2026 European Conference on Computing in Construction
Corfu, Greece
July 12–15, 2026

A Semantic Framework for Formalizing Stakeholder Information Requests in Construction

Timur Weilbach-Eyüboglu, Cornelius Preidel, and Simon Vilgertshofer
Hochschule München University of Applied Sciences, Munich, Germany

Abstract

The construction industry relies on answering recurring questions, but required information is fragmented across heterogeneous data sources, including digital models, schedules, and documentation. This paper introduces a semantic request framework implementing stakeholder questions as information objects that capture templates, bindings, constraints, and provenance for re-execution and diagnostics. Reusable question templates derived from an exemplar construction process are aligned with established domain ontologies such as *DiCon*, *ifcOWL*, and *BOT*, then operationalized through parameterized graph query patterns over federated knowledge graphs. Instantiated requests bind templates to project instances, enabling evidence-based evaluation of readiness, approvals, constraints, and resource availability.

Introduction

Construction projects require stakeholders to obtain situation-specific information on demand to make informed decisions. Such information needs arise dynamically, depending on the current project state, the actor's role, and the decision context. Typical questions include *"Is this area ready for the next activity?"*, *"Are the required approvals available?"*, or *"Can this task be released under the current conditions?"*. While their wording varies, the underlying intent of these questions is highly recurrent: across projects and roles, stakeholders repeatedly ask structurally similar questions but expect context-specific, evidence-based answers.

These questions are central to production coordination and decision-making. In Lean Construction, they directly relate to constraint analysis, work packaging, and reliable promise-making (Ballard, 2020; Koskela, 2000). From a situation awareness perspective, they express the information required to perceive, comprehend, and anticipate the state of the work system (Endsley, 1995). As such, they determine whether work can be executed safely, correctly, and without disruption. Despite their importance, answering such questions remains difficult in practice. Applied research and industry-informed studies consistently report that the evidence required to assess process readiness or approval conditions is distributed across heterogeneous systems, maintained by different stakeholders, and represented in various formats (Sacks et al., 2020; Čustović et al., 2025). This observation is further reinforced by practice-driven analyses that we conducted in preparation for this paper in close collaboration with industrial partners, based on structured workshops and expert interviews, in which even routine operational inquiries were found to require manual coordination across BIM models, schedules, quality assurance records, safety documentation, and domain-specific planning tools. As no single system provides a unified representation of process dependencies and current states, situational awareness is fragmented, and decision-making becomes time-consuming and error-prone. The core challenge is therefore not a lack of data, but the absence of a mechanism that consistently translates recurring operational questions into verifiable, cross-system information requests. Although construction projects repeatedly pose the same types of questions, these questions are reformulated anew for each system and context, preventing systematic reuse and comparability of answers.

This paper addresses this gap by proposing a semantic request layer for recurring operational information needs in construction. The approach separates stable question semantics from project-specific bindings, allowing the same underlying request structure to be specified consistently across heterogeneous systems and reused across projects. Three contributions are made: first, a formal representation of recurring stakeholder questions as reusable request templates; second, a binding and constraint mechanism that links these templates to project-specific entities and evidence requirements; and third, a prototype design for operationalizing such requests over federated graph-based project data. Execution-time readiness and work-release decisions serve as the exemplar application. Automated natural-language interpretation is discussed as a prospective interface to this layer, but is treated as future work rather than part of the present evaluation.

Related Work

From a lean production-control perspective, questions such as whether work can be released or whether an area is ready for the next task belong to make-ready planning, constraint analysis, and reliable promise-making (Ballard, 2020; Ballard and Howell, 1998; Koskela, 2000). These decision situations depend on multiple prerequisites spanning workforce, information, materials, equipment, prerequisite work, safety, and external conditions.

