На правах рукописи

ОТВЕТЧИКОВ ВЛАДИМИР ВЛАДИМИРОВИЧ

«ПОВЫШЕНИЕ КАЧЕСТВА УПРАВЛЕНИЯ ПРЕДПРИЯТИЕМ НА ОСНОВЕ РАЗРАБОТКИ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ПОДДЕРЖКИ ЖИЗНЕННОГО ЦИКЛА РАСПРЕДЕЛЕННЫХ ПРИЛОЖЕНИЙ».

Специальность 05.13.06 - «Автоматизация и управление технологическими процессами и производствами (технические системы)»

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Москва 2007 г.

Работа выполнена в ГОУ ВПО «Московский Государственный Технологический Университет «СТАНКИН»»

Научный руководитель: кандидат технических наук,
профессор Шемелин Владимир Константинович

Официальные оппоненты: доктор технических наук,
профессор Ковшов Евгений Евгеньевич

кандидат технических наук
Москалев Андрей Александрович

Ведущее предприятие: Институт конструкторско-технологической информатики РАН

Защита диссертации состоится «29» марта 2007 г. в _____ часов на заседании диссертационного совета К212.142.01 в Московском Государственном Технологическом Университете «СТАНКИН» по адресу: 127055, Москва, Вадковский переулок, д. 3а.

С диссертацией можно ознакомиться в библиотеке Московского Государственного Технологического Университета «СТАНКИН».

Автореферат разослан «26» февраля 2007 г.

Ученый секретарь
диссертационного Совета,
кандидат технических наук Тарарин Игорь Михайлович

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность работы

На современном уровне развития для предприятий возрастает роль информационной поддержки всех уровней функционирования, включая расширение информационных сервисных функций, особенно учитывая широкое применение услуг глобальной вычислительной сети Internet, составляющей основу информационной платформы для взаимодействия различных предприятий и частных лиц. Сеть Internet стала основой для построения распределенных корпоративных систем пользователей, в число которых входят и разнообразные промышленные предприятия и банки и офисы по продажам и социальные службы, которые используют различные уровни сервиса. Но в распределенных системах существуют ряд проблем для корпоративных пользователей.

В частности, многочисленные пользователи, как в рамках распределенных систем с использованием сети Internet, так и в структуре Intranet, используя универсальные средства доступа, в массовом порядке создают свои многочисленные распределенные приложения (в том числе и Web – приложения), пользуясь при этом универсальными громоздкими средствами и методами, что приводит к излишним затратам ресурсов в тех случаях, когда создаваемое приложение имеет простую архитектуру.

В этом контексте *актуальной задачей* является разработка методик и *компактных* инструментальных средств для быстрой и качественной разработки распределенных приложений вообще и Web – приложений в частности. Основным признаком, качественно влияющим на архитектурную простоту, сохранение стандарта на формы представления данных и доступность таких инструментальных средств является реализация механизма *жизненного цикла распределенного приложения* с точки зрения разработчика.

Цель исследования заключается в повышении эффективности информационного обеспечения предприятий на базе создания инструментальных средств поддержки жизненного цикла приложений, как методической основы деятельности разработчиков распределенных корпоративных систем.

В этом контексте объектом и предметом исследования в данной работе является разработка новых эффективных форм технологии разработки приложений для распределенных систем на основе методологии жизненного цикла приложения с точки зрения разработчика.

Научная новизна работы заключается:

- в разработке модели жизненного цикла распределенных приложений, как средства расширения возможностей разработчиков, с учетом специфики приложений;
- в разработке системы шаблонов проектирования приложений, основанной на рассмотрении распределенного приложения как

"реактивной системы" с иерархической структурой модулей, каждый из которых представляет собой конечный автомат;

- в разработке методики автоматической генерации кода контроллера распределенного приложения на основе высокоуровневого формального описания его функционирования.

Методы исследований.

При решении задач, поставленных в работе, были использованы следующие методы: методы моделирования, аппарат объектно-ориентированного подхода при разработке приложений, методы построения распределенных приложений в распределенных системах на базе сети Internet.

Практическая значимость

Практическая значимость работы заключается в создании методических и программных средств, в виде библиотеки базовых интерфейсов пользователя, реализующих систему шаблонов проектирования приложения на языке программирования Java и пакета инструментальных средств поддержки жизненного цикла распределенных приложений, что позволяет разработчику возможности рационального выбора средств разработки приложений.

Реализация работы:

Разработанные в работе модели, методика проектирования и реализации распределенных приложений (на примере Web – приложения) активно используются в учебном процессе студентов по специальности 220301 «Автоматизация технологических процессов и производств», при чтении дисциплин: «Распределенные системы управления», «Компьютерные технологии в области автоматизации и управления».

Апробация работы.

Основные положения и результаты диссертационной работы публиковались и докладывались на Международном форуме информатизации МФИ-2006// Труды международной научно-технической конференции «Информационные средства и технологии». 17 – 19 октября 2006 г., в 3-х т.т. Т3. – М.: Янус-К, 2006; на международном семинаре «Конкурентноспособность машиностроительной продукции и производств», Москва, ГОУ МГТУ «Станкин», 2005 г.

Публикации. По теме диссертационной работы опубликовано 4 работы.

