$$L = \frac{P_n P_n^0}{P_n' P_n''} \left(\frac{D_n}{P_n^{**}} - \frac{V_n}{P_n^*} \right) + \frac{\dot{q}_n^0}{q_n^0} (-1 - (D_n - V_n)) - \frac{1}{P_n^0} \left(1 + \left(\frac{D_n P_n'' P_n^*}{P_n' P_n^{**}} - \frac{V_n P_n' P_n^{**}}{P_n'' P_n^*} \right) \right),$$

$$P_i = p_i' + p_i''$$

ЗАКЛЮЧЕНИЕ

Таким образом, можно сделать вывод о том, что конкуренция оказывает укрепляющее действие, как на устойчивость рыночного равновесия, так и на его неустойчивость. Поэтому эффект конкуренции является не причиной, обеспечивающей устойчивое развитие рынка, а лишь его акселератором в ту или иную сторону.

Литература

1. Калитин Б. С. Математические модели экономики. Мн.: БГУ, 2004.
2. Малкин И. Г. Теория устойчивости движения. М.: Наука, 1966.
3. Калитин Б. С. Рынок типа эффективная конкуренция с неотрицательным запасом прочности // Вестник БГУ. 1998. Сер. 1. №2. С. 41–45.

УМЕНЬШЕНИЕ РАЗМЕРА JAVA-ПРОГРАММ

В. С. Горшунов

Введение

При выполнении Java-проектов, по технологиям, отличным от «тонкого клиента», Java-разработчики сталкиваются с проблемами распространения приложений и их обновления в условиях низкоскоростных каналов связи, конфликтов версий Java Runtime Environment (JRE), а так же защиты проектов от декомпиляции в исходный Java-код. Как правило, комплект поставки состоит из JRE, внешних библиотек и собственно приложения, что составляет от нескольких десятков до ста мегабайт. Из них около 20 мегабайт занимают пользовательские jar-библиотеки, 60–70 мегабайт JRE, а само приложение занимает в среднем всего 1–2 мегабайта. Очевидно, что части JRE и части используемых внешних библиотек являются избыточными для функционирования приложения. В работе предлагается сокращать их объем за счет удаления неиспользуемых class-файлов.

Целью данной работы является исследование технологических аспектов функционирования систем сжатия Java-проектов.

1. ОБЛАСТИ ПРИМЕНЕНИЯ

Рассмотрим области, в которых требуется уменьшение.

- Часто применяется технология «толстого клиента». Подобные проекты представляют из себя обычные Java-приложения, с

пользовательским интерфейсом Swing, AWT или SWT, внешними jar-библиотеками и, возможно, локальными базами данных. Для их запуска необходима определенная версия JRE. Объект поставки представляет собой инсталляционный файл, содержащий все компоненты приложения. Пользователь далеко не всегда в состоянии самостоятельно обеспечить такое приложение необходимой версией JRE, поэтому часто JRE включается в объект поставки. В итоге, как было сказано ранее, размер установленного приложения может достигать 100 мегабайт.

- Вариантом поставки отличной от инсталляционного пакета является Java Web Start развертывание. В этом случае для запуска приложения клиенту необходимо зайти на определенным образом сформированную Web-страницу. Скрипт в этой странице автоматически определяет установленную версию JRE на компьютере клиента, соотносит ее с потребностями разворачиваемого приложения и при необходимости предлагает скачать нужную версию JRE с сайта компании Sun. Далее закачивает в кэш клиенту jar-архивы необходимые для запуска приложения и запускает само приложение. Недостатком является существенная загрузка канала связи при передаче, например, 20 мегабайт библиотек приложения.

- Отдельно можно выделить RCP приложения – приложения, основанные на платформе eclipse. Они в свою очередь могут быть как «толстыми клиентами», так и распространяться через Java Web Start. Их особенностью являются разделение, в отличие от классических Java-приложений, физической и логической структуры модулей, поддержка вложенных модулей и наличие собственного механизма разрешения зависимостей между ними.

- Мидлеты – это Java-приложения для мобильных устройств, таких как мобильные телефоны и карманные компьютеры. Для этих приложений объем является критическим фактором, так как многие устройства не поддерживают мидлеты больше 1 мегабайта.

2. ТЕХНИКИ СЖАТИЯ

Основными техниками сжатия Java-приложений являются технология Obfuscation (запутывание), Shrinking (сжатие) и Optimization (оптимизация).

Shrinking – техника сжатия, основанная на вырезании неиспользуемых class-файлов из самой программы, а также из используемых ею библиотек. Такому виду сжатия также может подвергаться JRE. В среднем это приводит к уменьшению объема на 20–30%, максимально до 50%.

Obfuscation – техника, в большей степени используемая для защиты Java-программ от декомпиляции. Имена пакетов, классов, методов, полей

и внутренних переменных кодируются набором коротких, не имеющих определенной смысловой нагрузки, наборами символов (например: a,b,c...). Это уменьшает размер таблиц вызовов и ссылок в class-файлах. Как побочный эффект – незначительное уменьшение объема на 5–15%.

Optimization – техника, обеспечивающая оптимизацию байт кода class-файлов. В процессе оптимизации удаляется отладочная информация, длинные команды заменяются более короткими, удаляется поддержка старых виртуальных машин. Это приводит к незначительному уменьшению объема на 5-10% и усложняет декомпиляцию, так как из сигнатур методов удаляются имена передаваемых параметров.

Рассмотрим процесс разрешения зависимостей при работе Java приложения. За подгрузку class-файла отвечает Class Load Agent [1]. Когда во время выполнения программы происходит обращение к классу, Class Load Agent в текущем множестве jar-файлов ищет требуемый class-файл и подгружает его в память. Текущее множество jar-файлов создается при старте Java-приложения.

