РАЗРАБОТКА БИБЛИОТЕКИ ПАРСЕРА ДЛЯ ЯЗЫКА СПЕЦИФИКАЦИИ LIBSL

ЛАРИОНОВ Н.М.

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»

им. В.И. Ульянова (Ленина)

Аннотация. В статье описывается процесс проектирования и разработки библиотеки парсера для языка спецификации библиотек LibSL. Приводится краткое описание языка LiSL и рассматриваются существующие подходы к использованию языка спецификации. Обосновывается важность разработки библиотеки парсера на языке C++ и перечисляются необходимые инструменты, выбранные для этой задачи. Описываются процессы проектирования, разработки библиотеки при помощи генератора парсеров antlr4 и тестирования, после чего указывается, как библиотека может быть в дальнейшем модифицирована и в каких областях может использоваться.

Ключевые слова: Спецификация библиотек, Язык спецификации, Парсер грамматики, Генератор парсеров.

Введение

Использование сторонних библиотек является неотъемлемой частью процесса разработки современных программных продуктов. Однако, сторонние разработчики зачастую делают исходный код библиотек закрытым из-за чего у пользователя библиотеки не всегда есть возможность ознакомится со всеми нюансами работы библиотеки. Это может быть связано с ограниченными временными ресурсами, предоставляемыми сотрудникам на ознакомление с документацией, или же сопровождающая документация может описывать библиотеку не в полном объеме. В таких случаях, применение библиотек далеко не всегда положительно складывается на техническом качестве программного продукта.

Альтернативным подходом спецификации библиотек является применение языков спецификации. Одним из языков спецификации является язык спецификации библиотек LibSL [1]. LibSL позволяет описывать поведение библиотеки в виде системы взаимодействующих расширенных конечных автоматов. Описание поведения библиотеки при помощи языка спецификации должно являться менее ресурсоемкой задачей, чем составление полноценной текстовой спецификации. Помимо этого, полученная спецификация библиотеки может быть использована в различных автоматизированных процессах, вроде поиска ошибок интеграции проекта с библиотекой [2] или автоматизированной миграции проекта с одной библиотеки на другую [3].

Для языка LibSl разработаны инструменты для его использования и интеграции в проекты. Для проверки грамматики файлов, содержащих спецификацию LibSl, разработаны парсер в формате библиотеки, написанный на языке Kotlin и плагин для Visual Studio, использующий данную библиотеку [4]. Однако для работы они требуют наличия JVM с подходящей версией, что может вызывать трудности при интеграции парсера с проектом, использующим другую версию JVM или и вовсе другой язык программирования. Для расширения доступности применения языка спецификации LibSL была поставлена задача разработать библиотеку парсера языка LibSL на языке C++.

Проектирование библиотеки парсера

В соответствии с поставленной задачей были сформированы требования, которые определяют дальнейшую структуру библиотеки и используемые для выполнения задачи инструменты. Так как работа направлена на расширение доступности языка LibSL, то основным требованием является универсальность библиотеки – использование библиотеки парсера должно минимально зависеть от особенностей применяемой операционной системы и среды разработки. Вторым требованием является простота использования библиотеки – расширение доступности языка LibSL должно обеспечиваться за счет упрощенного взаимодействия пользователя с библиотекой парсера. Завершающим требованием является удобство сопровождение библиотеки – структура библиотеки парсера должна быть составлена таким образом, чтобы пользователи имели возможность устранять возможные неточности библиотеки парсера и модифицировать ее с минимальными усилиями.

Исходя из определенных ранее принципов, был составлен следующий набор решений, используемый для реализации библиотеки. Компилятор был выбран из набора инструментов MinGW-w64, так как он поддерживает работу и на ОС Windows и на дистрибутивах ОС Linux. Кроме этого, MinGW-w64 имеет в своем составе линкер, среду выполнения и стандартные библиотеки. В качестве средства автоматизированной сборки было выбрано средство Ninja, которое также доступно и на ОС Windows и на ОС Linux. Для работы со сборочными файлами был выбран генератор сборочных файлов CMake, который предоставляет возможность гибкой настройки выполняемых сборочных процессов и имеет встроенную поддержку генерации сборочных файлов для Ninja, что позволяет значительно упростить сборку библиотеки парсера.

