УДК 004.4'242

А.В. АЛЕЩЕНКО, асп., НТУУ "КПИ", Киев

ИСПОЛЬЗОВАНИЕ ПРОМЕЖУТОЧНОГО ЯЗЫКА ДЛЯ ТРАНСЛЯЦИИ ГРАФИЧЕСКИХ СХЕМ АЛГОРИТМОВ В ИСПОЛНЯЕМЫЙ КОД

Описаны функциональные требования к системе имитации для получения оценок эффективности трансляции графических схем алгоритмов в исполняемый код с использованием промежуточного языка трансляции. Приведен пример процесса тестирования. Описана возможность указания точек замера времени выполнения программ с целью учета или игнорирования определённых частей алгоритма. Ил.: 2. Библиогр.: 10 назв.

Ключевые слова: система имитации, оценка эффективности трансляции, исполняемый код, промежуточный язык трансляции, тестирование, время выполнения программы, алгоритм.

Постановка проблемы. В настоящее время широко распространены *CASE*-системы (*Computer-Aided System/Software Engineering*), которые позволяют автоматизировать процесс разработки программных продуктов. Однако такая автоматизация, как правило, не является полной, так как внутренние коды методов классов предлагается вписывать вручную [1]. Следовательно, актуальной является задача полной автоматизации процесса создания программных продуктов, в рамках которой реализуется собственная система автоматической генерации программных кодов по графической схеме алгоритма (ГСА), которая, например, в нотации UML соответствует диаграмме деятельности (*activity diagram*) [2].

При разработке данной системы, наряду с задачей представления ГСА и её верификации, ставится задача трансляции ГСА в исполняемый код. Такая трансляция возможна тремя способами. Первый из них – использование промежуточного языка высокого уровня, второй – трансляция в промежуточный унифицированный код (например, байт-код в языке *Java*), и третий – трансляция непосредственно в исполняемый код [3].

Для получения оценок способов трансляции ГСА и, в частности, эффективности промежуточного языка трансляции разрабатывается система имитации, так как аналитическая оценка сложности трансляции не представляется возможной.

Анализ литературы. В работе [4] описывается среда визуальной разработки "*PureBuilder*", которая, по словам автора, должна полностью избавить от исходного кода в текстовом виде и перейти к визуальному программированию на основе "универсального языка программирования". Автор использует "встроенный язык" для надписей на графических элементах среды визуальной разработки, который не наследует синтаксис какого-то из существующих языков программирования. Также в *PureBuilder* не применяются операторы досрочного выхода *exit, break, exit when, continue, return*, операторы генерации исключительной ситуации *throw* (допускается в отладочном режиме) и другие конструкции, которые расцениваются как небезопасные, в частности, таким документом, как *NASA Software Safety Guidebook*. В соответствии с функциональной парадигмой программирования, в *PureBuilder* не применяются глобальные и статические переменные. Отказ от использования существующих языков программирования в качестве промежуточных языков трансляции требует от разработчиков системы *PureBuilder* реализации, развития и поддержки собственных алгоритмов трансляции непосредственно в машинный код. С другой стороны, подобная концепция вынуждает пользователей отказаться от привычных конструкций, которые существуют во многих текстовых языках программирования.

В работе [5] описана парадигма языково-ориентированного программирования и её реализация автором в собственной системе "*Meta Programming System*". По словам автора, парадигма языково-ориентированного программирования даёт возможность работать непосредственно в терминах концепций и понятий проблемы, которая решается, вместо того чтобы переводить их в нотацию языка программирования общего назначения (классы, методы, циклы, ветвления и т.п.). Для этого нужен язык, специфичный для предметной области, который и предлагается создать на основе существующих языков программирования. Появление в процессе трансляции ещё одного промежуточного звена в виде специфичного языка для предметной области потенциально увеличит время генерации исполняемого кода. Также, с одной стороны, потребуется создание нового языка для каждой предметной области, а с другой – его изучение другими участниками процесса разработки. Кроме того, автор не рассматривает задачу выбора языка программирования, который будет использован в качестве основы для создания специфичного языка.