В RCP приложениях вводится новое понятие для разрешения зависимостей. Это bundle (пучок). Во время запуска RCP приложения происходит полное разрешение зависимостей по bundle. При сжатии RCP приложений необходимо полностью сохранять структуру зависимостей по bundle и уметь обрабатывать вложенные модули.

Таким образом, любой class-файл может быть изменен, удален или перемещен в другой jar-файл без существенных проблем [2]. Единственной трудностью остаются подписанные jar-файлы. После сжатия их необходимо переподписать, либо отказаться от подписи.

В данной работе разработан подход сжатия, учитывающий рассмотренные особенности. Перед сжатием проводится этап анализа сжимаемого приложения. Он состоит из построения трех покрытий:

- Статическое покрытие – покрытие зависимостей одного class файла от другого по таблице импорта;
- Class.forName() покрытие – статическое покрытие байт-кода class файлов для нахождения вызовов динамического создания объектов. При нахождении подобной структуры выполняется попытка нахождения константы вызова. Создается список зависимостей вызывающий класс - вызванный класс;
- Динамическое покрытие – покрытие множества загружаемых class файлов при работе приложения.

При сжатии происходит копирование структуры исходной папки в папку назначения. Все не jar-файлы переносятся без изменений. В jar-

файлах анализируются содержание их пакетов, и производится сжатие. Рассмотрим подробнее эту процедуру:

- Орграф статического покрытия пополняется вершинами и ребрами `Class.forName()` покрытия;
- Начальным множеством вершин волнового обхода считаем множество динамического покрытия;
- Запускается волновой алгоритм обхода;
- Покрытые вершины орграфа объединяются с множеством динамического покрытия, образуя рабочее множество;
- Запускается копирование файлов из источника в выходную папку.
- При копировании jar-файл, рекурсивно обходится, и если пакет принадлежит рабочему множеству – выполняется копирование class-файлов пакета в выходной jar-файл. Если jar-файл содержит вложенный jar-файл, то вложенный файл извлекается, сжимается и пакуется в выходной jar-файл. Все файлы отличные от class и jar файлов копируются без изменений.

Разработано приложение реализующее данную схему и проведены испытания на различных типах коммерческих Java приложений с использованием разных версий виртуальных машин. В таблице приведены некоторые результаты сжатия.

ЗАКЛЮЧЕНИЕ

Уменьшение объема Java-продуктов позволяет существенно сократить нагрузку на каналы связи при их передаче клиентам. Разработанное приложение имеет простую настройку, независимость от исходных Java кодов, позволяет сжимать как внешние jar-библиотеки, так и Java Runtime Environment, поддерживает работу с RCP-приложениями и полностью совместимо с Java 6.

Таблица

Результаты расчетов

Название Java-приложения	Версия JRE в поставке	Исходный размер, Мбайт	Сжатый размер, Мбайт	Процент сжатия, %
SES v1.41	нет	13,8	8,81	36
SES v1.41	1.6.0	79,3	31,6	60
SES v1.41	1.5.0_07	69,6	26,1	62,5
Report Generator v1.0.2	1.6.0	80,4	30,6	62
BNTD v1.0.6 (RCP-приложение)	нет	18,3	17,1	7

Литература

1. Интернет адрес: <http://java.sun.com/docs/books/jvms/VMSpecTOC.doc.html>.
2. Интернет адрес: [http://java.sun.com/docs/Class_\(file_format\)](http://java.sun.com/docs/Class_(file_format)).

АНАЛИЗ УСТОЙЧИВОСТИ БАНКОВСКОЙ СИСТЕМЫ НА ОСНОВЕ ЭКОНОМЕТРИЧЕСКОЙ МОДЕЛИ

А. А. Егоров

ВВЕДЕНИЕ

Стабильность банковской системы приобретает особую значимость в странах с переходной экономикой и имеющих развивающуюся банковскую систему, по ряду причин. Во-первых, это существенное вмешательство государства в банковский сектор. Во-вторых, административные методы воздействия на банковский сектор. Например, ограничения процентных ставок делает невозможным для коммерческих банков осуществление оптимальной политики риск-менеджмента. Проблема прогнозирования устойчивости банковского кризиса весьма важна и для банковской системы Республики Беларусь [1, с.30–35]. Для решения данной задачи используются различные подходы, основанные на эконометрических моделях [2, с.2–10; 3, с.3–6; 4, с.3–8]. Одной из таких моделей является логит-модель бинарного выбора по панельным данным. Целью данных исследований является оценка возможности использования данного подхода для анализа белорусской банковской системы в рамках систем раннего предупреждения.

1. ОПИСАНИЕ ЭКОНОМЕТРИЧЕСКОЙ МОДЕЛИ

Опишем логит-модель бинарного выбора по панельным годовым данным (panel logit model of binary choose), которая предназначена для оценивания вероятности банковского кризиса. Пусть $P(i, t)$ – зависимая переменная, принимающая значение 0 в случае, если в стране i в момент t нет кризиса, и значение 1 в противном случае. Пусть β – вектор неизвестных коэффициентов и $F(\cdot)$ – функция логистического распределения. Вероятность того, что в стране i в момент t присутствует кризис определяется как значение данной функции от произведения вектора β на вектор объясняющих переменных $x(i, t)$. Выбор объясняющих переменных объясняется экономической теорией и особенностями модели банковской системы. Тогда модель имеет следующий вид:

$$\text{Prob}(P(i, t) = 1) = F(\beta' x(i, t)) \quad i = 1, N \quad t = 1, T \quad (1)$$

где N – число стран, а T – количество лет, а: