www.w3.org/TR/shacl/
[9] Google, "CP-SAT Model Protocol (cp_model.proto)," OR-Tools, v9.14. https://github.com/google/or-tools/blob/v9.14/ortools/sat/cp_model.proto
[10] B. Bogaerts, E. Gamba, and T. Guns, “A Framework for Step-Wise Explaining How to Solve Constraint Satisfaction Problems,” Artificial Intelligence, vol. 300, art. 103550, 2021. https://doi.org/10.1016/j.artint.2021.103550
[11] E. Gamba, B. Bogaerts, and T. Guns, “Efficiently Explaining CSPs with Unsatisfiable Subset Optimization,” in Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI-21), pp. 1381–1388, 2021. https://doi.org/10.24963/ijcai.2021/191
[12] T. Lebo, S. Sahoo, and D. McGuinness, eds., “PROV-O: The PROV Ontology,” W3C Recommendation, 30 April 2013. https://www.w3.org/TR/prov-o/
[13] R. Dechter, Constraint Processing. San Francisco, CA: Morgan Kaufmann, 2003.
[14] A. K. Mackworth, “Consistency in Networks of Relations,” Artificial Intelligence, vol. 8, no. 1, pp. 99–118, 1977. https://doi.org/10.1016/0004-3702(77)90007-8
[15] A. Pardo and G. Siemens, “Ethical and Privacy Principles for Learning Analytics,” British Journal of Educational Technology, vol. 45, no. 3, pp. 438–450, 2014. https://doi.org/10.1111/bjet.12152

© 2026 TRTEE. Open Access under CC BY 4.0

[16] National Institute of Standards and Technology, “NIST Privacy Framework: A Tool for Improving Privacy through Enterprise Risk Management, Version 1.0,” NIST, 2020. https://doi.org/10.6028/NIST.CSWP.01162020
[17] S. Slade and P. Prinsloo, “Learning Analytics: Ethical Issues and Dilemmas,” American Behavioral Scientist, vol. 57, no. 10, pp. 1510–1529, 2013. https://doi.org/10.1177/0002764213479366
[18] W3C, “Web Content Accessibility Guidelines (WCAG) 2.2,” W3C Recommendation, 5 October 2023. https://www.w3.org/TR/WCAG22/

© 2026 TRTEE. Open Access under CC BY 4.0