At the same time, BIM and construction digital twin research consistently reports that the required evidence is distributed across domain tools, making the problem one of information access rather than data availability (Sacks et al., 2018, 2020; Čustović et al., 2025). Ontology-based approaches address this fragmentation by providing shared semantic foundations. Foundational ontologies such as *ifcOWL* (buildingSMART International, 2025) and the *Building Topology Ontology (BOT)* (Rasmussen et al., 2020) support graph-based representations of building elements and spatial structures. Workflow-oriented ontologies extend this scope toward construction processes, resources, states, and information objects, most notably the *DiCon* ontology suite (Zheng et al., 2021). Related work further specializes these ideas for production control, safety, and digital twin integration (Farghaly et al., 2024; Speiser et al., 2025; Schlenger et al., 2025). While these ontologies provide rich, machine-interpretable models of construction entities, processes, and states, their focus remains on the supply side of information modeling. They describe what data exists and how it can be linked. However, they are not designed to explicitly represent the structure of stakeholder questions or the decision logic for combining evidence from different domains. Competency questions used in ontology development primarily serve validation and illustrative purposes, but they are not formalized as reusable models of operational information needs (Zheng et al., 2021; Sacks et al., 2020). As a result, recurring process-oriented inquiries, such as readiness or approval checks, remain implicit and system-specific. In parallel, significant research has focused on improving information retrieval from BIM models. Early approaches introduced domain-specific query languages and visual query tools that operate directly on IFC or *ifcOWL* schemas, such as *BIMQL* (Mazairac and Beetz, 2013) and *QL4BIM* (Daum and Borrmann, 2014). These languages enable expressive, schema-aware queries for building models. However, they require users to be familiar with underlying data schemas and primarily address model-internal structures, geometric properties, and element relationships. More recent work increasingly uses natural language processing to translate user queries into formal graph queries. Ontology-aided semantic parsers and question-answering systems enable multi-constraint queries over BIM models for non-expert users (Yin et al., 2023; Nabavi et al., 2023). Recent work also explores LLM-based natural-language retrieval directly over IFC-encoded BIM models through agentic workflows (Hellin et al., 2025). These approaches improve access to BIM-centered information, but they generally focus on individual questions rather than on model data or closely related semantic extensions. By contrast, the present work formalizes recurring operational question types as reusable semantic request templates that can be instantiated across project contexts and resolved against federated cross-domain evidence. Thus, the contribution is not a direct NL-to-query interface, but an intermediate semantic request layer that such interfaces could use. To enable cross-system access, research on federated knowledge graphs proposes technical architectures for linking and querying heterogeneous, domain-specific graphs (Teclaw et al., 2024; Pauwels, 2014). Such federated approaches provide a robust infrastructure for cross-domain querying and align well with the digital twin vision. Nevertheless, they primarily address schema alignment and data availability. They do not specify how operational questions should be represented, nor how complex dependencies across federated sources can be captured in a reusable and decision-oriented manner. Users are still required to formulate explicit queries or rely on system-specific interfaces. Across these research streams, a common limitation becomes apparent. Existing ontologies, query systems, and federated graph architectures excel at modeling and accessing construction data, but they do not provide a structured semantic layer for representing recurring stakeholder questions. In particular, operational inquiries that combine spatial, temporal, procedural, and administrative dependencies, such as readiness to release work in lean production systems (Ballard and Howell, 1998), are not supported at this stage. This gap motivates the present work, which introduces a semantic request layer that formalizes recurring operational questions as reusable request templates, links them explicitly to distributed information dependencies, and connects lean work-release logic with existing ontologies and query infrastructures.

Methodology

We objectify stakeholder information needs by representing them as explicit *InformationRequest* entities in a semantic request layer, rather than ad-hoc *queries* across heterogeneous systems. Each request is a persistent, addressable object with an identifier and links to its *RequestTemplate*, bindings, constraints, and provenance, supporting reproducibility and auditability. We refer to (i) the natural-language question, (ii) its formal representation as an *InformationRequest*, and (iii) the corresponding graph-based retrieval specification (e.g., federated SPARQL). Because such questions recur in specific process situations (readiness, approval, compliance, coordination), we elicit reusable request types and associated constraints using an exemplar foundation process chain while keeping the framework process-agnostic; the foundation and readiness examples serve to instantiate the method, not to define its full scope. The methodology therefore proceeds from observed execution-time questions and process checkpoints to formal representations in a request layer that can be aligned with existing ontologies, such as *DiCon*, *ifcOWL*, *BOT*, and digital twin information models. Figure 1 summarizes how an *execution-time question* is mapped to a template, instantiated via placeholder bindings, and specified for graph-based retrieval over federated project data. Unlike NL-to-query approaches, the request layer specifies the decision condition and admissible evidence paths, not a single system-specific retrieval proce-

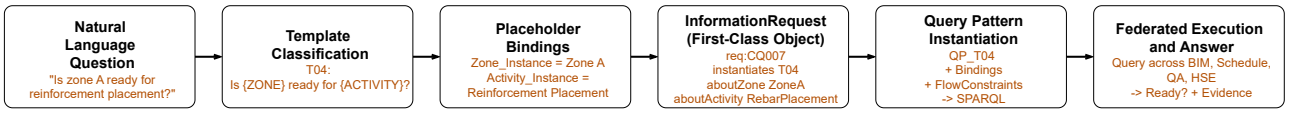

Figure 1: From a natural-language question to an operationalizable graph-based request via the proposed request framework

dure. A future natural-language front-end is envisioned as a candidate-generation layer rather than the locus of semantic interpretation itself. Its role is to rank likely request templates and propose bindings for typed placeholders (e.g., {Zone}, {Activity}, {Entity}) based on the user's utterance and the project context. The semantic request layer defined here then validates these candidates against ontology-based type constraints and admissible relation patterns before execution. Where multiple interpretations remain plausible, the system is intended to request clarification rather than execute an underspecified request.

