

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

На правах рукописи

Мосиенко Максим Алексеевич

АВТОМАТИЗИРОВАННЫЙ ПЕРЕВОД
УСТАРЕВШИХ ПРИЛОЖЕНИЙ
НА НОВЫЕ ЯЗЫКИ И ПЛАТФОРМЫ

05.13.11 – Математическое и программное обеспечение вычислитель-
ных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ

диссертации на соискание ученой степени
кандидата физико-математических наук

Санкт-Петербург
2006

Работа выполнена на кафедре системного программирования математико-механического факультета Санкт-Петербургского государственного университета.

Научный руководитель:

доктор физико-математических наук,
профессор Терехов Андрей Николаевич

Официальные оппоненты:

доктор физико-математических наук
Касьянов Виктор Николаевич
кандидат физико-математических наук
Кураленок Игорь Евгеньевич

Ведущая организация:

Государственное образовательное
учреждение высшего профессионального
образования "Санкт-Петербургский
государственный политехнический
университет"

Защита диссертации состоится "___" _____ 2006 года в _____ часов на заседании диссертационного совета Д 212.232.51 по защите диссертаций на соискание ученой степени доктора наук при Санкт-Петербургском государственном университете по адресу 198504, Санкт-Петербург, Старый Петергоф, Университетский пр., д. 28, ауд. 3536.

С диссертацией можно ознакомиться в Научной библиотеке Санкт-Петербургского государственного университета по адресу: 199034, Санкт-Петербург, Университетская наб., д. 7/9.

Автореферат разослан " " _____ 2006 года

Ученый секретарь диссертационного совета
доктор физико-математических наук,
профессор

Б.К. Мартыненко

Общая характеристика работы

Актуальность темы

Широкое распространение языков программирования высокого уровня с начала 60-х годов привело к появлению программных комплексов, используемых на протяжении нескольких десятилетий. Издержки на их сопровождение давно составляют большую часть затрат, связанных с ними, и с каждым годом стоимость поддержки только увеличивается [6].

В мировом масштабе по оценкам специалистов затраты на сопровождение старого программного обеспечения давно превысили затраты на создание нового ПО [9], только на одном из языков программирования долгоживущих систем, Коболе, написано до 30 % мирового программного обеспечения [4].

Одной из проблем сопровождения старых приложений является устаревание исходной платформы, что в свою очередь влечёт прекращение производителем поддержки операционной системы и аппаратного оборудования, отсутствие удобных инструментальных средств для внесения исправлений в программные комплексы и трудности в подборе технического персонала с необходимой квалификацией.

Возможным вариантом решения указанных проблем является автоматизированный перевод старой системы на иные языки и платформы. Кроме наличия специалистов, они обладают доступными средствами разработки программного обеспечения, поддерживающими рефакторинг [1] – структурные модификации существующего кода с сохранением функционального соответствия.

В случае если исходная и целевая среды близки, например, они используют разные диалекты одного языка, то процесс переноса системы хорошо изучен и может быть проведён высокоавтоматизированно [3,7].

Но в случае значительных различий исходной и целевой платформ, например, при использовании современных языков Джава, С++ или С#, существующие подходы к переводу либо обеспечивают невысокую сопровождаемость результатов [2,5], либо жертвуют полной функциональной эквивалентностью [8].

Таким образом, существует научная проблема высокоавтоматизированного перевода старых приложений на новые языки и платформы с получением сопровождаемых результатов.

Цели и задачи работы

Целью работы является исследование существующих и разработка новых подходов автоматизированного перевода для реинжиниринга старых приложений на новые языки и платформы. Как было показано выше, решения этой задачи имеют большое значение для облегчения сопровождения старых программных систем.

В рамках достижения цели были поставлены следующие задачи:

- Исследование существующих методов автоматизированных преобразований.
- Разработка новых подходов для решения проблем языкового перевода и улучшения качества результатов при различных сценариях реинжиниринга.
- Реализация предложенных подходов на практике.

Основные результаты

В работе получены следующие основные результаты:

1. Системный анализ отечественных и зарубежных работ, а также путей повышения их качества на основе практики известных исследований и проведенных автором натурных испытаний.
2. Подход к получению высококачественных результатов перевода.

3. Подход к преодолению трудностей языковых преобразований при переводе элементарных типов и операций над ними.
4. Стратегия создания классов на основе преобразования иерархии структурных описаний данных для инкапсуляции в одном классе связанных по реализации объявлений.
5. Метод перевода разделяемых описаний структур данных.
6. Метод управления зависимостью результатов перевода от внешней среды в процессе перевода и их дальнейшего сопровождения.
7. Промышленная реализация предложенных выше подходов и методов для перевода приложений на многих входных и выходных языках.
8. Апробация предложенных подходов и методов в рамках индустриальных проектов по переводу программных комплексов.

