Российская академия наук
Сибирское отделение
Институт Систем Информатики

На правах рукописи

Недоря Алексей Евгеньевич

Расширяемая переносимая система программирования, основанная на биязыковом подходе

05.13.11 – математическое и программное обеспечение вычислительных машин, комплексов, систем и сетей

Автореферат диссертации на соискание учёной степени кандидата физико-математических наук

Новосибирск – 1994

Работа выполнена в Институте систем информатики Сибирского отделения Российской академии наук (ИСИ СО РАН)

Научный руководитель:	И.В. Поттосин доктор физико-математических наук
Официальные оппоненты:	А.Н. Терехов доктор физико-математических наук
	С.Б. Покровский кандидат физико-математических наук
Ведущая организация:	Вычислительный Центр РАН

Защита диссертации состоится "11" марта 1994 г. в 14^{30} часов на заседании специализированного совета К003.93.01 в Институте систем информатики Сибирского отделения РАН по адресу: 630090, Новосибирск, пр. ак. Лаврентьева, 6.

С диссертацией можно ознакомиться в читальном зале библиотеки ВЦ СО РАН (пр. ак. Лаврентьева, 6).

Автореферат разослан "10" февраля 1994 г.

Ученый секретарь
специализированного совета
К 003.93 01
к.ф.-м.н. В.К. Сабельфельд

Общая характеристика работы

Актуальность темы. В работе рассматриваются вопросы построения систем программирования (СП), удобных для разработки прикладных систем. Построение СП является основной задачей инструментального программирования. Достаточно сложным (и в большой степени субъективным) является критерий "хорошести" СП. Не претендуя на полноту определения, постараемся привести наиболее важные критерии "хорошей" СП:

- надежность;
- переносимость;
- полнота;
- адаптируемость и расширяемость.

Целью данной работы являлась выделение принципов реализации "хорошей", а именно расширяемой и переносимой системы (РПС), удовлетворяющей приведенным критериям. Работа по достижению этой цели была разбита на две подзадачи:

- выбор подходящих языков, схемы трансляции и реализация переносимых компиляторов.
- исследования принципов построения расширяемых систем и разработка системы.

Научная новизна и практическая значимость работы заключается в следующем:

- Выделены критерии пригодности языков для реализации РПС; проведено сравнение языков с точки зрения их пригодности.
- Предложена новая методика реализации мало-языковых транслирующих систем.
- Реализовано семейство переносимых компиляторов и трансляторов с языков Модула-2 и Оберон-2.
- Проведен анализ и сравнение расширяемых систем.
- Разработаны принципы организации и структурирования РПС на базе одиночного наследования. Разработана архитектура системного ядра РПС.
- Реализовано ядро РПС Mithril.

Апробация работы. Результаты работы неоднократно докладывались на семинарах и конференциях, в том числе:

- 1-ая Всесоюзная школа семинар по языку Модула-2, Наманган, 1987.

- 2-ая Всесоюзная школа семинар по языку Модула-2, Севастополь, 1988.
- Рабочее совещание по архитектуре процессоров Кронос и программного обеспечения, Паланга, 1989.
- Рабочий семинар Institute of Computer Systems, ETH, Zurich, 1991.
- Международная конференция "Cooperative Standartisation", London, 1993.
- Рабочие семинары ИСИ СО РАН (1992, 1993).

По теме диссертации опубликованы 6 научных работ.

Структура и объем работы. Диссертация состоит из введения, четырех глав, списка литературы (33 наименований) и двух приложений.

Содержание работы

Во введении рассмотрены критерии качества систем программирования. Формулируются основные цели работы и практическая значимость результатов. Приводится краткое содержание диссертации по главам.

Первая глава посвящена определению расширяемых переносимых систем (РПС) и критериям выбора языка реализации таких систем.