Структура и объем диссертации. Диссертационная работа состоит из введения, четырех глав, основных результатов и выводов, изложенных на __151___страницах машинописного текста, содержит _32___рисунка и __7__таблиц, список использованной литературы из 24_____наименований и приложение на __14___ страницах. Общий объем работы – 151 __страница.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введении обосновывается актуальность решаемых в диссертации проблем и определяются цель исследования, новизна и решаемые в диссертации задачи.
В первой главе анализируются основные направления развития современных средств информационного сервиса в распределенных корпоративных системах, работающих на основе использования сети Internet; анализируются тенденции развития и повышения качества информационного сервиса для проектирования распределенных приложений.

Процесс проектирования распределенных приложений основан на использовании целого ряда сложных и громоздких инструментариев и системы шаблонов, причем отсутствуют методики рационального выбора тех или иных средств создания приложений в зависимости от объемов и спецификаций создаваемого приложения.

Поэтому *актуальной задачей* является создание моделей, методик и новых инструментальных средств, основанных на *концепции жизненного цикла* распределенных приложений, которые учитывают ранее созданные серии динамических прототипов приложения для уточнения функциональных требований заказчика и учитывают степень участия (роли) разработчика приложений.

Общее представление структуры распределенных приложений показано на рис.1.1.

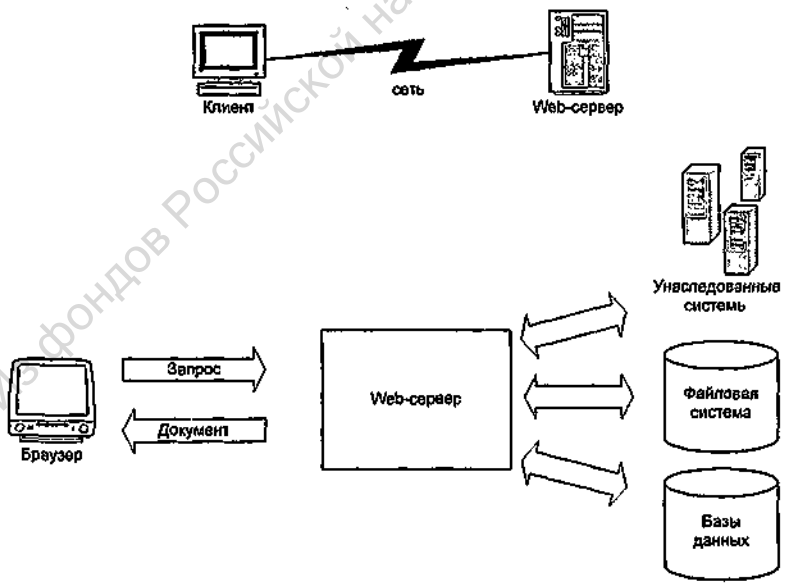

Рис. 1.1. Общее представление распределенного приложения

Фактически это клиент-серверная структура, где доступ к документам осуществлялся с помощью специальных программ - браузеров, работающих на клиентских компьютерах.

С помощью такой программы пользователь может запрашивать Web документы с других компьютеров сети и отображать их на экране своего компьютера. Для просмотра документа необходимо запустить браузер, а затем ввести имя документа и имя узлового компьютера, на котором он находится. Браузер отправляет этому узлу запрос на документ, который обрабатывается программным приложением, получившим название Web-сервера. Web-сервер обычно работает как служба или демон (англ. - daemon), отслеживающий сетевую активность через специальный порт (обычно порт с номером 80).

Браузер отправляет через этот порт запросы на документ (Web-страницу) в специальном формате. Web-сервер получает запрос, находит документ в своей файловой системе и отправляет его обратно браузеру. Но при разработке приложений, когда требуются от сервера не только данные, но и необходимы многочисленные манипуляции и изменения различных структур данных, важной категорией улучшения качества разработки является создание единого подхода, на основе, например, стратегии «жизненного цикла» распределенных приложений.

Специфические особенности жизненного цикла приложения с точки зрения разработчика приведены в таблице 1.1.

Таблица 1.1. Особенности разработки распределенных приложений

Этап жизненного цикла	Особенности
Определение требований	ограниченные начальные требования к функциональности; постоянная модификация функциональных требований в процессе разработки; ошибочные требования или полное отсутствие требований к производительности; ограниченные требования к информационной структуре и навигации; неопределенные требования к дизайну и эргономике пользовательского интерфейса.
Проектирование	отсутствие специализированных методов проектирования; необходимость обеспечения высокой гибкости и возможности быстрого наращивания функциональности; разнообразие, недостаточная стандартизованность и быстрая смена программных и аппаратных

	платформ и стандартов; повышенные требования к производительности, причем нагрузка не всегда предсказуема; необходимость обеспечения совместимости с множеством разнообразных внешних (клиентских) систем, на которые не всегда можно влиять; необходимость интеграции с имеющейся инфраструктурой (СУБД, корпоративные приложения, система защиты и т.д.).
Реализация	участие в реализации специалистов различного профиля (программист, дизайнер, HTML верстальщик, администратор БД); различие программно-аппаратных баз среды разработки и среды эксплуатации; сложность отладки, вызванная распределенной структурой среды разработки;
Тестирование	отсутствие специализированных методов тестирования функциональности; ограниченные возможности тестирования производительности; ограниченные возможности автоматизированного системного тестирования;
Эксплуатация и сопровождение	потребность в оперативном изменение дизайна пользовательского интерфейса; потребность в легкости добавления новой функциональности и изменении информационной структуры и навигации; непредсказуемые всплески нагрузки на серверах; увеличение нагрузки и как следствие потребность в оперативном масштабировании;

В рамках данной диссертационной работы рассматривается актуальный вопрос о качественном росте уровня Web-сервисов, на базе сети Internet, в среде распределенных корпоративных производственных систем за счет разработки моделей, методик и инструментальных средств проектирования приложений с учетом спецификаций создаваемого приложения.

