Отчет о проведенном аналитическом обзоре литературы

Студент	Неклюдов Семен Александровиич
Группа	РК6-71
Тип задания	аналитический обзор литературы
Тема исследования:	Особенности, технологии и методы генерации исходного кода программы.

Студент _____ **_Неклюдов С.А.__**
подпись, дата *фамилия, и.о.*

Преподаватель _____ **___А.П. Соколов_____**
подпись, дата *фамилия, и.о.*

Москва, 2018 г.

Оглавление

Задание

Аналитический обзор литературы проводился в рамках выполнения работ по разработке генератора кода GBSE решателей для распределенной вычислительной системы инженерного анализа GCD.

Объект исследования: особенности, технологии и методы генерации исходного кода программы.

Объект поиска: существующие решения в области генерации программного кода

Ключевые слова: Ключевые слова для поиска: модель-ориентированные методы генерации кода, автоматизация процессов разработки; быстрое прототипирование программного обеспечения; model-driven code generation; software engineering; model-to-text transformation;генерация кода; автоматизация программирования; программные процессоры; технологии разработки систем инженерного анализа.

Основная задача аналитического обзора литературы: изучить современные методы генерации программного кода.

Задачи аналитического обзора литературы (детально):

1) провести поиск источников литературы (преимущественно научных публикаций) согласно определенным ключевым словам;
2) определить историю развития объекта поиска и выявить основные тенденции развития;
3) определить наиболее перспективные современные разработки и направления развития объекта поиска;
4) определить перечень УДК, к которым относится объект поиска;
5) составить список найденных источников согласно ГОСТ Р 7.0.5-2008;

Начало поиска 01.10.18. Окончание поиска 07.10.18.

Введение

Разработка автоматизированных распределенных систем стратегического, корпоративного уровня, создание на их основе прикладных сервисов является сложным стохастическим процессом, требующим, в том числе, написания повторяющегося программного кода, а также типовых программных конструкций, реализующих шаблоны программирования, характерные для той или иной задачи. Современные средства разработки зачастую накладывают дополнительные идеологические, синтаксические, структурные ограничения. [1].

В настоящее время все более актуальной задачей становится создание программного продукта, позволяющего автоматизировать труд разработчика прикладного программного обеспечения. При проектировании разработчики стараются сделать структуру приложения максимально простой и стандартизированной - для облегчения программирования и сопровождения. Поэтому, при старте нового проекта, разработчику зачастую приходится выполнять однотипные действия по созданию базового шаблона той или иной конструкции. При правильных входных данных и корректной разработке генератора, большая часть кода может быть сгенерирована автоматически.

1. Результаты поиска источников литературы

В результате поиска были выявлены несколько источников, описываюшие подходы к генерации кода и организации структуры генераторов

1. Генераторы, построенные на шаблонах. Общая схема генерации программного кода на основе шаблонов включает в себя два этапа. На первом этапе происходит шаблонизация и конфигурирование. На вход шаблонизатора передаются конструкции языка и изменяемые поля, на выходе же получается шаблон. Для процесса конфигурирования входными параметрами являются модели предметной области, на выходе получается итоговая конфигурация. Модели предметной области формализуют структурные и функциональные аспекты автоматизируемой предметной области и являются источником

используемых в автоматизируемой системе имен, типов, ограничений. На втором этапе из полученных шаблона и конфигурации происходит рендеринг итогового программного кода.[1] Генераторы кода на основе шаблонов принято разделять в зависимости от типа применяемых шаблонов : а) предварительно определенные (англ. Predefined) или шаблоны, «зашитые» в генерирующую программу; б) основанные на формате генерируемого выходного объекта (англ. Output-based); в) основанные на правилах (англ. Rule-based).

