HOKKAIDO UNIVERSITY

Title	A Model Context Protocol Framework for Bridge Data Generation with Large Language Model Integration
Author(s)	Maharjan, Shreejan; Chun, Pang-jo
Relation	Proceedings of the 10th International Symposium on Reliability Engineering and Risk Management (ISRERM 2026), June 28 - July 1, 2026, Sapporo, Japan
Citation	Proceedings of the 10th International Symposium on Reliability Engineering and Risk Management (ISRERM 2026)
Issue Date	2026-07-01
Doc URL	https://hdl.handle.net/2115/100468
Type	conference paper
File Information	C000201_1.pdf

Hokkaido University Collection of Scholarly and Academic Papers : HUSCAP

10th International Symposium on Reliability Engineering and Risk Management (ISRERM 2026),
28 June-01 July 2026, Sapporo, Japan
Kitahara, T., Miyamori, Y. & Beer, M. (editors)

A Model Context Protocol Framework for Bridge Data Generation with Large Language Model Integration

Shreejan Maharjan[1] and Pang-jo Chun[1]*

[1]*Department of Civil Engineering, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan.*
E-mail: chun@g.ecc.u-tokyo.ac.jp
**Corresponding Author*

Large Language Models (LLMs) have demonstrated strong capabilities in natural language understanding and reasoning, however, integrating them with domain-specific engineering tools remains a persistent challenge. This paper introduces BrDSL-MCP, a Model Context Protocol (MCP) server designed to connect conversational AI interfaces with structural engineering workflows for bridge specification data generation. The proposed system supports real-time interaction between LLMs and bridge engineering tools, enabling parametric design generation, finite element analysis (FEA) via Abaqus, within a unified, natural-language-driven loop. We detail the architectural choices that facilitate reliable tool composition and demonstrate how the system supports bridge data generation through conversational interaction. Case studies indicate that engineers can complete design iterations faster than with conventional manual workflows, while retaining full authority over final engineering decisions.

Keywords: Model Context Protocol, Large Language Models, Bridge Design, Finite Element Analysis.

1 Introduction

The increasing proportion of infrastructure assets in Japan expected to exceed their service life by 2040 has made inspection, maintenance, and renewal planning a matter of growing urgency (MLIT, 2019). At the same time, limited maintenance budgets and the continued deterioration of bridge conditions have raised serious safety concerns not only in Japan but also worldwide, highlighting the need for more optimized maintenance and rehabilitation decision-making processes (Orcesi and Cremona, 2010). Among the many challenges in bridge management, inspection remains one of the most critical bottlenecks. In most countries, bridge condition assessment still relies mainly on periodic visual inspection by trained inspectors. Although widely practiced, this method is labor-intensive, subjective, and difficult to scale across large bridge inventories (Dorafshan and Maguire, 2018; Morgenthal et al., 2019). In Japan, these challenges are further intensified by limited technical and institutional capacity at the municipal level, which can hinder the efficient implementation of bridge inspection and maintenance practices. These limitations have encouraged growing interest in automated inspection frameworks that combine unmanned aerial systems, computer vision, and point cloud processing to support or partially replace conventional manual methods.

Bridge Information Modeling (BrIM) and bridge digital twins (DTs) have shown clear potential to centralize asset data, support lifecycle analysis, and improve the efficiency of maintenance planning (Costin et al., 2024; Yang et al., 2024; Nakamura, 2025). Recent work has demonstrated, for example, that automated point cloud-to-BIM reconstruction can streamline the creation of as-built digital models and link them directly to structural simulation (Fang et al., 2025), while point cloud-based damage detection combined with IFC semantic enrichment enables inspection findings to be integrated into BIM-based bridge management systems (Isailović et al., 2020). Despite these advances in data acquisition and processing, the integration of such inspection outputs into structured digital environments for lifecycle management remains an open challenge.

In principle, a digital twin can integrate geometric data, inspection records, sensor measurements, and deterioration predictions within a single interactive environment, which is difficult to achieve using conventional document-based management systems. However, the practical adoption of these technologies for existing bridges remains limited. A major reason is that information for older infrastructure is often scattered across paper-based documents, PDF reports, and disconnected databases, and in many cases the data are incomplete or missing. Without accessible and well-structured information, the full potential of digital twin technology is difficult to be realized.

