Chair of Computing in Civil and Building Engineering
TUM School of Engineering and Design
Technical University of Munich

Modeling:
Mathematics:
Programming:
Science:

Software Lab:

LSP Implementation for IFC

Description

Building Information Modeling (BIM) workflows rely heavily on the IFC standard as an open and vendor-neutral data format. IFC files are typically generated by BIM authoring tools rather than written manually in text editors. However, direct access to the textual representation remains highly valuable for **model inspection, debugging, education, research, and automated processing**. Despite its importance, developer tooling for raw IFC files remains limited. In particular, there is a lack of advanced editor support such as intelligent code completion, semantic navigation, and real-time validation.

The Language Server Protocol (LSP) enables advanced language features such as navigation, autocompletion, and error checking to work consistently across different code editors and IDEs by defining a generic client-server architecture.

The goal of this project is to design and implement a **Language Server Protocol (LSP) implementation for IFC**, enabling modern IDE features in any editor that supports the LSP standard (e.g., VS Code, IntelliJ, Neovim, Eclipse, Helix).

Students will develop a scalable and extensible language server that understands IFC schemas and model structures. The system will parse IFC files, build semantic representations, and provide real-time feedback to users. The project combines software engineering, formal language processing, and BIM domain knowledge.

Chair of Computing in Civil and Building Engineering
TUM School of Engineering and Design
Technical University of Munich

Task

- Familiarize yourself with the IFC specification and available open-source tooling (e.g., IfcOpenShell) as well as existing IFC developer tools (such as Alan Rynne's IFC Developer Tools) to understand current capabilities and limitations.
- Study the Language Server Protocol (LSP) specification and analyze existing language server implementations to understand architecture patterns and communication workflows.
- Collaboratively distribute responsibilities within the team (parser development, LSP integration, schema modeling, validation logic, testing, documentation).
- Design and implement a complete language processing pipeline including lexical analysis, parsing, AST construction, semantic model building, and incremental document updates.
- Integrate the language processing backend with the LSP interface and ensure compatibility with at least one reference editor (e.g., VS Code).
- Continuously test the system using real-world IFC datasets and keep in mind performance for large models.
- Prepare technical documentation and usage examples suitable for teaching and demonstration purposes.

Deliverables

The primary deliverable is a **fully functional IFC Language Server** that provides advanced editor support for IFC-based files and can be integrated into modern development environments.

The language server must include the following essential features:

- Syntax highlighting and semantic highlighting for IFC entities, attributes, and relationships
- Go-to-definition and find-references functionality
- Hover information displaying schema documentation and entity metadata
- Autocompletion suggestions based on schema context
- Real-time syntax and semantic validation diagnostics
- Document outline / symbol tree for structural navigation

The following features may be implemented to extend functionality:

- Schema version detection and compatibility checks (IFC2x3, IFC4, IFC4.3)
- Rule-based validation using EXPRESS constraints
- Performance optimization for large-scale models (incremental parsing, caching)

References:

- IFC Specification and Documentation: https://ifc43-docs.standards.buildingsmart.org/
- Language Server Protocol Specification: https://microsoft.github.io/language-server-protocol/specifications/lsp/3.17/specification/
- IFC Developer Tools: https://github.com/AlanRynne/ifc-developer-tools

Supervisor

Wolf Nepomuk, Chair of Computing in Civil and Building Engineering, nepomuk.wolf@tum.de