2025 European Conference on Computing in Construction

CIB W78 Conference on IT in Construction

Porto, Portugal
July 14-17, 2025

ACTOR-BASED FRAMEWORK FOR INTEROPERABLE BIM DATA MANAGEMENT

Edoardo De Santis[1] and Pieter Pauwels[2]
[1]Sapienza University of Rome, Rome, Italy
[2]Eindhoven University of Technology, Eindhoven, The Netherlands
edoardo.desantis@uniroma1.it

Abstract

This paper presents a novel framework designed to decentralise data management and enhance collaboration and interoperability in the Architecture, Engineering, and Construction (AEC) sector. By integrating BIM with a representation of building elements as independent actors in an actor-based system, it supports scalable and parallel workflows, allowing to concurrently access and modify distinct parts of the model. Semantic Web Technologies ensure that the data is linked and structured. The framework was tested in a controlled environment, demonstrating its ability to manage element-level actors, propagate changes asynchronously, and maintain data consistency. However, the complexity of maintaining decentralised components remains a challenge.

Introduction

The Architecture, Engineering, and Construction (AEC) industry is a cornerstone of economic development worldwide, yet it continues to struggle with productivity and efficiency (McKinsey Global Institute, 2017).
Recent studies, such as Almamlook et al. (2020), have investigated the factors influencing productivity in the construction industry in more detail, pointing out that these are, in particular, coordination problems and errors in communication and information transfer. These challenges are not new; they have persisted since the influential reports of Latham (1994) and Egan (1998).
An important aspect of the AEC industry is its interdisciplinary nature, involving experts from diverse fields. Throughout a building's lifecycle, stakeholders generate models, files, and data that need to be shared and exchanged. This dynamic nature of the industry emphasises the need for effective collaboration and communication (Nidhal Neamah et al., 2023).
Current solutions often rely on file-based exchanges which can lead to versioning conflicts and inconsistencies between different representations of the same model.
Studies have highlighted that interoperability between different software applications in the AEC industry remains a significant challenge (Costin and Eastman, 2019; Grilo and Jardim-Goncalves, 2010). Recent studies have highlighted the potential of using Linked Data to improve interoperability and facilitate data exchange between different software applications (Werbrouck et al., 2019).
One approach is the centralised model, where a BIM model is stored on the cloud or a local server. Every person can access this model, and the changes made to it are available and visible to everyone. However, Pauwels (2014) argues that the centralised model gives the false impression of seamless integration with surrounding systems. In reality, critical interface points persist, and 'standard' or 'neutral' formats often become just another file format that requires conversion. This makes a fully centralised information structure impractical for real-world, multi-tool AEC environments.
The alternative is a decentralised model, where each user has their own copy of the model, and changes are communicated between users. This approach can be more flexible and, although it requires efficient communication protocols and mechanisms for maintaining consistency, it eliminates the bottlenecks and interface issues inherent in centralised systems.
Actor-based programming models provide a scalable and resilient approach to managing complex systems, allowing for the independent execution of tasks and efficient communication between components (Hayduk et al., 2015). In the context of BIM, actor-based systems are particularly suitable for managing distributed and parallel workflows, enhancing collaboration by enabling multiple users to work on different building elements simultaneously (Carrara et al., 2017).
The proposed framework seeks to leverage the strengths of both actor-based systems and Semantic Web technologies to create a decentralised information structure that supports collaboration and data exchange in BIM workflows. Building elements, such as walls, windows, doors, and structural components, are treated as independent actors within the framework. Each actor is responsible for managing its own state, making independent updates to the model, and communicating with other actors as needed.
The research objectives are as follows:

1. To design a decentralised framework that manages building elements as independent actors, enabling scalable and parallel collaboration in BIM workflows.

2. To ensure distributed interoperability by leveraging Semantic Web technologies to structure and link BIM

data in a machine-readable format.

Background and related work

BIM, Linked Data, and Collaborative Workflows

BIM has emerged as a transformative approach within the AEC industries, facilitating enhanced collaboration and data management throughout the project lifecycle. However, existing multi-user BIM systems face several limitations in terms of collaboration and data exchange. These limitations often stem from the centralization of data management, which can create bottlenecks and hinder collaboration among stakeholders. Traditional BIM systems typically rely on a centralised model where all users access a single version of the project data, which can lead to issues such as version control conflicts and data silos (Yin, 2024; Esser et al., 2022).