2. Модель-ориентированная генерация. Модель — ориентированная парадигма — подход к разработке программного обеспечения, которая может затрагивать практически весь спектр этапов разработки продукта, начиная от сбора требований к программному инструментарию до поддержки системы[2]. Модели — основные единицы данного метода, созданные и используемые процессами жизненного цикла программного обеспечения. Модели обеспечивают эффективность путем принудительного разделения интересов: каждая модель представляет собой упрощенное представление системы, сфокусированное на проблемах, связанных с одной из этих задач разработки программного обеспечения, и в ней отсутствуют подробные сведения о системе, которые не нужны для выполнения этой задачи. Модель обычно представляется в виде документа на некотором языке моделирования(UML, IDEF,ARIS[4]) для построения модели объекта, ее интерпретации и конструирования заготовок исходного кода. Для манипулирования моделями используются преобразования. Преобразование действительно, если при заданных условиях преобразование модели M1 к M2 происходит так, что свойства модели M2 «переносятся» на M1.[5] В статье [5] были выделены 6 целей преобразований моделей:

- ограничительные запросы — выборка отдельных частей модели(подмоделей) из общего множества элементов; такие преобразования часто используются, когда необходимо отдать выходную

модель каким либо другим частям системы(например, при моделировании конечных автоматов[5]);

- уточнения — преобразование, создающее модель, конкретизирующую высокоуровневую спецификацию; является довольно абстрактным понятием; так, XML представление конкретизирует визуальную модель UML;
- трансляция и семантика трансляции — преобразование, транслирующее значения параметров исходной метамодели(модель, описывающая язык моделирования) на целевую метамодель; необходимо для возможности обеспечения делегирования функций; примером может послужить трансляция команд в машинный код, при котором высокоуровневые инструкции делегируют свои функции машинным командам; важным нюансом является невозможность выполнить какие либо действия, используя начальную модель(например, нельзя запустить программу на языке программирования, не использую трансляцию в машинные команды или интерпретацию);
- анализ — преобразование, реализующее алгоритмы анализа; примером такого алгоритма может быть алгоритм поиска неиспользуемого кода, использование которого позволит сократить размер программы;
- моделирование — преобразование, реагирующее на внешние возмущения, то есть модель M2 является полным отображением модели M1, за исключением компонент, на которых оказывалось внешнее воздействие;

Стадии разработки приложения, при использовании такого подхода к проектированию: а)разработка модели предметной области PIM; б)трансформация модели в платформо-зависимую модель(PSM); в)на основе PSM генерируется исходный код на целевом языке программирования.

Такой подход к генерации кода классифицируют по типу преобразований(трансформации) модели:

M2M(model to model) — преобразования между моделями; характеристикой такого преобразования служит явное представление моделей входной и выходной областей.

M2T(model to text) - формирование из исходной модели текстовых файлов; применяется в реверс-инжиниринге, генерации кода и документации, сериализации модели (включения обмена моделями), а также для визуализации и исследования модели[10]. Примером такой системы может служить разработка авторов статьи [11] AT2015: из модели, представленной UML диаграммой генерируется код на языке C#.

3) В работе [12] автор предлагает методику построения генератора исходного кода программ на основе многоуровнего набора правил. Методика основана на семиотическом подходе к построению информационных систем. В семиотических системах принимаются во внимание синтаксис(набор конструкций языка), прагматика(набор исходных требований к сгенерированному коду) и семантика(отношения между прагматикой и синтаксисом) решаемых задач. Такие системы удобно представлять в виде метаграфов. Отличительная особенность таких графов — наличие метавершин и(или) метаребер, иначе говоря — вложенных структур.

4) Преобразования M2T на основе шаблонов являются широко поддерживаемой техникой интеграции платформ в современных цепочках инструментов MDD. Существует множество реализаций языка шаблонов (таких как Eclipse Xpand, Xtend2, EGL, JET или Acceleo[10]). Для каждого из этих языков генераторы M2T и шаблоны генераторов могут быть реализованы различными способами. Например, одним из вариантов является использование документов спецификации с текстовым абстрактным синтаксисом и на основе дерева и промежуточным представлением, которое интерпретируется генератором.[17]

2. История развития объекта поиска. Основные тенденции развития

Генераторы программного кода были важной частью программного обеспечения практически с самых истоков эры вычислительной техники. Стартовой точкой развития генераторов программного кода можно считать появления первых компиляторов. Кодогенерация — важнейшая часть процесса трансляции программы, что является главной задачей компилятора.

