Chapter 1

Brief theoretical and conceptual background

1.1 Language Server Protocol (LSP)

The **Language Server Protocol (LSP)** is an open, JSON-RPC-based protocol designed to standardize communication between source code editors or integrated development environments (IDEs) and language servers that provide language-specific features such as code completion, syntax highlighting, error detection, and refactoring [1, 2]. LSP was originally developed by Microsoft for Visual Studio Code and has since become an open standard. The key motivation behind LSP is to decouple language intelligence from any specific editor, enabling language support to be implemented once and reused across multiple tools. This reduces the burden on both language providers and tool vendors, and allows for rapid integration of new languages into development environments.

LSP operates on a client-server model: the client (editor or IDE) sends requests and notifications about user actions to the server, which responds with language-specific services. Communication is performed using JSON-RPC messages, but the protocol does not mandate a specific transport mechanism, allowing flexibility in implementation [1, 2].

1.2 Graphical Language Server Platform (GLSP)

The **Graphical Language Server Platform (GLSP)** extends the LSP architectural pattern to graphical modeling and diagram editors [3, 4]. GLSP is a client-server framework for building web-based diagram editors, where the server encapsulates the language-specific logic for loading, interpreting, and editing diagrams. This architecture allows diagram editors to be easily integrated with various platforms such as VS Code, Eclipse Theia, or standalone web applications.

A GLSP server provides the backend logic, including model loading and transformation into a graphical model, which is then serialized and sent to the client. The client renders the diagram and requests information on editing capabilities. The client can then provide editing tools based on this information. When a user edits the diagram, the client notifies the server, which updates the source model and sends back the updated graphical model. GLSP is designed for extensibility and customizability, making it suitable for a wide range of graphical languages [3].

1.3 Visual Studio Code (VS Code)

Visual Studio Code (VS Code) is a popular open-source code editor developed by Microsoft, known for its extensibility and modular architecture [5]. VS Codes architecture is both layered and modular: The core provides utilities and UI building blocks, the platform manages services via dependency injection, and the *Monaco editor* handles text editing. The workbench layer hosts the editor and other UI components, all rendered by Electron.

VS Code's feature set is largely provided by extensions, which run in isolated processes for stability and security. Language support is delivered via LSP-compatible language servers, allowing VS Code to offer advanced features such as code completion, go-to-definition, and refactoring for a wide variety of programming languages [1, 5].

1.4 PlantUML

PlantUML is an open-source tool that enables users to create diagrams using a domain specific text based language [6]. It supports a wide range of diagram types, including UML diagrams (class, component, use case, activity, etc.), as well as other formats like Archimate, BPMN, and C4. PlantUML uses a domain-specific language (DSL) to describe diagrams, and relies on Graphviz for layout. It can output diagrams in various formats, including PNG, SVG, LaTeX, and ASCII art.

PlantUML's text-based approach makes it easy to version-control diagrams alongside source code, automate documentation generation, and integrate with other tools. It is widely used in software engineering education and industry for its simplicity and flexibility.

Bibliography

[1] Language Server Protocol. Wikipedia. https://en.wikipedia.org/wiki/Language_Server_Protocol

[2] Official page for Language Server Protocol - Microsoft Open Source. https://microsoft.github.io/language-server-protocol/

[3] Overview · Eclipse Graphical Language Server Platform. https://eclipse.dev/glsp/documentation/overview/

[4] Graphical Language Server Protocol Framework. https://github.com/eclipsesource/graphical-lsp

[5] VSCode - From Vision to Architecture. https://2021.desosa.nl/projects/vscode/posts/essay2/

[6] PlantUML. Wikipedia. https://en.wikipedia.org/wiki/PlantUML