БИНГО! Ты только что нашел решение своей проблемы! Только давай договоримся – ты прочтёшь текст до конца, окей? :)

Давай начистоту: тут один шлак, лучше закажи работу на **[author-24.pro](https://author-24.pro/)** и не парься – мы всё сделаем за тебя! Даже если остался **один** **день** до сдачи работы – мы справимся, и ты получишь «Отлично» по своему предмету! Только представь: ты занимаешься своим любимым делом, пока твои одногруппники теряют свои нервные клетки…

Проникнись… Это бесценное ощущение :)

Курсовая, диплом, реферат, статья, эссе, чертежи, задачи по матану, контрольная или творческая работа – всё это ты можешь передать нам, наслаждаться своей молодостью, гулять с друзьями и радовать родителей отличными оценками. А если преподу что-то не понравится, то мы **бесплатно** переделаем так, что он пустит слезу от счастья и поставит твою работу в рамочку как образец качества.

Ещё сомневаешься? Мы готовы подарить тебе **сотни часов свободного времени** за смешную цену – что тут думать-то?! Жизнь одна – не трать её на всякую фигню!

Перейди на наш сайт **[author-24.pro](https://author-24.pro/)** - обещаю, тебе понравится! :)

А работа, которую ты искал, находится ниже :)

**Введение**

Для успешной реализации проекта объект проектирования информационных систем должен быть прежде всего адекватно описан, должны быть построены полные и непротиворечивые функциональные и информационные модели информационных систем.

Накопленный к настоящему времени опыт проектирования информационных систем показывает, что это логически сложная, трудоёмкая и длительная по времени работа, требующая высокой квалификации участвующих в ней специалистов. Однако до недавнего времени проектирование информационных систем выполнялось в основном на интуитивном уровне с применение неформализованных методов, основанных на искусстве, практическом опыте, экспертных оценках и дорогостоящих экспериментальных проверках качества функционирования информационных систем. Кроме того, в процессе создания и функционирования информационных систем информационные потребности пользователей изменяться или уточняться, что ещё более усложняет разработку таких систем.

Одним из возможных подходов к разработке программного обеспечения в рамках спиральной модели ЖЦ является получившая в последнее время широкое распространение методология быстрой разработки приложений RAD (Rapid Application Development). Под этим термином обычно понимается процесс разработки программного обеспечения, содержащий 3 элемента:

* небольшую команду программистов (от 2 до 10 человек);
* короткий, но тщательно проработанный производственный график (от 2 до 6 мес.);
* повторяющийся цикл, при котором разработчики, по мере того, как приложение начинает обретать форму, запрашивают и реализуют в продукте требования, полученные через взаимодействие с заказчиком.
* Команда разработчиков должна представлять из себя группу профессионалов, имеющих опыт в анализе, проектировании, генерации кода и тестировании программного обеспечения с использованием CASE-средств. Члены коллектива должны также уметь трансформировать в рабочие прототипы предложения конечных пользователей.

1. **Подход RAD**

**1. Программное обеспечение по методологии RAD**

Жизненный цикл программного обеспечения по методологии RAD состоит из четырех фаз:

* фаза анализа и планирования требований;
* фаза проектирования;
* фаза построения;
* фаза внедрения.

На фазе анализа и планирования требований пользователи системы определяют функции, которые она должна выполнять, выделяют наиболее приоритетные из них, требующие проработки в первую очередь, описывают информационные потребности. Определение требований выполняется в основном силами пользователей под руководством специалистов-разработчиков. Ограничивается масштаб проекта, определяются временные рамки для каждой из последующих фаз. Кроме того, определяется сама возможность реализации данного проекта в установленных рамках финансирования, на данных аппаратных средствах и т.п. Результатом данной фазы должны быть список и приоритетность функций будущей информационной системы, предварительные функциональные и информационные модели информационных систем.

На фазе проектирования часть пользователей принимает участие в техническом проектировании системы под руководством специалистов-разработчиков. CASE-средства используются для быстрого получения работающих прототипов приложений. Пользователи, непосредственно взаимодействуя с ними, уточняют и дополняют требования к системе, которые не были выявлены на предыдущей фазе. Более подробно рассматриваются процессы системы. Анализируется и, при необходимости, корректируется функциональная модель. Каждый процесс рассматривается детально. При необходимости для каждого элементарного процесса создается частичный прототип: экран, диалог, отчет, устраняющий неясности или неоднозначности. Определяются требования разграничения доступа к данным. На этой же фазе происходит определение набора необходимой документации.