At the core of the approach is the notion of a *RequestTemplate* (a controlled-natural-language question template). Practitioners may ask many different *execution-time questions* in natural language, but many of these can be understood as instances of a compact set of underlying question types. For example, *"Has the excavation been completed?"*, *"Has the blinding layer been completed?"* and *"Has the reinforcement placement been completed?"* are phrased differently but share the same underlying structure: asking whether a given activity has reached a state of completion. Similarly, *"Has the blinding layer been inspected and approved?"*, *"Has the reinforcement for Footing "F1" been inspected and approved for concreting?"* and *"Has the formwork been inspected and approved?"* conform to a pattern that inquires whether an entity has passed through an approval process in a specific execution context. The methodology, therefore, begins by abstracting from individual wordings to generic question types.

In four workshops with two industry partners (design–build and general contracting), we collected execution-time questions along an exemplar foundation process, clustered them into request types, and iteratively refined template formulations and argument roles with practitioner feedback. The outcome is an initial template library covering completion, approval, readiness, and selected cross-domain request types, designed to be extended through additional templates, mappings, and query patterns. These workshops supported template elicitation and refinement, but they do not constitute a performance validation of the framework.

Each recurring execution-time question type is expressed as a *RequestTemplate*; each template references one or more query patterns (*QP_**) that are intended to become executable after placeholder binding. A template is written in a controlled natural language form with explicit placeholders for variable elements. For instance, a completion template may have the form *"Has **{Activity}** been completed?"*, a tolerance template may be written as *"Is the **{Property}** of **{Component}** within **{Requirement}**?"*, and a readiness template takes the form *"Is **{Zone}** ready for **{Activity}**?"*. The placeholders denote the template's arguments. They are not mere textual gaps but correspond to semantic roles that must be bound to specific project instances, such as activities, components, zones, requirements, or documents. Templates may introduce additional roles where required (e.g., resources, suppliers, time windows, hazards), each of which is typed and described as part of the template metadata. Each template is enriched with an explicit description of its arguments, specifying for each placeholder the type of entity it refers to.

Table 1 shows the resulting *RequestTemplate* representation for a readiness question type (T34), including typed arguments, ontology mappings, and selected template metadata such as flow type and target state.

Table 1: Request template T34 for readiness checking.

Request Template	
Context	Template_ID: T34
Question_Template	*Is {Entity} ready for {Activity}?*
Arguments	Entity: Entity Activity: Activity
Mapping	Entity: `ifcowl:IfcProduct` → `dice:Object` Activity: `dicp:Activity`
Flow_Type	Safety / Workspace; Prerequisite Work; Material; Equipment; External Condition
Target_State	Ready; Approved

Decision logic is captured in a configurable *FlowConstraint* profile linked to each template/request. At runtime, an *InformationRequest* instantiates the template by binding argument roles to project instances, e.g., CQ013 binds Entity=FootingF1, Activity=Concreting, Zone=Zone_FootingF1. Argument types are anchored in shared semantic representations through ontological mapping: activities to *DiCon* Processes (dicp:); components to *DiCon* Entities (dice:) or ifcOWL products (ifc:); and zones to BOT spaces (bot:) or *DiCon* locations (dice:). Similarly, documents map to *DiCon* Information (dici:), resources to equipment/material concepts (dice/dicp:), and actors to *DiCon* Agents (dica:). This structure enables binding placeholders to project-specific instances, such as schedule tasks, model identifiers, zone definitions, or supplier records.

In addition to the argument structure, each template is associated with a characterization of the dependency it addresses and the state being checked. The methodology employs a concise vocabulary of flow types and target states, informed by lean construction practice and represented us-

ing *DiCon* constraint and state constructs (dicv:, dice:). In particular, the flow vocabulary is derived from the eight main flows used in Lean Construction constraint analysis: people, information, shared understanding, equipment, materials, prerequisite work, safe workplace, and external conditions (buildingSMART Deutschland, Fachgruppe BIM und Lean Construction, 2025). In the present paper, this flow vocabulary serves as an exemplar structuring device for one important class of operational requests, rather than as a restriction on the generality of the request layer itself. Target states specify whether the required condition is completion, approval, conformity, cleanliness, safety, availability, confirmed booking, or acceptable external conditions. For approval *requests*, the template semantics reflect that approvals are often granted for a subsequent activity, for example, "approved for concreting", rather than representing a context-free approval status. This improves alignment with work-release decisions, in which the relevant interpretation of an approval is tied to the next planned operation.