More recently, advances in artificial intelligence, particularly Large Language Models (LLMs), have created new opportunities for infrastructure automation. Previous studies have shown that LLMs can support tasks such as extracting structured information from engineering documents, checking design compliance with codes, and generating finite element analysis scripts (Joffe et al., 2025; Chen and Bao, 2025; Liang et al., 2025). However, most of these studies still treat LLMs as standalone text-based tools (Avila et al, 2025). Their integration with simulation frameworks, validation workflows, and structured digital environments such as bridge digital twins remains limited. As a result, their practical contribution to real-world bridge infrastructure management is still not fully established.

To address this limitation, the authors previously developed the BrDSL (Bridge Domain-Specific Language) framework, which provides an end-to-end pipeline for generating validated, analysis-ready bridge specifications from minimal user inputs through a generate-validate-repair-simulate cycle (Maharjan and Chun, 2026). BrDSL achieves a high feasibility rate and produces specifications that pass automated Abaqus FEM verification under service loading.

However, like many domain-specific engineering frameworks, BrDSL still requires programmatic interaction, where users must write scripts, configure JSON inputs, and manage pipeline execution manually. This study extends the previous framework by introducing BrDSL-MCP, which integrates the BrDSL framework with the Model Context Protocol (MCP). By exposing BrDSL's capabilities as MCP tools, the proposed system enables an LLM to autonomously orchestrate bridge design generation, FEM simulation, result extraction, and code compliance checking through natural language, without requiring manual scripting or pipeline management.

2 BrDSL-MCP System Overview

2.1 *Bridge Domain Specific Language (BrDSL) overview*

BrDSL is a framework that transforms minimal user requirements x (span length, deck width, structure type) into a constraint-consistent, FEM-ready bridge specification y*. The framework is demonstrated for simply supported steel girder bridges with reinforced concrete deck slabs. The primary objective is formulated as:

$$y^* = \arg min_y J(y \mid x) \quad (1)$$

where $J(y \mid x)$ aggregates non-compliance across three constraint layers. Feasibility of a candidate specification y under user requirements x is defined through three Boolean predicates:

$$\boldsymbol{C}(y,x) = \boldsymbol{C}_{schema}(y) \wedge \boldsymbol{C}_{eng}(y,x) \wedge \boldsymbol{C}_{prior}(y) \quad (2)$$

$C_{\text{schema}}(y)$ verifies conformance to the canonical JSON schema (required keys, data types, hierarchical structure). $C_{\text{eng}}(y,x)$ enforces engineering constraints including slenderness limits ($\lambda_w = h/t \leq 120,\ \lambda_f = (b/2)/t \leq 9$), span–depth heuristics ($0.03L \leq h \leq 0.06L$), serviceability deflection ($\delta \leq L/800$), and capacity checks ($\phi M_n \geq M_u, V_n \geq V_u$). $C_{\text{prior}}(y)$ evaluates statistical plausibility against corpus-derived distributions, flagging parameters with $|z| > 3.5$ as outliers. The composite violation score:

$$J(y \mid x) = \lambda_1 V_{schema}(y) + \lambda_2 V_{eng}(y,x) + \lambda_3 V_{prior}(y) \quad (3)$$

BrDSL generates validated bridge specifications through a four-phase workflow consisting of prototype retrieval, LLM-based candidate generation, multi-level validation, and iterative repair. Using user requirements and a corpus of 1,021 canonical bridge designs, the framework retrieves relevant precedents, generates candidate JSON specifications, and refines them through schema, engineering, and statistical checks until a feasible solution is obtained across various span-width configurations. Once a hard-feasible specification is produced, BrDSL automatically converts it into an Abaqus FEM model for structural verification under service loading. The pipeline programmatically defines geometry, materials, mesh, boundary conditions, and loads, then evaluates stress and deflection criteria, with the full process requiring about 35-40 seconds per configuration.