С течением времени алгоритмы генерации стали усложняться. Стали широко применяться алгоритмы оптимизации, расширяя возможности компиляторов. Например, в статье [6] представлена разработка CVXGEN — программного инструмента, генерирующего простой код на языке C, практически не содержащий включаемых библиотек. В основе разработки лежит моделирование задач выпуклой оптимизации[7]. Также, одной из важнейших тенденций в данной сфере является развитие рынка многоядерных и графических процессоров. CUDA SDK — инструмент, позволяющий эффективно использовать ресурсы графических процессоров nVidea для задач с высокой степенью распараллеливания, в котором заложены алгоритмы кластеризации, целью которых является оптимальное распределение параллельных операций между ресурсами процессора. В статьях [8, 9] представлены генераторы, призванные скрыть реализацию параллельных алгоритмов. Существует ряд проблем, обусловленных неудобством традиционных подходов к параллельному программированию. Например, недетерменированность характера вычислительного процесса затрудняет выявление ошибок методами отладки [9]. Поэтому, актуально создание средств, позволяющих манипулировать «черными ящиками» - стандартными функциями, реализующими параллельные алгоритмы. Таким образом, конечный пользователь(разработчик) будет писать последовательные программы, не углубляясь в особенности параллельного программирования.

Еще одной важной ступенью в развитии генераторов кода является принятие на конференции OMG в 2001 году стандарта разработки, управляемой моделями - модель-ориентированного подхода к разработке программного обеспечения и создания модель-ориентированной архитектуры программного обеспечения. Его основной идеей является построение абстрактной метамодели управления и обмена метаданными, а также задании способа трансофрмации моделей. Дальнейшим шагом было принятие MOF - стандарта разработки MDD — модели программирования, в которой проект представляется в виде краткого и понятного языка[10]. Причиной появления такого подхода послужила необходимость создания архитектуры метамоделирования для UML. MOF представляет собой четырехуровневую архитектуру. Ядром является модель языка метамоделирования(пример метамодели - описание UML). На нижнизх уровнях находятся сами модели UML и данные для построения конкретных экземпляров на основе моделей. MDD позволяет проводить преобразования между этими уровнями, обеспечивая переходы от абстракций к конкретным реализациям.

Автор статьи [13] отмечает формирование намечающегося перехода с объектно - ориентированной технологии к языко-ориентированному(предметно-ориентированному) подходу. Идея такого подхода — формирование предметно-ориентированных языков, которые представляют собой некий интерфейс между пользователем и низкоуровневыми конструкциями.

3. Существующие перспективные современные разработки

1) Для разработчиков на платформе dotNet компания Microsoft разработала генератор кода Text Template Transformation Toolkit или T4 [5] на базе шаблонов. Для использования данного генератора необходимо написать файл-шаблон в формате *.tt. Существует большое количество готовых шаблонов для генерации стартового проекта под необходимые требования, однако из-за отсутствия единого репозитория поиск этих шаблоном крайне затруднен. Кроме того, T4 используется исключительно для генерации только C# кода. [1].

2) Генератор для системы Graphplus Templet, реализующий генерацию параллельных алгоритмов. Алгоритм генерации кода реализован в препроцессоре templet, совместная работа которого с интегрированной средой разработки Microsoft Visual Studio организуется, как описано ниже. Пусть код программы пишется на языке C++, что предусмотрено в текущей реализации. Модулем в данном случае является пара файлов: заголовочный файл с расширением h; файл реализации с расширением cpp. Код этих файлов помещается в общее пространство имён. Каждому модулю в проекте интегрированной среды разработки соответствует файл с XML спецификацией, который тоже включается в проект (обычно в папку ресурсов приложения). Для файла XML-спецификации в проекте указывается способ его обработки. Перед сборкой проекта (pre-built) такой файл должен быть обработан templet-препроцессором, который генерирует фрагменты кода, соответствующие модели программирования в файлах модуля. []

