Московский государственный технический университет
имени Н.Э. Баумана

Т.И. Вишневская, Т.Н. Романова

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

Часть 2

Методические указания
к лабораторному практикуму

Москва
Издательство МГТУ им. Н.Э. Баумана
2009

УДК 681.3.06 (076)
ББК 32.973-018.2
В 55

Рецензент

Вишневская Т.И., Романова Т.Н.

Технология программирования: Метод. указания к лабораторному практикуму. -Ч. 2. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2009. – с.

Сформулированы задания для лабораторных работ по курсу «Технология программирования» с учетом особенностей разработки Web-приложений, даны пояснения и примеры.

Для студентов, обучающихся по специальности «Информатика и вычислительная техника».

Прил. Библиогр. назв.

УДК 681.3.06 (076)
ББК 32.973-018.2

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

В настоящее время все возрастает доля программного обеспечения, предназначенного для работы в сети Internet (Web-приложений). При этом для разработки Web-приложений используются различные информационные технологии.

В первой части методических указаний «Технология программирования» [1] были предложены основные приемы моделирования сложных информационных систем с использованием языка визуального моделирования UML и рассмотрены вопросы объектно-ориентированного подхода к анализу и проектированию программного обеспечения.

Вторая часть методических указаний предназначена для изучения особенностей полного инженерного цикла разработки Web-приложений на платформе Microsoft .NET.

Практикум предназначен для студентов специальности «Информатика и вычислительная техника», изучающих информационные технологии в рамках дисциплины «Технология программирования».

Авторы выражают благодарность студентам факультета «Информатика и системы управления» МГТУ им. Н.Э. Баумана Бушминкину К.В., Ребрикову А.В., Старостину А.Н. за помощь в разработке программных проектов по предложенной методике.

БАЗОВЫЕ СВЕДЕНИЯ О MICROSOFT .NET

Введение в платформу .NET

.NET – эта платформа, разработанная компанией Microsoft [2]. Платформа, которая, с одной стороны, предоставляет средства для разработки новых приложений, а с другой – содержит большое количество средств, для обеспечения работы приложений. .NET ориентирована на работу как с обычными Desktop приложениями, так и с Web-приложениями, работающими через Internet. Так одним из способов взаимодействия приложений на платформе .NET является язык XML, который позволяет передавать данные в виде текстовых файлов, что подходит для транспортного протокола HTTP.

Платформа Microsoft .NET содержит следующие составляющие:

- .NET Framework (MSF) – это фундамент платформы, который представляет собой среду для создания, развертывания и запуска приложения;
- .NET Enterprise серверы – это все современные серверы Microsoft (например, SQL,Server, BizTalk Server);
- Visual Studio .NET – это среда для разработки приложений;
- Клиенты – системы, позволяющие обеспечить всю необходимую инфраструктуру для работы приложения .NET;
- XML Web-службы – блоки, расположенные по всей сети Internet, с которыми могут взаимодействовать приложения, используя язык XML.

Что такое .NET Framework

.NET Framework (MSF) – представляет собой среду для создания, развертывания и запуска приложения [3]. Фактически MSF можно разбить на две части: библиотеку классов и Common Language Runtime (общеязыковую среду исполнения, CLR).

Библиотека классов включает в себя следующие элементы:

- классы, предназначенные для разработки Windows приложений;
- ADO.NET – это технология доступа к данных, реализуемая в виде набора классов.
- ASP.NET – технология, позволяющая разрабатывать Web-приложения. Эта технология включает набор классов, описывающих элементы управления, а также классы по управлению состоянием, защитой и работой Web-приложения целиком;

- средства для разработки XML Web-служб – тут содержаться классы, позволяющие обрабатывать файлы в формате XML, а также передавать эти файлы по сети.

Common Language Runtime – это набор утилит и служб, которые включают в себя все необходимые средства для создания и запуска приложений.

Среда Common Language Runtime позволяет загрузить классы приложения в память, создать объекты этих классов, настроить работу приложения в определенном контексте, а также управлять сборкой мусора. Кроме этого, среда имеет средства для компилирования приложения на специальный промежуточный язык Microsoft - MSIL (Microsoft Intermediate Language). Откомпилированную программу можно переносить на любую платформу, поддерживающую .NET Framework. **Приложения, написанные под .NET Framework не зависят от платформы и аппаратной части.** Для того чтобы приложение работало, необходимо иметь CLR на той машине, где мы используем приложение. Кроме того, CLR позволяет создавать приложения из компонентов, написанных на разных языках программирования.

Модель процессов .NET Framework

Модель процессов определяет порядок проектирования и описывает жизненный цикл проекта. Модель процессов MSF состоит из пяти четко определенных этапов:

- создания общей картины приложения;
- проектирования;
- разработки;
- стабилизации;
- развертывания.

Каждый этап завершается контрольной точкой:

- утверждение документа общей картины и области действия проекта;
- утверждение дизайна проекта;
- окончательное утверждение области действия проекта;
- подтверждение готовности проекта к выпуску;
- приложение развернуто.

Технология распределенных объектов

Многие достижения в области средств создания Web-приложений и других распределенных систем стали возможны благодаря объединению объектно-ориентированных методов и технологии распределенных систем [4]. Это объединение часто называют технологией распределенных объектов (distributed object technology – DOT). В этой техноло-

гии объект – это модуль вычисления и распределения в гетерогенной многомашинной вычислительной среде. Этот подход использует два ключевых свойства объектов: их способность инкапсулировать как данные, так и операции в одном модуле вычислений, а также отделять интерфейс от реализации. С помощью DOT- технологии приложения можно распределить по гетерогенной среде и обеспечить возможность выполнять каждый компонент на наиболее подходящей платформе. Технология распределенных объектов обеспечивается и поддерживается промежуточным программным обеспечением (ПО).

Промежуточное ПО — это определенное сервисное звено между приложением и базовой платформой (операционной системой и сетевым ПО). Оно предоставляет независимые от приложения услуги, которые позволяют различным процессам выполняться на одной или нескольких платформах. Существуют различные типы промежуточного ПО.

- Технология построения распределенных объектных приложений (Common Object Request Broker Architecture - CORBA). CORBA определяет стандартную архитектуру для брокеров объектных запросов (object request brokers — ORBs), форму промежуточного ПО, которое управляет связями между объектами в распределенной среде. Стандарт CORBA был разработан и поддерживается группой управления объектами (OMG). Интерфейсы между объектами определяются с использованием OMG-языка описания интерфейсов (Interface Definition Language— IDL). Объекты могут реализовываться в различных языках, выполняться на различных аппаратных средствах и под управлением различных операционных систем.
- Модель компонентных объектов (Component Object Model— COM) и распределенная модель компонентных объектов (Distributed Component Object Model— DCOM). Модели COM и DCOM ссылаются на спецификацию и реализацию технологии ORB, которая была разработана корпорацией Microsoft. COM позволяет взаимодействовать компонентам (объектам) на одном компьютере, в то время как DCOM поддерживает взаимодействие компонентов по сети. Чтобы установить связь, компоненты должны соответствовать бинарной структуре, определенной корпорацией Microsoft. При этом компоненты могут быть реализованы в различных языках, главное, чтобы они придерживались этой бинарной структуры.
- Java 2, Enterprise Edition (J2EE). J2EE поддерживает многоуровневую архитектуру приложений, основанную на использовании трех типов Java-компонентов: аплетов, сервлетов (включая JSP) и Enterprise JavaBeans. Спецификация J2EE определяет программные интерфейсы J2EE-приложение может быть развернуто в любой среде, которая реализует эту платформу.