Recent research highlights the challenges and opportunities in BIM-based collaboration and data exchange, focusing on solutions to improve interoperability, security, and workflow efficiency. Cloud-based BIM services offer potential for enhanced data integration, but face challenges related to standardisation and system compatibility (Afsari et al., 2017). Sattler et al. (2021) propose a query-based framework to facilitate multi-domain collaboration, addressing semantic differences and data heterogeneity through intuitive BIM data analysis and enrichment.

Research indicates that decentralised BIM collaboration models may offer a solution to these challenges by allowing multiple users to work on their respective components of a project independently while still maintaining a coherent overall model. Decentralised systems leverage cloud technologies to facilitate real-time updates and data sharing, thus improving the efficiency of information flow among project participants (Hui, 2023; Hsu, 2024).

However, the transition from centralised to decentralised systems is not without its challenges. Issues such as interoperability between different software platforms and the need for standardised protocols for data exchange remain significant barriers to effective collaboration (Hagedorn et al., 2022; Wijekoon et al., 2018; Hsu, 2024).

Recent studies (Senthilvel and Beetz, 2020; Schlachter et al., 2022; Werbrouck et al., 2019) focus on the use of Linked Data as a possible solution to improve interoperability and steer the industry towards a data-driven approach. The adoption of such an approach could facilitate automated compliance verification and model validation, offering a more effective way to manage complex processes in the AEC sector.

Actor-Based Systems for Collaboration

Actor-based programming is a paradigm that facilitates the development of scalable and resilient systems, particularly suited for parallel and distributed workflows. In this model, "actors" are the fundamental units of computation, encapsulating state and behavior, and communicating exclusively through asynchronous message passing. This design allows for high levels of concurrency and decouples the components of a system, which enhances modularity and fault tolerance (Charousset et al., 2016; Chechina et al., 2017; Hayduk et al., 2015).

One of the primary benefits of actor-based systems is their scalability. They can efficiently manage a large number of concurrent operations, making them ideal for data-intensive applications. For instance, the C++ Actor Framework is capable of scaling to millions of actors across numerous processors, which demonstrates the model's adaptability to various hardware configurations (Charousset et al., 2016).

In the realm of data-intensive applications, actor-based models have been successfully implemented in various domains. For example, Akka.NET[1], an actor-based framework for .NET, has been utilised to build high-performance applications capable of handling significant loads, such as in real-time data processing and streaming applications (Tokpayev, 2023). Akka's design allows developers to create systems that can scale horizontally by adding more nodes, thus accommodating increasing workloads without significant architectural changes.

The principles of actor-based systems also align with collaborative frameworks in construction projects, where distributed workflows and scalability are critical. For example, the Bimserver.org project (Beetz et al., 2010) provides an affordable gateway for shared building models, enabling the storage, upkeep, and retrieval of building information models based on IFC. This system facilitates collaboration among stakeholders by leveraging distributed workflows, akin to actor-based systems. Similarly, the collaborative design framework introduced by Rosenman et al. (2007) uses a virtual world platform for real-time modifications, supported by an IFC-based database and an agent-based system managing user interactions. These frameworks highlight the importance of scalable, distributed systems in enabling collaboration and data management.

More recent studies (Novembri and Rossini, 2020) introduce a novel support system for building and architectural design that integrates BIM and distributed multi-agent optimization technologies. This system employs evaluator agents (Agent Swarm) interconnected with the "BIM World" via an API, functioning proactively to assess solutions, identify optimal choices, and modify BIM models.

Methodology

This section outlines the methodology used to develop a framework that integrates BIM with an actor-based system and Semantic Web technologies. This framework supports collaborative workflows by managing building elements as independent actors. Each actor manages its own state and responds to updates asynchronously through message passing. When a change occurs, the actor processes the update and communicates with other, actors and with the