2.2 *MCP Integration*

Although BrDSL provides a robust engineering methodology for generating and validating bridge designs, it operates as a programmatic pipeline that requires scripting knowledge and manual control over each workflow step. The main contribution of this study lies in the MCP integration layer, which transforms BrDSL from a developer-oriented tool into a conversational engineering framework that engineers can interact with through natural language.

The MCP (Anthropic, 2025a) is an open standard that allows LLMs to discover and invoke external software tools during a conversation. For instance, when a connection is established, the MCP server exposes each BrDSL capability (such as design generation, simulation, or code checking) as a callable tool, along with a description of what it does and what inputs it requires. This allows the LLM to select and sequence the appropriate tools based on the engineer's request, without the user needing to write any code. For engineering applications, this architecture offers three key properties: discoverability of newly added tools without reconfiguration, composability of tools into flexible multi-step workflows, and accessibility through natural language rather than scripted commands. In the present study, these properties allow engineers to access bridge design generation, Abaqus-based finite element analysis, and code compliance checking through a single conversational interface.

Accordingly, the MCP integration developed in this work addresses three core challenges: (i) decomposing the BrDSL pipeline into independently callable tools while preserving the engineering logic that connects them, (ii) managing long-running Abaqus simulations within the conversational interaction model through status polling, and (iii) designing tool interfaces that give the LLM sufficient context to orchestrate multi-step workflows flexibly and correctly.

2.3 *System architecture*

The proposed system architecture, shown in Figure 1, is designed to connect natural language interaction with automated bridge design and structural verification. The process begins when the user provides a design or analysis request through the LLM client (Claude (Anthropic, 2025b)). This request is then handled through the BrDSL-MCP server, which acts as the coordination interface between the conversational front end and the underlying engineering tools. Through MCP, the server exposes 11 tools that support bridge generation, design modification, finite element simulation, code compliance checking, and design comparison.

These tools are organized according to their functional role in the workflow. The design-related tools are used for generating bridge specifications and performing parametric modifications based on the BrDSL methodology. The

simulation-related tools manage Abaqus execution through an asynchronous mechanism, where the solver runs in the background and the LLM checks the job status until completion. The verification-related tools evaluate structural performance against AASHTO and Eurocode, while the utility tools support design inspection, retrieval, and comparison.

The backend executes the bridge generation pipeline, validation routines, and design modification procedures, whereas Abaqus is used for structural modelling and finite element analysis. A distinguishing feature of the proposed architecture is that workflow progression is not prescribed through a fixed hard-coded sequence. Instead, the LLM determines tool selection, execution order, and branching logic dynamically according to the user request and intermediate outputs produced during execution. As a result, the system supports a closed-loop iterative refinement process in which design modification, simulation, and verification are performed adaptively through conversational interaction.

