Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Робототехники и комплексной автоматизации(РК)

КАФЕДРА Систем автоматизированного проектирования

Отчет

О прохождении преддипломной практики

в организации:

НИИ АПП

Студент РК6-81 _____ **С.А. Неклюдов**
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель преддипломной практики _____ **А.П. Соколов**
(Подпись, дата) (И.О.Фамилия)

Руководитель практики от предприятия _____ **И.А. Киселев**
(Подпись, дата) *(И.О.Фамилия)*

2019 г.

АННОТАЦИЯ

В данном отчете будет представлена информация об организации “НИИ автоматизации производственных процессов”, описание проекта, над которым производилась работа, а также описание выполненной работы.

СОКРАЩЕНИЯ

НИИ АПП — научно исследовательский институт автоматизации производства

РВС GCD – распределенная вычислительная система GCD

UML – унифицированный язык моделирования

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В преддипломной практике проводились работы над распределенной вычислительной системой GCD. Была разработана архитектура программной подсистемы генерации исходного кода программ. Потребность в генерации кода обусловлена особенностями разработки прикладного программного обеспечения инженерного анализа, а именно: необходимостью написания существенного объема исходного кода программ требующих специальной подготовки; потребностью обеспечения слаженной работы коллектива разработчиков; проблемой согласованного повторного использования исходного кода. Одной из прикладных задач генераторов исходного кода является генерация документации к программному средству. [1]

Известно 2 метода генерации кода:

1) преобразование модели алгоритма в исходный код[2];

2) генерация кода на основе шаблонов[3];

В данной работе реализовывался подход преобразования модели в текст программы(M2T), была построена программная архитектура в нотации диаграммы классов UML.

1. ОПИСАНИЕ РАСПРЕДЕЛЕННОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ GCD

Распределенная вычислительная система GCD — программный комплекс, разрабатываемый усилиями сотрудников и студентов МГТУ им. Н.Э. Баумана, а также студентами МГТУ и других отечественных высших учебных заведений.

Основной задачей данной работы является ознакомление с библиотекой comsdk. Библиотека позволяет создавать программное обеспечение, имеющее возможность решать сложные технические и инженерные задачи. Процесс создания такого программного обеспечения выглядит следующим образом:

1) задача декомпозируется на несколько этапов, на каждом из которых производятся некоторые вычисления; 2) составляется графовая модель алгоритма решения задачи; 3) если выделенные на каждом этапе подзадачи не реализованы — они реализуются в виде функций; 4) пишется программное обеспечение, позволяющее запустить каждую из функций.

Так как процесс разработки таких программ является рутинным — предлагается автоматизировать процесс получения исходных кодов подобных решателей. Входными данными послужит графовая модель алгоритма в формате aDot.

2. АРХИТЕКТУРА ПРОГРАММНОЙ РЕАЛИЗАЦИИ

Были спроектированы семь классов, работающих со строковыми полями.

- ProgramGenerator — главный класс, содержащий поля — составные части файла исходного кода на языке C. Были выделены следующие части программы:

1. header — содержит директивы препроцессора, а также объявления прототипов функций. Пример header строки:
   "**#include <anymap.h>**\n**typedef int processorFuncType(AnyMap&)**;\n**typedef bool predicateFuncType(const AnyMap&);**\n**template<processorFuncType* tf, predicateFuncType* tp>**\n**int F(AnyMap& p_m)**\n{\n\t**return (tp(p_m))? tf(p_m):tp(p_m);**\n}\n**int main(){**\n\t";
2. footer — заверщающая строка: "\treturn 0;\n}"
3. programm_name — имя программы;
4. data_ini — объект Anymap, ассоциированный с программой;
5. loaderSet — набор строк, сериализуемых методом класса loader; генерирует строку программы, в которой осуществляется поиск библиотек для используемых функций решателя;
6. callerSet — набор строк, сериализуемых методом класса caller; генерирует строку программы, в которой осуществляется вызов функций;
7. getterSet — набор строк, сериализуемых методом serialize() класса getter; генерирует строку программы, в которой осуществляется поиск функции в заданной библиотеке;