После детального определения состава процессов оценивается количество функциональных элементов разрабатываемой системы и принимается решение о разделении информационных систем на подсистемы, поддающиеся реализации одной командой разработчиков за приемлемое для RAD-проектов время - порядка 60 - 90 дней. С использованием CASE-средств проект распределяется между различными командами (делится функциональная модель). Результатом данной фазы должны быть:

* общая информационная модель системы;
* функциональные модели системы в целом и подсистем, реализуемых отдельными командами разработчиков;
* точно определенные с помощью CASE-средства интерфейсы между автономно разрабатываемыми подсистемами;
* построенные прототипы экранов, отчетов, диалогов.

Все модели и прототипы должны быть получены с применением тех CASE-средств, которые будут использоваться в дальнейшем при построении системы.

Данное требование вызвано тем, что в традиционном подходе при передаче информации о проекте с этапа на этап может произойти фактически неконтролируемое искажение данных. Применение единой среды хранения информации о проекте позволяет избежать этой опасности.

**1.2. Методология RAD**

Один из подходов к разработке ПО в рамках спиральной модели ЖЦ - получившая широкое распространение методология быстрой разработки приложений RAD (Rapid Application Development), она включает в себя три составляющие:

* небольшую команду программистов (от 2 до 10 человек);
* короткий, но тщательно проработанный производственный график (от 2 до 6 мес.);
* повторяющийся цикл, при котором разработчики по мере того, как приложение начинает обретать форму, запрашивают и реализуют в продукте требования, полученные через взаимодействие с заказчиком.

Команда разработчиков должна представлять собой группу профессионалов, имеющих опыт в анализе, проектировании, генерации кода и тестировании ПО с использованием CASE-средств, способных хорошо взаимодействовать с конечными пользователями и трансформировать их предложения в рабочие прототипы.

Жизненный цикл ПО по методологии RAD состоит из четырех фаз: анализа и планирования требований; проектирования; построения; внедрения.

На фазе анализа и планирования требований пользователи системы определяют функции, которые она должна выполнять, выделяют наиболее приоритетные из них, требующие проработки в первую очередь, описывают информационные потребности. Формулирование требований к системе осуществляется в основном силами пользователей под руководством специалистов-разработчиков. Ограничивается масштаб проекта, устанавливаются временные рамки для каждой из последующих фаз. Кроме того, определяется сама возможность реализации проекта в заданных размерах финансирования, на имеющихся аппаратных средствах и т.п. Результатом фазы должны быть список расставленных по приоритету функций будущей ИС, предварительные функциональные и информационные модели ИС.

На фазе проектирования часть пользователей принимает участие в техническом проектировании системы под руководством специалистов-разработчиков. CASE-средства используются для быстрого получения работающих прототипов приложений. Пользователи, непосредственно взаимодействуя с ними, уточняют и дополняют требования к системе, которые не были выявлены на предыдущей фазе. Более подробно рассматриваются процессы системы. Анализируется и при необходимости корректируется функциональная модель. Каждый процесс рассматривается детально. Если требуется, для каждого элементарного процесса создается частичный прототип: экран, диалог, отчет, устраняющий неясности или неоднозначности. Устанавливаются требования разграничения доступа к данным. На этой же фазе происходит определение необходимой документации.

После детального определения состава процессов оценивается количество функциональных элементов разрабатываемой системы и принимается решение о разделении ИС на подсистемы, поддающиеся реализации одной командой разработчиков за приемлемое для RAD-проектов время (60 - 90 дней). С использованием CASE-средств проект распределяется между различными командами (делится функциональная модель). Результатом данной фазы должны быть:

* общая информационная модель системы;
* функциональные модели системы в целом и подсистем, реализуемых отдельными командами разработчиков;
* точно определенные с помощью CASE-средств интерфейсы между автономно разрабатываемыми подсистемами;
* построенные прототипы экранов, отчетов, диалогов.

Все модели и прототипы должны быть получены с применением тех CASE-средств, которые будут использоваться в дальнейшем при построении системы. Данное требование вызвано тем, что в традиционном подходе при передаче информации о проекте с этапа на этап может произойти фактически неконтролируемое искажение данных. Применение единой среды хранения информации о проекте позволяет этого избежать.