[1] https://getakka.net/

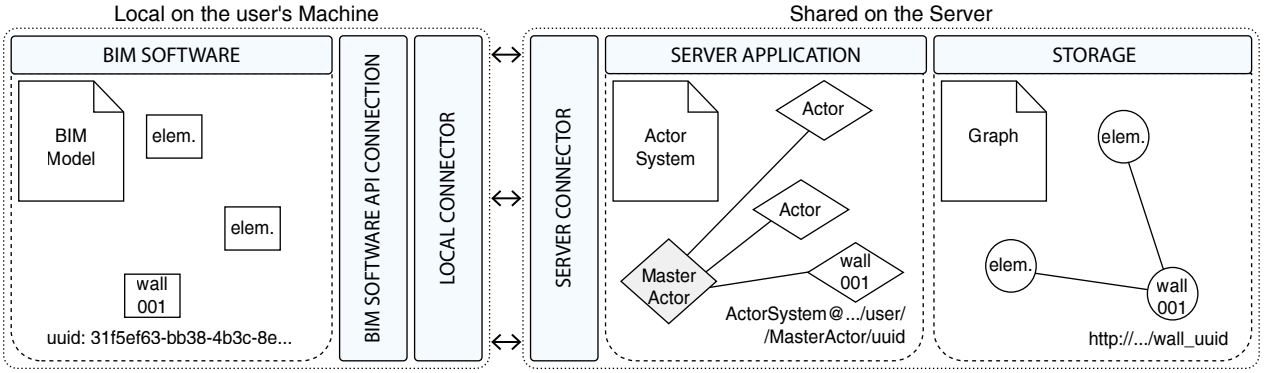

Figure 1: Diagram of the proposed system architecture

user, to maintain consistency. The adoption of Semantic Web technologies further enriches the system by structuring, linking, and managing data in a semantic and interconnected manner. To ensure interoperability across software platforms, the system employs non-proprietary data schemas that can then be exported to open data schemas and ontologies, such as the Building Ontology Topology (BOT), through the use of mapping files.

This framework moves away from a centralised model by distributing building elements as independent actors. This decentralisation is implemented on multiple levels: it includes distributing actors across multiple machines to handle increased workloads and ensure scalability, while also decentralising data storage through Linked Data, where each actor's data is stored in separate repositories that are interlinked. The decentralised approach enhances performance by allowing concurrent updates to different building elements without relying on a single central model, preventing bottlenecks. This is particularly beneficial in large AEC projects, where multiple stakeholders need to access and modify distinct parts of the model simultaneously.

Requirements analysis

The requirements analysis phase identifies the functional and non-functional needs of the system to ensure it meets the objectives of improving collaboration and interoperability in AEC workflows. Functionally, the system must integrate with BIM tools, such as Revit, to enable seamless data exchange and synchronization. To support collaborative workflows, each building element should be managed as an independent actor, enabling parallel and scalable collaboration. Additionally, BIM data must be converted into formats that comply with linked data standards to enable queries.

Non-functional requirements include scalability to handle large datasets and multiple users, security to ensure data integrity and controlled access, and user-friendliness to provide intuitive interfaces for non-technical stakeholders. These requirements collectively ensure that the system is robust, efficient, and accessible to all participants in the AEC process.

System architecture

The key element of this framework lies in the management of data associated with building elements in a decentralised repository managed by an Actor System. In this way data can be easily aggregated and users can retrieve only the information that they need, based on their permissions, and avoiding the need to export the entire model. This ensures stakeholders always have access to the latest model information for informed decision-making.

The system architecture is designed to facilitate seamless interaction between BIM tools, a local processing application, and a remote actor-based system, with a storage serving as the repository for building data. Figure 1 summarises the system's components.

At the core of the system is the BIM Authoring Tool, which is used to create and manage the BIM model. The BIM Authoring Tool provides the necessary environment for defining geometry, parameters, and marking changes to building elements. Each building element is represented in a proprietary format accessible via APIs. For this reason, the system includes an API connector/plugin that enables the interaction between the BIM Authoring Tool and the rest of the system. This connector is responsible for extracting data from the BIM model, detecting changes, and packaging them for transmission to the Actor System.

Then, a Local Connector that runs on the user's machine serves as the interface between the multiple BIM tools used by the user and the Actor System. This connector facilitates data export, change detection, and synchronization. It handles critical tasks such as mapping parameters, packaging changes, and managing data exchange to ensure accurate transmission of updates. The Local Connector runs a BIM-to-Actor mapping file that translates BIM data into a standardised format understood by the Actor System. The Local Connector securely communicates with the Actor System.

The Actor System manages building elements as independent actors. Each actor corresponds to a specific building element in the BIM model and is responsible for processing changes, updating its state, and interacting with the storage repository. The Actor System ensures that data

is processed and synchronised in a scalable, decentralised, and efficient manner, through supervisory actors that manage the lifecycle of individual actors. The Actor System also runs an Actor-to-Graph mapping file that aligns actor parameters with domain ontologies, ensuring that data is exported to the storage repository in a compliant format.