- COM+, COM+ корпорации Microsoft — это расширение модели COM, которое представляет собой как архитектуру объектно-ориентированного программирования, так и набор служб операционной системы. COM+ объединяет COM и сервер транзакций корпорации Microsoft (Microsoft Transaction Server — MTS), а также добавляет ряд таких новых возможностей, как организация очередей сообщений и обработка событий с подписью.

2. ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ WEB-ПРИЛОЖЕНИЙ

Web-приложения и распределенные системы

В Web-приложениях обычно используется многоуровневая архитектура, которая включает "внешнее" клиентское звено, управляемое Web-браузером, "внутреннее" звено с источниками данных (или уровень информационных источников), а также один или несколько средних уровней, которые обеспечивают функции Web-сервера и приложений. Именно по этой причине, Web-приложения являются распределенной системой.

Существует четыре фундаментальных различия между локальным и распределенным вычислением, которые необходимо учитывать при разработке Web-приложения [4]:

- латентность;
- доступ к памяти;
- частичный отказ;
- параллелизм.

Латентность — это разница во времени реакции между вызовом локальной и удаленной операции. Источники латентности включают скорость передачи данных по сети, расходы на промежуточное ПО и расходы на связь вследствие принадлежности объектов различным адресным пространствам. Эта разница может составлять четыре или пять порядков. Такую ситуацию невозможно значительно улучшить, поскольку ускорение будет компенсировано повышенным трафиком и дополнительными требованиями к ПО.

Доступ к объектам предоставляется по-разному, в зависимости от того, локальные они или удаленные. К объектам, относящимся к одному и тому же адресному пространству, можно эффективно обращаться посредством указателей. Если же объекты располагаются в различных адресных пространствах, то к ним можно получить доступ с помощью менее эффективных объектных ссылок. Различие в способе доступа к локальным и удаленным объектам требует, чтобы:

- либо программист имел информацию о местоположении объекта (локальный он или удаленный) и принимал соответствующие меры (это усложняет задачу программирования);
- либо среда выполнения ПО обеспечивала универсальный механизм для доступа к объектам независимо от их местоположении (в этом случае программистам придется научиться использовать этот новый механизм). Во избежание ошибок этот механизм должен использоваться для всех видов доступа к объектам.

В распределенных системах некоторые компоненты могут давать сбои. Возможен **отказ по сети**, в результате которого отдельные процессоры будут изолированы один от другого, или же может отказать конкретный процессор, в то время как другие будут продолжать нормально работать и обмениваться информацией. Существует два варианта "борьбы" с частичными отказами:

- со всеми объектами следует обращаться так, как если бы они были локальными. Это решение при частичном отказе ведет к недетерминированному поведению.
- со всеми объектами следует обращаться так, как если бы они были удаленными. Это решает проблему недетерминированного поведения, но вносит дополнительную латентность при доступе к локальным объектам. Это также неоправданно усложняет работу с объектами, которые никогда не будут удаленными.

В распределенной среде методы любого объекта могут вызываться **параллельно.** Чтобы предотвратить несогласованность данных или их искажение, эти объекты должны определять и поддерживать критические разделы для управления параллельным доступом к их данным. Для того чтобы со всеми объектами можно было обращаться единообразно (некоторым унифицированным способом), необходимо использовать один из трех возможных подходов:

- игнорировать проблему. К сожалению, при этом подходе также теряется информация о характере отказа в системе, если несколько объектов одновременно попытаются обновить одни и те же данные;

- сделать все объекты однопоточными. Но если проявятся проблемы с производительностью, могут потребоваться серьезные изменения, связанные с преобразованием объектов, благодаря которому они могли бы корректно выполняться в многопоточной среде;
- включить семантику параллелизма во все объекты, безотносительно к их местоположению. Это лишь усложнит объекты, к которым никогда не будет организован параллельный доступ, и увеличит расходы на него.

Квинтэссенция архитектуры программной системы распределенной системы — это размещение объектов, а также организация связей между ними плюс синхронизация их работы. Один из аспектов распределенных систем, который может иметь существенное влияние на производительность, представляет собой расходы на коммуникацию и синхронизацию между распределенными объектами. При разработке Web-приложения необходимо учитывать ряд следующих факторов, влияющих на его производительность:

- прогнозирование объема деятельности;
- выбор механизма для выполняемых файлов (например, CGI-программы или сервлеты);
- распределение выполняемых файлов по узлам обработки;
- механизм доступа к базе данных;
- механизм обработки данных длительного хранения;
- механизм предоставления доступа к данным и/или их защита;
- оснащение внутрипрограммными средствами контроля для определения коэффициента использования данного Web-приложения;
- размещение базы данных (например, в узле сервера или в отдельном узле);
- взаимодействия с промежуточным программным обеспечением;
- интерфейсы с существующими системами;
- влияние загрузки аплетов или другого программного кода.

Приложения, основанные на технологиях WWW, несут в себе новые проблемы, как для разработчиков, так и для тестеров. К этим проблемам можно отнести:

- короткие циклы выпусков;
- постоянно изменяющиеся технологии;
- большое количество пользователей при начальном запуске Web-узла;
- невозможность контроля пользовательской среды запуска;
- доступность Web-узла в течение 24-х часов.

О производительности Web-приложений судят по двум "меркам": реактивности и расширяемости [4]. **Реактивность -** это способность системы соответствовать требуемым

значениям времени отклика системы. О важности реактивности достаточно сказать, что в наше время пользователи не станут терпеть большое время реакции системы, а просто уйдут на другой сайт. **Расширяемость** - это способность системы соответствовать значениям времени реакции при возрастающих требованиях к функциям программного обеспечения. Расширяемость не менее важна для поддержки реактивности при увеличении числа посетителей сайта.

Работа 1. Разработка технического задания

Техническое задание (ТЗ) на разработку Web- приложения – это документ, оформленный в установленном порядке [5] и определяющий цели создания приложения, требования к функционированию приложения и основные исходные данные, необходимые для его разработки, а также план-график создания приложения.

Проектирование Web- приложения необходимо начинать с системного анализа предметной области и выявления потребности в создании приложения с определенными функциями. В ТЗ необходимо четко сформулировать цель создания приложения и определить основные требования к нему.

Программные требования (*Software Requirements)* – это свойства, которые должны быть надлежащим образом представлены для решения конкретных практических задач. Данная область знаний касается вопросов извлечения (сбора), анализа, специфицирования и утверждения требований. Требования к ПО подразделяются на функциональные и нефункциональные (рис.1).

Функциональные требования (Functional Requirements) определяют функциональность (поведение) программной системы, которая должна быть создана разработчиками для предоставления возможности выполнения пользователями своих обязанностей в рамках бизнес-требований и в контексте пользовательских требований. Группа функциональных требований задает, что система должна делать. Часто функциональные требования представляют в виде сценариев (вариантов использования, Use-Case).

Нефункциональные требования задают, с соблюдением каких условий, должна функционировать система (например, скорость отклика при выполнении заданной операции).

Бизнес-требования (Business Requirements) – определяют высокоуровневые цели организации или клиента (потребителя) – заказчика разрабатываемого программного обеспечения.