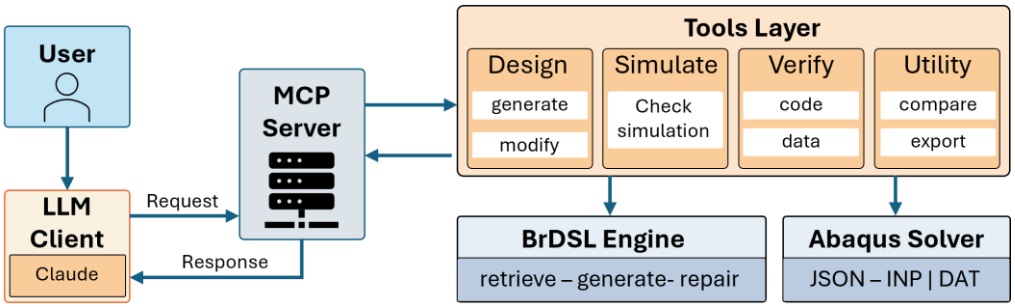


Figure 1. System architecture of proposed BrDSL-MCP framework

3 Case Study and results

3.1 *Experimental Setup*

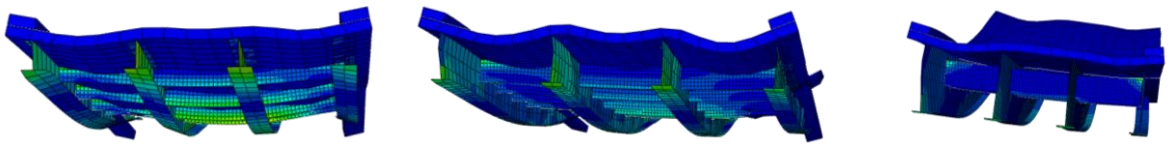
To evaluate BrDSL-MCP, the LLM (Claude) was given a single natural language instruction to perform design space exploration. Specifically, it was asked to generate, simulate, and verify five steel girder bridges with spans of 15, 20, 25, 30, and 35 m, each having a width of 8 m, in sequential order. The instruction required that the full workflow for one bridge be completed before proceeding to the next, with AASHTO and Eurocode compliance checks carried out for each design. All cases were generated using the generation mode of BrDSL. FEM models used S4R quadrilateral shell elements with 500 mm nominal mesh size. Loading comprised a uniform live load of P_1 = 10.0 kN/m², P_2 = 3.5 kN/m², with impact factors ranging from 0.222 to 0.250. Material properties were linear elastic: E_steel = 200 GPa, ν = 0.3; E_concrete = 25 GPa, ν = 0.17.

3.2 *Results*

To evaluate the performance of the proposed BrDSL-MCP system, a five-case bridge design and analysis study was carried out. Table 1 summarizes the main design parameters and Abaqus FEM results for the five bridge configurations. All five bridges satisfied both the AASHTO deflection limit of L/800 and the Eurocode limit of L/250. The AASHTO utilization ratios ranged from 0.142 to 0.206, indicating that all generated designs remained well within the allowable serviceability limits. Figure 2 illustrates some examples of simulation model generated from the BrDSL-MCP system.

Table 1. Summary of design parameters and Abaqus FEM results for the five bridges.

Span (m)	Web depth (mm)	Web thickness (mm)	Max deflection (mm)	L/800 (mm)	Deflection ratio	L/250	Span-depth ratio	Max deck stress (MPa)
15	656.5	14.0	3.45	18.75	0.184	PASS	22.8	1.182
20	910	12.0	4.70	25.00	0.188	PASS	22.0	1.372
25	1111	14.0	6.43	31.25	0.206	PASS	22.5	1.867
30	1800	15.5	5.34	37.50	0.142	PASS	16.7	1.428
35	2100	18.0	6.61	43.75	0.151	PASS	16.7	1.596



15 m X 8 m Bridge 25m x 8m Bridge 35m x 8m Bridge

Figure 2. Simulation model generated from case study on bridges.

The BrDSL generator framework increased the web height progressively with span length, from 656.5 mm for the 15 m bridge to 2100 mm for the 35 m bridge, generally following the expected span-to-depth relationship. A non-monotonic deflection trend was observed, with the 30 m bridge showing lower deflection than the 25 m bridge. This behavior is attributed to the larger increase in web height for spans of 30 m and above, which resulted in higher flexural stiffness. Peak von Mises stresses remained below 1.9 MPa in all cases, confirming elastic structural response under the applied loading.

3.3 *Parametric study via iterative design modification*

To demonstrate the feedback-loop capability of MCP, a parametric study was carried out on the 20 m × 8 m bridge by modifying selected design parameters using the modify_design tool. Unlike the design space exploration presented in Section 3.2, where each bridge was generated from scratch through the BrDSL pipeline, this study focused on modifying an already verified design and re-running the simulation. This was done to illustrate how MCP can support iterative and conversational design refinement within an existing workflow.

Starting from the original generated bridge design, two modified configurations were examined. In Variant A, the deck slab thickness increased from 212 mm to 250 mm, while the web height was kept unchanged. This modification was intended to isolate the influence of increased deck stiffness on the structural response of the bridge system. In Variant B, the deck thickness was again increased to 250 mm, but the web height was also increased from 910 mm to 1200 mm. This second case therefore combined the effect of a stiffer concrete deck with the effect of a larger girder moment of inertia. The modified JSON files for both variants were saved separately so that the original verified design could be retained for direct comparison. The comparative FEM results for the three configurations are summarized in Table 2.