В работе [6] также используется парадигма языково-ориентированного программирования, рассмотрены инструментальные средства разработки и сопровождения программного обеспечения на

основе автоматической генерации кода, реализованные на языке *Python* с использованием комбинаторной библиотеки *PyParsing*. Наполнение компонентной библиотеки и формирование конкретной реализации программного приложения происходит в результате генерации программного кода, формирующего исходный код на языке *C++*. Однозначность выбора промежуточного языка трансляции потенциально ограничивает пользователя описываемой в работе системы.

В работах [7] и [8] сравниваются различные языки программирования на конкретных тестовых задачах. По причине неиспользования системы генерации кодов, для решения общей тестовой задачи на каждом языке программирования авторы вручную писали отдельные текстовые программы. Кроме необходимости владения всеми тестируемыми языками, ещё одним недостатком подобного подхода есть то, что написанные таким образом исходные коды могут терять свою актуальность ("устаревать") в процессе развития соответствующих языков программирования и их трансляторов по причине появления новых более эффективных конструкций.

Всё вышеперечисленное делает актуальным создание системы имитации на основе генерации кодов, которая позволит пользователю оценить и сравнить эффективность различных трансляторов и языков программирования в качестве промежуточных языков трансляции ГСА в исполняемый код.

Цель статьи – разработка системы имитации для получения оценок эффективности промежуточного языка трансляции ГСА в исполняемый код.

Описание эксперимента и аспектов системы имитации. Любой транслятор (компилятор или интерпретатор) является автоматом перевода текста на некотором языке в соответствующую последовательность команд процессора ЭВМ. От длины и структуры этой последовательности существенно зависит скорость выполнения программы пользователя. Например, программу, написанную на каком-либо языке программирования, можно перевести на язык команд процессора двумя различными компиляторами $К_1$ и $К_2$. Более эффективным следует считать тот компилятор, после обработки которым исходной программы, последняя выполняется быстрее [9].

С целью измерения скорости выполнения алгоритма, реализованного на разных языках, в системе генерации [10] была построена тестовая ГСА (см. рис. 1). Эта графическая схема описывает реализацию алгоритма сортировки простыми обменами ("пузырьком").