C++ позволяет реализовывать статические и динамические варианты библиотек. При использовании, статические библиотеки компилируются целиком вместе с использующей их программой, а динамические позволяют подгружать только вызываемые компоненты при их вызове. Однако, динамические библиотеки имеют особенности, связанные с структурой заголовочных файлов, поэтому было принято решение разработать первичный вариант библиотеки парсера в статическом экземпляре для использования на ОС Windows.

Разработка библиотеки парсера

Язык спецификаций библиотек LibSL описан при помощи языка описания грамматик ANTLR. Таким образом, сформировать файлы, описывающие логику парсера языка LibSL на языке C++ можно при помощи генератора парсеров ANTLR4, что значительно упрощает разработку библиотеки парсера. Полученный парсер языка LibSL состоит из двух компонентов – лексера и парсера. Лексер получает на вход поток данных на языке LibSL и преобразует его в поток лексических токенов. Парсер обрабатывает поток токенов, проверяет их на предмет совпадения с грамматикой и далее на их основе строит лексическое дерево.

Для работы такого парсера необходимы сторонние компоненты – библиотека, содержащая генератор парсеров и среда выполнения, необходимая для работы парсера. Для сборки готовой библиотеки парсера необходимо сгенерировать файлы с логикой парсера, собрать среду выполнения ANTLR4, скомпилировать главный класс библиотеки, который линкуется вместе со средой выполнения в единый исполняемый файл. Полученную структуру библиотеки можно рассмотреть на рисунке 1.

Рис. 1. Структура библиотеки

На рисунке изображена структура библиотеки, используемой тестовой программой executable. Файлы библиотеки расположены в папке libsl-parser, состоящей из папок generated, grammar, include и runtime. Файлы, сгенерированные на основе файлов грамматики языка LibSL, содержащихся в папке grammar, хранятся в папке generated. Они генерируются при помощи библиотеки генератора парсера ANTLR, содержащейся в папке libs. Классы LibSLLexer и LibSLParser обеспечивают работу лексера и парсера соответственно. Главный класс LibSL.cpp позволяет использовать методы из сгенерированных файлов. Файлы среды выполнения содержатся в папке runtime. Во время сборки библиотеки парсера, главный класс библиотеки парсера и сгенерированные файлы парсера линкуются вместе со средой выполнения runtime в единый исполняемый файл, подключаемый и используемый в тестовой программе. Тестовая программа может решать различные задачи, например проверять файлы спецификации на языке LibSL из папки examples на предмет их соответствия грамматике языка спецификации.

Вывод

В ходе работы была разработана статическая библиотека парсера языка спецификации LibSL на языке C++ для использования на ОС Windows. Библиотека парсера позволяет пользователю работать пользователю со спецификацией на языке LibSL, обеспечивая проверку соответствия спецификации грамматике LibSL. Библиотека парсера может быть в дальнейшем доработана доступными средствами для использования на дистрибутивах

ОС Linux. В случае необходимости, может быть разработан также динамический вариант библиотеки. Разработанная библиотека парсера может быть в будущем использована в различных проектах, например для проверки файлов спецификации, используемых для автоматизированной миграции приложения с одной библиотеки на другую.

Список литературы

1. Ицыксон В. М. LibSL-язык спецификации компонентов программного обеспечения //Программная инженерия. – 2018. – Т. 9. – №. 5. – С. 209.
2. Feofilaktov V., Itsykson V. SPIDER: Specification-Based Integration Defect Revealer //International Conference on Software Testing, Machine Learning and Complex Process Analysis. – Cham : Springer Nature Switzerland, 2021. – С. 107-119.
3. Aleksyuk A. O., Itsykson V. M. Semantics-driven migration of Java programs: a practical application //Automatic Control and Computer Sciences. – 2018. – Т. 52. – С. 581-588.
4. Мартюшев В.Д. Плагин для VSCode для языка спецификации библиотек LibSL. – URL: https://nnov.hse.ru/ba/se/students/diplomas/925717980 (Дата обращения: 18.03.2025)