1. SonarQube

**Website:** https://www.sonarsource.com//

1. **Code Comprehension Support:**

SonarQube is an excellent tool for static code analysis. It gives precise information on code quality, including defects, vulnerabilities, and code smells. Its dashboards and visualizations allow developers to immediately grasp the state of their codebase.

1. **Code Translation Support:**
2. SonarQube does not directly allow code translation. It prioritizes code analysis to improve quality and maintainability over code translation from one language to another.
3. **Support for Extracting Design Artifacts**:
4. SonarQube's analysis can extract certain design artifacts, such as duplicated code and architectural smells, which can indicate design flaws.

**Advantages:**

1. Comprehensive analysis with support for multiple languages.
2. Integration with various CI/CD pipelines.
3. Extensible through plugins.

**Disadvantages:**

1. Analysis can be resource-intensive.
2. It requires some configuration and setup effort.
3. Support for legacy languages is limited.
4. **JArchitect**

**Website:** *[https://www.jarchitect.com/](https://www.jarchitect.com/%20/)*

**Code Comprehension Support:**

JArchitect is intended for the analysis of Java code. It offers thorough insights into the dependencies, structure, and quality of the code, which makes maintenance and comprehension easier.

**Code Translation Support:**

Code translation is not supported by JArchitect. Its main objective is to analyse Java codebases for architectural and quality problems.

**Support for Extracting Design Artifacts:**

Developers can better understand the design and architecture of their code by using JArchitect, which can extract a variety of design artefacts such dependency graphs, code metrics, and architecture diagrams.

**Advantages:**

thorough examination specific to Java.   
aids in codebase refactoring and maintenance.   
thorough reporting and graphics.

**Disadvantages:**

limited to Java.   
can be difficult to set up and adjust.   
Higher price for versions with all the features.

1. **cppdepend**

**website**: <https://www.cppdepend.com/>

1. **Code comprehension support:**

Yes, CppDepend provides robust support for code comprehension. It includes features such as dependency graphs, code query languages (CQLinq), and code metrics that help developers understand complex codebases. These features enable users to visualize and navigate through the code structure, identify dependencies, and understand the relationships between different parts of the code.

1. **Code translation**

CppDepend does not primarily focus on code translation. Its core functionalities are centered around code analysis, quality assessment, and comprehension. While it can analyze and provide insights on the code, translating code from one language to another is not within its main feature set.

1. **Support for extracting artifacts design**

Yes, CppDepend supports the extraction of design artifacts from code. It can generate various diagrams such as dependency graphs and sequence diagrams that represent the architecture and design of the software. These visual tools help in understanding the design and architecture, making it easier to extract and document design artifacts.

1. **advantages**

· C**omprehensive Code Analysis:** Provides detailed code metrics, dependency graphs, and technical debt estimations that help in understanding and improving code quality.

· **Customizable Queries:** The CQLinq feature allows users to write custom queries to explore code in ways that are specific to their needs, providing flexibility in code analysis.

· **Visualization Tools:** Offers various visualization tools like dependency graphs and sequence diagrams, which help in understanding the code structure and design.

· **Integration:** Integrates well with Visual Studio and CI/CD pipelines, making it a seamless part of the development workflow.

· **Technical Debt Management:** Helps in identifying and managing technical debt through detailed reporting and actionable insights.

1. **disadvantages**

· **Learning Curve:** Due to its extensive features and capabilities, there can be a steep learning curve for new users.

· **Complexity for Small Projects:** For smaller projects, the tool might be overkill, as the detailed analysis and metrics may be more than necessary.

· **Cost:** CppDepend is a commercial tool, which may be a disadvantage for small teams or individual developers due to the associated costs.

· **Performance Overhead:** Running comprehensive analyses on large codebases can be resource-intensive and may impact performance.

1. **Sourcetrail:**

**Website:** https://www.sourcetrail.com/

1. **Code Comprehension Support:**

Source trail's interactive dependency graph provides strong support for code comprehension. Users are able to efficiently browse the code base, visualize code relationships, and comprehend complicated inter dependencies.

1. **Code Translation Support:**

Translation is not Source trail's primary goal; rather, it is code understanding. It doesn't offer direct code translation features, but it does let users investigate code bases in-depth.

1. **Support for Extracting Design Artifacts**:

With its ability to visualize inheritance structures, class hierarchies, and method invocations, Sourcetrail is an excellent tool for extracting design artifacts from code. This aids in comprehending the patterns and design concepts applied throughout the program.

**Advantages:**

* Dependency graph that is interactive and intuitive for simple code exploration.
* supports a variety of programming languages, such as Python, C/C++, Java, and so on.
* Large codebase indexing that is quick and lightweight. Frequent updates and receptive community assistance.

**Disadvantages:**

* Restricted assistance in reworking or translating code.
* A premium subscription may be needed for certain functionality, such as live code synchronization.
* May occasionally have problems with specific IDEs or codebases in terms of compatibility.

1. Understand

**Website:** *<https://scitools.com/>*

**Code Comprehension Support:**

With its powerful code analysis features, Understand offers thorough support for code comprehension. To help in understanding complex code bases, it provides cross-referencing, visualization tools, and extensive analytic.

**Code Translation Support:**

Understands primary functions are code analysis and comprehension, but because to its refactoring capabilities, it also offers some assistance with code translation. The tool allows users to carry out simple code rearrangement activities.

**Support for Extracting Design Artifacts:**

Through the creation of intricate dependency matrices, call graphs, and class diagrams, Understand excels in removing design artifacts from code. This aids in the analysis of software architecture and the discovery of design errors.

**Advantages:**

* Capabilities for deep code analysis with intricate metrics and visualization.
* Supports a large number of programming languages, such as Python, Java, C/C++, and others.
* Scripting and plugins allow for customization and expansion.Suitable for both small and large code bases.

**Disadvantages:**

* Steeper learning curve than that of certain other instruments.
* Comparatively more expensive enterprise licenses.
* Occasionally, very huge code bases might cause performance concerns.