These elements are formalized in a request ontology. In this ontology, *RequestTemplates* are represented as instances of a *RequestTemplate* class, with properties that capture argument roles, associated flow types, and target states, as well as links to the corresponding classes in *DiCon*, *ifcOWL*, and *BOT*. The request ontology sits on top of the existing domain ontologies and defines the structure of information requests independently from the particular data instances of a project. It does not introduce new construction entities but explicitly represents the demand side. This allows the framework to refer to question types in the same way that *DiCon* relates to activity or entity types, and it enables reasoning about which data and states must exist to answer a given question type.

A second element of the methodology is to define graph-based query patterns for each template. While the request ontology defines the semantic structure of the *InformationRequest*, the query pattern describes how an instance of that *template type* is intended to be resolved over a set of federated knowledge graphs. For each template, the methodology identifies the relevant paths through the *DiCon* modules and related ontologies that correspond to the flows and states referenced by the template. For a completion *template*, the pattern follows the link from a process to its state and checks whether it is *Completed*. For a tolerance *template*, it follows from a component to a variable representing a measured property. It compares it to a constraint that encodes an allowed range and provides traceability to the originating requirement or specification reference. For booking and availability queries, patterns integrate process time windows with resource allocation or booking states and verify that commitments cover the relevant period.

For readiness *requests*, the pattern becomes more complex because readiness is defined as a conjunction of multiple flow conditions. In the framework, readiness is operationalized as an explicit set of constraints attached to a request, each specifying a flow type, a required target state, and the topic it concerns. The readiness query pattern evaluates these constraints by searching for evidence paths in the federated graph. This supports evidence-based assessment of work release and enables identification of which constraint lacks supporting evidence when a request cannot be satisfied.

These query patterns are defined as reusable graph-traversal templates and, for selected question types, are specified in SPARQL. They specify which node and relationship types must be traversed in the graph, but they do not hard-code project-specific identifiers. Instead, they are designed so that, once a concrete information request binds the template placeholders to specific project entities, the pattern can be instantiated into a graph query specification in SPARQL or another query language.

Scope and limitations. The methodology targets *execution-time* decision support in construction, where stakeholders assess readiness, approvals, constraints, resource commitments, and safety/compliance conditions under short-horizon operational planning. While the request layer is process-agnostic by design, the elicitation and exemplar instantiations in this study focus on foundation construction; extending coverage to additional project phases (e.g., design), trades, or process types primarily requires adding further templates, mappings, and query patterns.

The methodology stops short of implementing the full machinery for automatically translating arbitrary natural-language questions into template instances and for extracting evidence from all operational systems. Instead, it focuses on defining the semantic structure and logical content of templates and patterns, demonstrating how they can be aligned with existing ontologies and support cross-system reasoning about readiness, approvals, constraints, and resource availability.

By proceeding from observed questions in practice through abstraction into templates, formalization in a request ontology, and definition of graph-based query patterns, the methodology provides a coherent framework for representing stakeholder information needs in a manner compatible with ontology-based data integration efforts. It establishes a layer between human questions and machine-interpretable models, serving as a foundation for subsequent work on natural language interfaces, interactive disambiguation of underspecified requests, automated query generation, and integration with digital twin platforms.

Prototype Design and Illustrative Instantiation

This section outlines the current prototype design for operationalizing *InformationRequest*s over federated construction knowledge graphs and illustrates how selected request types are intended to be instantiated. The companion repository (see Data Availability) provides the request ontology, template library, example request instances, and

an evolving query-pattern collection that together specify the proposed execution logic. At the current stage, these artifacts demonstrate how recurring operational questions can be formalized, bound to project-specific entities, and linked to graph-based evidence paths, but they do not yet constitute a fully implemented end-to-end demonstrator. In particular, automated execution across multiple distributed sources, persistence of intermediate check results, and comparative evaluation against manual information retrieval remain future work. Accordingly, the contribution of this section is to make the proposed operationalization concrete and inspectable rather than to report a completed system evaluation. The examples that follow are therefore intended as illustrative instantiations of the request logic, not as an empirical evaluation of end-to-end system performance.