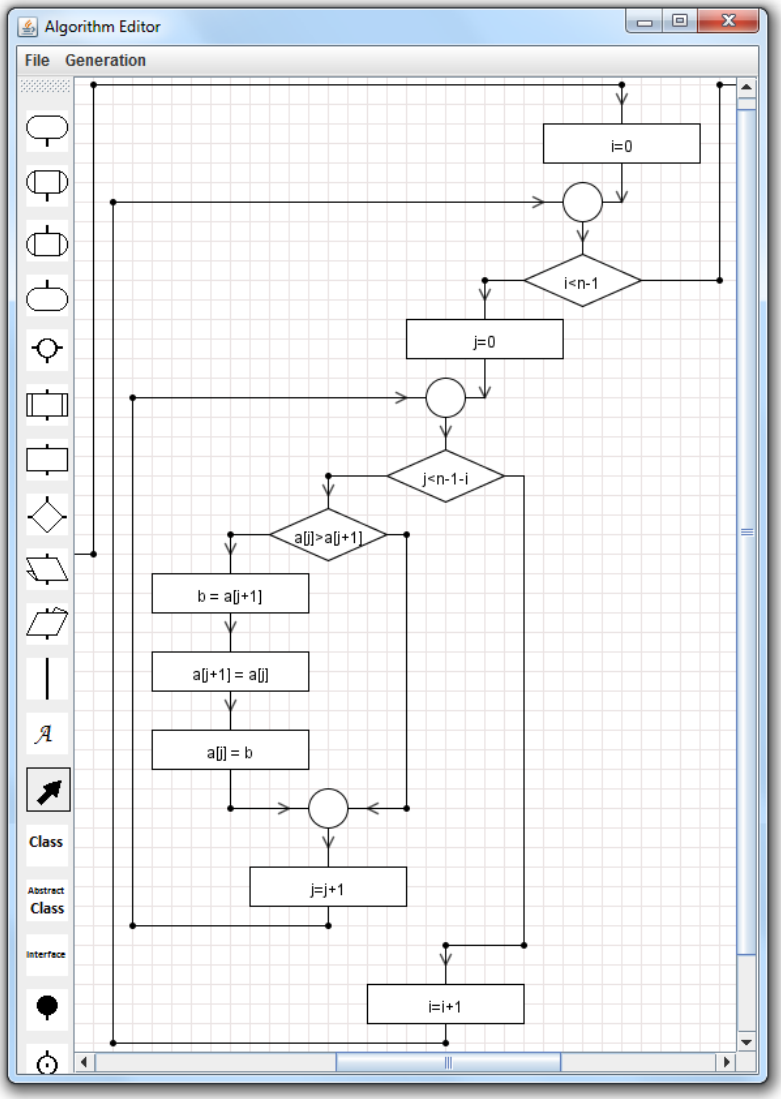

Рис. 1. Главное окно системы генерации с ГСА сортировки простыми обменами ("пузырьком")

Главное окно системы генерации позволяет задать графическую схему алгоритма с помощью панели инструментов, которая размещена

вертикально у левого края окна. Главное меню позволяет создать новую ГСА (удалить схему, которая находится в данный момент), сохранить и загрузить ГСА, а также сгенерировать программные коды по текущей ГСА.

С помощью системы генерации были получены по ГСА программные коды на языке *Java* и *Pascal*. На данный момент система ещё не генерирует код на языке *C++*, но программная реализация алгоритма на этом языке была также создана методом ручной трансляции с кода программы на языке *Java*.

В трёх программных реализациях были добавлены операторы измерения времени выполнения и ввод данных был изменён с клавиатурного на файловый. Предварительно был создан файл с множеством случайных чисел, который служил источником данных для сортировки. Эти дополнительные действия носят рутинный характер и могут быть автоматизированы средствами разрабатываемой системы имитации.

Результаты многократного выполнения данных программных реализаций приведены на диаграмме (см. рис. 2), построение которой также предусматривается функционалом системы имитации. Данная диаграмма демонстрирует изменение времени выполнения алгоритма (ось ординат) для разных языковых реализаций в ходе отдельных экспериментальных запусков (ось абсцисс). Эта информация может быть отображена и в табличной форме.

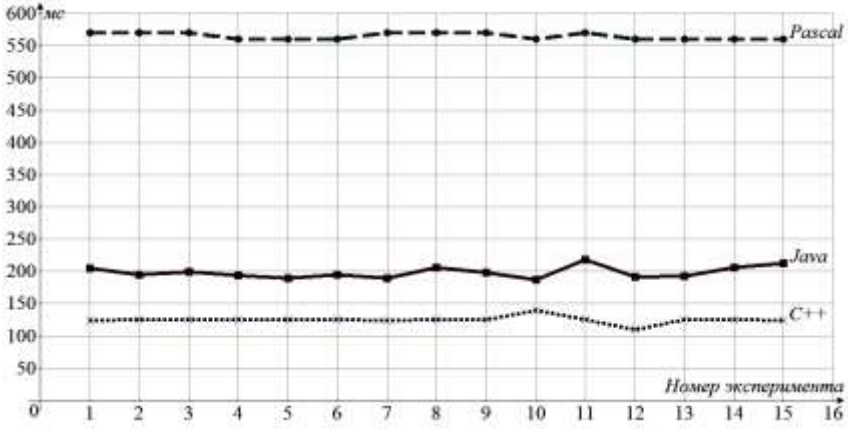

Рис. 2. Результирующая диаграмма тестирования времени выполнения программных реализаций

В процессе тестирования измерялся интервал времени выполнения непосредственно сортировки данных без "накладных расходов" на ввод и вывод данных. Полезной функцией разрабатываемой системы имитации будет возможность указания на ГСА точек замера времени выполнения самим пользователем с целью учитывания или игнорирования определённых частей алгоритма.

Кроме времени выполнения существуют и другие метрики эффективности трансляции, например, размер исполняемого файла, информация о котором также может быть предоставлена системой имитации.

Конкретные цифровые результаты тестирования, объёмы входных данных, задача выбора тестируемого алгоритма и набора тестируемых языков программирования не являются предметом данной статьи. Приведенное тестирование носит концептуальный характер и отображает функциональные требования к системе имитации для получения оценок эффективности промежуточного языка трансляции ГСА в исполняемый код.