Отсюда ***целью работы*** является повышение эффективности информационного обеспечения предприятий на базе создания инструментальных средств поддержки жизненного цикла приложений, как

методической основы деятельности разработчиков распределенных корпоративных систем.

Для достижения поставленной цели необходимо решить следующие ***научные задачи:***

1. Произвести анализ методик построения современных средств использования информационных ресурсов и выделение наиболее используемого архитектурного шаблона распределенных приложений с идентификацией особенностей этапов разработки современных приложений.
2. На основе анализа существующих моделей разработки программного обеспечения создать модель проектирования распределенных приложений на основе принципов жизненного цикла, с учетом особенностей и специфики приложения. Данная модель должна использовать ранее созданные серии динамических прототипов приложения для уточнения функциональных требований заказчика.
3. На основе модели этапов разработать модель потоков данных и процессов в нотации IDEF0, а также ролевую модель процесса разработки.
4. Разработать библиотеки базовых интерфейсов пользователя, реализующие систему шаблонов разработки распределенных приложений с иерархической структурой модулей, каждый из которых представляет собой конечный автомат.
5. Разработать инструментальные средства поддержки жизненного цикла приложений, как средства повышения качества разработок, учитывающего специфику разрабатываемого приложения.
6. Оценить факторы повышения эффективности деятельности предприятий при применении методик и технологии разработки распределенных приложений на основе дисциплины жизненного цикла с точки зрения разработчика приложений.

Во второй главе рассматриваются современные методы и средства разработки распределенных приложений на основе набора архитектурных шаблонов, дается анализ, основные спецификации и средства использования шаблонов.

В приложении можно использовать распределенные объекты или аплеты Java, и при этом Web-сервер и сервер приложений могут размещаться на одном и том же компьютере. Эти распространенные технологии позволяют улучшить базовую архитектуру распределенного приложения.

На достаточно высоком уровне абстракции можно выделить существующие в настоящее время архитектурные шаблоны приложений. Архитектурный шаблон отражает фундаментальную структурно-организационную схему программных систем. Он предоставляет набор предопределенных подсистем, описывает спектр их обязанностей, а также

представляет правила и рекомендации для организации взаимодействия между ними. Рассмотрим три наиболее известных шаблона:

1. *Thin Web Client* (на основе "тонкого" Web-клиента) используется в большинстве приложений Internet и предоставляет ограниченные возможности по управлению конфигурацией клиента. В распоряжении клиента должен быть только стандартный браузер, поддерживающий формы. Все операции, связанные с бизнес-логикой, выполняются на сервере.
2. *Thick Web Client* (на основе "толстого" Web-клиента) предполагает, что значительная часть бизнес-логики выполняется на клиентской машине. Обычно для выполнения бизнес-логики клиентом используется динамический HTML, аплеты Java или управляющие элементы ActiveX. Взаимодействие с сервером по-прежнему происходит через протокол HTTP.
3. *Web Delivery* (на основе механизма Web-доставки). При взаимодействии клиента и сервера, кроме протокола HTTP, используются и другие протоколы, такие как IOOP и DCOM, которые могут применяться для поддержки системы распределенных объектов. В данном случае браузер функционирует как контейнерный модуль системы распределенных объектов.

Архитектура на основе "тонкого" Web-клиента наиболее подходит для приложений, сервер которых должен передавать ответы в течение интервала времени, приемлемого для пользователя, а также до истечения интервала ожидания клиентского браузера. Обычно это время не превышает нескольких секунд. Применение шаблона Thin Web Client оказывается далеко не лучшим решением, если приложение должно обеспечивать возможность запуска пользователем продолжительного процесса. Для выполнения таких задач могут использоваться технологии, которые выполняют периодический опрос сервера.

Еще одной важной особенностью этого архитектурного шаблона являются ограниченные возможности по созданию сложного интерфейса пользователя. Поскольку браузер предоставляет целостный интерфейс пользователя, все элементы управления должны им поддерживаться. В большинстве стандартных браузеров и в спецификации HTML множество доступных элементов управления ограничивается несколькими полями для ввода текста и кнопками. С другой стороны, упрощенный пользовательский интерфейс может оказаться и преимуществом, поскольку в этом случае команда разработчиков не сможет неоправданно его усложнить.