В отличие от традиционного подхода, при котором использовались специфические средства прототипирования, не предназначенные для построения реальных приложений, а прототипы выбрасывались после того, как выполняли задачу устранения неясностей в проекте, в подходе RAD каждый прототип развивается в часть будущей системы. Таким образом, на следующую фазу передается более полная и полезная информация.

На фазе построения выполняется непосредственно сама быстрая разработка приложения. На данной фазе разработчики производят итеративное построение реальной системы на основе полученных в предыдущей фазе моделей, а также требований нефункционального характера. Программный код частично формируется при помощи автоматических генераторов, получающих информацию из репозитория CASE-средств. Конечные пользователи на этой фазе оценивают получаемые результаты и вносят коррективы, если в процессе разработки система перестает удовлетворять определенным ранее требованиям. Тестирование системы осуществляется в процессе разработки.

После окончания работ каждой отдельной команды разработчиков производится постепенная интеграция данной части системы с остальными, формируется полный программный код, выполняется тестирование совместной работы данной части приложения, а затем тестирование системы в целом. Завершается физическое проектирование системы:

* определяется необходимость распределения данных;
* осуществляется анализ использования данных;
* производится физическое проектирование базы данных;
* определяются требования к аппаратным ресурсам;
* определяются способы увеличения производительности;
* завершается разработка документации проекта.

Результатом фазы является готовая система, удовлетворяющая всем согласованным требованиям.

На фазе внедрения производятся обучение пользователей, организационные изменения и параллельно с внедрением новой системы осуществляется работа с существующей системой (до полного внедрения новой). Так как фаза построения достаточно непродолжительна, планирование и подготовка к внедрению должны начинаться заранее, как правило, на этапе проектирования системы. Приведенная схема разработки ИС не является абсолютной. Возможны различные варианты, зависящие, например, от начальных условий, в которых ведется разработка: разрабатывается совершенно новая система; уже было проведено обследование предприятия и существует модель его деятельности; на предприятии уже существует некоторая ИС, которая может быть использована в качестве начального прототипа или должна быть интегрирована с разрабатываемой.

Следует, однако, отметить, что методология RAD, как и любая другая, не может претендовать на универсальность, она хороша в первую очередь для относительно небольших проектов, разрабатываемых для конкретного заказчика. Если же разрабатывается типовая система, которая не является законченным продуктом, а представляет собой комплекс типовых компонентов, централизованно сопровождаемых, адаптируемых к программно-техническим платформам, СУБД, средствам телекоммуникации, организационно-экономическим особенностям объектов внедрения и интегрируемых с существующими разработками, на первый план выступают такие показатели проекта, как управляемость и качество, которые могут войти в противоречие с простотой и скоростью разработки. Для таких проектов необходимы высокий уровень планирования и жесткая дисциплина проектирования, строгое следование заранее разработанным протоколам и интерфейсам, что снижает скорость разработки.

Методология RAD неприменима для построения сложных расчетных программ, операционных систем или программ управления космическими кораблями, т.е. программ, содержащих большой объем (сотни тысяч строк) уникального кода.

Не подходят для разработки по методологии RAD приложения, в которых отсутствует ярко выраженная интерфейсная часть, наглядно определяющая логику работы системы (например, приложения реального времени), и приложения, от которых зависит безопасность людей (например, управление самолетом или атомной электростанцией), так как итеративный подход предполагает, что первые несколько версий наверняка не будут полностью работоспособны, что в данном случае исключается.

Оценка размера приложений производится на основе так называемых функциональных элементов (экраны, сообщения, отчеты, файлы и т.п.). Подобная метрика не зависит от языка программирования, на котором ведется разработка. Размер приложения, которое может быть выполнено по методологии RAD, для хорошо отлаженной среды разработки ИС с максимальным повторным использованием программных компонентов определяется следующим образом:

* менее 1000 функциональных элементов - один человек;
* 1000 - 4000 функциональных элементов - одна команда разработчиков;
* более 4000 функциональных элементов - 4000 функциональных элементов на одну команду разработчиков.
* Итак, перечислим основные принципы методологии RAD:
* разработка приложений итерациями;
* необязательность полного завершения работ на каждом из этапов жизненного цикла;
* обязательность вовлечения пользователей в процесс разработки ИС;
* необходимость применения CASE-средств, обеспечивающих целостность проекта;
* применение средств управления конфигурацией, облегчающих внесение изменений в проект и сопровождение готовой системы;
* необходимость использования генераторов кода;
* использование прототипирования, позволяющее полнее выяснить и удовлетворить потребности конечного пользователя;
* тестирование и развитие проекта, осуществляемые одновременно с разработкой;
* ведение разработки немногочисленной хорошо управляемой командой профессионалов;
* грамотное руководство разработкой системы, четкое планирование и контроль выполнения работ.

**2. Применение подхода RAD. Его отличие**

**2.1 Отличие подхода RAD от традиционного**

В отличие от традиционного подхода, при котором использовались специфические средства прототипирования, не предназначенные для построения реальных приложений, а прототипы выбрасывались после того, как выполняли задачу устранения неясностей в проекте, в подходе RAD каждый прототип развивается в часть будущей системы. Таким образом, на следующую фазу передается более полная и полезная информация.

На фазе построения выполняется непосредственно сама быстрая разработка приложения. На данной фазе разработчики производят итеративное построение реальной системы на основе полученных в предыдущей фазе моделей, а также требований нефункционального характера. Программный код частично формируется при помощи автоматических генераторов, получающих информацию непосредственно из репозитория CASE-средств. Конечные пользователи на этой фазе оценивают получаемые результаты и вносят коррективы, если в процессе разработки система перестает удовлетворять определенным ранее требованиям. Тестирование системы осуществляется непосредственно в процессе разработки.

После окончания работ каждой отдельной команды разработчиков производится постепенная интеграция данной части системы с остальными, формируется полный программный код, выполняется тестирование совместной работы данной части приложения с остальными, а затем тестирование системы в целом. Завершается физическое проектирование системы:

* определяется необходимость распределения данных;
* производится анализ использования данных;
* производится физическое проектирование базы данных;
* определяются требования к аппаратным ресурсам;
* определяются способы увеличения производительности;
* завершается разработка документации проекта.

Результатом фазы является готовая система, удовлетворяющая всем согласованным требованиям.

На фазе внедрения производится обучение пользователей, организационные изменения и параллельно с внедрением новой системы осуществляется работа с существующей системой (до полного внедрения новой). Так как фаза построения достаточно непродолжительна, планирование и подготовка к внедрению должны начинаться заранее, как правило, на этапе проектирования системы. Приведенная схема разработки информационных систем не является абсолютной. Возможны различные варианты, зависящие, например, от начальных условий, в которых ведется разработка: разрабатывается совершенно новая система; уже было проведено обследование предприятия и существует модель его деятельности; на предприятии уже существует некоторая информационная система, которая может быть использована в качестве начального прототипа или должна быть интегрирована с разрабатываемой.

### 2.2. Опыт применения методологий RAD в конкретных проектах

### (на примере Национального Банка)

При использовании методологии проектирования от данных при разработке системы надзора для Национального Банка были получены схемы взаимодействия отдела надзора, деятельность которого автоматизировалась, с внешними организациями, модели деятельности отдела в виде диаграмм бизнес-процессов, временные диаграммы выполнения процессов, концептуальная модель данных автоматизируемой задачи. Результатами обследования деятельности отдела явились состав и содержание ролевых функций отдела, состав и содержание форм документов, выпускаемой отделом, описание программных средств, используемых в отделе. Анализ результатов обследования выявил направления совершенствования информатизации отдела и позволил сформулировать требования к доработке существующих систем и к интеграции приложений.

Использование современных технологий разработки и сопровождения программного обеспечения (в частности подход RAD) дало новые возможности:

* Гибкость ориентации на "ролевые" функции
* Переносимость ПО на другие платформы без перепрограммирования
* Более высокая производительность в разработке и сопровождении
* Упрощение реализации новых функциональных задач без перепрограммирования

На этапе проектирования системы концептуальная модель данных была преобразована в реляционную модель данных (структуру базы данных), создана архитектура информационной системы: схемы навигации экранов приложений, модели данных приложений, модели интерфейса (данных, спецификации, представления). (см. Приложение рис. 1).