В первом разделе дается определение РПС. Мы будем называть расширяемой переносимой системой (РПС) систему, построенную по следующим принципам:

расширяемость:

- ядро системы определяет набор базовых понятий;
- этот набор понятий может быть расширен без изменения ядра (и без перекомпиляции);
- новые понятия могут быть определены на базе уже определенных понятий;
- отсутствуют различия между базовыми и вновь определенными понятиями;

переносимость:

- все машинно-зависимые и системно-зависимые части РПС текстуально выделены;
- объем таких частей существенно меньше объема ядра системы;

- система может быть настроена таким образом, чтобы (почти) полностью использовать ресурсы конкретной аппаратуры и ОС.

При разработке любой системы необходимо обращать внимание на надежность системы. Для РПС критерий надежности приобретает новые черты, так как мы должны обеспечить надежное функционирование не только базовых механизмов, но и их возможных расширений и переопределений. Для обеспечения надежности в традиционных системах применяется принцип разделения системной части (супервизор) и пользовательской, при этом достаточно обеспечить надежность и независимость системной части для устойчивого функционирования системы. В расширяемых системах такое разделение не существует, поэтому особенно важным становится использование языка программирования, обеспечивающего достаточный уровень надежности.

Во втором разделе перечисляются основные требования к языку реализации РПС, в том числе, и требования по надежности языка. Проводится анализ соответствия языков программирования двум формальным моделям объектно-ориентированного программирования. Обосновывается выбор необходимых для реализации расширяемой системы языков программирования. Подтверждается важность таких свойств языка, как:

- сильная статическая типизация;
- динамическая типизация;
- сборка мусора.

Проведенный в разделе анализ подтвердил правильность выбора языка Оберон-2 для реализации РПС. Некоторой проблемой является отсутствие низкоуровневых средств в этом языке. Мы считаем это скорее сильной чертой языка, чем недостатком. Язык программирования высокого уровня не должен содержать "опасных" средств. Возникает вопрос о языках программирования таких низкоуровневых частей системы, как динамическая поддержка, сборка мусора, реакция на программные прерывания и т.д. Эти части системы не являются переносимыми, и могут быть написаны на разных языках для различных платформ. Но мы считаем, что трудоемкость переноса может быть существенно уменьшена, если система содержит также язык программирования низкого уровня. В качестве такого языка мы используем язык Модула-2. Может показаться странным использования Модулы-2 в качестве языка низкого уровня. Но заметим, что Модула-2 обладает очень важным свойством, присущим скорее языку низкого уровня, чем ЯВУ: отсутствие собственной динамической поддержки.

Мы считаем, что наличие пары языков в системе, близких по синтаксису и семантике, но отличающихся некоторыми существенными свойствами (Оберон-2: расширение типа, сборка мусора; Модула-2: низкоуровневое программирование, отсутствие динамической поддержки) позволяет значительно увеличить общий потенциал системы программирования.

Во второй главе описывается реализация переносимой компилирующей системы, являющейся необходимым инструментом создания РПС.

В первом разделе выполняется анализ сравнения различных методов построения переносимых компиляторов и на его основе вводится понятие *биязыковой транслирующей системы*. Описываются преимущества биязыковой системы над многоязыковой. Обосновывается выбор схемы компиляции, при которой фаза анализа компилятора строит промежуточное представление единицы компиляции в памяти, и затем это промежуточное представление используется фазой синтеза.

Рассмотрим немного более подробно способы реализации пары переносимых компиляторов. Эти компиляторы можно было бы реализовывать полностью независимо, но желательно было найти способы, позволяющие уменьшить затраты на разработку. Очевидное решение – это реализовать многоязыковую транслирующую систему, позволяющую для *N* языков и для *M* целевых машин реализовать *N* фаз анализа, транслирующих некоторой язык программирования в промежуточное представление, и *M* фаз синтеза, транслирующих промежуточное представление в коды конкретной ЭВМ, и тем самых существенно сократить затраты на реализацию (с *N* * *M* до *N* + *M*).

Такой общий подход принят в системе Бета. Промежуточное представление определяется языком ВЯ3. В рамках проекта были выполнены реализации языков Паскаль, Симула-67 и Фортран и подмножества языков Модула-2 и Ада. Наибольшая нагрузка в проекте выпадает на внутреннее представление (внутренний язык). Требования, предъявляемые к ВЯЗу, весьма противоречивы. С одной стороны, все конструкции языков программирования должны достаточно удобно отображаться в него, с другой стороны он должен быть достаточно удобен для генерации кода. Соответственно уровень его должен быть достаточно низок (очевидно, что все операторы управления могут быть представлены в терминах переходов управления), и достаточно высок, для того чтобы можно было выполнять оптимизацию и генерацию кода на разные машины.

ВЯЗ построен как объединение конструкций входных языков системы. Попытки совместить противоречивые требования привели к большому объему как внутреннего языка, так и системы в целом.

Отрицательный результат экспериментов с многоязыковыми системами (не только с системой Бета) является отражением того факта, что проблема в целом не разрешима (или пока не разрешима) и для достижения успеха необходимо пойти на некоторые ограничения. Если для систем типы системы Бета использовать обозначение $N \rightarrow M$, то естественно рассмотреть случаи, когда N или M равно 1, то есть компиляция одного языка на множество машин, или компиляция множества языков на одну машину. Введение таких ограничений оказывается вполне достаточным. Примером удачной системы $N \rightarrow 1$ является система TopSpeed, в которой реализованы языки Модула-2, Паскаль, C, C++ для IBM PC, а примером удачной системы $1 \rightarrow M$ является Оберон компилятор OP2, который в настоящее время перенесен на такие существенно различные архитектуры, как SPARC, MC 680x0, MIPS, RS/6000 и i80386.

Успех таких систем обусловлен тем, что в обоих случаях существует определенность на одном конце системы. Для системы $1 \rightarrow M$ определен входной язык, и вся дальнейшая работа может выполняться в его терминах. Промежуточное представление соответствует входному языку. Для системы $N \rightarrow 1$ задана целевая архитектура, и каждый входной язык системы транслируется в некоторое промежуточное представление, близкое к целевой архитектуре. Необходимо также отметить неявное задание семантики, что существенно облегчает разработку промежуточного представления. В случае неограниченной системы ($N \rightarrow M$) все семантические особенности языков должны быть явно отражены в промежуточном представлении.

Проведенные анализ показывает нам трудности разработки многоязыковых систем. С другой стороны, мы не можем ограничиться схемой $1 \rightarrow M$ или $N \rightarrow 1$. Выход из этого тупика дает анализ необходимых нам входных языков. Язык Оберон-2 является расширенным подмножеством языка Модула-2. Семантика большинства конструкций этих языков совпадает.

Таким образом, мы приходим к отображению $1+\varepsilon \rightarrow M$, где ε определяет количество дополнительных конструкции, необходимых для реализации второго входного языка ($\varepsilon << 1$). При таком подходе мы будем (как и в Бете) строить промежуточное представление как отображение объединения двух языков, но в отличие от системы Бета, набор входных языков определен изначально и не может быть расширен и, что очень важно, семантика этих языков для большинства конструкций совпадает. Так как промежуточное представление является общим для обоих языков, то и фазы синтеза для каждой целевой архитектуры совпадают, а при реализации фазы анализа можно выделить все общие компоненты компилятора, сократив тем самым и затраты на разработку фазы анализа.

Во втором разделе описывается подход к реализации входных языков системы. Основной проблемой реализации является отсутствие общепринятых стандартов входных языков системы.

В третьем разделе описывается внутреннее представление компилятора, построенное как объединение внутренних представлений двух входных языков системы.

Для внутреннего представления нами выбрано *синтаксическое дерево* (СД) единицы компиляции. Это представление не является идеальным для выполнения оптимизаций и качественной генерации. Впрочем, для разных оптимизаций требуется различная форма представления. Мы полагаем, что для выполнения оптимизаций дерево может перестраиваться в некоторую удобную форму. Такое преобразование часто удобней выполнять с деревом, чем с другими видами промежуточных представлений. В двух реализованных генерациях для оптимизации структур управления по дереву строится граф управления. Для простой, неоптимизирующей генерации СД является оптимальным.