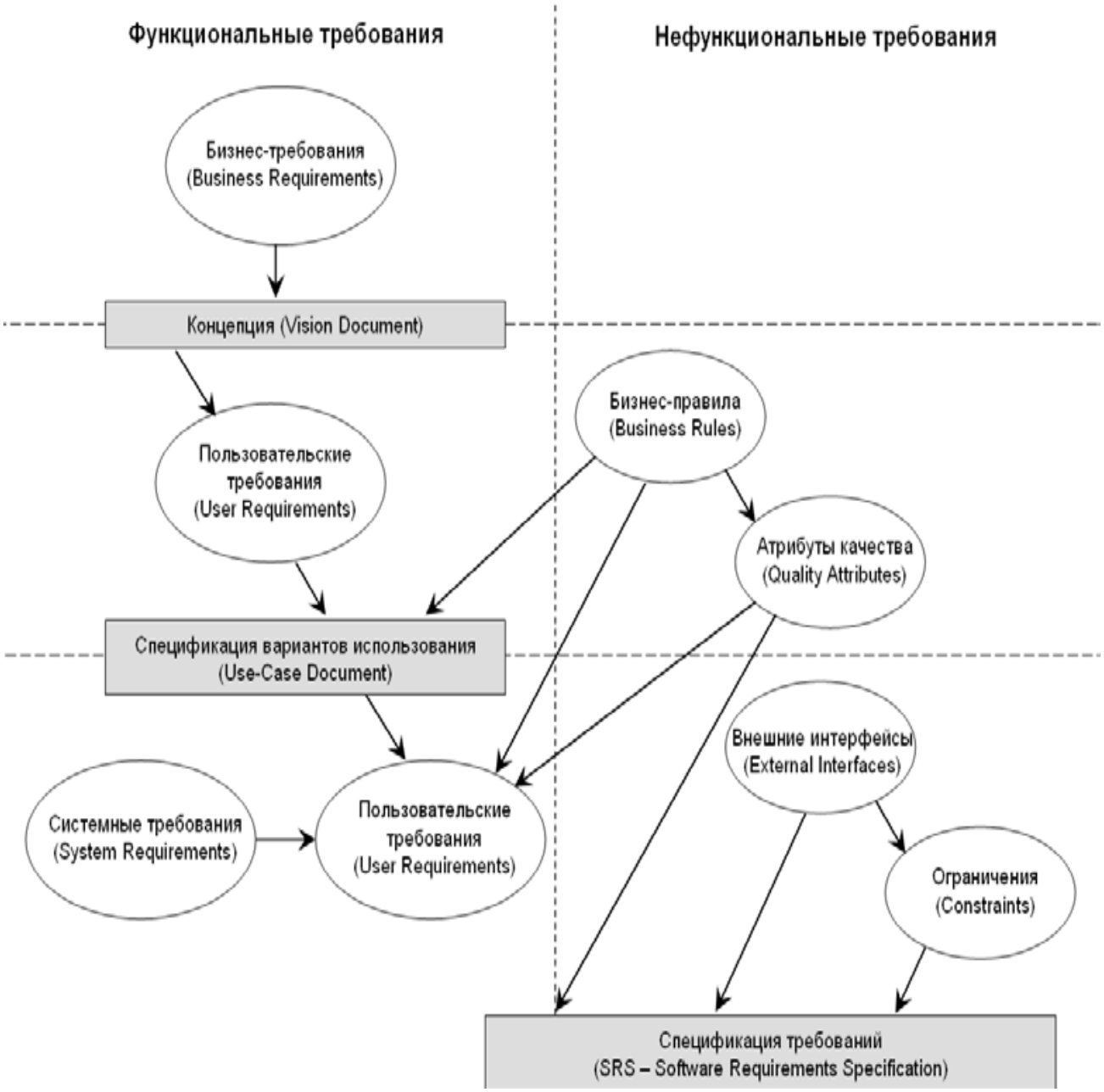

Рис. 1. Пример высокоуровневого структурирования групп требований к ПО.

Пользовательские требования (User Requirements) – описывают цели/задачи пользователей системы, которые должны достигаться/выполняться пользователями при помощи создаваемой программной системы.

Бизнес-правила *(Business Rules)* включают или связаны с корпоративными регламентами, политиками, стандартами, законодательными актами, внутрикорпоративными инициативами. Бизнес-правила часто определяют распределение ответственности в системе, отвечая на вопрос "кто будет осуществлять конкретный вариант, сценарий использования" или диктуют появление некоторых функциональных требований.

Внешние интерфейсы часто подменяются "пользовательским интерфейсом". Конкретизация аспектов взаимодействия с другими системами, операционной средой (например, запись в журнал событий операционной системы), возможностями мониторинга при эксплуатации – все это не функциональные требования (к которым ошибочно приписывают иногда такие характеристики), а вопросы интерфейсов, так как функциональные требования связаны непосредственно с функциональностью системы, направленной на решение бизнес - потребностей.

Атрибуты качества (Quality Attributes) – описывают дополнительные характеристики продукта в различных "измерениях", важных для пользователей и разработчиков. Атрибуты касаются вопросов портируемости, интероперабельности (прозрачности взаимодействия с другими системами), целостности данных, отказоустойчивости программного продукта.

Ограничения (Constraints) – формулировки условий, модифицирующих требования или наборы требований, сужая выбор возможных решений по их реализации. В частности, к ним могут относиться параметры производительности, влияющие на выбор платформы реализации и/или развертывания (протоколы, серверы приложений, баз данных), которые, в свою очередь, могут относиться, например, к внешним интерфейсам.

Системные требования (System Requirements) – иногда классифицируются как составная часть группы функциональных требований (не путайте с как таковыми "функциональными требованиями"). Описывают высокоуровневые требования к программному обеспечению, содержащему несколько или много взаимосвязанных подсистем и приложений. При этом, система может быть как целиком программной, так и состоять из программной и аппаратной частей. В общем случае, частью системы может быть персонал, выполняющий определенные функции системы, например, авторизация выполнения определенных операций с использованием программно-аппаратных подсистем.

Независимые свойства (Emergent Properties) – требования, которые не могут быть адресованы тому или иному компоненты программной системы, но которые должны быть

соблюдены, например, в контексте сетевой инфраструктуры или регламентов работы пользователей.

Требования с количественной оценкой – требования, имеющие количественное определение/измерение, например, система должна производить поиск документов <определенного вида> за время, не превышающее 5 секунд. Большинство требований с количественной оценкой относится к атрибутам качества

Формулирование требования в форме: "Система должна обеспечить рост пропускной способности" без указания конкретных количественных характеристик является просто некорректно определенным требованием!

Важную роль на этапе написания ТЗ играет архитектурное проектирование ПО. Для определения особенностей разрабатываемого Web-приложения необходимо отобразить в самом общем виде среду, в которой создаваемое приложение будет функционировать. Среда функционирования Web-приложение может содержать множество компонент. Реальную конфигурацию определяет администратор сети. Конфигурация клиент-серверной архитектуры может состоять из одного или нескольких компьютеров пользователя, подсоединенных к серверу посредством локальной (Local Area Network – LAN) или глобальной (Wide Area Network – WAN) сети. Модели клиент/серверных архитектур могут быть различные: одноуровневые, двухуровневые или трехуровневые. Важно указать какая модель сети будет использоваться, и по каким протоколам будет выполняется передача информации между компонентами сети.

Например, рассмотрим простую конфигурацию, состоящую из нескольких компонент. Пользователь просматривает Web-узел с помощью ***браузера***, подключенного к сети Internet. Программное обеспечение Web-узла может выполняться на браузере пользователя, сервере сети, подсоединенном в каком-то другом месте сети Internet, или на сервере приложений – в зависимости от того, как разработчики спроектировали систему. ***Брандмауэр***, который представляет собой комбинацию аппаратного и программного обеспечения, существует для того, чтобы ограждать сети от злоумышленников. Пользователь также может установить брандмауэр, чтобы защитить компьютер от внешних атак. ***Прокси-сервер*** – это программное обеспечение, цель которого заключается в том, что он является единственным соединением частной сети и Internet. Прокси-сервер выполняет множество функций (например, предотвращает проникновение некоторых файлов или покидание ими сети, улучшает характеристики системы путем кэширования и т. д.). Многие Web-приложения используют базы данных для хранения информации, необходимой для запуска Web-узла. На рис. 2 изображена описанная выше конфигурация среды.

