Research On Program Code Analysis Techniques

1. Project Overview

This section provides a comprehensive description of the end-to-end solution designed for the master’s thesis titled “Research on Program Code Analysis Techniques.” The proposed system aims to perform detailed dependency analysis of TypeScript source code to assist developers in understanding and improving code modularity and structure.

The solution is structured as a complete workflow that begins with the collection of source code and ends with the generation of dependency matrices, graphs, and structured reports. It employs static and semantic analysis techniques without incorporating artificial intelligence or machine learning components.

1. Xs of the System (Inputs)

The inputs (X) of the system represent the data structures and artifacts that are analyzed to extract dependencies and relationships within the TypeScript codebase.

|  |  |
| --- | --- |
| Input Type | Description |
| TypeScript Source Code Repository | The complete set of .ts files forming the project under analysis. |
| Abstract Syntax Tree (AST) | A tree structure generated from the source code using the TypeScript Compiler API. |
| Symbol and Type Tables | Compiler-generated metadata that stores information about variables, functions, and their scopes. |
| Module Resolution Map | Mapping of imports and exports between TypeScript modules. |
| Dependency Graph Edges | Connections representing relationships between classes, functions, and modules. |

1. Ys of the System (Outputs)

The outputs (y) represent the results produced by the system after the dependency analysis is performed. They provide valuable insights into the internal structure and modular quality of the analyzed project.

|  |  |
| --- | --- |
| Output Type | Description |
| Dependency Matrix (DSM) | A matrix representation displaying relationships between modules. |
| Dependency Graph Visualization | A graphical depiction of inter-module and intra-module dependencies. |
| Semantic Dependency Classification | Categorization of dependencies into types such as import, export, or circular. |
| Structural Integrity Report | A summary describing code coupling, modularity, and dependency density. |
| Developer Report (JSON/HTML) | Generated reports with readable metrics and visual summaries of the analysis results. |

1. Example (X, y) Pairs for the /data/ Directory

The following examples illustrate how the system processes input data (X) and produces corresponding outputs (y). These examples will be documented and placed in the /data/ directory of the GitHub repository.

|  |  |  |
| --- | --- | --- |
| Example ID | X (Input) | y (Output) |
| Example 1 | Small TypeScript project with 3 modules and basic imports | Dependency Graph JSON highlighting a circular dependency between module A and C. |
| Example 2 | AST dump of a medium-sized TypeScript project | Dependency Matrix (DSM) summarizing coupling strengths between modules. |
| Example 3 | Code containing mixed import types | Classification report indicating 70% runtime and 30% type-only dependencies. |
| Example 4 | Project with nested module hierarchy | Structural Report identifying deep coupling between submodules. |
| Example 5 | Codebase before and after refactoring | Comparison output showing reduction in dependency count post-refactoring. |

1. End-to-end Workflow

The system operates as a sequential pipeline, where each stage transforms the input into a more structured form of analysis data. The workflow is described as follows:

1.Data Collection → Import TypeScript source code from the repository.  
2. Parsing Stage → Convert the code into Abstract Syntax Trees using the TypeScript Compiler API.  
3. Semantic Analysis → Resolve symbols, types, and scope relationships.  
4. Dependency Extraction → Detect import/export and function-level dependencies.  
5. Graph and Matrix Generation → Build the Dependency Graph and DSM representations.  
6. Structural Evaluation → Identify circular dependencies and coupling issues.  
7. Report Generation → Create structured outputs and visual summaries for developers.