В четвертом и пятом разделе описывается структура компилятора и организация взаимодействия фазы анализа и фазы синтеза.

Компилятор состоит из интерфейсного слоя, парсера, генератора и утилитного слоя.

Интерфейсный слой содержит операции взаимодействия с системой, утилитный слой определяет операции чтения исходного текста и выдачу сообщений компилятора. Меняя утилитный слой, мы можем перейти от обычного пакетного компилятора к турбо-компилятору, читающему текст из буфера редактора и так далее.

Парсер выполняет синтаксический и семантический анализ, строит СД единицы компиляции и возвращает его корень. Если при этом были обнаружены ошибки, то компиляция завершается. Генератор всегда работает с корректным синтаксическим деревом. Генератор выполняет проход по объектам модуля, вычисляя необходимые атрибуты: размер типов, размещение переменных, размер области параметров и т.д. После чего выполняется запись симфайла и затем генерация кода.

Вычисление атрибутов объектов перед записью симфайла необходимо, так как в симфайле должна присутствовать информация о них. Так для генерации доступа к переменной из другого модуля необходимо знать размещение этой переменной.

Структура генератора для разных платформ может сильно различаться. Генератор может выполняться в несколько проходов по СЛ, и может включать в себя фазу машинно-зависимой оптимизации.

Машинно-независимая оптимизация может быть вставлена между фазами анализа и синтеза.

Шестой раздел посвящен описанию реализации семейства компиляторов. Для каждой платформы описаны особенности реализации фазы синтеза. В настоящее время в состав семейства входят компиляторы для рабочих станций Кронос и для машин на базе i386/486 и трансляторы в ANSI C.

В седьмом разделе описываются некоторые особенности реализации языка Оберон-2, такие как динамическая типизация и канонизация симфайлов.

И, в последнем разделе, приводятся характеристики семейства компиляторов и сравнение его представителей. Отметим основные особенности реализации семейства:

- промежуточное представление является объединением представления двух языков;
- только синтаксический анализ реализован отдельно для каждого входных языков; все остальные компоненты компиляторов являются общими; только в некоторых случаях (достаточно редко) семантический анализ и/или генерация различается для входных языков;
- синтаксический анализ выполнен методом рекурсивного спуска;
- для реализации таблицы символов и алгоритмов идентификации используются бинарные деревья;
- при записи симфайла используется оригинальный алгоритм канонизации дерева;
- модель взаимодействия парсера и генератора позволяет получить полностью независимый от платформы парсер;
- наличие в составе семейства транслятора в ANSI C позволяет осуществить быстрый перенос ПО практически на любую платформу;
- все семейство реализовано на языке Модула-2.

Успешный перенос компиляторов и их использование дает нам основание утверждать, что идеи, заложенные в основу семейства, являются весьма продуктивными. Тесно связанная реализация двух языков позволяет разрабатывать ПО, используя в каждом случае достоинства каждого языка. Дальнейшее развитие семейства может идти в двух направления: перенос на другие платформы и реализация машинно-независимых (или машинно-ориентированных) оптимизирующих преобразователей дерева.

В третьей главе выполняется анализ расширяемых систем на примере системы Оберон. В четырех разделах анализируется структура системы и возможности ее расширения. На основании анализа перечисляются слабые места системы.

Разработка системы Оберон была начата Н. Виртом и Ю. Гуткнехтом в 1985 году. Целью проекта была разработка современной и переносимой ОС для персональных рабочих станций. Одновременно с разработкой системы был спроектирован и реализован язык Оберон. В первую очередь система и Оберон компилятор были реализованы для рабочих станций Ceres. Далее был выполнен перенос на такие платформы, как Mac II, SparcStation/Unix, DecStation/Unix, i386/AIX, Chameleon. На всех этих платформах (кроме Ceres и Chameleon) система Оберон является не операционной системой в традиционном смысле, а некоторым окружением, работающим над базовой ОС.