При создании системы была реализована спиральная модель жизненного цикла, было получено четыре прототипа системы, разработка каждого прототипа заняла четыре недели. В конце четвертого цикла прототипирования была получена полностью функциональная система, которая бала передана заказчику для опытной эксплуатации.

При создании первого прототипа были зафиксированы внешние представления системы, собраны замечания и предложения заказчика по реализованному составу ролевых функций, внешнему виду и функциональности системных модулей, замечания и предложения по проекту документации сопровождения. В состав первого прототипа вошли работающая версия системы, уточненная диаграмма автоматизируемых ролевых функций, список реализованных ролевых функций, реляционная модель данных системы и пояснительная записка к ней, архитектура системы в виде схемы навигации модулей, модели данных автоматизируемых функций и интерфейсных элементов, внешний вид интерфейсных элементов (на бумажном носителе), исходные тексты прикладного программного обеспечения (на магнитном носителе).

В результате предъявления первого прототипа заказчику были получены протокол рассмотрения итогов создания прототипа, список замечаний по составу ролевых функций, список замечаний по составу реализованных в базе данных системы сущностей предметной области, соглашение по внешнему виду системы с приложением к протоколу в виде бумажных копий экранов с замечаниями на них.

При реализации второго прототипа были зафиксированы перечень и состав ролевых функций системы и способы их реализации, структура базы данных системы, проверены выполнения требований к первому прототипу и сбор замечаний по функциональности экранов и логикам формирования отчетов.

Результатами второго прототипа явились протокол рассмотрения итогов создания 2-го прототипа, список замечаний по функциональности экранов и логикам формирования отчетов, приложение к протоколу в виде бумажных копий экранов с замечаниями на них.

Результаты третьего прототипа были аналогичны.

Перед вводом системы в опытную эксплуатацию были созданы рабочая база данных системы, рабочие места пользователей, системы автоматизированного версионного управления версиями ПО и автоматизированного сбора замечаний, проведено обучение пользователей.

Таблица 1. Таблица характеристик проекта

|  |  |
| --- | --- |
| **Объекты проекта** | **Общее количество** |
| Файлы экранов | 217 |
| Файлы отчетов | 74 |
| Файлы меню | 33 |
| Файлы моделей архитектуры | 41 |
| Файлы моделей данных | 18 |

**Характеристики проекта** при традиционном подходе: Сложность проекта ~ 1850 ф.т. Типичная норма выработки - 18 ф.т. на человеко-месяц Проект потребует - 103 человеко-месяца 5 человек - более 20 месяцев при спиральной модели жизненного цикла: Сложность проекта ~ 1850 ф.т. Проект потребовал - 38 человеко-месяцев Норма выработки - 48,5 ф.т. на человеко-месяц. (см. Приложение рис. 2)

Высокая производительность проекта достигалась за счет использования средств автоматизации анализа и проектирования (CASE SilverRun), языка четвертого поколения (JAM), за счет автоматизации всей технологической цепочки (мост SR-(JAM), использования средств конфигурационного управления (PVCS), средства автоматизации тестирования QualityWorks.

**2.3. Применение подхода RAD в других областях**

Методология RAD неприменима для построения сложных расчетных программ, операционных систем или программ управления космическими кораблями, т.е. программ, требующих написания большого объема (сотни тысяч строк) уникального кода.

Не подходят для разработки по методологии RAD приложения, в которых отсутствует ярко выраженная интерфейсная часть, наглядно определяющая логику работы системы (например, приложения реального времени) и приложения, от которых зависит безопасность людей (например, управление самолетом или атомной электростанцией), так как итеративный подход предполагает, что первые несколько версий наверняка не будут полностью работоспособны, что в данном случае исключается.

Оценка размера приложений производится на основе так называемых функциональных элементов (экраны, сообщения, отчеты, файлы и т.п.) Подобная метрика не зависит от языка программирования, на котором ведется разработка. Размер приложения, которое может быть выполнено по методологии RAD, для хорошо отлаженной среды разработки информационных систем с максимальным повторным использованием программных компонентов, определяется следующим образом:

Таблица 2. Определение размера приложения по методологии RAD

|  |  |
| --- | --- |
| **< 1000** функциональных элементов | один человек |
| **1000-4000** функциональных элементов | одна команда разработчиков |
| **> 4000** функциональных элементов | 4000 функциональных элементов на одну команду разработчиков |

**Заключение**

Недостатки традиционного (каскадного) подхода к построению жизненного цикла программного обеспечения хорошо известны. Жестко зафиксированные требования к разрабатываемой системе в самом начале жизненного цикла, отсутствие в процессе реализации участия конечного пользователя приводят к тому, что проект растягивается на годы, выходит за рамки установленного бюджета, полученная в результате система не удовлетворяет требованиям пользователей.