For data storage and management, the system employs a graph repository that stores BIM data in a machine-readable format, such as RDF. The repository serves as a centralised data store that enables queries, data linking, and interoperability. The graph format interconnects building elements and enables querying with standard languages like SPARQL. This enables stakeholders to access, analyse, and exchange data efficiently.

In an ideal scenario, the system would allow multiple users to work on different machines on their own BIM software. Furthermore, the Actor System could be deployed either on the local machine or on one or multiple remote servers. Similarly, the GraphDB repository could be hosted on a local or on one or multiple remote server. This flexibility allows for a scalable and distributed architecture that can accommodate various project sizes and team structures.

It is hereby very important that in this agent-based approach, small-scale updates should be targeted, where information for individual elements are communicated, rather than complete models. This aligns with a web-based communication strategy, and an approach where snippets are sent back and forth between servers and clients (or peers), using small, per-element files (JSON, RDF, etc.).

Framework Implementation

In Table 1 the components and tools used for the implementation of the system are listed. In particular, on the user side, Autodesk Revit was used as the BIM authoring tool, a custom Revit plugin was developed to interact with its APIs, and the Local Connector was implemented as a custom C# application. On the server side, the Actor System was built using the Akka.NET framework, and the storage was implemented using GraphDB as an RDF repository. Custom mapping files were developed to convert BIM data into actor representations, and from this to RDF format.

Workflow

The system was implemented and tested in two main phases: initial setup and model editing. The initial setup phase involved creating actors for each building element, exporting RDF data, and uploading it to the RDF repository. The model editing phase focused on detecting changes in the BIM model, packaging updates, and uploading them with the RDF repository.

Initial Setup. The initial setup begins with a new or existing Revit model being opened. The Local Connector is automatically lauched and it receives the model data from the Revit plugin. The Local Connector packages these updates using the Revit-to-Actor mapping file. This file translates proprietary Revit parameters into a format understood by the Actor System. For each element with an assigned UUID, a message is then sent to the remote Actor System. Upon receiving the message, this generates a unique actor for each building element, assigning its properties to a dictionary. In Figure 2, a code snippet shows the creation of an actor for a building element, when a message is received.

```
namespace ActorSystemServer
{
    public class ElementActor : ReceiveActor
    {
        private string? Uri { get; set; }
        public string Uuid { get; private set; }
        public string Name { get; private set; }
        public string Category { get; private set; }
        public Dictionary<string, string> Properties { get; set; } = new();

        public ElementActor(string category, string uuid)
        {
            Uuid = uuid;
            Category = category;
            Properties = ActorParameters.List[Category].ToDictionary(parameter => parameter, _ => "");


            foreach (var parameter in elementParameters)
            {
                Properties.Add(parameter, "");
            }

            Receive<ActorSetup>(msg =>
            {
                Uuid = msg.Uuid;
                Name = msg.Name;
                Category = msg.Category;

                foreach (var (key, value) in msg.Properties)
                {
                    Properties[key] = value;
                }

                Uri = CreateUri(this, IfcExport.Namespace);
            });
        }
    }
}
```

Figure 2: Code snippet showing the creation of an actor for a building element

Then, a full RDF export of its data, including geometry, properties, and relationships is done by each actor, mapping actor properties to known ontologies such as BOT. An Actor-to-Graph mapping file aligns the Actor System's generic parameters with domain ontologies, ensuring compliant RDF exports and semantic interoperability. This RDF data is uploaded to GraphDB, establishing a distributed repository. The system is now ready to manage building elements as independent actors and synchronise changes with the repository.

Model Editing. During editing, the plugin detects changes to Revit elements via hash string comparisons,

Table 1: Components and tools used in the system

Component	Tool	Description
BIM Authoring Tool	Autodesk Revit	Used to create and manage the BIM model.
BIM API Connector	C# Revit Plugin	Interfaces between the BIM tool and Local Connector, facilitating data export and change detection.
Local Connector	Custom C# Application	Interfaces between the BIM tool and the remote Actor System, mapping BIM data to actor representations.
Server Application	Akka.NET Actor System	Manages building elements as independent actors, exchanging messages.
Mapping Files	Custom JSON files	BIM-to-Actor and Actor-to-Graph mapping files for data conversion.
Storage	GraphDB	Stores and manages BIM data in RDF format.

identifying differences in geometry and property values. In the current implementation, changes to building elements are synchronised to the RDF graph by updating the corresponding triples representing the element's current state. The system currently does not implement full historical versioning; previous states are overwritten upon new updates from the Actor System. Packaged changes are sent to the Actor System, where a Master Actor routes them to the relevant Element Actors by UUID. Upon receiving updates, actors modify their internal states and execute SPARQL queries to synchronise changes in GraphDB.