В ТЗ следует описать особенности функционирования будущей системы в рамках конкретной конфигурации сети и обосновать основные технические решения:

- решения по структуре системы, подсистем, средствам и способам связи для информационного обмена между компонентами системы, подсистем;
- состав функций, комплексов задач, реализуемых системой (подсистемой);
- решения по взаимосвязи разрабатываемого ПО со смежными системами, обеспечение ее совместимости;
- решения по режимам функционирования, диагностированию работы системы;
- решения по составу информации, объему, способам ее организации, видам машинных носителей, входным и выходным документам и сообщениям, последовательности обработки информации и другим компонентам;
- задание потребительских характеристик системы (подсистемы), определяющих ее качество. Для интерактивных систем это, прежде всего, скорость и надежность взаимодействия.

Задание. Написать техническое задание для разработки Web-приложения в соответствии с ГОСТ 19.201-78. ЕСПД «Техническое задание. Требование к содержанию». Пример оформления технического задания дан в Приложении 1.

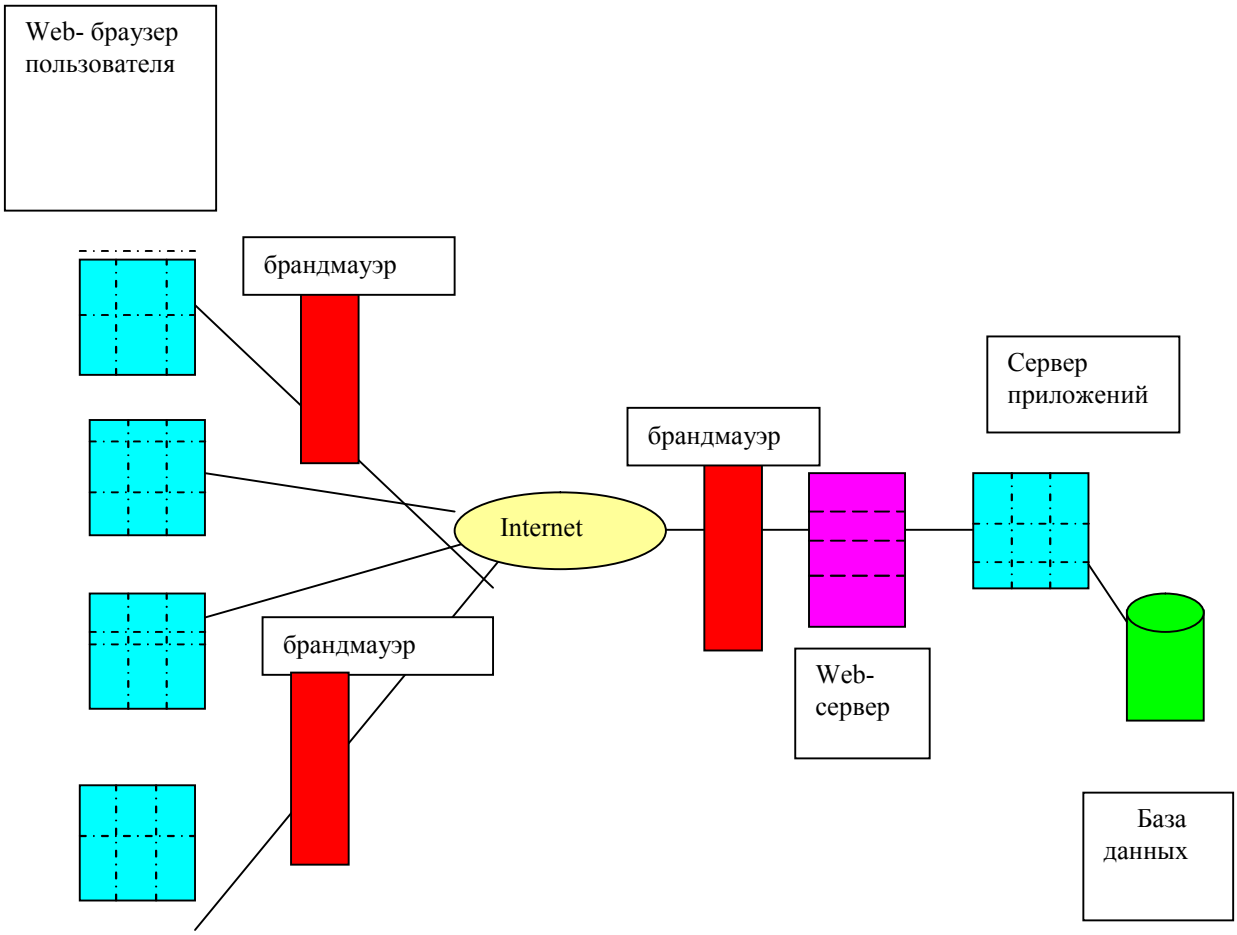

Рис. 2. Пример Web-среды

3. ЭТАП СОЗДАНИЯ ОБЩЕЙ КАРТИНЫ ПРИЛОЖЕНИЯ

Характеристика этапа

Цель создания общей картины приложения - выработать единое понимание проекта среди всех основных его участников.

Основные задачи, которые должны быть решены на этапе создания общей картины приложения:

- определение структуры приложения;
- определение бизнес-целей;
- оценка существующей ситуации;
- создание документа общей картины и области действия проекта;
- определение требований и профилей пользователей;
- оценка риска;
- подготовка отчетного документа этапа.

Результаты этапа создания общей картины приложения:

- формулировка задач и бизнес-целей;
- анализ существующих процессов;
- наиболее общее определение требований пользователей;
- профили пользователей, определяющие, кто будут работать с продуктом;
- стратегии проектирования приложения;
- список предварительно определенных рисков;
- планы устранения или снижения влияния выявленных рисков.

Этап создания общей картины приложения завершается контрольной точкой «Утверждение документа общей картины и области действия проекта». Данный документ согласуется с заказчиком.

Работа 2. Разработка документа общей картины и области действия приложения

В документе общей картины и области действия приложения основное внимание следует уделить пониманию и определению проблемы. В этом документе необходимо указать:

- формулировки задач; профили пользователей;
- область действия проекта; требования к проекту;
- концепцию решения; цели проекта (проектные и бизнес-цели);
- анализ рисков; календарный график работ.

Формулировки задач – это краткое описание задач, которые заказчик планирует решить за счет реализации проекта. Этот раздел документа должен создаваться на основе текущего состояния бизнес-операций.

Например, необходимо увеличить число посетителей, регистрирующихся на Web-сайте, за счет упрощения системы навигации.

Профили пользователей определяют потребности той или иной группы пользователей. При создании пользовательских профилей следуйте нескольким рекомендациям:

- опишите задачи, выполняемые пользователями;
- опишите информационные потоки между пользователями;
- опишите физическое и географическое местоположение пользователей, а также число пользователей в каждом месте, пропускную способность и загрузку линий связи между сайтами;
- укажите особенности поведения, которыми, по мнению пользователя, должен обладать продукт.

Область действия проекта – это четкое определение того, что входит в рамки проекта. При определении область действия проекта необходимо оценить важность бизнес-задач и выбрать те, которые будут реализованы в будущих версиях проекта.

Требования к проекту определяют запросы заказчика к приложению.

Концепция решения включает концептуальную модель программной и аппаратной архитектуры системы. Например, при решении проблемы хостинга Web-сайта возможны варианты: размещать его на локальных серверах или на сервере Интернет-провайдера.

Бизнес-цели проекта определяют задачи, которые клиент планирует решить, внедрив новый продукт.

Например, список бизнес-целей проекта для электронной коммерции:

- расширение географии сбыта компании за счет использования Web-ресурсов;
- расширение рынка сбыта компании за счет привлечения молодых потребителей, посещающих онлайновые магазины;
- сокращение времени на сбыт за счет развертывания эффективных онлайновых сайтов.