Table 2. Parametric study results for the 20 m × 8 m bridge. Variant A modifies only deck thickness; Variant B modifies both deck thickness and web height. Percentage changes are relative to the original design.

Parameter	Original	Variant A	Variant B
Deck thickness (mm)	212	**250 (+18%)**	**250 (+18%)**
Web height (mm)	910	910 (unchanged)	**1200 (+32%)**
Web thickness (mm)	12.0	12.0	12.0
Max deflection (mm)	4.704	4.165 (−11.5%)	2.804 (−40.4%)
Max Mises stress (MPa)	1.372	1.205 (−12.2%)	0.922 (−32.8%)
AASHTO L/800 limit (mm)	25.0	25.0	25.0
AASHTO utilization	0.188	0.167 (−11.2%)	0.112 (−40.4%)
Eurocode utilization	0.059	0.052 (−11.9%)	0.035 (−40.7%)
Span-to-depth (L/H_w)	22.0	22.0	16.7
Code compliance	PASS	PASS	PASS
MCP tool used	brdsl:pipeline	brdsl:modify_design	brdsl:modify_design

Increasing only the deck thickness in Variant A resulted in a moderate improvement in structural performance. The maximum midspan deflection decreased from 4.704 mm to 4.165 mm, corresponding to a reduction of 11.5%, while the maximum von Mises stress decreased from 1.372 MPa to 1.205 MPa, giving a reduction of 12.2%. This behavior can be attributed to the additional composite stiffness provided by the thicker deck slab, which shifts the neutral axis upward and increases the effective stiffness of the composite girder-deck system.

In contrast, Variant B produced a much more pronounced structural improvement. The maximum deflection was reduced from 4.704 mm to 2.804 mm, representing a reduction of 40.4%, and the maximum von Mises stress decreased from 1.372 MPa to 0.922 MPa, corresponding to a reduction of 32.8%. The increase in web height was the main contributing factor in this case, since the flexural stiffness of the girder section is highly sensitive to sectional depth. Correspondingly, the AASHTO utilization ratio decreased from 0.188 to 0.112, indicating that the modified design used only a small fraction of the allowable deflection limit. Although this confirms excellent structural performance, it also suggests that the configuration may be overly conservative and could be further optimized from the viewpoint of material efficiency.

From the MCP workflow perspective, this study highlights two important capabilities. First, the MCP tool allows specific design parameters to be adjusted without requiring regeneration of the full bridge specification, while preserving all other validated design attributes such as flange dimensions, cross-beam arrangement, bearing layout, material properties, and loading conditions. Second, the modification step can be directly linked with Abaqus FEM execution, allowing the user to obtain updated structural results from a single conversational instruction. In practical terms, a request such as increasing the deck thickness and web height, followed by simulation, can be completed in one interaction. In a conventional scripted workflow, the same process would typically require manual editing of the JSON file, saving a new configuration, launching the FEM analysis, and then extracting and interpreting the output results.

3.4 *Error handling*

To examine the robustness of the proposed system under code compliance failure, an additional experiment was carried out by deliberately introducing a structural violation. Starting from the validated 25 m × 8 m bridge configuration shown in Table 1, the design was intentionally weakened by reducing the web height to 400 mm and the web thickness to 8 mm using the modify_design tool, followed by re-simulation. Table 3 presents the results for the three design states, namely the original design, the failed design, and the corrected design.

Table 3. Error-handling experiment results for the 25 m × 8 m bridge. The failed design was created by deliberately reducing the web height and thickness to induce deflection failure. The LLM autonomously identified the violation and corrected the design without explicit instructions.

Parameter	Original	Failed design	Corrected
Web height (mm)	1111	400	800
Web thickness (mm)	14.0	8.0	12.0
Max. deflection (mm)	8.7	39.5	12.2
AASHTO limit L/800 (mm)	31.2	31.2	31.2
Deflection ratio	0.28	1.26	0.39
AASHTO check	PASS	FAIL	PASS
Eurocode L/250	PASS	PASS	PASS
LLM action	Generation	Detect violation	Autonomous recovery

The modified bridge produced a deflection failure during the code compliance check. The maximum deflection increased to 39.5 mm, exceeding the allowable L/800 limit of 31.2 mm and resulting in a utilization ratio of 1.26. In response to this outcome, the LLM selected the modify_design tool and revised the section dimensions by increasing the web height to 800 mm and the web thickness to 12 mm. After re-simulation and re-evaluation, the corrected design showed a maximum deflection of 12.2 mm with a utilization ratio of 0.39, thereby satisfying both the AASHTO and Eurocode serviceability requirements. The complete recovery sequence involved 18 tool calls, as illustrated in Figure. 3.

