DOI 10.37882/2223-2966.2023.05.17

СОВРЕМЕННЫЕ ПОДХОДЫ К АВТОМАТИЧЕСКОМУ ПРОГРАММИРОВАНИЮ

CURRENT APPROACHES TO AUTOMATIC PROGRAMMING

Yu. Latin

Summary: The article is devoted to the study of modern approaches to automatic programming. The advantages and advantages of automatic code generation are outlined in the course of the study. The capabilities and features of some methods and frameworks that are used for automatic programming are considered. The article also describes the author's patented Automated Code Generation System «Autocode». The algorithm of its implementation is highlighted separately.

Keywords: automated programming, code, framework, generation, speed, error.

Латин Юрий Эдуардович
Генеральный директор Bell Integrator (АО Бэлл Интегратор);
Казахский Государственный Национальный Университет им. Аль-Фараби (Факультет Физика), Казахстан, г. Алматы
yury.latin@gmail.com

Аннотация. Статья посвящена изучению современных подходов к автоматическому программированию. В процессе исследования обозначены преимущества и достоинства автоматической генерации кода. Рассмотрены возможности и особенности некоторых методов и фреймворков, которые используются на современном этапе. Также в статье описана авторская запатентованная разработка Автоматизированная система кодогенерации «Автокод». Отдельно выделен алгоритм ее реализации и отличительные черты по сравнению с аналогами.

Ключевые слова: автоматическое программирование, код, фреймворк, генерация, скорость, ошибка.

С непрерывным углублением информационного века, развитием цифровизации и проникновением передовых технологий во все сферы жизни, значительно расширился спектр видов и типов программного обеспечения, кроме того, к нему выдвигаются все более высокие требования [1]. В данном контексте вопрос о том, можно ли повысить эффективность создания программ при соблюдении жестких требований к их качеству, чтобы быстро реагировать на требования рынка, является актуальным и практически значимым.

Хотя большинство существующих подходов к разработке программного обеспечения предполагает использование различных моделей, таких как водопадная, фонтанная, спиральная и инкрементная, моделирование системы в основном направлено на анализ и ее проектирование на ранней стадии разработки, в то время как последующая реализация по-прежнему требует от программистов ручного написания кода, что на самом деле не повышает эффективность процесса разработки [2].

Поэтому изучение технологий автоматической генерации кода, методов повышения уровня абстракции разрабатываемого приложения и подходов к освобождению разработки программы от многочисленных рамок, имеет важное значение для ускорения прототипирования программного обеспечения, решения проблем повторного использования кода и повышения эффективности его разработки. Необходимость решения данных задач и определяет выбор темы данной статьи.

Над изучением особенностей и подходов к проектированию программного обеспечения на основе моделей трудятся такие авторы как Тутов И.А., Гительман В.С., Воскобойникова О.Б., Li, Dongcheng; Pan, Sean; Koh, Liang-Seng; Li, Shenglong.

Возможности системы APTS, которая была предложена Бобом Пейджем, для создания генераторов кода, которые конструируют исходный код на языке Си, описываются в работах Бойцова Г.В., Петрова А.В., Zheng, Chen; Xing, Jiajian; Wang, Zhanxi; Qin, Xiansheng.

Однако несмотря на то, что автоматическое программирование является целью компьютерной науки и искусственного интеллекта с тех пор, как первый программист столкнулся с трудностями программирования, не все предметные области в данной сфере проработаны в достаточной мере. Ряд вопросов требует особого внимания и более углубленного исследования. В частности, в дополнительной проработке нуждаются способы нахождения жизнеспособных разделов моделирования в разнородных аппаратных средах. Также особого внимания заслуживают подходы к автоматическому разрешению конфликтов на основе заданных пользователем правил.

Таким образом, цель статьи заключается в проведении анализа современных подходов к автоматическому программированию.