Проектные цели ориентированы на то, что команда будет делать для реализации бизнес-целей. Например,

- сократить среднее время загрузки страницы до 5 секунд;
- сократить время и усилия, затрачиваемые пользователем на онлайновую регистрацию;

- обеспечить доступность сервиса на уровне 99,9%.

Риск – это результат неопределенности результатов проектных решений. В MSF определены шесть стадий управления рисками, основные из них:

- определение рисков;
- анализ и определение важности рисков;
- отчетность и мониторинг рисков - наблюдение за рисками и документирование планов мероприятий по их устранению.
- управление рисками – процесс выполнения планов мероприятий по управлению и устранению рисков.

Количественная оценка рисков определяется в числовом выражении двумя значениями – вероятность риска и влияние риска (серьезность возможных потерь). Произведение этих чисел – подверженность риску. По данному показателю можно сравнивать риски, определять их относительную серьезность и предусмотреть мероприятия по предотвращения или снижения риска.

Для разработки Web-приложений возможны риски:

- недостаток опыта разработки, развертывания и поддержки Web-сайтов;
- риски, связанные с безопасностью, в том числе вероятность вторжения в систему злонамеренных пользователей и воровство конфиденциальных данных клиентов.

Задание. Разработать документ общей картины и области действия Web-приложения. Образец диаграмм вариантов использования приложения приведен в Приложение 2.

4. ЭТАП ПРОЕКТИРОВАНИЯ

Характеристика этапа

Цель этапа проектирования – создание архитектуры и дизайна приложения.

Основные задачи, которые должны быть решены на этапе проектирования:

- разработка дизайна и архитектуры приложения;
- создание функциональных спецификаций;
- разработка календарного графика этапа;
- подготовка отчетного документа этапа.

Дизайн приложения документируется в виде функциональной спецификации, которая описывает поведение и вид каждой функции будущего проекта, а также определяет архитектуру и дизайн всех функций.

Результаты этапа проектирования:

- функциональные спецификации;
- план управления рисками; календарный план проекта.

Функциональные спецификации – это виртуальное хранение проекта и связанных с проектом артефактов, которые создаются на этапе планирования в модели процессов MSF. Артефакты представляют собой результат процессов концептуального, логического и физического дизайна, выполняемых на этапе проектирования. К артефактам относятся UML- модели, такие, как диаграммы вариантов использования и состояний системы, диаграммы классов и другие информационные модели. Многие артефакты – это документы в электронной форме или данные, хранимые в базах данных.

Работа 3. Проектирование приложения

Этап проектирования приложения состоит из трех составляющих.

- **Концептуальный дизайн.** Задача рассматривается с точки зрения пользовательских и бизнес-требований, и определяется в виде сценариев использования системы.
- **Логический дизайн.** Задача рассматривается с точки зрения проектной команды, и решение определяется как набор сервисов.
- **Физический дизайн.** Задача рассматривается с точки зрения разработчиков (программистов). На этой стадии уточняются технологии, интерфейсы компонентов и сервисы решения.

Концептуальный дизайн – это процесс сбора, анализа и определение приоритетных особенностей проекта с точки зрения пользователей. На этапе концептуального дизайна уточняются артефакты, созданные на этапе создания общей картины.

В функциональных спецификациях, разрабатываемых на этапе концептуального дизайна, необходимо включить следующие артефакты.

1. Требования к системе, после их уточнения и классификации:
   - пользовательские требования;
   - системные требования;
   - процедурные требования;
   - Бизнес-требования.
2. Диаграммы вариантов использования системы.
3. Сценарии использования системы.
4. Доступные технологии и архитектура приложения.
5. Макет пользовательского интерфейса.

Логический дизайн – это процесс описания проекта в терминах структуры и взаимодействия его частей с точки зрения проектной команды. Результаты логического дизайна таковы:

- логическая модель объектов и соответствующих им сервисов, атрибутов и отношений;
- высокоуровневый дизайн пользовательского интерфейса;
- логическая модель данных.

Объекты – это люди или предметы, описанные в сценариях использования системы. Например, объектами могут быть: сотрудник, контракт, платеж, номер клиента.

Сервис – это схема поведения бизнес-объекта, то есть операции, которые он выполняет. Сервисы используются для реализации бизнес-целей, манипулирования данными и доступа к информации. Например, сервис - определение номера клиента.

Атрибуты – это свойства объекта. **Отношения** отображают связи между объектами.

Логическая модель данных может быть представлена в виде диаграммы последовательности [1]. Логическая модель данных – в виде диаграммы классов [1] или дизайна базы данных.

Высокоуровневый дизайн пользовательского интерфейса содержит подробную информацию о компоновке используемых компонентов пользовательского интерфейса. Компоненты пользовательского интерфейса служат для отображения информации на экране, ввода и проверки пользовательских данных, а также интерпретации действий пользователя. В схеме «модель – вид – элемент управления» (Model – View – Controller, MVC) компоненты пользовательского интерфейса играют роль представления и/или элемента управления. Согласно этой схеме приложение (или даже интерфейс приложения) делится на три части: модель (объект-приложение), представление (пользовательский вид) и механизм управления (пользовательский элемент управления).

Ниже перечислены характеристики удачно спроектированного и эффективного пользовательского интерфейса.

- Продуманное использование площади экрана.
- Интуитивно понятный дизайн.
- Простота навигации.
- Заполнение полей данных по умолчанию.
- Проверка входных данных на корректность.
- Наличие меню, панели инструментов и справочной системы.
- Эффективная обработка событий (минимальное время отклика приложения).

- Масштабируемость (возможность легко настраивать и расширять как интерфейс, так и само приложение при увеличении числа пользователей, рабочих мест, объема и характеристик данных).
- Адаптивность к действиям пользователя - приложение должно допускать возможность ввода данных и команд множеством разных способов (клавиатура, мышь, другие устройства) и многовариантность доступа к прикладным функциям (иконки, «горячие клавиши», меню и т.д.). Кроме того, программа должна учитывать возможность перехода и возврат от окна к окну, от режима к режиму, и правильно отрабатывать такие ситуации.
- Независимость в ресурсах - для создания пользовательского интерфейса должны предоставляться отдельные ресурсы, направленные на хранение и обработку данных, необходимых для поддержки пользователя (пользовательские словари, контекстно-зависимые списки, наборы данных по умолчанию или по последнему запросу, истории запросов и прочее).
- Переносимость – при переходе на другую аппаратную (программную) платформу, должен осуществляться автоматический перенос и пользовательского интерфейса, и конечного приложения.
- Стилевая гибкость – возможность использовать различные интерфейсы с одним и тем же приложением. Для web-интерфейсов на практике реализуется с помощью таблицы стилей, в том числе возможность в выборе пользователем собственных установок пользовательского интерфейса (цвет, иконки, подсказки и пр.).
- Совместное наращивание функциональности – возможность развивать приложение без разрушения, то есть, оставаясь в рамках существующего интерфейса.

В Microsoft .NET пользовательский Web-интерфейс разрабатывается средствами ASP.NET [3]. Эта технология предоставляет богатую функциями среду, позволяющую создавать сложные Web-интерфейсы. Вот лишь некоторые из возможностей ASP.NET:

- унифицированная среда разработки;
- привязка данных к пользовательскому интерфейсу;
- интерфейс на основе компонентов с элементами управления;
- встроенная модель безопасности каркаса .NET.
- доступность, производительность и масштабируемость Web-обработки данных.

Физический дизайн – это описание компонентов, сервисов и технологий проекта с точки зрения требований к разработке. Суть физического дизайна – это реализация логического дизайна в рамках конкретных информационных технологий для обеспечения:

- масштабирования;
- надежности и доступности;
- производительности
- способности к взаимодействию;
- безопасности.