Научная новизна

Все основные научные результаты диссертации являются новыми.

Практическая и теоретическая ценность

Предложенные в диссертации подходы и методы могут быть использованы при создании новых средств автоматизированного преобразования программ.

На практике предложенные подходы и методы были использованы для создания промышленного средства реинжиниринга Relativity Modernization Workbench, а также при проведении проектов по преобразованию конкретных устаревших систем, проводившихся с использованием этого инструментального средства.

Апробация работы

Результаты диссертации докладывались на семинаре кафедры прикладной математики Санкт-Петербургского Политехнического Университета (2006 год, Санкт-Петербург), а также на конференции по сопровождению и реинжинирингу программно-

го обеспечения (CSMR 2003, Беневенто, Италия) и семинаре по анализу и модификации программного кода (SCAM 2003, Амстердам, Нидерланды).

Предложенные методы и полученные результаты были использованы в программном продукте Relativity Modernization Workbench.

Публикации

Основные результаты работы изложены в 4 работах [1,2,3,4], перечисленных в конце автореферата.

Структура и объем диссертации

Диссертация состоит из введения и 4 глав со сквозной нумерацией разделов, рисунков и таблиц, а также одного приложения. Текст диссертации изложен на 122 страницах. Список литературы содержит 129 наименований.

Содержание работы

Во **введении** демонстрируется актуальность выбранной темы исследований, излагается исторический контекст, цели и задачи диссертационной работы. Кратко перечислены тезисы, выносимые на защиту.

В **первой** главе рассмотрены теоретические основы автоматизированного перевода в задаче реинжиниринга программного обеспечения.

Произведён анализ предмета сопровождения в контексте жизненного цикла ПО, приведены виды деятельности и проблемы старых систем в этом контексте.

Показана роль реинжиниринга как средства преодоления проблем сопровождения, даны определения различным видам деятельности, используемым в сопровождении. Рассмотрены во-

просы применимости, рисков, эффективности и альтернатив использования реинжиниринга.

Продемонстрировано место автоматизированного перевода при выполнении реинжиниринга, показаны его задачи и применение. Дано сравнение автоматизированного перевода и компиляции, показаны различные подходы к переводу и их характеристики. Описаны проблемы перевода и существующие способы их преодоления.

Рассмотрены различные виды перевода: межъязыковые преобразования, реструктуризация, использование объектно-ориентированного подхода и рефакторинг. Отдельно приведены недостатки существующих подходов в контексте перевода старых приложений на новые языки и платформы, такие как: отсутствие решения проблемы конвертации разделяемых описаний, низкая способность модификации результатов перевода.

Во **второй** главе уделяется внимание научному и техническому контексту разработки средства автоматизированного реинжиниринга Modernization Workbench, в рамках которого происходили исследования и практическая апробация данной работы.

Показана область выполненного исследования, дан исторический обзор старого языка Кобол и нового языка Джава, перечислены другие входные и выходные языки, использованные для апробации.

Перечислены конкретные трудности перевода из одного языка в другой, а именно: полное различие типов, зависимость операций от физического представления памяти, не структурный поток управления, переиспользование описаний данных, основанное на файлах включения и т.д.

В **третьей** главе рассмотрена реализация автоматизированного перевода из Кобола в Джаву. Описаны предыдущие результаты исследования проблемы перевода, а именно использование пе-

ревода уровня реализации и перевода встроенных средств целевого языка. На конкретных примерах показана низкая понятность и модифицируемость порождённого кода при использовании первого подхода и недостатки результатов перевода при использовании второго подхода.

Для достижения компромисса между автоматизацией перевода и качеством его результатов автором предложен общий подход к применению автоматизированного перевода, заключающийся в следующих этапах:

- получения легко читаемых результирующих текстов с высокой степенью соответствия исходным текстам;
- ручные исправления фиксированного объёма для реализации кода, зависящего от внешней среды;
- рефакторинг отдельных фрагментов результирующих текстов для улучшения структуры приложения;

Данный подход позволяет осуществлять постепенный перевод, поскольку две стадии могут быть осуществлены в короткое время, а третья стадия может быть выполнена итеративно по потребности, при этом система находится в работающем состоянии, улучшаясь постепенно.