При использовании шаблона на основе "толстого" Web-клиента очень важно обеспечить переносимость приложения между различными реализациями браузеров. Не все HTML-браузеры поддерживают сценарии JavaScript и VBScript. Кроме того, управляющие элементы ActiveX могут применяться лишь на клиентских компьютерах под управлением операционной системы Windows компании Microsoft. Даже если используются браузеры

известных производителей, всегда присутствуют, пусть незначительные, но все же отличия в реализации модели DOM.

При использовании клиентских сценариев, управляющих элементов и аплетов очень важно, чтобы группой тестирования был выполнен полный набор тестов для всех поддерживаемых клиентских конфигураций. Если на клиенте размещена важная часть бизнес-логики, то необходимо удостовериться в ее корректном и непротиворечивом выполнении на всех типах используемых браузеров. Различные браузеры будут по-разному обрабатывать один и тот же исходный код, и, более того, один и тот же браузер будет по-разному работать в различных операционных системах. Например, Microsoft Internet Explorer 4.01 в системах Windows 95 и Windows NT 4.0 функционирует по-разному.

При использовании шаблона Web Delivery очень важно обеспечить переносимость между различными реализациями браузеров. Кроме того, применение шаблона Web Delivery требует надежной сети, поскольку в данном случае соединения между клиентскими и серверными объектами гораздо более продолжительны по сравнению с HTTP-соединениями. Так что неожиданный разрыв соединения, который не станет пробяемой при использовании двух других архитектур, в приложении на основе шаблона Web Delivery приведет к серьезным нарушениям его функционирования.

В главе также проанализированы различные модели проектирования приложений: каскадная модель, эволюционная модель, модель формальной разработки. При этом каскадная модель принимается как базовая модель жизненного цикла по разработке распределенных приложений (см. рис. 2.1).

Это первая модель процесса создания программного обеспечения, порожденная моделями других инженерных процессов. Эту модель также иногда называют *моделью жизненного цикла программного обеспечения (ПО).*

Основные принципиальные этапы (стадии) этой модели отражают следующие базовые виды деятельности, необходимые для создания ПО.

1. Анализ и формирование требований. Путем консультаций с заказчиком ПО определяются функциональные возможности, ограничения и цели создаваемой программной системы.
2. Проектирование системы и программного обеспечения. Процесс проектирования системы разбивает системные требования на требования, предъявляемые к аппаратным средствам, и требования к программному обеспечению системы. Разрабатывается общая архитектура системы. Проектирование ПО предполагает определение и описание основных программных компонентов и их взаимосвязей.

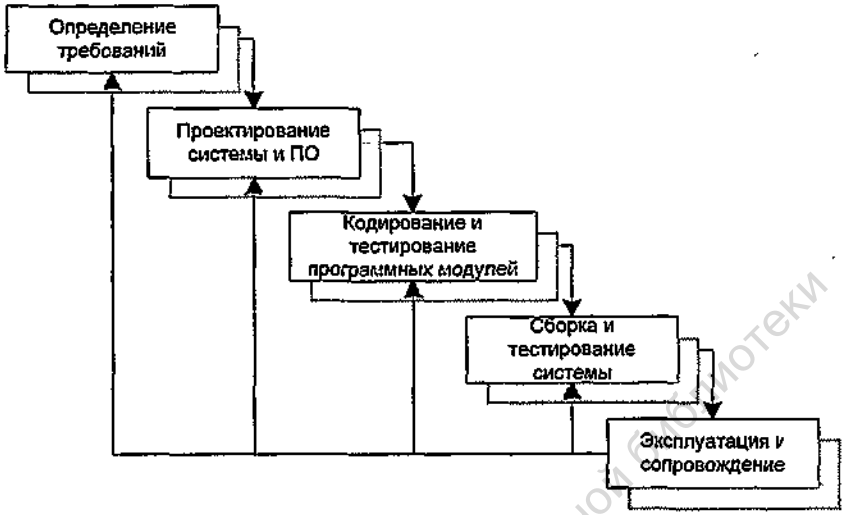

Рис.2.1 Базовая модель жизненного цикла программного обеспечения

3. Кодирование и тестирование программных модулей. На этой стадии архитектура ПО реализуется в виде множества программ или программных модулей. Тестирование каждого модуля включает проверку его соответствия требованиям к данному модулю.
4. Сборка и тестирование системы. Отдельные программы и программные модули интегрируются и тестируются в виде целостной системы. Проверяется, соответствует ли система своей спецификации.
5. Эксплуатация и сопровождение системы. Обычно (хотя и не всегда) это самая длительная фаза жизненного цикла ПО. Система инсталлируется, и начинается период ее эксплуатации. Сопровождение системы включает исправление ошибок, которые не были обнаружены на более ранних этапах жизненного цикла, совершенствование системных компонентов и "подгонку" функциональных возможностей системы к новым требованиям.

При сравнительном анализе моделей применительно к разработке web-приложений установлены следующие положения.

При создании систем, как правило, приходиться использовать различные подходы к разработке разных частей системы, т.е. в целом к разработке системы применяются гибридные (смешанные) модели. Конкретная модель зависит от специфики разрабатываемого ПО и определенных обстоятельств на момент разработки (например квалификация персонала).

На основании представленного анализа разработана гибридная модель процесса разработки программного обеспечения распределенных приложений, представленная на рис. 2.2.