Масштабируемость – это возможность быстро и легко модернизировать систему, чтобы она могла обслуживать больше операций или пользователей за счет увеличения системных ресурсов. Стандартные методы масштабирования – это вертикальное масштабирование (увеличение аппаратных ресурсов) и горизонтальное масштабирование (распределение нагрузки между несколькими серверами). Для обеспечения масштабируемости необходимо следовать нескольким рекомендациям:

- процессы не должны конкурировать за ресурсы (преимущество отдавайте наиболее доступным и емким ресурсам);
- процессы не должны ожидать отклика дольше, чем необходимо (используйте асинхронные процессы);
- необходимо разделять ресурсы и сервисы;
- следует создавать взаимозаменяемые компоненты (такие компоненты освобождают ресурсы и инициализируются снова для использования новым клиентом);

Доступность – это мера отказоутойчивости приложения, то есть соотношение времени, когда приложение обслуживает запросы, к запланированному времени его работы. Для реализации требований по доступности используйте следующие методики:

- сокращение времени простоя (применяется «горячее» обновление с использованием дополнительного сервера);
- сокращение внеплановых простоев за счет кластеризации (использование кластера – сети из нескольких компьютеров);
- балансировка сетевой нагрузки (распределение трафика между доступными серверами);
- использование копий данных для возможности при отказе одного из дисков приложение автоматически переадресовать на зеркальный диск;
- изоляция критически важных приложений (размещение их на выделенной инфраструктуре);
- использование очереди для приема сообщений (очередь гарантирует доставку сообщения, независимо от наличия связи и загруженности приложения).

Производительность приложения измеряется определенными метриками, например, время отклика системы. Выбор метрики зависит от назначения приложения. Как правило, Web-приложения используются в вычислительных средах, построенных на различных технологиях. Для оценки производительности приложений, при интеграции их в вычислительные среды, необходимо учитывать ряд типов взаимодействия:

- сетевое взаимодействие;
- взаимодействие с различными источниками данных;
- взаимодействие приложений;
- взаимодействия, связанные с управлением (данными пользователей, мониторингом системы).

Планирование безопасности приложения предусматривает:

- определение опасностей (разработка модели опасностей);
- создание политики безопасности (использование функций безопасности).

Существует множество механизмов защиты приложения: брандмауэры, прокси-серверы и защищенные каналы связи. Однако чтобы обойти систему безопасности, злоумышленнику достаточно отыскать одну брешь в системе. Брешь – это слабое место в системе защиты, которую использует нападающий для получения доступа к корпоративной сети или отдельным ее ресурсам. Вот наиболее типичные, уязвимые места приложений:

- слабые пароли;
- передача данных в незашифрованном виде;
- возможность переполнения буфера;
- секретные данные прописываются в коде;
- неправильная конфигурация ПО (возможность использования одного из сервисов приложения для получения доступа к системе).

На этапе проектирования разрабатывается модель опасностей, наиболее популярная из них STRIDE. STRIDE – это метод, применяемый для определения и классификации опасностей, грозящих приложению. Каждая буква в сокращении STRIDE обозначает определенную категорию опасности: подмена сетевых объектов (Spoofing identity), модификация данных (Tampering), отказ от причастности (Repudiation), раскрытие информации (dIsclosere), отказ от обслуживания (Denial of service) и повышение привилегий (Elevation of privilege) [2].

Основными методами по реализации функций безопасности приложения являются:

- аутентификация и авторизация;
- защищенная связь; фильтрация;
- защита секретной информации;

- защита аудита; качество сервиса;
- конфиденциальные протоколы;
- минимизация привилегии;
- ограничение числа входящих запросов;
- цифровые подписи; безопасные протоколы.

По завершению физического дизайна необходимо подготовить следующие документы:

- диаграммы классов;
- компонентная диаграмма, диаграммы последовательностей;
- схема базы данных;
- диаграмма развертывания приложения;
- спецификации компонентов;
- планирование безопасности приложения.

Задание. Разработать функциональные спецификации дизайна приложения. Примеры оформления функциональных спецификаций даны в Приложениях 3-5.

5. ЭТАП РАЗРАБОТКИ

Характеристика этапа

Цель этапа разработки – разработать и документировать код продукта.

Основные задачи, которые должны быть решены на этапе разработки:

- создание прототипа приложения;
- подготовка кода и документации;
- реализация функций безопасности в соответствии с функциональной спецификацией.

Результаты этапа разработки:

- исходный текст и исходные файлы;
- сценарии для установки и конфигурации для развертывания;
- завершенная функциональная спецификация;
- сценарии тестирования.

Работа 4. Создание прототипа приложения

При разработке приложения важным аспектом является полная реализация спецификаций дизайна приложении. Web-приложения, созданные на базе .NET, реализуют один или более логических сервисов, которые могут базироваться на разных технологиях, например, Microsoft ASP.NET, Enterprise Services, XML Web Services, удаленном взаимодействии (remoting), Microsoft ADO.NET и Microsoft SQL Swrver. Эффективное использо-

вание конкретных технологий позволит успешно реализовать функциональные спецификации дизайна.

Например, технология .NET Framework поддерживает следующие функции безопасности приложения [3]:

- аутентификация и авторизация;
- контроль типов (ограничение по доступу к коду приложения);
- подписание кода (создание реквизитов кода);
- шифрование и подписание данных;
- изолированное хранилище данных (использование виртуальной файловой системы на клиенте).

Задание. Разработать код Web-приложения в соответствии с функциональной спецификацией.

6. ЭТАП СТАБИЛИЗАЦИИ

Характеристика этапа

Цель этапа стабилизации – проверка готовности приложения для развертывания в промышленной среде.

Основные задачи, которые должны быть решены на этапе стабилизации:

- тестирование приложения;
- пилотная эксплуатация (тестирование приложения в реальных условиях его эксплуатации).

Результаты этапа стабилизации:

- финальная версия кода продукта;
- информативные документы о версии;
- результаты и инструменты тестирования.

Работа 4. Тестирование приложения

Тестирование позволяет выявить и устранить неполадки до начала развертывания приложения. Оно подтверждает, что компоненты приложения соответствуют техническому заданию, а также позволяет убедиться в корректности и полноте функциональности тестируемого приложения.

Команда разработчиков, которая пишет код, должна тестировать его помодульно и в процессе каждой сборки.

Команда тестировщиков отвечает за проектирование тестов, за создание сценариев автоматизированного тестирования, за выполнение тестирования и за подготовку от-

четной документации по тестированию приложения. Эта команда также несет ответственность за принятие решения о готовности приложения к эксплуатации.

В .NET Framework определены два вида тестирования: базовое тестирование и тестирование пользовательских функций [2].

Базовое тестирование – это низкоуровневое тестирование кода. Как правило, его выполняет автор, хотя возможно и внешнее базовое тестирование, выполняемое тестировщиком.

Тестирование пользовательских функций – это тестирование на пригодность к использованию и считается высокоуровневым, его выполняют тестировщики.

Сформулируем некоторые рекомендации по тестированию.

1. Для принятия решения об успешности разработки приложения необходимы критерии. Создание **критериев успешности** – это определение условий, при выполнении которых считается, что приложение соответствует техническому заданию. Критерии успешности иногда называют ключевыми показателями производительности.
2. Для оценки неполадок или ошибок при разработке приложения в .NET Framework рекомендуется использовать следующие параметры [2].

   Повторяемость – количество ошибок в единицу времени.

   Обнаруживаемость – вероятность обнаружить ошибку.

   Серьезность – степень влияния неполадки на приложение (целое число в диапазоне от 1 до 10). Например, уровень 10 соответствует аварийному завершению приложения с потерей данных и невозможностью их восстановления, 9 - аварийному завершению приложения с потерей данных, которые удастся восстановить.