В целях достижения высокой читаемости и функциональной эквивалентности результатов перевода автором предложена комбинированная схема выбора проекций перевода, основанная на следующих критериях:

- области применения исходного языка (десятичная арифметика, записи данных для языка Кобол);
- идентификации практического применения низкоуровневых примитивов и структурных случаев использования операторов исходного языка, обнаруженных в реальных программных системах;
- сокрытия реализации изменчивой функциональности внутри классов со стабильным интерфейсом после перевода.

Приведены примеры перевода исходных текстов, содержащих операторы безусловного перехода, использование перекрытий

памяти и оператора пересылки. Показано, что результаты перевода удовлетворяют критериям функциональной эквивалентности, читаемости и модифицируемости.

Для преодоления различия в типах данных исходного и целевого языках автором предложено в процессе перевода осуществлять разделение представления элементарного значения от его операций, т.е. создавать *свойство* встроенного типа целевого языка, методы доступа которого скрывают специфику его представления. Данный подход является оригинальным в преодолении трудностей языковых преобразований, позволяющим достичь понятности результатов перевода в силу использования встроенных типов при сохранении функциональной эквивалентности типов и операций над ними.

Для достижения высокой степени автоматизации и постепенности перевода автором предложено разделить процесс создания классов на автоматическое создание классов-“кандидатов” и последующий рефакторинг попутно с более глубоким пониманием приложения.

При переводе определённых операторов исходного языка (например, файловые операции Кобола) и встроенных языков, использующих структуры данных исходного языка, автором предложено создавать методы в классах, порождённых из записей, для поддержки этого типа операций, а сами операции переводить на вызовы созданных методов класса. Поскольку многие операции выполняются несколько раз, то происходит локализация их реализации после перевода. В работе приведены примеры результатов перевода файловых операций с использованием указанной техники.

Автором предложена оригинальная стратегия построения классов с сокрытием реализации. Каждая запись первого уровня трансформируется в класс, а все вложенные поля становятся свойствами объемлющего класса. Если элементарное поле было частью массива, то оно становится индексированным свойством

после перевода. Таким образом, реализация поддержки свойств после перевода оказывается внутри одного класса. Если запись связана по памяти с другой записью, то происходит объединение классов перевода, что позволяет скрыть реализацию неструктурных случаев поддержки памяти также внутри одного класса.

Важным отличием обсуждаемого автоматизированного перевода от аналогичных подходов является метод перевода разделяемых описаний структур данных, состоящей из семантически управляемого слияния оптимизированных образов индивидуальных трансляций. Использование предложенного автором метода позволяет обеспечить повторное использование результатов перевода разделяемых описаний структур данных при независимой трансляции программных модулей.

В целях обеспечения поддержки постепенного перевода крупных приложений автором предложена специальная поддержка межпрограммных вызовов, а именно выполнение их посредством вызова через класс контекста приложения. Поскольку для каждой программы возможно создание класса-оболочки, который осуществляет вызов удалённой программы, а, кроме того, интерфейс программного класса и созданной оболочки совпадают, то в процессе перевода появилась возможность изменять реализацию модуля без модификации его клиентов.

Для возможности лёгкой настройки системы перевода на различные конфигурации целевой среды, а также для возможности модификации кода, зависящего от внешней среды, *после* проведения автоматизированного перевода автором предложен следующий метод. В процессе перевода кроме создания классов происходит порождение файлов описаний связи с внешней средой. Из них с помощью специальной трансформации получается код, зависящий от внешней среды, который затем внедряется в реализацию классов. Поскольку в процессе перевода указанный код был локализован в виде методов, то происходит замещение только его реализаций. При необходимости, предложенную

процедуру можно многократно повторить после проведения перевода с модифицированными описаниями трансформаций.

Таким образом, с помощью предложенного метода результаты перевода *параметризованы зависимостью от внешней среды*, что позволяет существенно уменьшить их ручные исправления в процессе постепенного перевода и дальнейшего адаптивного сопровождения.

Автором предложено осуществлять тестирование интерактивных приложений методом “чёрного” ящика с помощью вышеуказанной возможности заменять реализацию отдельных методов. На первом этапе в код взаимодействия с пользователем добавляются команды сбора данных и осуществляется прогон всех сценариев работы. На втором, тестовом, этапе реализация обмена данных с пользователем заменяется на код, имитирующий пользователя с помощью собранной информации и сравнивающий текущие данные приложения с эталонными. Возможность осуществления регрессионного тестирования существенно упрощает проведение дальнейших структурных преобразований приложений и их дальнейшего сопровождения после перевода.