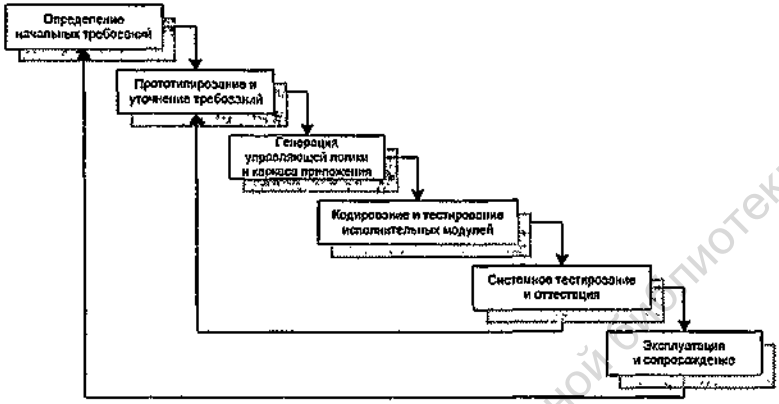

Рис.2.2 Гибридная модель процесса разработки ПО распределенных приложений

Представленная гибридная модель содержит основные преимущества рассмотренных моделей создания программного обеспечения.

В главе также представлены характеристики и спецификации современных инструментальных средств разработки программного обеспечения и в том числе приложений, с помощью - CASE-средств.

Третья глава посвящена рассмотрению типовых шаблонов проектирования, используемые для создания многоуровневых систем с явным разделением обязанностей. Затем на базе рассмотренных шаблонов предлагается структурная схема распределенного приложения.

В главе анализируется наиболее применяемый архитектурный шаблон построенный по принципу "Модель-Представление-Контроллер" (Model-View-Controller- MVC) повсеместно используемый для создания гибких и повторно используемых систем.

Разделение компонента или подсистемы на три логические части (модель, представление и контроллер) производится с целью облегчения модификации или настройки каждой части в отдельности.

Шаблон MVC полезен в тех случаях, когда компонент или подсистема имеет некоторые из следующих характеристик:

- Компонент или подсистему можно представлять несколькими различными способами. При этом внутреннее представление компонента или подсистемы в системе может полностью отличаться от представления на экране.

- Необходимо реализовывать несколько различных типов поведения. Иными словами, один и тот же компонент может инициализировать какие-то операции по запросам, поступающим из разных источников, и в зависимости от того, каков именно этот источник, поведение компонента может быть разным.
- Поведение или представление компонента изменяется в процессе его использования.
- Следует обеспечить возможность адаптации или повторного использования компонента в разных системах с минимальными изменениями в его программном коде.

Компонентная диаграмма шаблона MVC представлена на рис. 3.1.
Для описания данного шаблона используется компонентная диаграмма. Каждая из трех частей шаблона MVC представляет собой компонент, содержащий, в свою очередь, много классов и интерфейсов.

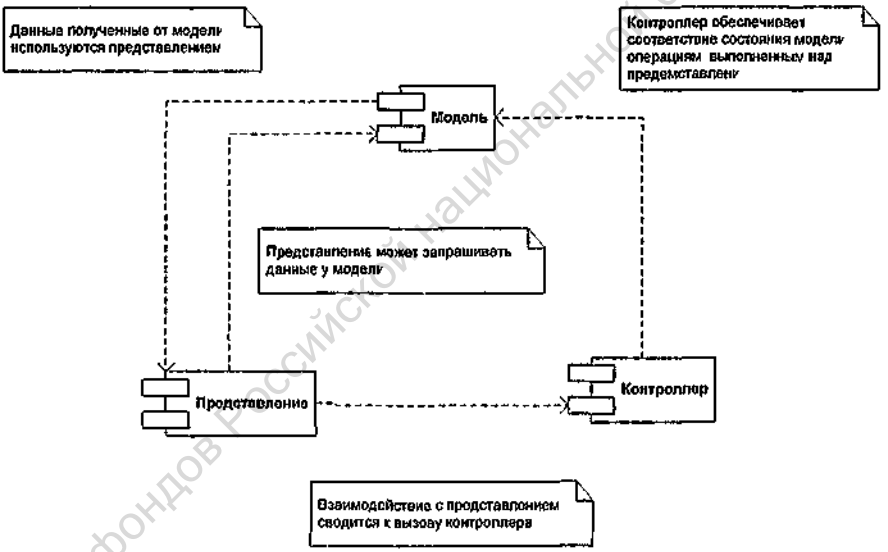

Рис.3.1, Компонентная диаграмма шаблона MVC

Шаблон MVC предоставляет отличный способ создания гибких и адаптируемых к различным новым ситуациям элементов приложения. При этом гибкость может использоваться как статически, так и динамически. Под динамической гибкостью понимается возможность замены объекта представления или контроллера во время работы приложения.

Обычно самая большая сложность при реализации шаблона MVC заключается в том, как правильно выбрать базовую презентацию элемента. Иными словами, нужно правильно разработать интерфейсы между моделью, представлением и контроллером. Элемент, выполненный в соответствии с шаблоном MVC, часто, как и большинство других программных объектов, должен удовлетворять какому-то определенному набору требований. Поэтому для реализации элемента таким образом, чтобы он не нес на себе отпечатка специфических особенностей приложения, необходимо иметь определенное видение и выполнить тщательный анализ.