В четвертой главе описана экспериментальная РПС Mithril. При описании системы особое внимание уделяется критериям выбора проектных решений. Название системы взято из книги Р.Р. Толкиена "Властелин колец". Мифрил – это название подлинного серебра (the True Silver), которое добывалось гномами в недрах Мории. Удивительно легкий и прочный металл обогатил и прославил гномов.

Мы полагаем, что слово Mithril сконструировано Толкиеном из словосочетания mythical rill (мифический источник). Так это или нет, но этот смысл хорошо соответствует идеи РПС.

Мы предполагаем, что система Mithril будет использоваться как база для разработки переносимого прикладного ПО. Для того чтобы упростить разработку прикладного ПО, необходимо решить несколько важных задач:

- Необходимо создать комфортную обстановку для программиста (языки, отладка, поддержка проекта);
- Система должна включать набор средств для создания пользовательского интерфейса (окна, мышь, конструктор интерфейсов);
- Система должна поддерживать одинаковую обстановку на разных платформах (т.е. на разных машинах и под разными ОС).

В первом разделе описываются принципы построения и структура системы, а также влияние сборки мусора на построение системы. Анализируются пути повышения адаптируемости и переносимости системы.

Во втором разделе описывается динамическая поддержка языка

Oberon-2, включая сборку мусора и поддержку метаязыкового программирования.

Третий раздел посвящен описанию основных механизмов системного ядра, построенных на базе одиночного наследования:

- понятие "объекта" и финализации объекта;
- контекст ошибки;
- расширяемый механизм ввода/вывода, построенный на идее разделения "носителей" ввода/вывода и объектов доступа;
- динамическая загрузка;
- постоянные (persistent) объекты;
- файлы;
- понятие текста со встроенными объектами;
- оконная система;
- функционирование системы.

В четвертом разделе описывается надстройка над системным ядром – оболочка системы. Оболочка системы содержит набор расширений базовых понятий, необходимых для реализации пользовательского интерфейса, и набор необходимых команд. Необходимо отметить, что на базе ядра системы можно построить совершенно различные оболочки. Более того, оболочка системы реализована таким образом, что любой ее объект может быть изменен или расширен, и это не повлияет на работоспособность остальных объектов.

В пятом разделе описывается разработка прикладной задачи в системе на примере простого расширяемого редактора математических формул. В этом редакторе формула представляется в виде встроенного в текст объекта. Реализованное ядром РПС понятие *текста* позволяет осуществить вставку (хранение, редактирование) в тексте произвольных объектов, реализуя тем самым мощную базу для разработки систем гипертекста и систем подготовки документов.

Раздел содержит детальное описание реализации расширяемого редактора формул, причем ядро редактора состоит из 300 строк на языке Оберон-2. Редактор позволяет получать изображение математической формулы по описанию на простом функциональном языке, так например:

```
sqrt(div("a+b","c+d"))
```

должно быть отображено в

$$\sqrt{\frac{a+b}{c+d}}$$

В шестом разделе описывается опыт переноса системы Mithril. В настоящее время выполнен перенос системы в среду MS-DOS для старших моделей PC. Для переноса использовались переносимые компиляторы с генерацией для i386/486.

Основной сложностью была реализация специальной утилиты – загрузчика (DOS extender), который переводит процессор в защищенный режим и обеспечивает доступ к базовым функциям операционной системы. Реализация такого загрузчика была выполнена О. Шатохиным.

Полученный при переносе опыт позволил существенно улучшить переносимость системы и обеспечить полную совместимость приложений на уровне исходных текстов.

В седьмом разделе описываются основные особенности РПС Mithril:

- поддержка расширяемости на всех уровнях системы;
- динамическое создание объекта по типу;
- универсальный механизм финализации;
- унифицированный механизм сохранения контекста ошибки;
- поддержка постоянных объектов;
- универсальный расширяемый механизм ввода/вывода;
- поддержка базового набора графических примитивов;
- оконная система, поддерживающая иерархические пере-крывающиеся окна.