Система генерации и система имитации предполагают расширение множества промежуточных языков трансляции. Создав ГСА для решаемой задачи, пользователь может автоматически сгенерировать и протестировать целый ряд текстовых программных реализаций. Система имитации, используя систему генерации, будет способна определить наиболее эффективный язык и транслятор для реализации заданного алгоритма. Кроме того, запуск данных систем на оборудовании и операционной системе, которые должны исполнять итоговый программный код, даёт возможность получить наиболее актуальные сведения. Разработка описываемых систем на кроссплатформенном языке *Java* позволит реализовать генерацию и тестирование на разных операционных системах.

Необходимым условием получения правильных и актуальных результатов тестирования есть своевременное обновление транслятора из ГСА в промежуточный язык программирования высокого уровня. Причиной этого есть тот факт, что появление новых, более эффективных конструкций языков и реализация их в новых версиях трансляторов существенно влияет на результаты тестирования.

Выводы. В результате проделанной работы были описаны функциональные требования к системе имитации для получения оценок эффективности промежуточного языка трансляции ГСА в исполняемый код. Данная система, основываясь на реализованной системе генерации,

позволяет автоматически транслировать исходную ГСА и определить наиболее эффективный язык и транслятор из поддерживаемых этими системами. Разработка систем на языке *Java* позволяет реализовывать генерацию кодов и сбор данных (тестирование) на различных операционных системах.

Приведен пример процесса тестирования, в ходе которого использовались промежуточные языки *Pascal*, *Java* и *C++*. Описана возможность указания пользователем точек замера времени выполнения программ с целью учитывания или игнорирования определённых частей алгоритма.

Дальнейшими перспективами развития есть увеличение множества поддерживаемых системами промежуточных языков высокого уровня, разработка эффективного способа добавления нового промежуточного языка трансляции и усовершенствование процесса создания ГСА в разработанном редакторе.

Список литературы: 1. *Канжелев С.* Автоматическая генерация кода программ с явным выделением состояний / *С. Канжелев, А. Шалыто* // *Paths to Competitive Advantage: Software Engineering Conference*. – М., 2006. – С. 60-63. **2.** *Бузовский О.В.* Система верификации графических схем алгоритмов и генерации программных кодов / *О.В. Бузовский, А.В. Алещенко* // Проблеми інформатизації та управління. – 2015. – Т. 2. – № 50. – С. 32-35. **3.** *Бузовский О.В.* Разработка системы генерации кодов по графическим схемам алгоритма с промежуточным языком трансляции / *О.В. Бузовский, А.В. Алещенко* // Технологический аудит и резервы производства. – 2015. – № 4/2 (24). – С. 15-19. **4.** *Прохоренко С. PureBuilder*. Проект: Среда визуальной разработки без исходного кода [Электронный ресурс] / *С. Прохоренко* – Режим доступа: *https://sites.google.com/site/purebuilder/* (08.10.2015) – Загл. с экрана. **5.** *Дмитриев Сергей.* Языково-ориентированное программирование: следующая парадигма [Электронный ресурс] / *Сергей Дмитриев* // *RSDN Magazine* – Санкт-Петербург. – 2005. – № 5. *https://rsdn.ru/article/philosophy/LOP.xml* **6.** *Александров А.Е.* Инструментальные средства разработки и сопровождения программного обеспечения на основе генерации кода / *А.Е. Александров, В.П. Шильманов* // Бизнес-информатика. – 2012. – № 4. – С. 10-17. **7.** *Цилюрюк О.* Производительность языков программирования. Часть 1 [Электронный ресурс] / *Олег Цилюрюк.* – 2014. – Режим доступа: *https://www.ibm.com/developerworks/ru/library/ManySpeed_08_1/* (08.10.2015) – Загл. с экрана. **8.** *Юдин С.Ю.* Выбор языка программирования для научных работников [Электронный ресурс] / *С.Ю. Юдин.* – 2007. – Режим доступа: *http://modsys.narod.ru/Stat/Stat_Prog/Vibor/Vibor2.html* (08.10.2015) – Загл. с экрана. **9.** Средства ресурсной поддержки. Программное обеспечение ЭВМ. Языки программирования [Электронный ресурс] – Режим доступа: *http://ecocyb.narod.ru/410-417/inrs2_17.htm* (08.10.2015) – Загл. с экрана. **10.** *Бузовский О.В.* Система визуального проектирования и генерации программных кодов / *О.В. Бузовский, А.В. Алещенко* // Вісник Національного технічного університету "ХПІ". Збірник наукових праць. Серія: Механіко-технологічні системи та комплекси. – Х.: НТУ "ХПІ", 2015. – № 22 (1131) – С. 14-17.