3) Десять лет назад группа компаний « ИВС» разработала технологическую программную платформу , которая автоматизировала процесс проектирования информационных систем[1]. Разработчикам удалось значительно снизить временные затраты программистов на реализацию программных инструментов. Идея данной платформы была максимально простой и заключалась в следующем: по аналитическим моделям бизнес - процессов , происходящих

в той или иной компании , составляется диаграмма классов в нотации языка UML, которая в дальнейшем преобразуется в исходный код (C# или SQL). На выходе системы разработчик получает готовый скомпилированный прототип системы управления информационными процессами компании со следующей функциональностью :

1) наличие графического интерфейса (WinForms- приложение , веб - при - ложение);

2) система авторизации пользователей ;

3) стандартные операции по работе с данными , такие как чтение , запись , редактирование , удаление , фильтрация и многое др.

После генерации данный прототип дорабатывается вручную с учетом требований, прописанных в техническом задании. При большом разнообразии провайдеров баз данных [2] (Microsoft SQL, MySQL, PostgreSQL, Oracle Database) прототип системы управления был совместим с любым из перечисленных провайдеров (это дает возможность в будущем без особых усилий менять источник данных). Такая универсальность получилась за счет наличия некоторого промежуточного слоя между источником данных и системой управления. Для автоматизации процесса программирования в области проектирования АСУТП было принято решение о разработке специализированной прикладной IDE, основной задачей которой будет автоматизированная генерация исходного кода и документации на таких языках программирования , как C#, Qt, Java, C[1]

4) Наиболее известным инструментом для быстрого создания стартового проекта является скаффолдер Yeoman. Скаффолдинг - это метод метапрограммирования, при котором проект генерируется на базе анализа требований разработчика [4]. Yeoman базируется на трех основных компонентах: менеджер зависимостей пакетов, необходимый для загрузки, обновления и удаления дополнительных пакетов (bower); инструмент для сборки java-script проектов с использованием заранее написанных задач (grunt);

базовое приложение, отвечающее за генерацию базы для нового приложения (yo).[1]

5) В [14] статье представлена разработка генератора Genesys, основанная на объединении сервис — ориентированного и модель — ориентированных подходов. Автор приводит пример генерации HTML кода документации при помощи на основе. Genesys является частью программной системы поддержки жизненного цикла программного продукта jABC. При моделировании с помощью jABC системы или приложения собираются на основе (расширяемой) библиотеки четко определенных многократно используемых строительных блоков SIB. Из таких моделей можно генерировать код для различных платформ (например, платформ на основе Java, мобильных устройств и т. д.). Поскольку генераторы кода для этих моделей также встроены в jABC, такие концепции, как начальная загрузка и повторное использование существующих компонентов, обеспечивают быструю эволюцию библиотеки генерации кода.

6) Исследователи из Университета Малаги представили среду для проверки контрактов преобразований M2M, M2T, T2M Tracts.[15] Разработка мотивирована тенденцией к развитию модель-ориентированного подхода, что влечет за собой необходимость тестирования, валидации и верификации преобразований. В Tracts для тестирования преобразований применяется M2M трансформации, позволяющие сопоставлять структуру моделей. M2T разработчики системы применяли для генерации кода, а T2M для встраивания текста в модель. Таким образом преобразования M2T и T2M могут сводиться к M2M.

7) DWE(Data Warehouse Evolution) — разработка в сфере BI(Business Intelligence). Необходимость данного программного инструментария заключается в том, чтобы обеспечивать автоматическое распространение изменений в модели OLAP на многомерную модель DW [16]. Преобразование M2M представляется в виде концепции QVT(Query, View, Template) и используется вместе с M2T, реализованного при помощи шаблонов Acceleo.

8) CVXGEN - это программный инструмент, который на высоком уровне описывает семейство выпуклых задач оптимизации и автоматически генерирует пользовательский код на языке C, который компилируется в надежный, высокоскоростной решатель для целого набора задач. CVXGEN генерирует простой, плоский, безбиблиотечный код, подходящий для встраивания в приложения реального времени. Сгенерированный код практически не содержит ветвей и поэтому имеет предсказуемое поведение во время выполнения. [6]

Заключение

1) Генерация кода широко применяется в современных программных продуктах, в том числе при построении приложений, основанных на модель-ориентированном подходе.