Развитие информационных технологий, появление средств автоматизации практически всех этапов жизненного цикла привели к возникновению разнообразных моделей жизненных циклов современных информационных систем. Современные методологии разработки программных продуктов позволяют создавать в сжатые сроки системы высокого качества, удовлетворяющие требованиям пользователей. Многие из современных методологий поддерживаются современными инструментальными средствами.

Наиболее широкое распространение получила методология быстрой разработки приложений RAD (rapid application development), которая охватывает все этапы жизненного цикла современных информационных систем.

Следует, однако, отметить, что методология RAD, как и любая другая, не может претендовать на универсальность, она хороша в первую очередь для относительно небольших проектов, разрабатываемых для конкретного заказчика. Если же разрабатывается типовая система, которая не является законченным продуктом, а представляет собой комплекс типовых компонент, централизованно сопровождаемых, адаптируемых к программно-техническим платформам, СУБД и т.д., на первый план выступают такие показатели проекта, как управляемость и качество, которые могут войти в противоречие с простотой и скоростью разработки. Для таких проектов необходимы высокий уровень планирования и жесткая дисциплина проектирования, строгое следование заранее разработанным протоколам и интерфейсам, что снижает скорость разработки.

В качестве итога перечислим основные принципы методологии RAD:

* разработка приложений итерациями;
* необязательность полного завершения работ на каждом из этапов жизненного цикла;
* обязательное вовлечение пользователей в процесс разработки информационных систем;
* необходимое применение CASE-средств, обеспечивающих целостность проекта;
* применение средств управления конфигурацией, облегчающих внесение изменений в проект и сопровождение готовой системы;
* необходимое использование генераторов кода;
* использование прототипирования, позволяющее полнее выяснить и удовлетворить потребности конечного пользователя;
* тестирование и развитие проекта, осуществляемые одновременно с разработкой;
* ведение разработки немногочисленной хорошо управляемой командой профессионалов;
* грамотное руководство разработкой системы, четкое планирование и контроль выполнения работ.

**Список литературы**

1. Автоматизированные информационные технологии в экономике/Под ред. И.Т. Трубилина. – М.: Финансы и статистика, 2000.
2. Автоматизированные информационные технологии в экономике: Учебник/Под ред. Г.А. Титоренко. – М.: Компьютер: ЮНИТИ, 1998.
3. Благодатских В.А. Экономика, разработка и использование программного обеспечения ЭВМ. – М: Финансы и статистика, 1995.
4. Благодатских В.А., Волнин В.А., Поскакалов К.Ф. Стандартизация разработки программных средств. – М: Финансы и статистика, 2003.
5. Вендров А.М. Пректирование программного обеспечения экономических информационных систем – М: Финансы и статистика, 2002.
6. Вендрова А.М. Практикуме по проектированию программного обеспечения экономических информационных систем – М: Финансы и статистика, 2002.
7. Информатика. Базовый курс // Под ред. С.В. Симоновича, СПб., 2000.
8. Компьютерные технологии обработки информации./Под. ред. С.В. Назарова. – М.: Финансы и статистика, 1995.
9. Котов С.Л. Нормирование жизненного цикла программной продукции. – М.: ЮНИТИ-ДАНА, 2002.
10. Липаев В.В. Надежность программных средств – М: СИНТЕГ, 1998.
11. Орлов С.А. Технологии разработки программного обеспечения: Разработка сложных программных систем: Учебное пособие для студентов вузов, обуч. по напр. Подготовки акалавров и магистров «Информатика и выч.техника». – СПб.: Питер, 2002.
12. Черемных С.В., Семенов И.О., Ручкин В.С. Моделирование и анализ систем: IDEF-технологии: практикум – М: Финансы и статистика, 2002.
13. Черемных С.В., Семенов И.О., Ручкин В.С. Структурный анализ систем: IDEF-технологии – М: Финансы и статистика, 2001.