Подсистема "контроллер" в различных классах программных систем может быть реализована по разному, основываясь на различных принципах взаимодействия подсистем "модель" и "представление".

Рассматривая web-приложения на основе шаблона "тонкого клиента", можно отнести их к классу реактивных систем, т.е. систем изменяющих свое состояние и выполняющих определенные действия под влиянием внешних воздействий.

Данное решение очевидно, исходя из того, что web-приложения на основе "тонкого клиента" базируются на протоколе HTTP не поддерживающим соединения и состояния.

Разработка подсистемы "контроллер" в реактивных системах наиболее удобна на основе модели конечного автомата, включающей следующие понятия:

- **Состояние**. Ситуация в жизни объекта, на протяжении которой он удовлетворяет некоторому условию, выполняет определенную деятельность или ожидает какого-то события.
- **Событие**. Спецификация существенного факта, имеющего место в пространстве и во времени и инициирующего переход из одного состояния в другое;
- **Переход**. Отношение между двумя состояниями, показывающее, что объект, находящийся в первом состоянии, должен выполнить определенные действия и перейти во второе состояние, как только произойдет указанное событие и будут удовлетворены определенные условия перехода.
- **Условия перехода**. При возникновении события, вычисляется некоторое булевское выражение условия, если оно верно, то данный переход выполняется;
- **Действие**. Атомарное вычисление, которое приводит к изменению состояния модели или возврату значения;

Данная модель автомата основана на модели машины Мили, которая представляет собой шестерку объектов:

$A = \langle S, X, Y, s0, \delta, \lambda \rangle$,

где

S – конечное непустое множество состояний;

X – конечное непустое множество входных сигналов;

Y – конечное непустое множество выходных сигналов;
$s0 \in S$ – начальное состояние;
δ: $S \times X \rightarrow S$ – функция переходов;
λ: $S \times X \rightarrow Y$ – функция выходов;
Функции переходов и выходов автомата Миля имеют вид:

$$s(t+1) = \delta(s(t), x(t));$$

$$y(t) = \lambda(s(t), x(t)).$$

Подобные модели применяются в различных нотациях используемых при разработке программного обеспечения, например в объектно-ориентированном языке UML.

Таким образом, контроллер распределенного приложения можно представить в виде совокупности конечных автоматов указанного выше типа. При этом каждому элементу подсистемы "представление" будет соответствовать один автомат подсистемы "контроллер".

В главе представлена диаграмма процессов разработки и сопровождения web-приложения в нотации структурных моделей IDEF0 и подробно представлен этап разработки и уточнения требований. Собранный прототип поступает на оценку заказчику. В процессе оценки часть требований утверждается и на их основе подготавливаются подробные спецификации предметной области, а также планы системного тестирования функциональных возможностей распределенного приложения. Другая часть требований изменяется и передается вновь на этапы подготовки прототипа. После уточнения всех требований и отсутствии модификаций считается, что определение требований закончено.

При этом у разработчиков имеется функциональный прототип приложения. Добавление к нему исполнительных модулей образует законченное приложение. На основе модели потоков данных и процессов, *разработана ролевая модель разработки web-приложения*

В зависимости от размера приложения в разработке могут принимать участие различное количество сотрудников, выполняющих различные роли, а также совмещающие несколько ролей сразу.

Четвертая глава посвящена технической реализации инструментальных средств поддержки жизненного цикла распределенных приложений при решении задач повышения качества управления в распределенных корпоративных системах. Представлена разработка библиотеки базовых интерфейсных модулей пользователя (модулей API) в структуре приложений, представлены разработанные инструментальные средства поддержки жизненного цикла распределенных приложений.

Разработана библиотека программных модулей, на базе которой возможно построение приложений, следуя описанной модели разработки.
Элементы библиотеки размещены в следующих пакетах:

- **com.kapych.webapp** – исходный пакет библиотеки, содержащий вложенные пакеты и прототип головного модуля приложений (См. 4.1)
- **com.kapych.webapp.module** – пакет библиотеки, содержащий прототипы основных модулей web-приложения: фрагментов, условий, генераторов представления, команд, объектов доступа к данным (См. 4.2) Код модулей на языке Java представлен *в приложении 1.*
- **com.kapych.webapp.pool** – пакет библиотеки, содержащий механизм повторного использования объектов
- **com.kapych,webapp.util** – пакет библиотеки, содержащий вспомогательные модули web-приложения: механизм журнализации, интерфейс к файлам конфигурации и др.

Разработка приложения происходит посредством реализации и сборки основных модулей, унаследованных от прототипов из пакета module.

Код модулей на языке Java представлен *в приложении диссертации.*

Разработка распределенного приложения происходит посредством реализации и сборки основных модулей, унаследованных от прототипов из пакета module. На рис. 4.1 представлена диаграмма классов исходного пакета библиотеки API. В главе разработаны также: диаграмма классов пакета прототипов основных модулей; диаграмма классов механизма повторного использования объектов; диаграмма классов пакета вспомогательных модулей, а на Рис. 4.2 Диаграмма взаимодействия основных модулей приложения при обработке запроса.