Figure 3 shows a simplified version of the sequence inteactions during the creation of a new element in the BIM environment. When the BIM software plugin detects a new element, it setups element data in a package, that is then sent to the Local Connector. The Local Connector, upon receiving the message, maps the parameters through the Revit-to-Actor mapping file and sends a message containing the new package to the Actor System. The Actor System receives the message and creates a new Element Actor for the new element, redirecting the package to it. The Element Actor then initialises its internal state and properties, maps the parameters through the Actor-to-Graph mapping file, and creates a new RDF representation of the element. Finally, the Element Actor and uploads its data to the GraphDB repository.

Results

The proposed framework was implemented and tested in a controlled environment using a small-scale BIM model. The model consisted of some building elements, including walls, windows, and doors, with a limited number of parameters. The BIM model was running on Autodesk Revit on a local machine. The Local Connector was installed on the same machine as Revit to facilitate communication between the BIM tool and the Actor System. The Actor System and the GraphDB repository were hosted on a remote server to simulate a distributed environment.

The system was tested by one user who performed a series of edits on the BIM model, including adding new elements and modifying existing ones. The Actor System created new actors for the added elements and updated the states of modified elements.

The system successfully demonstrated its ability to manage building elements as independent actors, communicate updates asynchronously, and ensure data consistency through RDF-based storage. The dual-layer mapping approach, using BIM-to-Actor and Actor-to-Graph files, facilitated interoperability between Revit and the repository, enabling easy data exchange and propagation of changes.

Quantitative data such as response times or system throughput under multi-user editing scenarios was not collected, but it will be part of future work.

Discussion

The implemented framework shows a novel approach to managing BIM data through an actor-based system. The combination of decentralised actors, RDF storage, and service-oriented architecture provides a robust solution for handling BIM models while maintaining data consistency, accessibility, and interoperability.

The dual-layer mapping approach, utilizing both Revit-to-Actor and Actor-to-Graph mappings, addresses the challenge of vendor lock-in by providing a standardised intermediate format. This approach facilitates interoperability between different BIM authoring tools while ensuring compliance with established ontologies in the AEC domain.

Platforms like Speckle[2] have proven effective in enabling data exchange and collaboration across diverse tools and workflows, streamlining processes by avoiding full model exports and supporting granular updates. Our framework builds on these principles but extends them further by introducing semantic interoperability and decentralised data management and, by leveraging RDF, our system enables richer data integration.