   Эти три значения параметров используются для определения важности ошибки.

   Важность ошибки обычно вычисляется по следующей формуле:

   Важность = (Повторяемость + Обнаруживаемость) * Серьезность
3. По итогам тестирования на каждом этапе необходимо оформлять отчет по тестированию, определяя матрицу важности ошибок и делая вывод о дальнейшей работе по созданию приложения.
4. При выборе приоритетов тестирования приложений необходимо основываться на типе тестируемого Web-узла. Для информационных основным свойством является доступность, в то время как для интерактивных узлов важна скорость и надежность взаимодействия.

Рассмотрим основные направления тестирования Web-приложений [6].

Модульное и интеграционное тестирования

Модульное и интеграционное тестирования – это особые формы внутреннего базового тестирования, в которых применяются средства автоматизации.

Модульное тестирование – это тестирование классов [1]. Основной принцип модульного тестирования заключается в независимом тестировании отдельных функций (по одной за один раз).

Интеграционное тестирование – это тестирование правильности взаимодействия классов, входящих в различные модули. Методики по интеграционному тестированию представлены в части 1 методических указаний [1].

В ASP.NET Framework автоматическое тестирование наиболее целесообразно проводить с помощью средств, интегрированных в среду разработки Microsoft Visual Studio 2008.

Тестирование функциональных возможностей

Тестирование функциональных возможностей заключается в подтверждении того, что в Web-приложении реализованы функциональные требования, описанные в ТЗ. Оформление тестов возможно в виде таблиц, например, Таблица 1.

Таблица 1. Пример оформления результатов тестирования.

Номер теста	Описание теста	Входные данные	Ожидаемые результаты	Реальные результата	Результат тестирования

Тестирование практичности

При тестировании практичности оценивается дружелюбие узла к пользователям**.** Ключом к тестированию практичности является изучение реальных действий пользователя. Тестер фиксирует действия пользователя и его реакцию, чтобы определить, с каким типом трудностей сталкивается пользователь и как он их преодолевает.

Тестовые примеры по практичности являются четко заданным и несложным набором задач, которому должны следовать участники. Выполнение тестовых примеров по практичности состоит в наблюдении за реакциями и эмоциями участников во время выполнения заданных задач. Со стороны тестировщика не должно поступать никакой ин-

формации и не должна предоставляться начальная помощь. Тестировщик фиксирует следующую информацию:

- успешно ли пользователь выполнил задачу;
- сколько времени понадобилось пользователю на выполнение задачи;
- число страниц, к которым нужен был доступ для завершения каждой задачи;
- в каких местах пользователь сталкивался с трудностями или где он делал ошибки;
- как пользователь искал помощи в случае неудач;
- предоставляет ли интерактивная справка достаточное количество информации;
- щелкал ли пользователь по страницам или использовал возможность поиска;
- как пользователь реагировал на время загрузки отдельных страниц;
- место, в котором пользователи запутывались или даже были не в состоянии выполнить задачу;
- количество щелчков между задачами, время между щелчками, а также число просмотренных страниц и то, какие именно страницы просматривались.

Результаты тестирования практичности, должны быть документированы.

Тестирование форм

При тестировании форм подтверждается, что каждое поле работает так, как планировалось разработчиком. Тестирование форм состоит из следующих действий:

- использование клавиши табулятора для проверки того, что форма проходит по полям в положенном порядке как вперед, так и назад;
- тестирование граничных значений для данных, вводимых на форме;
- проверка корректного отлавливания формами неверных данных, в особенности форматов даты и числовых форматов;
- проверка правильности обновления формами информации.

Тестирование содержимого страницы

При тестировании содержимого страницы подтверждается, что информация, предоставляемая узлом, корректна. Тесты, подтверждающие корректность содержимого страницы с точки зрения пользователя делятся на две категории: подтверждение адекватного функционирования каждой компоненты и подтверждение того, что содержимое каждой из них корректное.

Задача первой категории тестов – проверить, что:

- все изображения и графики правильно отображаются в различных браузерах;

- все содержимое представлено согласно требованиям;
- структура страниц остается постоянной в различных браузерах;
- динамические страницы сохраняют содержимое от версии к версии;
- представлены все части таблицы или формы, а их расположение правильное;
- ссылки на важное содержимое внутри и снаружи узла точные;
- объекты с текстом всплывающей подсказки корректны;
- Web-страницы внешне выглядят привлекательно.

Чтобы убедиться в корректности содержимого, необходимо проверить содержимое на точность, оценить правильность орфографии, грамматики и терминологии.

Тестирование навигации

При тестировании навигации путем проверки доступа к узлу, изображениям, ссылкам и другим компонентам узла подтверждается, что пользователь может выполнить желаемую задачу. Оценка навигации является наиболее значимой частью тестирования практичности. Охарактеризуем основные направления при тестировании навигации:

- Переход на страницу и с нее;
- Прокрутка страниц;
- Щелканье на развернутых и свернутых изображениях, чтобы убедиться, что они работают;
- Тестирование всех ссылок (как внутри, так и снаружи Web-узла), чтобы подтвердить их обоснованность и корректность;
- Подтверждение того, что извлекаемые страницы – именно те, которые предполагалось увидеть;
- Просмотр таблиц и форм, чтобы убедиться в правильности их расположения (для разных браузеров размещение таблиц и форм может быть различным);
- Подтверждение того, что окна с большим количеством блоков обрабатываются так, как если бы каждый из них содержал только один блок;
- Измерение времени загрузки каждой страницы;
- Подтверждение совместимости и согласованного использования клавиш, быстрых сочетаний клавиш и действий мыши.
- возможность вернуться в предыдущее состояние или на домашнюю страницу;

Тестирование баз данных

Тестирование баз данных часто является очень важной частью тестирования Web-приложения, поскольку информация на Web-узлах обычно хранится в базах данных. В не-

которых приложениях есть данные, вводимые пользователем, которые затем становятся частью баз данных.

К ключевым проблемам, возникающим при тестировании баз данных, относятся:

- целостность данных;
- достоверность данных (подходящая форма при вводе в базы данных);
- манипуляции с данными и обновления (обновления значения количества данных и т.д.).

Целостность данных – это основное условие успешной реализации Web-узла. Организация должна установить средства для контроля искажения данных. Искажение данных часто обнаруживается только на поздних стадиях, поскольку искажения происходят в подходе “создание блоков”. Все начинается с малого, но со временем (или с увеличением числа транзакций) проблема усугубляется. Установка подходящих точек проверки может помочь ослабить искажение данных. Например, если осуществлять ежедневный пересмотр журнала транзакций, то тем самым можно облегчить отслеживание изменений в базах данных.

При тестировании баз данных также нужно учитывать достоверность данных. Достоверность данных гарантирует предоставление клиентам точной информации и возврат этой информации в базу данных. Как правило, доля дефектов в данных значительно больше, чем в самом приложении. В таком случае необходимо изучить последовательность выполняемых действий и проверить базу данных в точках изменений. Этот подход заключается в изоляции действий, которые модифицируют базы данных, а также в инспектировании измененного содержимого на предмет корректности. Важно, чтобы тестер осознавал области риска и правильно расставлял приоритеты при тестировании.

Тестирование баз данных выполняется на двух уровнях: административных и пользовательских функций. Администратор баз данных может выполнить ограниченные действия, недоступные для клиентов Web-узла.

Тестирование безопасности

Тестирование безопасности предназначено для обнаружения уязвимых мест в приложении, которые могли бы позволить несанкционированный доступ пользователя к системе. При разработке тестов, связанных с безопасностью, могут помочь ответы на следующие вопросы.

