УДК 004.4'23

ОБОБЩЕННАЯ ОБЪЕКТНАЯ МОДЕЛЬ ПРОГРАММНОГО ПРОЕКТА

Ильинская О.В. (Университет ИТМО)
Кореньков Ю.Д. (Университет ИТМО)
Научный руководитель – кандидат технических наук Кореньков Ю.Д.
(Университет ИТМО)

В современном программировании одной из ключевых задач является эффективное управление структурой программного проекта и процессом его сборки. Различные системы сборки предлагают разнообразные инструменты и подходы, однако из-за их разнообразия при разработке проектов, сочетающих созданные в рамках разных систем сборки компоненты, возникает необходимость в создании универсального средства, способного объединить и координировать работу таких систем.

Целью данного исследования является разработка обобщенной объектной модели программного проекта, которая будет обеспечивать согласованность при управлении структурой и сборке программного проекта прозрачно контролируя отдельные системы сборки, задействованные для отдельных компонентов проекта. В рамках данной работы будет рассмотрена модель жизненного цикла программного проекта, включая этапы инициализации, запуска, конфигурации и другие аспекты, учитывая параллельность и особенности различных систем сборки.

В рамках исследования были рассмотрены несколько популярных систем сборки, таких как Apache Maven, Gradle, CMake и прочие. Каждая из этих систем имеет свои особенности и преимущества. Например, Apache Maven обеспечивает управление зависимостями и построением проекта на основе XML-конфигураций, в то время как Gradle предлагает гибкую систему сценариев на языке Groovy.

Были рассмотрены основные проблемы, возникающие при интеграции различных систем сборки:

1. Форматы конфигурационных файлов: Различные системы сборки могут использовать разные форматы конфигурационных файлов (например, XML, JSON, Groovy и т. д.), что затрудняет переход от одной системы к другой и требует дополнительной работы по конвертации файлов.
2. Зависимости и плагины: Каждая система сборки имеет свои собственные механизмы для управления зависимостями и плагинами. Это может привести к конфликтам при интеграции проектов, особенно если они используют разные версии зависимостей или плагинов.
3. Синтаксис команд и скриптов: Различные системы сборки имеют различный синтаксис для выполнения команд и написания сборочных скриптов.

Модель жизненного цикла программного проекта, рассмотренная в исследовании, включает этапы инициализации, компиляции, тестирования, упаковки и развертывания. Каждый из этих этапов имеет свои особенности и требования, которые необходимо учитывать при разработке обобщенной объектной модели.

При анализе параметров сборки было уделено внимание таким ключевым аспектам, как управление версиями зависимостей, настройки компилятора, параметры оптимизации компилируемого программного кода и другие параметры, которые могут влиять на процесс сборки и работу программного продукта

Основными компонентами, реализующими разработанную обобщённую модель программного проекта и обеспечивающими интеграцию различных систем сборки, являются:

1. Модуль управления конфигурацией. Отвечает за хранение и управление конфигурационными данными. Реализует операции чтения——и записи конфигурационных файлов.
2. Модуль управления зависимостями. Отвечает за управление внешними зависимостями проекта, их загрузку ~~р~~разрешение конфликтов.
3. Модуль выполнения операций сборки. Отвечает за выполнение операций сборки проекта, таких как компиляция кода, упаковка приложения, управление автоматизированными тестами и другие операции. Также включает в себя механизмы контроля выполнения операций.
4. Модуль управления проектом. Отвечает за управление структурой и состоянием проекта, его внутренними компонентами, и зависимостями между ними.

В результате работы была разработана обобщенная объектная модель программного проекта, учитывающая особенности различных систем сборки, реализуемая в виде набора библиотек. Создаваемое с использованием данных библиотек средство поддержки программирования и создания программных проектов может обеспечить повышению эффективности и надежности процесса сборки программных проектов.

Список использованных источников:

1. Apache Maven. (2021). Apache Maven Documentation. https://maven.apache.org/guides/index.html
2. CMake. (2021). CMake Documentation. https://cmake.org/documentation/
3. Gradle. (2021). Gradle User Guides. https://docs.gradle.org/current/userguide/userguide.html