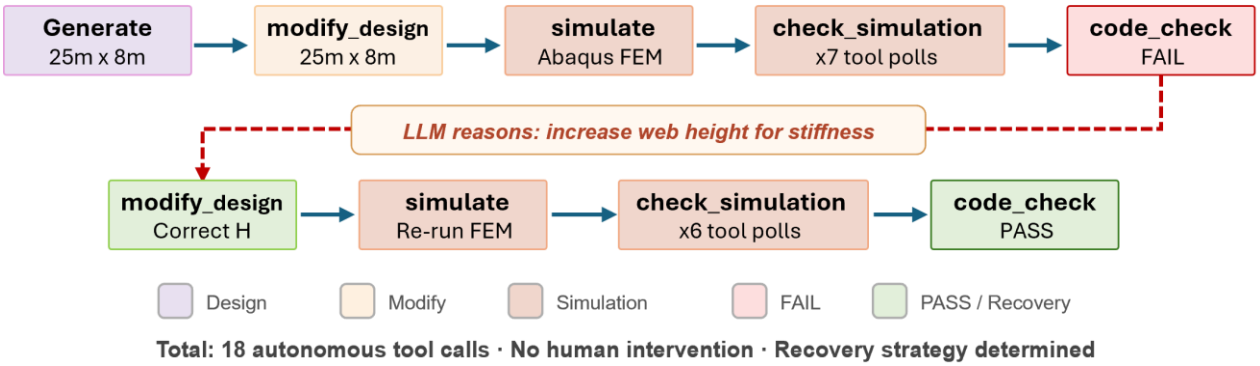


Figure 3. LLM-driven error-handling trajectory for a deliberately failed 25 m × 8 m bridge design. Top: the design is weakened by reducing the web height to 400 mm, triggering a deflection failure (ratio = 1.26). Bottom: the LLM autonomously increases the web height to 800 mm and re-verifies compliance (ratio = 0.39).

It should be noted that no explicit guidance was provided to the LLM regarding which parameters should be modified or the extent of the required adjustment. The recovery process emerged from the model's interpretation of the code_check output, through which it identified the stiffness-related cause of the violation and selected a suitable corrective modification. This result demonstrates that the MCP-based workflow can support autonomous failure detection and iterative recovery within the bridge design and verification process.

4 Discussion

Compared with the standalone BrDSL workflow, the MCP-integrated system offers three main advantages. First, it removes the scripting barrier by allowing bridge generation, simulation, and code checking to be initiated through natural language rather than direct Python-based execution. Second, it supports flexible post-processing and comparative analysis across multiple design cases without requiring additional user-written scripts. Third, it enables iterative design refinement, where a previously verified bridge can be modified and re-simulated through a single conversational instruction while preserving all unchanged parameters.

Despite these advantages, several limitations remain. The asynchronous polling strategy adopted for long-running Abaqus simulations introduces additional latency and increases the overall execution time. Another practical limitation observed during the experiment was related to the output length constraint of the LLM. During the five-bridge workflow, the conversation reached the maximum output limit between the fourth and fifth bridge cases, and a continuation prompt was required to resume execution. Although this interruption did not affect the underlying engineering process, as the workflow continued from the same point, it nevertheless reduced the continuity of an otherwise autonomous interaction. This issue arises because repeated tool calls and their responses accumulate in the conversation history and gradually consume the available context window. For larger design studies, this constraint may become more significant.