A central design choice is that the framework does not assume universal, one-size-fits-all definitions of readiness, approvals, constraints, and resource availability. Instead, each template is paired with a *constraint profile* derived from decision checkpoints in an exemplar process chain, as described in the Methodology. This keeps request semantics stable across projects while allowing project-specific specialization of what counts as sufficient evidence (e.g., adding/removing constraints or tightening evidence requirements).

Consider the natural language question *''Is zone A ready for reinforcement placement?''*. The framework does not treat *ready* as a literal attribute to retrieve. Instead, the utterance is mapped to template T04, *''Is {Zone} ready for {Activity}?''*. This readiness mechanism is defined for multiple scopes, including zone-level readiness (T04) and entity-level readiness (T34, Figure 2). In the current repository state, zone-level readiness (T04) is specified more concretely than the entity-level readiness pattern (T34), whereas the latter currently remains illustrative. The request ontology constrains this mapping by typing the placeholders: *{Zone}* must resolve to a spatial entity, for example a *BOT* space or an aligned spatial entity concept, and *{Activity}* must resolve to a process concept in the workflow model, for example a *DiCon* process instance or type. These types act as semantic guardrails. They ensure that query construction relies on relations and states that exist in the aligned graph model, rather than producing queries that are formally valid but semantically ungrounded.

The template becomes operationalizable through bindings. For the corresponding request instance, *Zone_Instance = Zone A* and *Activity_Instance = Reinforcement Placement* are bound to concrete graph nodes such as *ex:ZoneA* and *ex:RebarPlacement_ZoneA*. The binding step links the domain language to project identifiers, including schedule IDs, identifiers from digital building models (for example, IFC GUIDs or curated element sets), and document references. In practice, this does not require authoring each request by hand: zones can be resolved from a location breakdown structure or spatial definitions, while activities can be resolved from schedule activities, work packages, or Last Planner tasks. What matters is that bindings persist as explicit links, preserving traceability from the question text to the graph entities used in execution. Operationally, readiness is evaluated as a *set of prerequisites* rather than a single status value: an *InformationRequest* is satisfied only if each required prerequisite can be justified by an evidence path in the federated graph. Accordingly, each prerequisite presupposes that corresponding evidence objects (e.g., inspection records, delivery confirmations, reservations, or safety documents) are maintained in the relevant source systems with sufficient identifier quality and timeliness to support decision-making. In Lean Construction constraint analysis, such prerequisites are commonly organized into recurring flow categories (often referred to as the ''eight flows''): *People* (qualified workforce), *Information* (execution documents), *Shared Understanding* (aligned interpretation and commitments), *Equipment*, *Materials*, *Prerequisite Work* (prerequisite work and trade interfaces), *Safe Workplace* (space availability and safety measures), and *External Conditions*. Table 2 summarizes the mapping from these flow categories to *FlowConstraint* types and typical evidence objects in a federated graph. Within

Table 2: Mapping of Lean ''8 main flows'' to controlled FlowConstraint types and typical evidence objects.

8 main flows	**Flow Constraint Type**	**Evidence Object Types (examples)**
People	Staffing	Crew allocation; Shift record
Information	Information	Latest drawings; Permits
Shared Understanding	Commitment	Weekly plan promise; Handover criteria
Equipment	Equipment	Reservation; Availability confirmation
Materials	Material	Delivery confirmation; On-site stock record
Prerequisite Work	Prerequisite Work	Completion state; Predecessor handover
Safe Workplace	Safety / Workspace	Risk assessment; Traffic setup
External Conditions	External Condition	Weather check; Site restrictions

Note: Flow types and target states are controlled vocabulary terms; labels shown here are human-readable.

the request ontology, these flow categories are encoded via *FlowConstraint* objects attached to an *InformationRequest*. Each *FlowConstraint* specifies a flow type, a target state, and the topic it concerns, with approvals modeled