Работа программного обеспечения — это работа с данными, а работающие данные — это в основном таблицы баз данных или объекты сущностей [3]. Процесс разработки — это процесс создания и наложения программных страниц, реализуемых операцией, причем ко-

личество программных страниц является ограниченным множеством. Согласно теории конечных автоматов, процесс генерации кода информационной системы управления можно представить в виде квинтеплета:

$$M=(P, D, O, \delta, p0),$$

где P — представляет собой ограниченный набор страниц;

D — ограниченный набор данных;

O — ограниченное множество операций;

$\delta: (P \times D \times O) \rightarrow P$ — функция перехода;

$p0 \in P$ — начальное состояние, то есть состояние, когда целевой исходный код не был сгенерирован.

Таким образом, в основе концепции автоматической генерации кода лежит возможность генерирования отдельных компонентов программы по заданным параметрам [4]. Согласно обозначенному абстрактному определению процесса автоматической генерации кода, инструмент его реализации может быть разработан следующим образом: предполагая, что он находится в начальном состоянии сырого кода, когда механизм кода принимает входные инструкции операций, механизм кода объединяет правила генерации страниц (то есть, шаблон кода). После завершения преобразования функции перемещения, целевой исходный код может быть сгенерирован, затем вводится следующее состояние, то есть, $\delta(pn-1,d,o)=pn$ ($n \in N^*$).

Примером автоматического генератора кода является инженерная среда ALBA (генератор приложений на основе местоположения для Android), которую разработчики могут использовать для автоматического создания приложений Android на основе местоположения на Java. ALBA состоит из трех компонентов: (1) предметно-ориентированного языка моделирования (DSML), который поддерживает концепции приложений Android на основе местоположения; (2) графического редактора, позволяющего разработчикам моделировать приложение Android на основе местоположения; (3) подключаемый модуль Eclipse, который генерирует окончательный код приложения из модели на основе предопределенных преобразований.

Разработчики используют редактор моделирования ALBA для разработки модели приложения на основе предварительно определенных требований. Модель проверяется на соответствие предопределенным ограничениям, а редактор предотвращает создание недопустимых моделей. Затем код приложения автоматически генерируется из моделей с использованием преобразований, встроенных в плагин генерации кода. Таким образом, платформа ALBA может значительно облегчить разработку мобильных приложений на основе местоположения. Это связано с высоким уровнем абстракции, который обеспечивает язык моделирования ALBA, а также с автоматической генерацией кода, обеспечиваемой фреймворком. ALBA продвигает разработку приложений на основе местоположения, используя преимущества парадигмы, основанной на модели и коде.

Также на практике широко используются фреймворки для автоматизации процесса разработки программного обеспечения. Примером автоматического генератора кода является Yii-фреймворк, позволяющий сгенерировать простейшие формы ввода, вывода и изменения информации в базе данных, создавая при этом необходимые контроллеры, модели и отображения [5]. При этом программисту необходимо через веб-интерфейс просто указать заглавие таблицы, для которой следует сгенерировать форму. Одним из самых мощных инструментов Yii-фреймворка является модуль кодогенерации, основной задачей которого является экономия времени программиста по созданию основы связи веб-приложения с базой данных, основного каркаса модуля, модели, расширения или файлов операции CRUD.

Значительную популярность получил фреймворк Flatlogic, который позволяет создавать, размещать и разрабатывать полностью работоспособное веб-приложение CRUD с интерфейсом, серверной частью и базой данных. Сгенерированное приложение является основой для дальнейшей разработки программного обеспечения с аутентификацией пользователей, управлением данными и готовой базовой структурой. Flatlogic дает возможность ускорить процесс разработки приложений и защитить код от человеческих ошибок.

Учитывая имеющиеся на сегодняшний день подходы и методы, для оптимизации и упрощения генерации кода согласно заранее установленным параметрам на основании выделенных бизнес-сущностей, автором предложена запатентованная Автоматизированная система кодогенерации «Автокод» [6].

Главным отличием разработанной системы является то, что она позволяет покрыть все стадии производства (генерацию скриптов развертывания инфраструктуры, анализ требований, генерацию тестов, генерацию кода backend и frontend), а также совместима с наиболее популярными языками программирования. В качестве начальных данных для Автокода могут выступать текстовые документы, в которых содержится описание задачи, а также ожидаемые параметры приложения.