[2]https://www.speckle.systems/

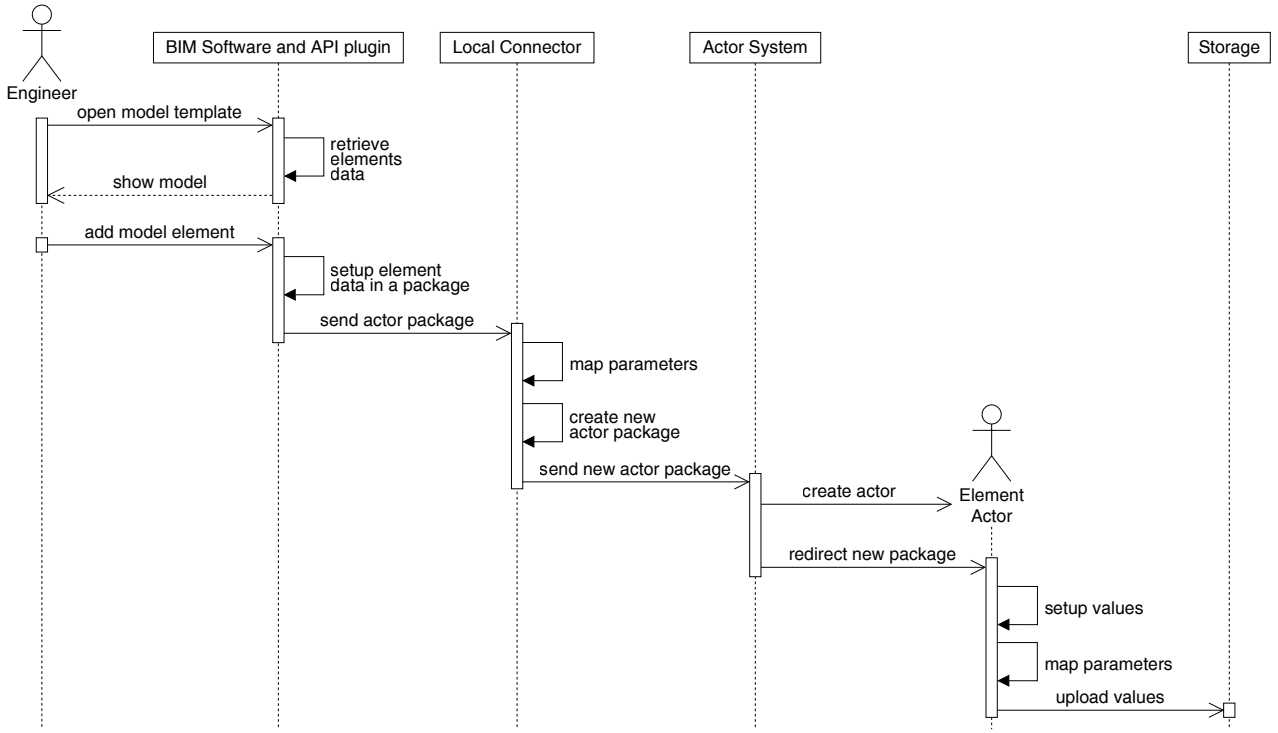

Figure 3: Simplified sequence diagram showing only key steps in element and actor creation

Unlike Speckle, which relies on a centralised server architecture for data synchronization, our framework employs an actor-based system that decentralises data management. This allows building elements to function as independent entities, communicating asynchronously, which enhances scalability and fault tolerance—particularly beneficial for large-scale projects with multiple stakeholders.

Another key distinction lies in data representation. Speckle uses a lightweight JSON-based format optimised for performance and ease of use, while our framework adopts RDF and linked data technologies. This enables queries and seamless integration with domain ontologies. In this context, the ongoing development of IFC5 represents a promising direction[3]. IFC5 aims to align more closely with linked data principles, potentially bridging the gap between traditional object-based schemas and graph-based, semantically rich data models. Incorporating IFC5 in future iterations could streamline the mapping process, improve interoperability, and further reduce reliance on proprietary formats.

Challenges and limitations

In this research, some challenges and limitations were identified, which relate to performance, granularity, and maintenance. These are described in detail below.

Performance Overhead. The translation between proprietary BIM formats and RDF could introduce significant processing overhead. This is particularly noticeable in large-scale models with frequent updates, where the system must handle a high volume of data transformations. While the use of RDF and ontologies enables semantic interoperability, the additional computational steps required for data conversion and validation can impact system responsiveness, especially in real-time collaboration scenarios.

Granularity of Control. Determining the optimal granularity for actor representation is a key challenge. Representing each building element as an independent actor ensures precise control over changes but increases system complexity and communication overhead. Conversely, grouping elements into larger actors simplifies communication but may reduce the system's ability to track and propagate fine-grained changes. Striking the right balance between granularity and performance is critical for scalability, particularly in projects with thousands of elements and multiple stakeholders.

It is worth noting that although actor-based systems support distribution across machines, this is not a requirement. In our current implementation, actors are hosted on a single machine for simplicity and easier debugging. However, the architecture is designed to support distributed deployment. Distributing actors across nodes could offer performance improvements, scalability, and fault tolerance, particularly in scenarios involving large models, multiple users, or geographically dispersed teams. Distribution strategies may include colocating actors based on model hierarchy, project phase, or access frequency, allowing for flexible optimisation based on specific needs. This would enable a more fine-grained granularity of con-

[3]https://www.buildingsmart.es/2024/12/03/the-evolution-of-ifc-the-path-to-ifc5/

trol over the model.