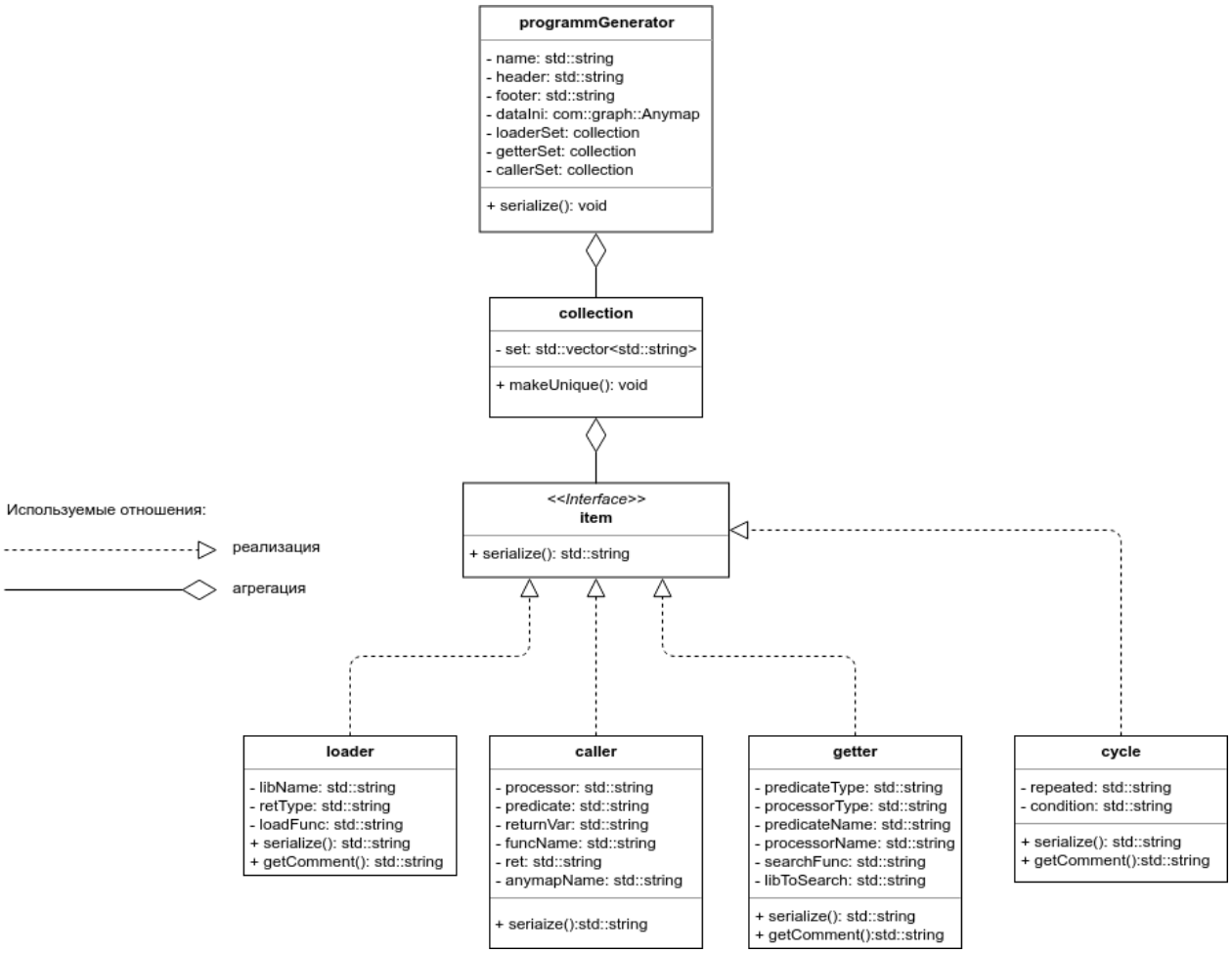

Рис. 1. Архитектура генератора исходного кода

Для тестирования был выбран следующий файл описания графовой модели алгоритма:

```
digraph JUGR_GraphModel
{
    HOM_POST [predicate=CHECK_BC]
    CHECK_BC [type=boolean, binname=jugr, entry_func=predicate_1]
    FUNC_1 [binname=jugr, entry_func=function_1]
    FUNC_2 [binname=jugr, entry_func=function_2]
    FUNC_3 [binname=jugr, entry_func=function_3]
    FINALIZE [binname=jugr, entry_func=function_3]

    __BEGIN__ -> INPUT
    INPUT -> HOM_POST [edge=FUNC_1]
    HOM_POST -> SOLVED_2 [on_predicate_value=false, edge=FUNC_2]
    HOM_POST -> SOLVED_3 [on_predicate_value=true, edge=FUNC_3]
    SOLVED_2 -> FINILEZED [edge=FINALIZE]
    FINILEZED -> __END__
}
```

Листинг 1. Модель алгоритма

В результате вызова метода serialize() класса programGenerator был получен следующий файл исходного кода:

```
#include <anymap.h>
typedef int processorFuncType(AnyMap& );
typedef bool predicateFuncType(const AnyMap& );
template<processorFuncType* tf, predicateFuncType* tp>
int F(AnyMap& p_m)
{
    return (tp(p_m))?tf(p_m):tp(p_m);
}
int main(){
        //Загрузка библиотеки lib_name
    HMODULE lib_lib_name=LoadLibrary(L"lib_name");
    //Загрузка библиотеки lib_name
    HMODULE lib_lib_name=LoadLibrary(L"lib_name");
    //Загрузка библиотеки lib_name
    HMODULE lib_lib_name=LoadLibrary(L"lib_name");
    //Поиск функции обработчика func1 и функции-предиката pred1 в библиотеке lib_name
    processorFuncType *proc_func1 = (processorFuncType *)GetProcAddress(lib_name,"func1");
    predicateFuncType *pred_pred1 = (predicateFuncType *)GetProcAddress(lib_name, "pred1");
    //Поиск функции обработчика func2 и функции-предиката pred2 в библиотеке lib_name
```

```
    processorFuncType *proc_func2 = (processorFuncType *)GetProcAddress(lib_name,"func2");
    predicateFuncType *pred_pred2 = (predicateFuncType *)GetProcAddress(lib_name, "pred2");
    //Поиск функции обработчика func3 и функции-предиката pred3 в библиотеке lib_name
    processorFuncType *proc_func3 = (processorFuncType *)GetProcAddress(lib_name,"func3");
    predicateFuncType *pred_pred3 = (predicateFuncType *)GetProcAddress(lib_name, "pred3");
    //Вызов функции-обработчика func1 с предикатом pred1
    auto res = F<func1, pred1>(input);if (res!=0) return res;
    //Вызов функции-обработчика func2 с предикатом pred2
    auto res = F<func2, pred2>(input);if (res!=0) return res;
    //Вызов функции-обработчика func3 с предикатом pred3
    auto res = F<func3, pred3>(input);if (res!=0) return res;
    return 0;
}
```

Листинг 2. Исходный код сгенерированной программы

3. Заключение

По результатам прохождения практики был разработан функционал позволяющий генерировать исходный код линейных программ.

4. Список 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.