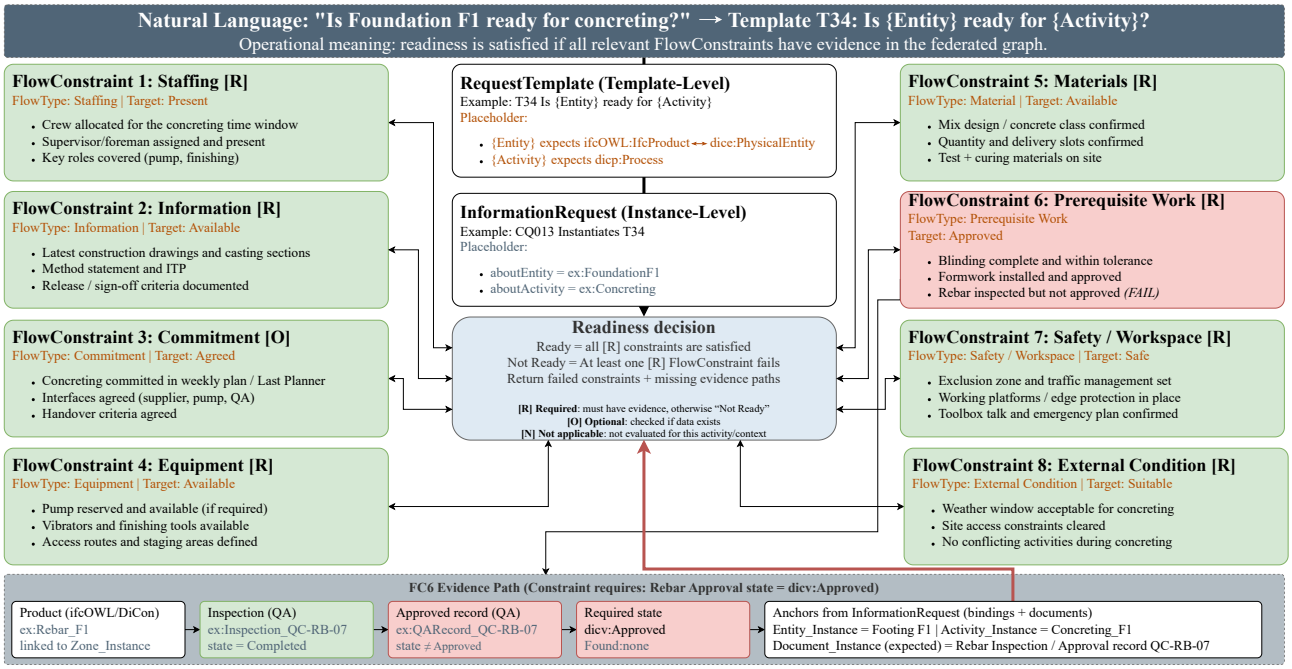

Figure 2: Illustrative template-driven readiness specification with evidence-path diagnostics at object level.

as required target states of the relevant evidence objects (e.g., inspection records) rather than as a separate flow type. For reinforcement placement in a foundation zone, this can include (i) the completion state of prerequisite work (e.g., blinding), (ii) an approval state for the relevant inspection record, and (iii) an implemented state for required safety measures. In this design, a *constraint profile* serves as a reusable default for a given template, specifying which flow constraints are required at a particular checkpoint. An *InformationRequest* then references the applicable profile and materializes the corresponding *FlowConstraint* instances for request-specific evaluation logic; project-specific specialization is expressed by adding, removing, or tightening these constraints at the request level (while keeping the template semantics stable). The key point is that the constraint profile is not constructed at query time; it is derived from the exemplar process checkpoints and refined based on partner feedback, so that the template encodes reusable readiness logic while remaining adaptable to project-specific requirements.

To make evaluation and diagnostics explicit, Figure 2 illustrates the same mechanism as shown in Figure 1, but on an object-level readiness request for concreting (T34) instead of zone-level readiness (T04), in which a single failing constraint can be traced to missing evidence that should justify work release. This highlights a practical requirement of operational question answering: a negative outcome should be explainable in terms of identifiable blockers rather than an undifferentiated ‘‘not ready’’ label.

In the intended execution logic, the bound request is evaluated against the attached constraint set. A readiness request is satisfied only if each required constraint can be justified by a matching evidence path in the federated graph; otherwise, the corresponding unsatisfied constraint and expected evidence type are identified. In this way, negative outcomes remain actionable by exposing specific blockers rather than an undifferentiated ‘‘not ready’’ label.

In the pattern library, analogous blocks are specified to resolve state checks (e.g., approved inspections), safe workplace, information, and logistics constraints through the corresponding document, resource, and process links in the aligned ontologies. Because constraints are explicit objects, the same request can, in principle, return diagnostics by selecting the constraints that fail and exposing missing evidence paths. Conceptually, a response can therefore be treated as a structured answer object that includes (i) the boolean result (ready / not ready), (ii) the constraint set evaluated, (iii) minimal evidence paths for satisfied constraints, and (iv) for unsatisfied constraints an expected path signature indicating what evidence type is missing (e.g., an inspection record in state *Approved* for a specific object and checkpoint). This turns negative outcomes into actionable blockers and preserves traceability back to the originating question, bindings, and sources.