Maintenance Overhead. The system requires regular updates to mapping files when new parameters, relationships, or ontologies are introduced. This maintenance overhead can be a barrier to adoption, especially in dynamic projects where data schemas evolve frequently. Additionally, concurrent modifications to the same building element by multiple users can lead to synchronization conflicts that require resolution. Developing robust conflict resolution mechanisms is essential to ensure data consistency and integrity in collaborative workflows.

Future work

While the decentralised design prevented central model bottlenecks, performance issues were also observed in the Local Connector when processing large batches of updates. Future work should investigate optimisation of this component for parallel processing. The system could benefit from performance optimizations to reduce the overhead of RDF translations and Actor System communications, particularly when dealing with large-scale models. This might involve refining the actor hierarchy and message routing strategies, as well as exploring data streaming techniques to enhance scalability.

Furthermore, data ownership and governance issues should be addressed to ensure that stakeholders have clear roles and responsibilities in managing the shared data. This could involve implementing role-based access control to track data modifications and ensure accountability. Also, Ontology for Property Management (OPM) [4] could be integrated to support versioning and temporal tracking of building element properties within the RDF repository. OPM is specifically designed to represent how properties change over time as a building design evolves. Incorporating OPM would allow the system to preserve the history of each element's state transitions rather than overwriting values, supporting time-aware queries, design audits, and improved traceability throughout the project lifecycle.

Lastly, the development of testing methodologies for validating data transformations across system components would also enhance reliability and maintainability. Compliance actors could be introduced to ensure that data is compliant with industry standards and ontologies, providing a mechanism for automated validation and verification of BIM data. These actors could be responsible for checking the integrity and consistency of data before it is uploaded to the RDF repository, ensuring that only valid and compliant data is stored. Furthermore, they could be used to check data against predefined rules or constraints, such as Information Delivery Specification (IDS) [5] requirements or project-specific guidelines. This would help maintain data consistency and quality across the system, reducing the risk of errors and inconsistencies in the BIM model.

[4] https://w3c-lbd-cg.github.io/opm/

[5] https://www.buildingsmart.org/standards/bsi-standards/information-delivery-specification-ids/

Conclusions

This research presented a novel approach to integrating BIM with actor-based frameworks and Semantic Web technologies, addressing challenges in interoperability, collaboration, and data consistency. By decentralizing data management through independent actors and leveraging RDF for semantic interoperability, the framework enables scalable collaboration and data exchange. The dual-layer mapping approach mitigates vendor lock-in, ensuring compatibility across diverse BIM tools.

The implementation of an actor-based system enhances collaboration by enabling asynchronous updates and efficient propagation of changes. This eliminates the inefficiencies of traditional manual data exports and fosters more informed decision-making through immediate access to updated information. The approach not only improves the efficiency of collaborative workflows but also aligns with the industry's push towards digital transformation and interconnected processes.

However, challenges remain in terms of performance, granularity, and maintenance. Addressing the limitations through further research and testing in real-world case studies will be critical for the system's broader applicability and success.

Acknowledgments

The author E. De Santis acknowledges the financial support by Project ECS 0000024 Rome Technopole, - CUP B83C22002820006, NRP Mission 4 Component 2 Investment 1.5, Funded by the European Union - NextGenerationEU.

Author Contributions

E. De Santis: Conceptualization, Methodology, Software, Writing - Original Draft, Writing - Review & Editing.
P. Pauwels: Supervision, Writing - Review & Editing.

References

Afsari, K., Eastman, C., and Shelden, D. (2017). Building Information Modeling data interoperability for Cloud-based collaboration: Limitations and opportunities. International Journal of Architectural Computing, 15(3):187–202.

Almamlook, R., Bzizi, M., Al-Kbisbeh, M., Ali, T., and Almajiri, E. (2020). Factors Affecting Labor Productivity in the Construction Industry. American Journal of Environmental Science and Engineering, 4(2):24–30.

Beetz, J., van Berlo, L., de Laat, R., and van den Helm, P. (2010). Bimserver.org - an Open Source IFC model server. In Proceedings of the CIP W78 Conference, pages 1–8. International Council for Research and Innovation in Building and Construction (CIB).

Carrara, G., Fioravanti, A., Loffreda, G., and Trento, A. (2017). Knowledge Collaboration Design: Theory,

Techniques and Applications for Collaboration in Architecture. Gangemi editore SpA international, Roma.

Charousset, D., Hiesgen, R., and Schmidt, T. C. (2016). Revisiting Actor Programming in C++. Computer Languages Systems & Structures, 45:105–131.