Отдельные фрагменты результатов перевода могут быть подвергнуты рефакторингу. Автором выделены следующие направления структурных модификаций:

- дальнейшее улучшение понимания кода (преобразования: переименования, введение константы/переменной, выделение метода);
- дальнейшее улучшение объектной структуры (преобразования: извлечение базового класса/интерфейса, изменение сигнатуры метода)
- выделение общего кода между программными классами и вынесения его в отдельные классы (преобразования: сделать метод статическим, перенести метод, сделать статический метод методом экземпляра);

Автором описано применение предложенных подходов и методов для автоматизированного перевода программ с другими входными (ПЛ1, Натурал) и выходными языками (Visual Basic и C++).

Четвертая глава посвящена промышленной апробации построенной реализации автоматизированного перевода. Описаны результаты использования предложенного метода на реальных приложениях, в том числе для промышленного приложения объёмом около 2-х миллиона строк кода, состоящего из около 3000 программных модулей.

Применение разработанной системы автоматизированного перевода на промышленном проекте продемонстрировало возможность её лёгкой адаптации к требованиям заказчика в направлениях, предложенных автором, а именно: получения высококачественных результатов перевода, преодоления трудностей языковых преобразований для элементарных типов, перевода разделяемых описаний и управления зависимостью результатов перевода от внешней среды.

Приведены данные тестирования производительности результатов перевода, описан предложенный автором метод кэширования промежуточных объектов в реализации динамической поддержки, позволяющий ускорить работу приложения в 2-5 раз.

В заключении, приведена общая характеристика диссертационной работы и основные выводы по её результатам.

Список литературы

1. M. Fowler, K. Beck, J. Brant, et. al., "Refactoring: Improving Design of Existing Code", Addison Wesley Longman, 1999
2. R. Gray, T. Bickmore, S. Williams, "Reengineering Cobol Systems to Ada," In Proceedings of Air Force/Army/Navy Software Technology Conference, US Dept. of Defense, 1995.
3. M. Harsu "Reengineering Legacy Software through Language Conversion", Report A-2000-8, Department of Computer and Information Sciences, University of Tampere (Doctoral thesis), June 2000, 150 pp.
4. C. Jones "The Year 2000 Software Problem – Quantifying the Costs and Assessing the Consequences", Addison-Wesley, 1998
5. LegacyJ's PercCobol - Cobol to Java compiler
6. B. Lientz, E. B. Swanson "Problems of application software maintenance", Communications of the ACM, Vol. 24, No. 11, 1981, pp. 763–769.
7. H. M. Sneed "Object-Oriented COBOL Recycling", In Proceedings of Working Conference on Reverse Engineering, IEEE Computer Society Press, 1996, pp. 169-178
8. A. A. Terekhov, C. Verhoef "The Realities of Language Conversions", IEEE Software, November/December 2000, Vol. 17, No. 6, pp. 111–124.
9. C. Verhoef "Software Development is a Special Case of Maintenance", In Proceedings of International Conference on Software Engineering and Applications, IEEE Computer Society Press, 1999

Работы автора по теме диссертации

1. М. Мосиенко "Построение динамической поддержки для задач реинжиниринга", в сб. "Автоматизированный реинжиниринг программ", СПб, изд-во С.-Петербургского университета, 2000. С. 145–164.

2. M. Mossienko “Automated Cobol to Java Recycling”, In Proceedings of 7th IEEE Conference on Software Maintenance And Reengineering, IEEE Computer Society Press, 2003, pp. 40-49
3. M. Mossienko, O. Khaschansky, D. Antonov, O. Smirnov, A. Gubanov “Towards managing environment dependence during legacy systems maintenance and renovation”, In Proceedings of 3rd IEEE Workshop on Source Code Analysis and Manipulation, IEEE Computer Society Press, 2003, pp. 131-140
4. М. Мосиенко, А. Тиунова “Интеграция программной логики с пользовательским интерфейсом в процессе реинжиниринга” в сб. "Труды кафедры системного программирования", СПб, изд-во С.-Петербургского университета, 2004. С. 199–224.

ЛР № 040815 от 22.05.97

Подписано к печати . Формат бумаги 60x90 1/16.

Бумага офсетная. Печать ризографическая.

Объем 1 п.л. Тираж 100 экз. Заказ .

Отпечатано в отделе оперативной полиграфии НИИХ СПбГУ
с оригинал-макета заказчика.

198504, Санкт-Петербург, Старый Петергоф, Университетский пр., 26.

Бесплатно