Bibliography (transliterated): 1. *Kanzhelev S.* Avtomaticheskaja generacija koda programm s javnym vydeleniem sostojanij / *S. Kanzhelev, A. Shalyto* // Paths to Competitive Advantage: Software Engineering Conference. – M., 2006. – P. 60-63. **2.** *Buzovskij O.V.* Sistema verifikacii

graficheskih shem algoritmov i generacii programmnyh kodov / *O.V. Buzovskij, A.V. Aleshhenko* // Problemi informatizaciї ta upravlinnja. – 2015. – T. 2, № 50. – P. 32-35. **3.** *Buzovskij O.V.* Razrabotka sistemy generacii kodov po graficheskim shemam algoritma s promezhutochnym jazykom transljacii / *O.V. Buzovskij, A.V. Aleshhenko* // Tehnologicheskij audit i rezervy proizvodstva. – 2015. – № 4/2 (24). – P. 15-19. **4.** *Prohorenko C.* PureBuilder. Proekt: Sreda vizual'noj razrabotki bez ishodnogo koda [Jelektronnyj resurs] / *C. Prohorenko* – Rezhim dostupa: https://sites.google.com/site/purebuilder/ (08.10.2015) – Zagl. s jekrana. **5.** *Dmitriev Sergej* Jazykovo-orientirovannoe programmirovanie: sledujushhaja paradigma [Jelektronnyj resurs] / *Sergej Dmitriev* // RSDN Magazine – Sankt-Peterburg. – 2005. – № 5. – Rezhim dostupa: https://rsdn.ru/article/philosophy/LOP.xml (08.10.2015) – Zagl. s jekrana. **6.** *Aleksandrov A.E.* Instrumental'nye sredstva razrabotki i soprovozhdenija programmnogo obespechenija na osnove generacii koda / *A.E. Aleksandrov, V.P. Shil'manov* // Biznes-informatika. – 2012. – № 4. – P. 10-17. **7.** *Ciljurjuk O.* Proizvoditel'nost' jazykov programmirovanija. Chast' 1 [Jelektronnyj resurs] / *Oleg Ciljurjuk*. – 2014. – Rezhim dostupa: https://www.ibm.com/developerworks/ru/library /ManySpeed_08_1/ (08.10.2015) – Zagl. s jekrana. **8.** *Judin S.Ju.* Vybor jazyka programmirovanija dlja nauchnyh rabotnikov [Jelektronnyj resurs] / *S.Ju. Judin.* – 2007. – Rezhim dostupa: http://modsys.narod.ru/Stat/Stat_Prog/Vibor/Vibor2.html (08.10.2015) – Zagl. s jekrana. **9.** Sredstva resursnoj podderzhki. Programmnoe obespechenie JeVM. Jazyki programmirovanija [Jelektronnyj resurs] – Rezhim dostupa: http://ecocyb.narod.ru/410-417/inrs2_17.htm (08.10.2015) – Zagl. s jekrana. **10.** *Buzovskij O.V.* Sistema vizual'nogo proektirovanija i generacii programmnyh kodov / *O.V. Buzovskij, A.V. Aleshhenko* // Visnik Nacional'nogo tehnichnogo universitetu "HPI". Zbirnik naukovih prac'. Serija: Mehaniko-tehnologichni sistemi ta kompleksi. – H.: NTU "HPI", 2015. – № 22 (1131) – P. 14-17.

Поступила (received) 09.10.2015

Статью представил д.т.н., проф. НТУУ "КПИ" Бузовский О.В.

Aleshchenko Oleksii, PhD student
National Technical University of Ukraine
"Kiev Polytechnic Institute"
Prospect Peremohy, 37, 03056, Kyiv-56, Ukraine
tel./phone: (096) 833-89-99, e-mail: alexey.aleshchenko@gmail.com
ORCID ID: 0000-0002-6528-3748