Chechina, N., MacKenzie, K., Thompson, S., Trinder, P., Boudeville, O., Fördős, V., Hoch, C., Ghaffari, A., and Hernandez, M. M. (2017). Evaluating Scalable Distributed Erlang for Scalability and Reliability. Ieee Transactions on Parallel and Distributed Systems, 28(8):2244–2257.

Costin, A. and Eastman, C. (2019). Need for Interoperability to Enable Seamless Information Exchanges in Smart and Sustainable Urban Systems. J. Comput. Civ. Eng., 33(3):04019008.

Egan, J. (1998). Rethinking construction: The report of the construction task force. Technical report, Department for Trade and Industry, London: HMSO.

Esser, S., Vilgertshofer, S., and Borrmann, A. (2022). Graph-based version control for asynchronous BIM collaboration. Advanced Engineering Informatics, 53:101664.

Grilo, A. and Jardim-Goncalves, R. (2010). Value proposition on interoperability of BIM and collaborative working environments. Automation in Construction, 19(5):522–530.

Hagedorn, P., Block, M., Zentgraf, S., Sigalov, K., and König, M. (2022). Toolchains for Interoperable BIM Workflows in a Web-Based Integration Platform. Applied Sciences, 12(12):5959.

Hayduk, Y., Sobe, A., and Felber, P. (2015). Dynamic Message Processing and Transactional Memory in the Actor Model. In Bessani, A. and Bouchenak, S., editors, Distributed Applications and Interoperable Systems, pages 94–107, Cham. Springer International Publishing.

Hsu, C. L. (2024). Modelling Exchange With Blockchain for the Collaborative Design of a Building Envelope in BIM. The Open Construction and Building Technology Journal, 18(1).

Hui, S. (2023). The Impacts of 5D Building Information Modelling Towards Cost Management in the Construction Industry. Journal of Advanced Research in Applied Sciences and Engineering Technology, 32(3):471–487.

Latham, M. (1994). Constructing the team. Technical report, HMSO, London.

McKinsey Global Institute (2017). Reinventing construction: A route to higher productivity. Technical report, McKinsey & Company.

Nidhal Neamah, A.-S., Raimar, S., and Karsten, M. (2023). From static to dynamic information containers. In Computing in Construction, volume 4, Heraklion, Greece. European Council on Computing in Construction.

Novembri, G. and Rossini, F. L. (2020). Swarm modelling framework to improve design support systems capabilities. ITcon, 25:398–415.

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

Rosenman, M. A., Smith, G., Maher, M. L., Ding, L., and Marchant, D. (2007). Multidisciplinary collaborative design in virtual environments. Automation in Construction, 16(1):37–44.

Sattler, L., Lamouri, S., Pellerin, R., Fortineau, V., Larabi, M., and Maigne, T. (2021). A query-based framework to improve BIM multi-domain collaboration. Enterprise Information Systems, 15(10):1395–1417.

Schlachter, A., Rasmussen, M. H., and Karlshøj, J. (2022). Using Linked Building Data for managing temporary construction items. Automation in Construction, 139:104258.

Senthilvel, M. and Beetz, J. (2020). A Visual Programming Approach for Validating Linked Building Data. In EG-ICE 2020 Workshop on Intelligent Computing in Engineering, volume EG-ICE, pages 403–411. RWTH Aachen University.

Tokpayev, K. (2023). Handling HTTP Flood Attacks in High-Load Applications Using Akka Actors Model. Ingénierie Des Systèmes D Information, 28(3):655–662.

Werbrouck, J., Pauwels, P., Beetz, J., and van Berlo, L. (2019). Towards a decentralised common data environment using linked building data and the solid ecosystem. In Kumar, B., Rahimian, F., Greenwood, D., and Hartmann, T., editors, Advances in ICT in Design, Construction and Management in Architecture, Engineering, Construction and Operations (AECO) : Proceedings of the 36th CIB W78 2019 Conference, pages 113–123, Newcastle, UK.

Wijekoon, C., Manewa, A., and Ross, A. (2018). Enhancing the Value of Facilities Information Management (FIM) Through BIM Integration. Engineering Construction & Architectural Management, 27(4):809–824.

Yin, X. (2024). Research on Collaborative Management of BIM (Building Information Modeling) in Engineering Projects. Advances in Economics Management and Political Sciences, 115(1):49–55.