2) Генерация кода позволяет реализовывать миграцию систем со старых архитектур.

3) Использование генераторов исходного кода существенно снижает время разработки программ.

4) Генераторы кода позволяют создавать предметно — ориентированные языки, при помощи которых можно доверить разработчику средней квалификации сложные вычислительные задачи

5) Генераторы могут применяться при развертывании приложений.

Список найденных источников литературы

1) Д.В. Жевнерчук, А.С. Захаров Семантическое моделирование генераторов программного кода распределенных автоматизированных систем// Информатика и управление в технических и социальных системах. - 2018. - №1. — С. 23-30.

2) J. Klein, H. Levinson, J. Marchetti Model-Driven Engineering: Automatic Code Generation and Beyond // Carnegie Mellon University Software Engineering Institute. — 2015. — №1. — С. 1-51.

3) Nikiforova, O. Comparison of BrainTool to Other UML Modeling and Model Transformation Tools / O. Nikiforova, K. Gusarov // Applied Computer Systems. — 2013. — №-1 — С. 31-42.

4) V.Y. Rosales-Morales, G. Alor-Hernández, J.L. García-Alcaráz An analysis of tools for automatic software development and automatic code generation.//Revista Facultad de Ingenieria.. — 2015. — №77. — С. 75-87.

5) L. Lúcio, M. Amrani, J. Dingel Model transformation intents and their properties // Springer-Verlag Berlin Heidelberg. — 2014. — №3. — С. 647- 684.

6) J. Mattingley, S. Boyd CVXGEN: a code generator for embedded convex optimization / // Optim Eng. — 2011. — №13. — С. 1-27.

7) Ю.В. Нестеров Методы выпуклой оптимизации / Ю.В. Нестеров; под науч. ред. “Nesterov-final”. — Москва : МЦНМО, 2010. — 281 с.

8) Verdoolaege S., Carlos Juega J. Polyhedral Parallel Code Generation for CUDA// ACM Transactions on Architecture and Code Optimization. — 2013. — №9. — С. 54-77.

9)Востокин, С.В., А.Р. Хайрутдинов Программный комплекс параллельного программирования Graphplus Templet// ACM Transactions on Architecture and Code Optimization. — 2011. — №4. — С. 146-153.

10) L.M. Rose, N. Matragkas A Feature Model for Model-To-Text Transformation Languages// MiSE '12 Proceedings of the 4th International Workshop on Modeling in Software Engineering / Switzerland; Zurich: IEEE Press Piscataway, 2012. — С. 57-63.

12) Э.Н. Самохвалов, Г.И. Ревунков, Ю.Е. Гапанюк Генерация исходного кода программного обеспечения на основе многоуровнего набора правил// Вестник МГТУ им. Н.Э. Баумана. — 2014. — №5. — С. 77-87.

13) А.Е. Александров, В.П. Шильманов Инструментальные средства разработки И сопровождения программного обеспечения на основе генерации кода// БИЗНЕС-ИНФОРМАТИКА. — 2012. — №4. — С. 10-17.

14) S. Jörges, B. Steffen Building Code Generators with Genesys: A Tutorial Introduction// Springer-Verlag Berlin Heidelberg. — 2011. — №6491. — С. 364-385.

15) L. Burgueno, B. Steffen Testing M2M/M2T/T2M Transformations// Springer Science+Business Media. - 2011. — С. 203-219.

16) S. Taktak, J. Feki Model-Driven Approach to Handle Evolutions of OLAP Requirements and Data Source Model // Springer International Publishing AG. — 2019. — №880. — С. 401-425.

17)S. Taktak, J. Feki Higher-Order Rewriting of Model-to-Text Templates for Integrating Domain-specific Modeling Languages // MODELSWARD 2013. — 2013. — №-. С. 1-13.