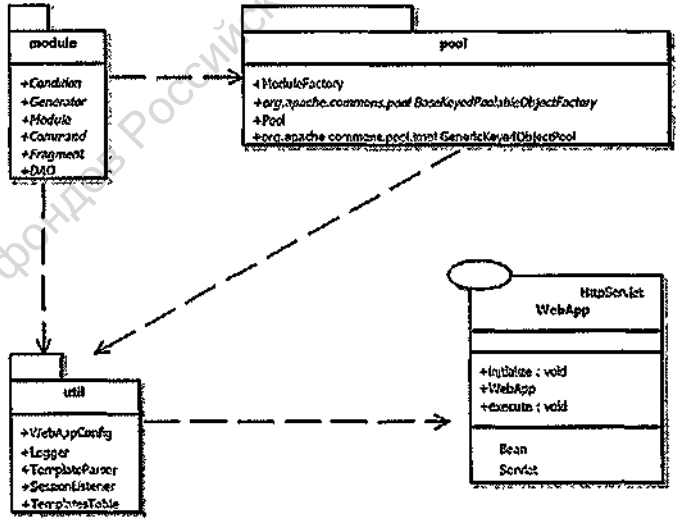

Рис.4.1. Диаграмма классов исходного пакета библиотеки базовых API

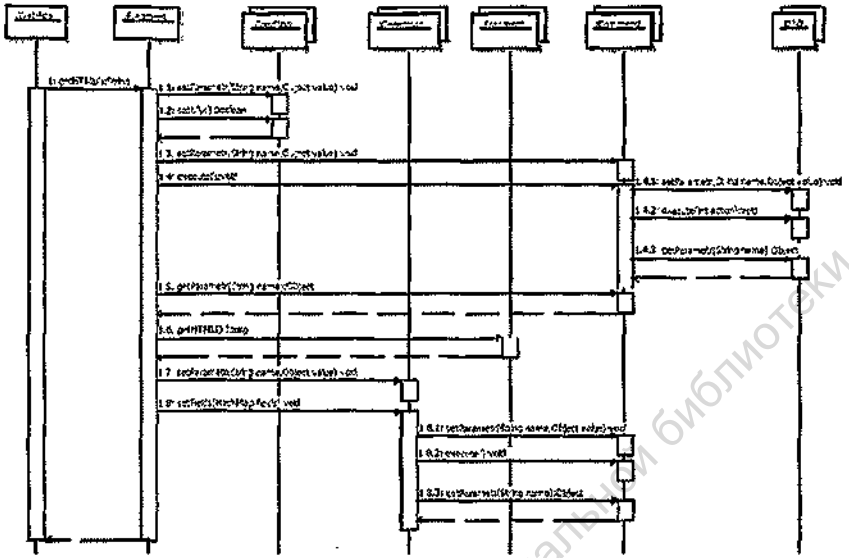

Рис.4.2 Диаграмма взаимодействия основных модулей приложения при обработке запроса

Редактор диаграмм обеспечивает возможность составления формальных спецификаций управляющей логики путем построения графов переходов автоматов контроллера web-приложений. На рис. 4.3 представлено окно редактора диаграмм.

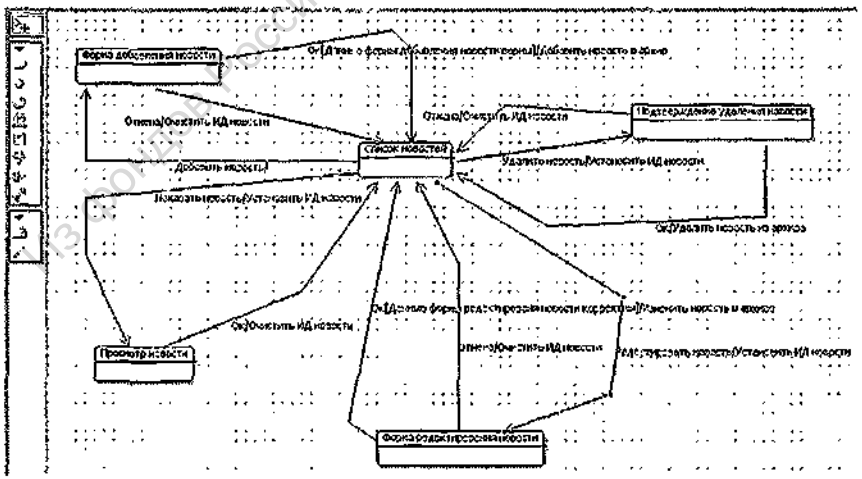

Рис.4.3. Окно редактора диаграмм

Изображение окна менеджера проекта представлено на рис. 4.4. В качестве менеджера проекта использован стандартный менеджер среды Eclipse.