Алгоритм работы системы кодогнерации «Автокод» включает в себя следующие последовательные этапы.

1. Настройка сценариев, которые включают определенный порядок действий, связанных с выделением бизнес-сущностей, их использованием и генерацией необходимых артефактов.

2. Анализ документа и выделение бизнес-сущностей. Пользователь может использовать следующие форматы документов doc, docx, rtf, pdf, txt.
3. Корректировка в ручном режиме обозначенных на предыдущем этапе сущностей и взаимосвязей между ними. Когда анализ текстовой постановки задачи будет завершен, выделенные бизнес-сущности, а также распознанные между ними связи с использованием редактора могут быть откорректированы.
4. Настройки генерации кода (backend, frontend, тесты). На этом этапе выбираются параметры для разработки приложения. Например, такими параметрами могут быть: язык программирования, подключаемые библиотеки, архитектура проекта, параметры отлова необработанных исключений и т.д.
5. Настройка генерации инфраструктуры. В рамках данного этапа определяется необходимость применения подхода infrastructure-as-a-code. Этап может быть пропущен, если нет поребности в создании скриптов для поднятия инфраструктуры проекта.
6. Распределение сущностей по модулям/микросервисам. Задачей данного этапа является определение состава модулей/микросервисов, которые необходимы, чтобы реализовать функционал управлениями всеми обозначенными бизнес-сущностями. Для решения этой задачи могут быть использованы дефолтные настройки, которые позволяют обеспечить автоматическое распределение сущностей. Либо же пользователь может прибегнуть к ручной разработке каркаса приложения с использованием графического редактора.
7. Старт генерации кода проекта.
8. Генерация кода backend. На этом этапе для каждого модуля/микросервиса, которые выделены на этапе 5, разрабатывается каркас кода, благодаря ему может быть реализована функциональность по управлению соответствующими бизнес-сущностями.
9. Генерация кода frontend. Если приложение не предполагает графического интерфейса, то этот этап пропускается.
10. Миграция технологического стека приложения. Данный этап необходим в том случае, если миграция является частью исполняемого сценария.
11. Генерация скриптов развертывания инфраструктуры. Этот этап предполагает разработку скриптов развертывания системы на базе установленных параметров.
12. Генерация тестов. В рамках данного этапа генерируются код интеграционных тестов, нагрузочных тестов, модульных (unit) тестов и BDD. Если в генерации тестов нет необходимости, то этот этап автоматически пропускается.

Таким образом, подводя итоги, можно сделать следующие выводы. На сегодняшний день в связи с усложнением программного обеспечения, крайне важной является задача предоставить разработчикам простые технологии генерации кода, которые снижают входные барьеры. Достичь этой цели позволяет автоматическая генерация кода, которая обычно приводит к созданию программного обеспечения с меньшим количеством ошибок, чем код, который разрабатывался вручную.

В процессе исследования рассмотрены различные методы и фреймворки автоматической генерации кода. Также детально описана авторская запатентованная разработка Автоматизированная система кодогенерации «Автокод».

ЛИТЕРАТУРА

1. Zheng, Chen Knowledge-based program generation approach for robotic manufacturing systems // Robotics and computer-integrated manufacturing. 2023. Volume 73; pp 89–93.
2. Минакова О.В. Построение генератора программного кода для решения инженерных задач // Вестник Воронежского государственного технического университета. 2020. № 3. С. 14–19.
3. Dey, Bappaditya Code Generation Using Machine Learning: A Systematic Review // IEEE access: practical innovations, open solutions. 2022. Volume 10; pp 82434–82455.
4. Савельев И.Е. Автоматическая генерация программного кода смартконтрактов // Инновации и инвестиции. 2018. № 6. С. 229–231.
5. Летута Л.А. Преимущества CASE-технологии перед традиционными подходами при разработке программных обеспечений // Вестник современных исследований. 2019. № 1.8 (28). С. 113–115.
6. Патент № 2022684119, 12.12.2022. Автоматизированная система кодогенерации «Автокод» // Патент России № 2022681905. 2022. Бюл. №12 / Латин Ю.Э.

Журнал «Современная наука: актуальные проблемы теории и практики»