Bindings determine which concrete project instances a request refers to (e.g., which zone, activity, object, or document). In practice, bindings can be obtained through three complementary mechanisms: (i) user-assigned bindings, for example by selecting a zone, activity, and checkpoint in an interface; (ii) context- or rule-derived bindings from existing project structures, such as linking scheduled activities to locations via a location breakdown structure, mapping work packages to curated element sets, or resolving document instances from metadata (e.g., inspection type and revision); and (iii) language-assisted proposals that extract candidate entities from the utterance and surrounding

context. In ambiguous cases, proposed bindings are expected to be confirmed or clarified by the user.

To preserve traceability and support evidence-based diagnostics, each binding is explicitly stored in the *InformationRequest* and annotated with provenance metadata that capture its origin and creation context (source system/-dataset, timestamp, and confirmer). At minimum, this includes the source system or dataset from which the binding was obtained, the identifier/URI used in the graph, a timestamp, and (if applicable) the role or actor who confirmed the binding. Thereby, the provenance metadata supports reproducibility and auditing of decision-relevant answers, and helps distinguish missing evidence in the data from underspecified or incorrectly bound requests. Beyond readiness, the same request semantics covers other execution-time decision questions across the eight flows. The template library includes approval and release checks (linking components or activities to inspection records and release documents), tolerance and requirement conformance queries (linking measured variables to specification constraints with traceability to the originating document), and logistics and resource commitment requests (linking activities and time windows to bookings for equipment, deliveries, or crews). These requests follow the same operational pattern: placeholders are bound to project instances, and the corresponding query pattern searches for satisfiable evidence paths, potentially across federated sources such as scheduling systems, Quality Assurance (QA) tools, health-and-safety management systems, procurement, or supplier dispatch data. When evidence is missing or inconsistent, the request returns unsatisfied constraints and their missing paths, enabling targeted follow-up actions rather than manual, system-by-system search.

Discussion and Future Work

The proposed framework highlights that many execution-time questions are fundamentally representation problems: decision-level inquiries require combining evidence across domain systems. The framework addresses these information needs through reusable requests: templates capture stable semantics, while bindings link them to project-specific instances, and the intended response structure exposes evidence-grounded blockers.

The readiness example illustrates why this separation is useful. A question such as *"Is zone A ready for reinforcement placement?"* cannot be reliably answered by searching a single system or treating readiness as a single status value. It requires explicit aggregation of prerequisite conditions aligned with lean constraint analysis flows, and it benefits from diagnostics that expose which prerequisites lack supporting evidence. Representing prerequisites as typed constraints and evaluating them via evidence paths enables the system to return actionable blocker traces, rather than undifferentiated negative answers. At the same time, the approach is compatible with federated settings, where evidence remains in its native systems, while requests provide a uniform mechanism to verify whether the required information is present and consistent.

The current scope also implies limitations. Templates and query patterns were derived from an exemplar process chain and refined with partner feedback, but broader generalization requires validation across additional trades, contracting arrangements, and delivery settings. Practical deployment depends not only on semantic formalization but also on data governance: binding resolution requires sufficiently stable identifiers, classifications, and contextual cues across tools, and missing records must not be conflated with negative evidence. In federated settings, evaluation must also account for access constraints, latency, and conflicting versions across sources. We have not yet evaluated the approach through end-to-end execution on noisy real-world project data; current validation is limited to curated example graphs and synthetic instances in the companion repository. Robustness under missing, conflicting, or partially linked evidence across sources remains future work. The intended behavior under such conditions is to return explicit blockers (missing evidence paths) rather than produce ungrounded answers silently.

In this sense, the framework can also be understood as an explicit specification of project information obligations: it makes visible which evidence must be available, from which source, and at what level of reliability to support a given operational decision.

Future work will focus on closing the loop from natural language to operationalizable, evidence-grounded requests. This includes template selection and placeholder binding from user utterances and project context, clarification of underspecified requests, and connectors for retrieving and linking evidence from real project systems. Evaluation across multiple projects will assess the correctness of answers, time-to-answer, and the practical usefulness of diagnostics for resolving blockers.

Conclusion

This paper addressed the persistent difficulty of formalizing execution-time process questions in construction when relevant evidence is distributed across heterogeneous systems. It introduced a semantic request layer that represents recurring operational information needs as reusable request templates, separates stable request semantics from project-specific bindings, and links these requests to graph-based evidence structures across federated sources. Lean work-release and readiness questions served as an exemplar application domain for eliciting and structuring the required dependencies.