In addition, the current MCP implementation does not provide direct access to visual FEM outputs such as stress contours or deformation plots, limiting interpretation to extracted numerical summaries. The scope of BrDSL-MCP is also constrained by the underlying BrDSL framework, which currently supports only linear static analysis of simply supported steel girder bridges with RC decks. Finally, although the LLM can coordinate workflow execution and summarize results, its interpretations should still be reviewed by qualified engineers.

Table 4 compares BrDSL-MCP with the standalone BrDSL workflow for the five-bridge study. The results show that MCP improves accessibility, interaction flexibility, and iterative refinement capability, although at the expense of longer wall-clock time. Importantly, the structural results remain identical in both workflows because both rely on the same BrDSL procedures and Abaqus FEM models.

Table 4. Summary of key metrics comparing BrDSL-MCP with the standalone BrDSL workflow for the five-bridge study.

Metric	BrDSL (standalone)	BrDSL-MCP
User input mode	Python script	Natural language
Lines of user code required	~50–100	0
Total tool/function calls	5 (scripted)	87 (autonomous)
Wall-clock time (5 bridges)	~3–4 min	~8 min
Human intervention after start	None	None
Emergent analysis (trends, charts)	No (only scripted)	Yes (autonomous)
Dual-code compliance	If scripted	Autonomous
Design space exploration	Requires loop code	Natural language
Iterative refinement (parametric)	Edit JSON + re-run script	Single modify_design call

The 87 autonomous tool calls recorded during the five-bridge study reflect the LLM's real-time orchestration decisions. For each bridge, the core workflow consisted of calls to generate, simulate, repeated check_simulation polling until completion, code_check, and export_report. The majority of the 87 calls were attributable to asynchronous polling, which accounted for the largest share since the LLM issued multiple status queries per simulation at self-determined intervals. Additional calls to utility tools such as read_design and compare_designs were made by LLM for cross-case comparison and result synthesis. It should be noted that the exact number and sequence of tool calls is non-deterministic because the LLM selects tools through reasoning rather than a fixed script, a repeated run of the same experiment could produce a slightly different call count while arriving at identical structural results.

5 Conclusion and Future Work

This study presented BrDSL-MCP, an MCP-based integration framework that connects conversational LLM interaction with the BrDSL bridge generation and simulation-validation system. By exposing BrDSL functions as engineering tools through MCP, the proposed system enables bridge design generation, FEM simulation, and code compliance checking to be executed through natural language interaction. The integration also supports long-running Abaqus analysis within the conversational workflow through an asynchronous polling mechanism.

The case study on five steel girder bridge configurations demonstrated that the system can autonomously manage the full workflow with no human intervention after task initiation. All generated bridges satisfied the required code-based deflection limits, and the structural results were identical to those obtained from standalone BrDSL execution. In addition, the MCP-based workflow enabled comparative analysis and iterative design interaction within the same framework.

Future work will focus on extending the proposed framework toward broader and more practical bridge engineering applications. Priority areas include support for additional bridge typologies, incorporation of advanced FEM analyses such as nonlinear and dynamic response, and inclusion of visual result interpretation within the MCP toolset. To overcome the context window and session-timeout limitations observed in this study, future implementations will utilize the server-side state persistence already supported by the BrDSL-MCP server, where design outputs and simulation results are maintained on the filesystem independently of the conversation history. This would allow workflow checkpoints to be stored externally, enabling the LLM to resume multi-bridge studies after context

interruptions without loss of progress. In addition, replacing the current active polling mechanism with an event-driven completion notification would improve the efficiency of longer-running analyses.

Another important direction is the integration of BrDSL-generated models with point cloud-based reconstruction and inspection workflows. Recent studies have shown that image-point cloud fusion can support automated damage detection and three-dimensional spatial registration of bridge defects (Lin et al., 2025), while semantic segmentation can enable automated dimensional estimation of bridge components (Chen et al., 2025). Integrating these capabilities with BrDSL would make it possible to reconstruct as-built geometry from field data, map it onto a BrIM model, and link it directly with the FEM simulation framework. Such an extension would support the development of a digital simulation model that reflects the actual condition of the structure and would move the system closer to a closed-loop digital twin for infrastructure maintenance.