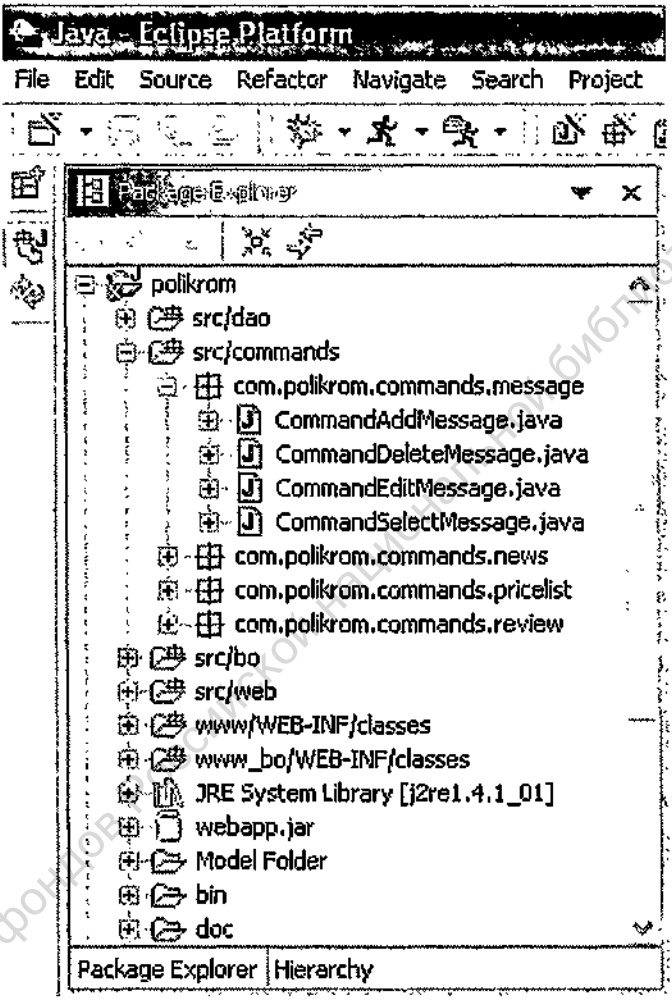

Рис.4.4. Окно менеджера проекта

В диссертации также представлен разработанный интерфейс текстового редактора и транслятор формальных спецификаций и вся совокупность представленные средств позволяет эффективно разрабатывать различные распределенные приложения, чаще всего Web-приложения.

Точный расчет экономического эффекта от внедрения программного обеспечения для разработки распределенных приложений является проблемой, поскольку не существует стандарта на стоимость единицы программного продукта, в виде исходного кода.

Поэтому обычно используют приблизительные методы расчета, больше учитывающие сложившуюся практику отношений разработчика и клиента.

В частности, результат статистического анализа разработанной по представленной в диссертации методике системы доступа к данным, как Web-приложения, на одном из предприятий показал, что время обработки информации, за счет исключения дублирования информации о пользователях, сократилось примерно на 30%.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ ДИССЕРТАЦИОННОЙ РАБОТЫ

1. На основе анализа методик построения современных средств использования информационных ресурсов определены наиболее используемые архитектурные шаблоны проектирования распределенных приложений с идентификацией особенностей этапов разработки современных приложений.

2.Рассмотрены основные модели этапов процесса разработки программного обеспечения и на их основе предложена смешанная модель этапов процесса разработки распределенных приложений. Данная модель предполагает использование ранее созданных серий динамических прототипов приложения для уточнения функциональных требований заказчика.

3.На основе модели этапов предложена модель потоков данных и процессов в нотации IDEF0, а также ролевая модель процесса разработки.

4. Разработана библиотека базовых интерфейсов пользователя, реализующей систему шаблонов разработки распределенных приложений с иерархической структурой модулей, каждый из которых представляет собой конечный автомат.

5. Разработаны инструментальные средства поддержки жизненного цикла распределенных приложений, как средства повышения качества разработок, учитывающего специфику разрабатываемого приложения.

6.Определены факторы повышения эффективности деятельности предприятий при применении методик и технологии разработки приложений на основе дисциплины жизненного цикла с точки зрения разработчика приложений.

ОСНОВНОЕ СОДЕРЖАНИЕ ДИССЕРТАЦИИ ОПУБЛИКОВАНО В СЛЕДУЮЩИХ РАБОТАХ:

1. Ответчиков В.В., Шемелин В.К. Методы разработки web-приложений как средство повышения качества управления бизнес-процессами. // Известия Вузов. Северо-Кавказский регион. Техн. Науки. Спец. Вып. Математическое моделирование и компьютерные технологии. 2006, с.87-90.

2. Ответчиков В.В. Разработка инструментальных средств поддержки жизненного цикла web-приложений./Труды международной научно-технической конференции «Информационные средства и технологии». 17 – 19 октября 2006 г., в 3-х т.т. Т3. – М.: Янус-К, 2006.

3. Ответчиков В.В., Шемелин В.К. Идентификация и моделирование процессов разработки web-приложений.// – М.: Объединенный научный журнал, 2006, № 10, стр. 55-56.

4. Ответчиков В.В., Шемелин В.К. Разработка концепции средств поддержки жизненного цикла web-приложений. // – М.: Объединенный научный журнал, 2006, №27, стр. 56-57.

Подписано в печать 23.01.2007

Формат 60x90 1/16	Бумага 80 гр/м^2	Гарнитура Times
Объем 1,5 п.л.	Тираж 50 экз.	Заказ № 4

Отпечатано в Издательском Центре ГОУ ВПО МГТУ «СТАНКИН»
Лицензия на издательскую деятельность ЛР №01741 от 11.05.2000
127055, Москва, Вадковский пер., д.3а