The proposed request layer complements supply-side ontologies such as *DiCon*, *ifcOWL*, and *BOT* by explicitly representing the demand side of project information: which question is being asked, which entities and constraints it refers to, and which evidence paths would be required to support an answer. The illustrative examples show how natural-language questions can be grounded in explicit bindings and operationalized as evidence-oriented request specifications, providing a more transparent and

traceable basis for decision support. Automated natural-language interpretation and end-to-end execution over live project systems remain future work.

Data availability statement

The data supporting this study's findings are available at `https://gitlab.lrz.de/tweilbach/semantic_request_framework`,
ensuring that they are Findable, Accessible, Interoperable, and Reusable (FAIR).

Acknowledgments

The authors gratefully acknowledge the support and resources provided by the Bavarian Construction Industry Association (Bayerischer Bauindustrieverband), enabling our research and its practical application.
In addition, we would like to thank Vollack Gruppe GmbH & Co. KG and Riedel Bau AG.

References

Ballard, G. (2020). The last planner system. In Lean construction, pages 45–53. Routledge.

Ballard, G. and Howell, G. (1998). Shielding Production: Essential Step in Production Control. Journal of Construction Engineering and Management, 124(1):11–17.

buildingSMART Deutschland, Fachgruppe BIM und Lean Construction (2025). BIM und Lean Construction. Number Schriftenreihe 1.0. buildingSMART Verlag.

buildingSMART International (2025). ifcOWL. `https://github.com/buildingsmart-community/ifcOWL`. Accessed: 2025-11-27.

Čustović, I., Cao, J., Hall, D. M., and Wamelink, H. (2025). Enhancing Situation Awareness of Construction Managers using Human-centered Digital Twins. Advanced Engineering Informatics, 67:103557.

Daum, S. and Borrmann, A. (2014). Processing of Topological BIM Queries using Boundary Representation Based Methods. Advanced Engineering Informatics, 28(4):272–286.

Endsley, M. R. (1995). Toward a Theory of Situation Awareness in Dynamic Systems. Human Factors, 37(1):32–64.

Farghaly, K., Soman, R., and Whyte, J. (2024). cSite ontology for production control of construction sites. Automation in Construction, 158:105224.

Hellin, S., Nousias, S., and Borrmann, A. (2025). Natural Language Information Retrieval from BIM Models: An LLM-Based Multi-Agent System Approach. In Proceedings of the 2025 European Conference on Computing in Construction, volume 6 of Computing in Construction, Porto, Portugal. European Council on Computing in Construction.

Koskela, L. (2000). An exploration towards a production theory and its application to construction: Dissertation. PhD thesis, Aalto University, Finland.

Mazairac, W. and Beetz, J. (2013). BIMQL – An open query language for building information models. Advanced Engineering Informatics, 27(4):444–456.

Nabavi, A., Ramaji, I., Sadeghi, N., and Anderson, A. (2023). Leveraging Natural Language Processing for Automated Information Inquiry from Building Information Models. Journal of Information Technology in Construction, 28.

Pauwels, P. (2014). Supporting Decision-Making in the Building Life-Cycle Using Linked Building Data. Buildings, 4(3):549–579.

Rasmussen, M. H., Lefrançois, M., Schneider, G. F., and Pauwels, P. (2020). BOT: The building topology ontology of the W3C linked building data group. Semantic Web, 12(1):143–161.

Sacks, R., Brilakis, I., Pikas, E., Xie, H. S., and Girolami, M. (2020). Construction with digital twin information systems. Data-Centric Engineering, 1:e14.

Sacks, R., Eastman, C., Lee, G., and Teicholz, P. (2018). BIM handbook: A guide to building information modeling for owners, designers, engineers, contractors, and facility managers. John Wiley & Sons.

Schlenger, J., Pluta, K., Mathew, A., Yeung, T., Sacks, R., and Borrmann, A. (2025). Reference architecture and ontology framework for digital twin construction. Automation in Construction, 174:106111.

Speiser, K., Seiß, S., Boukamp, F., Melzner, J., and Teizer, J. (2025). From fragmented data to unified construction safety knowledge: A process-based ontology framework for safer work. Automation in Construction, 176:106293.

Teclaw, W., O'Donnel, J., Kukkonen, V., Pauwels, P., Labonnote, N., and Hjelseth, E. (2024). Federating cross-domain BIM-based knowledge graph. Advanced Engineering Informatics, 62:102770.

Yin, M., Tang, L., Webster, C., Xu, S., Li, X., and Ying, H. (2023). An ontology-aided, natural language-based approach for multi-constraint BIM model querying. Journal of Building Engineering, 76:107066.

Zheng, Y., Törmä, S., and Seppänen, O. (2021). A shared ontology suite for digital construction workflow. Automation in Construction, 132:103930.