References

Anthropic. (2025a). Model Context Protocol Specification. Retrieved from https://modelcontextprotocol.io

Anthropic. (2025b). Claude: A family of large language models. Retrieved from https://www.anthropic.com

Avila, C., Ilbay, D., & Rivera, D. (2025). Human–AI Teaming in Structural Analysis: A Model Context Protocol Approach for Explainable and Accurate Generative AI. Buildings, 15(17), 3190. https://doi.org/10.3390/buildings15173190

Chen, J., & Bao, Y. (2025). Multi-agent large language model framework for code-compliant automated design of reinforced concrete structures. Automation in Construction, 177, 106331. https://doi.org/10.1016/j.autcon.2025.106331

Chen Y, Lin C, Li X, et al (2025) Automated dimension estimation of bridge components using semantic segmentation and geometric fitting of point cloud data. https://doi.org/10.1016/j.engstruct.2025.120837

Costin, A., Adibfar, A., & Bridge, J. (2024). Digital twin framework for bridge structural health monitoring utilizing existing technologies: New paradigm for enhanced management, operation, and maintenance. Transportation Research Record: Journal of the Transportation Research Board, 2678(6), 1095-1106.

Dorafshan S, Maguire M (2018) Bridge inspection: human performance, unmanned aerial systems and automation. J Civ Struct Health Monit 8:443–476. https://doi.org/10.1007/s13349-018-0285-4

Fang, Y., Mitoulis, S., Boddice, D., Yu, J. and Ninic, J. (2025). Scan-to-BIM-to-Sim: Automated reconstruction of digital and simulation models from point clouds with applications on bridges, Results in Engineering, 25, 104289. https://doi.org/10.1016/j.rineng.2025.104289

Isailovic, D., Stojanovic, V., Trapp, M., Richter, R., Hajdin, R. and Döllner, J. (2020). Bridge damage: Detection, IFC-based semantic enrichment and visualization, Automation in Construction, 112, 103088. https://doi.org/10.1016/j.autcon.2020.103088

Joffe, I., Felobes, G., Elgouhari, Y., Kalaleh, M. T., Mei, Q., & Chui, Y. H. (2025). The framework and implementation of using large language models to answer questions about building codes and standards. Journal of Computing in Civil Engineering, 39(4), 05025004. https://doi.org/10.1061/JCCEE5.CPENG-6037

Liang, H., Zhou, Y., Talebi Kalaleh, M., & Mei, Q. (2025). Automating structural engineering workflows with large language model agents. arXiv preprint arXiv:2510.11004.

Lin C, Chen Y, Itakura K, et al (2025) Bridge inspection using image–point cloud fusion with image filtering, damage detection and 3D registration. Autom Constr 180:. https://doi.org/10.1016/j.autcon.2025.106538

Maharjan, S., & Chun, P.-J. (2026). A large language model-driven framework for automated bridge specification generation and simulation validation. Computer-Aided Civil and Infrastructure Engineering, 100014. https://doi.org/10.1016/j.cacaie.2026.100014

Ministry of Land, Infrastructure, Transport and Tourism (MLIT). (2019). *White paper on land, infrastructure, transport and tourism in Japan*. Retrieved from https://www.mlit.go.jp/en/statistics/white-paper-mlit-2019.html

Morgenthal G, Hallermann N, Kersten J, et al (2019) Framework for automated UAS-based structural condition assessment of bridges. Autom Constr 97:77–95. https://doi.org/10.1016/j.autcon.2018.10.006

Nakamura, K., Tsukada, Y., Teraguchi, T., Kurokawa, C., & Imai, R. (2025). Integrated data model for bridges with 3D geometry and maintenance information. Computer-Aided Civil and Infrastructure Engineering, 40, 4994-5009. https://doi.org/10.1111/mice.70084

Orcesi AD, Cremona CF (2010) A bridge network maintenance framework for Pareto optimization of stakeholders/users costs. Reliab Eng Syst Saf 95:1230–1243. https://doi.org/10.1016/j.ress.2010.06.013

Yang, Y., Zhu, Y., & Cai, C. (2024). Research progress and prospect of digital twin in bridge engineering. Advances in Structural Engineering, 27(2), 333-352. https://doi.org/10.1177/13694332231218764