Основные результаты работы

1. Выработан набор критериев, которым должен удовлетворять язык реализации РПС. Показано, что язык Оберон-2 удовлетворяет этим критериям.
2. Предложена методика реализации мало-языковых транслирующих систем, заключающаяся в фиксации языкового надмножества набора семантически близких языков, построении анализатора для этих языков в единое внутреннее представление и реализации генерации с внутреннего представления для каждой платформы, на которую необходимо перенести компиляторы. Показано, что для пары близких языков (в нашем случае Модула-2 и

Оберон-2) трудозатраты на реализацию семейства переносимых компиляторов не на много превышают затраты на разработку переносимого компилятора для одного языка. Выполнен анализ схем трансляции и обоснован выбор такой схемы, при которой внутреннее представление между фазами анализа и синтеза является синтаксическим деревом в памяти.

3. Проведена разработка внутреннего представления и реализованы анализаторы языков Модула-2 и Оберон-2. Эти языки достаточно близки как синтаксически, так и семантически, что позволило выделить общую часть, включающую такие компоненты, как лексический анализ, обработка ошибок, семантический анализ, включая алгоритмы идентификации и межмодульный импорт и экспорт.
4. Автором выполнена реализации генерации кода для ЭВМ Кронос и генерация в текст на языке ANSI C, что позволяет переносить транслирующую систему на любую платформу, для которой есть ANSI C компилятор (то есть на любую платформу, кроме ЭВМ Кронос). Простота реализации такой генерации в первую очередь, обусловлена выбором синтаксического дерева в качестве внутреннего представления.[1]
5. Проведен анализ расширяемых систем на примере системы Оберон. Выявлен ряд недостатков системы Оберон. Показана важность реализации в ядре системы механизмов финализации и постоянных объектов.
6. Выполнено проектирования ядра РПС. Показано, как с использованием только одиночного наследования можно построить универсальный расширяемый механизмы ввода/вывода, сохранения контекста ошибок, и т.д.
7. Совместно с А. Никитиным[2] и А. Хапугиным[3] реализовано ядро РПС Мифрил.
8. Выполнен перенос системы Мифрил в среду MS-DOS.

Основные положения диссертации опубликованы в следующих работах:

1. Кузнецов Д.Н., Недоря А.Е., Тарасов Е.В., Филиппов В.Э. КРОНОС – автоматизированное рабочее место

[1] А. Денисовым и О. Шатохиным реализована генерация кода для процессоров i386/486.
[2] оконная подсистема, графика
[3] динамическая поддержка, загрузчик

профессионального программиста. В сб.: Автоматизированное рабочее место программиста. Новосибирск, 1988, с. 49-67.

2. Кузнецов Д.Н., Недоря А.Е. Проектирование таблиц символов для языков со сложными правилами видимости. В сб.: Методы трансляции и конструирования программ. ВЦ СО АН СССР, Новосибирск, 1984, с. 153-158.
3. Кузнецов Д.Н., Недоря А.Е. Симфайлы как интерфейс операционной системы. В сб.: Информатика. Технологические аспекты. ВЦ СОАН СССР, Новосибирск, 1987, с. 68-75.
4. Недоря А.Е., Никитин А.Г., Mithril – переносимая Оберон-2 система. В сб.: Среда программирования: методы и инструменты. ИСИ СО РАН, Новосибирск, 1992, с. 99-109.
5. Никитин А.Г., Недоря А.Е., Оконная поддержка в системе Mithril. В сб.: Среда программирования: методы и инструменты. ИСИ СО РАН, Новосибирск, 1992, с. 119-125.
6. A. Nedorya. Mithril – A Portable Oberon-2 Environment. The Modular Mag, 1993, No 1, p. 24-27.

Подписано в печать 17.01.94

Формат бумаги 60 x 90 1/16 Объем 0,9 уч-изд.л.

Заказ 9 Тираж 100 экз.

Отпечатано на ротапринте Вычислительного Центра СО РАН

630090, Новосибирск, 90 проспект Академика Лаврентьева, 6