- Какие существуют меры предосторожности для предотвращения или ограничения атак со стороны хакеров?

- Настраиваются ли в браузере установки, гарантирующие максимально безопасную защиту?
- Как Web-узел оперирует правами доступа?
- Есть ли в приложении программа обнаружения вирусов для пользователей, вошедших в систему посредством сетевого теледоступа или стороннего внутрисетевого провайдера?
- Оперирует ли приложение тайными действиями транзакций, т.е. изменением первоначального адреса отправки?
- На месте ли системный журнал, как часто он проверяется, автоматически определяя подозрительное поведение?
- Как Web-узел шифрует данные?
- Каким образомWeb-узел аутентифицирует пользователей?
- Дает ли возможность просмотр исходного кода выявить наиболее важную информацию?
- Можно ли получить доступ к системе и нанести ей ущерб путем прямого вызова баз данных по коммутируемым линиям связи посредством модема?
- Как защищены кредитные карточки или информация пользователя? Какая защищенность считается достаточной?
- Допускает ли приложение файлы, обозначаемые цифрами? (Тут используется технология шифрования с открытым ключом, гарантирующая, что стороннее лицо не нанесет вред файлу.)

Тестирование надежности и доступности

При тестировании надежности и доступности оценивается доступность Web-узла в любое время по запросу пользователя. Такая оценка достигается путем тестирования приложения в пиковое время использования (в периоды маркетинговой стимуляции и циклов высокой активности). Важно, чтобы тестеры понимали архитектуру системы для того, чтобы провести адекватное тестирование доступности и надежности. Например, если в системе содержится два Web-сервера для выравнивания нагрузки, то тестеры должны быть в состоянии описать характеристики системы с одним Web-сервером. Если по каким-либо причинам один из Web-серверов будет удален из продукта (например, для обслуживания), то Web-сайт должен продолжать функционировать на более низком, но все же приемлемом уровне.

Для расчета степени доступности используется формула [2]:

Доступность = (t1/(t1+t2))*100,

где t1 – средний промежуток бесперебойной работы (время работы приложения, деленное на число сбоев);

t2 – средний промежуток времени, необходимый для восстановления приложения после сбоя (время восстановления приложения, деленное на число сбоев).

Задание. Выполнить тестирование Web-приложения по предложенным направлениям. Пример отчета по результатам тестирования приведен в Приложении 6.

7. ЭТАП РАЗВЕРТЫВАНИЯ

Характеристика этапа

Цель этапа развертывания – запуск решения в промышленную эксплуатацию.

Основные задачи, которые должны быть решены на этапе развертывания:

- создание документации по развертыванию проекта;
- развертывание окружения и самого решения;
- анализ проекта совместно с заказчиком.

Результаты этапа развертывания:

- системы сопровождения и поддержки (процессы, базы знаний, отчеты);
- документы (код и документация проекта);
- план обучения;
- отчет о проекте.

Работа 5. Оформление отчета о выполнении лабораторных работ

Отчет о выполнении лабораторных работ должен содержать:

- титульный лист;
- оглавление;
- тексты лабораторных работ;
- список литературы;
- приложения.

Задание. Оформить отчет о выполнении лабораторных работ.

Приложение 1

ПРИМЕР ТЕХНИЧЕСКОГО ЗАДАНИЯ ПРОЕКТА
«Система диагностики гетерогенных компьютеризированных систем»

Глоссарий

Термин	Определение
Валидация данных	Проверка на корректность, полноту и непротиворечивость входных, выходных и обрабатываемых данных
Гетерогенная система	Функции и подсистемы, интегрируемые в единую информационную систему, реализованные на различных программно-аппаратных платформах с разными архитектурами, расположенных в различных подразделениях предприятия, возможно, территориально удаленных друг от друга.
Децентрализация	Тип архитектуры программных систем без создания выделенного единственного центра
Диагностируемый параметр	Количественная характеристика диагностируемого узла
Диагностируемый узел	Техническое устройство, принимающее участие в работе вычислительной системы, параметры которого необходимо контролировать
Работоспособность узла	Состояние диагностируемого узла, при котором диагностируемые параметры диагностируемого узла не выходят за допустимые значения
Хеширование	Преобразование входного массива данных произвольной длины в выходную битовую строку фиксированной длины
WEB-интерфейс	Интерфейс пользователя, предоставляемой системой через Web-браузер

Принятые сокращения

Сокращение	Расшифровка
АРМ	Автоматизированное рабочее место
ОС	Операционная система
СПО	Специальное программное обеспечение

Введение

Данное техническое задание составляется для проектирования СПО «Система диагностики гетерогенных компьютеризированных систем» (далее – СПО). Техническое задание выполняется в соответствии со стандартом ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы».

Краткое описание предметной области

Для эффективного выполнения задачи по обработке большого объёма информации необходимо привлекать системы, состоящие из нескольких узлов, соединенных каналами связи. Примерами таких задач могут служить научные исследования с большими объема-

ми данных и огромными потребностями в вычислительных мощностях, хранилища данных с большими объемами и высокой доступностью, базы данных с высокой доступностью. При этом количество узлов в системах, обеспечивающих решение таких задач, может варьироваться от единиц до сотен тысяч. Подобные информационные системы используют такие корпорации, как Google, Yahoo, Microsoft и многие другие.

В общем случае, такие системы состоят из некоторого количества узлов, объединённых каналами связи. Схема предметной области представлена на Рис. П1.1. Под узлами понимаются технические устройства, непосредственно используемые для решения части общей задачи. С ростом количества узлов растёт и вероятность выхода из строя системы в целом во время работы. Потеря работоспособности какого-либо из узлов – неизбежное событие, но можно избежать разрушительных последствий путём быстрого восстановления работоспособности вышедшего из строя узла. Для быстрого восстановления узла после сбоя необходимо, прежде всего, оперативно обнаружить факт выхода узла из строя. Для повышения скорости обнаружения привлекают специальные системы диагностики и контроля. Привлекаемые системы диагностики должны обладать свойствами распределённых систем, как географически (способность системы диагностировать узлы, находящиеся в разнесённых центрах обработки данных), так и распределенных в смысле хранения данных (информация о состоянии диагностируемых узлов может поступать с высокой интенсивностью, что не позволяет реализовать систему в виде монолитного сервера).

Для выполнения диагностирования узлов системы необходимо производить сбор параметров различного рода. В качестве параметров могут выступать электромеханические характеристики аппаратного обеспечения узла (напряжение, температура, скорость вращения вентилятора и т.д.), различные программные параметры операционной системы (объем свободного места на диске, загрузка процессора), а также параметры запущенных сервисов (состояние сервиса почты, количество активных сессий WEB-сервера). Можно выделить основные типы значений диагностируемых параметров: целое число (количество подключённых пользователей, количество открытых сессий и т.д.); вещественное число (загрузка процессора, используемая пропускная способность канала и т.д.); логическое значение (включено или выключено устройство и т.д.). Остальные типы могут быть смоделированы на основе трёх базовых типов, определённых выше.

Существующие аналоги

В качестве аналогов проектируемой системы можно выделить такие системы, как Zabbix, Nagios, Cacti. Все эти системы мониторинга направлены на мониторинг серверов и сетевого оборудования, причем в них отсутствует способность диагностирования периферийных устройств. Все эти системы реализованы в виде одного сервера, что усложняет масштабирование системы при диагностировании большого количества узлов.

Описание системы

Главное назначение СПО «Система диагностики гетерогенных компьютеризированных систем» – диагностирование состояния отдельных узлов. Узлы могут находиться под управлением различных операционных систем (например, Linux, Windows, FreeBSD и т.д.) и иметь различные технические платформы (например, различные процессоры: x86, Cell, Power, ARM), а также узлы могут быть географически разнесены (различные центры обработки данных).

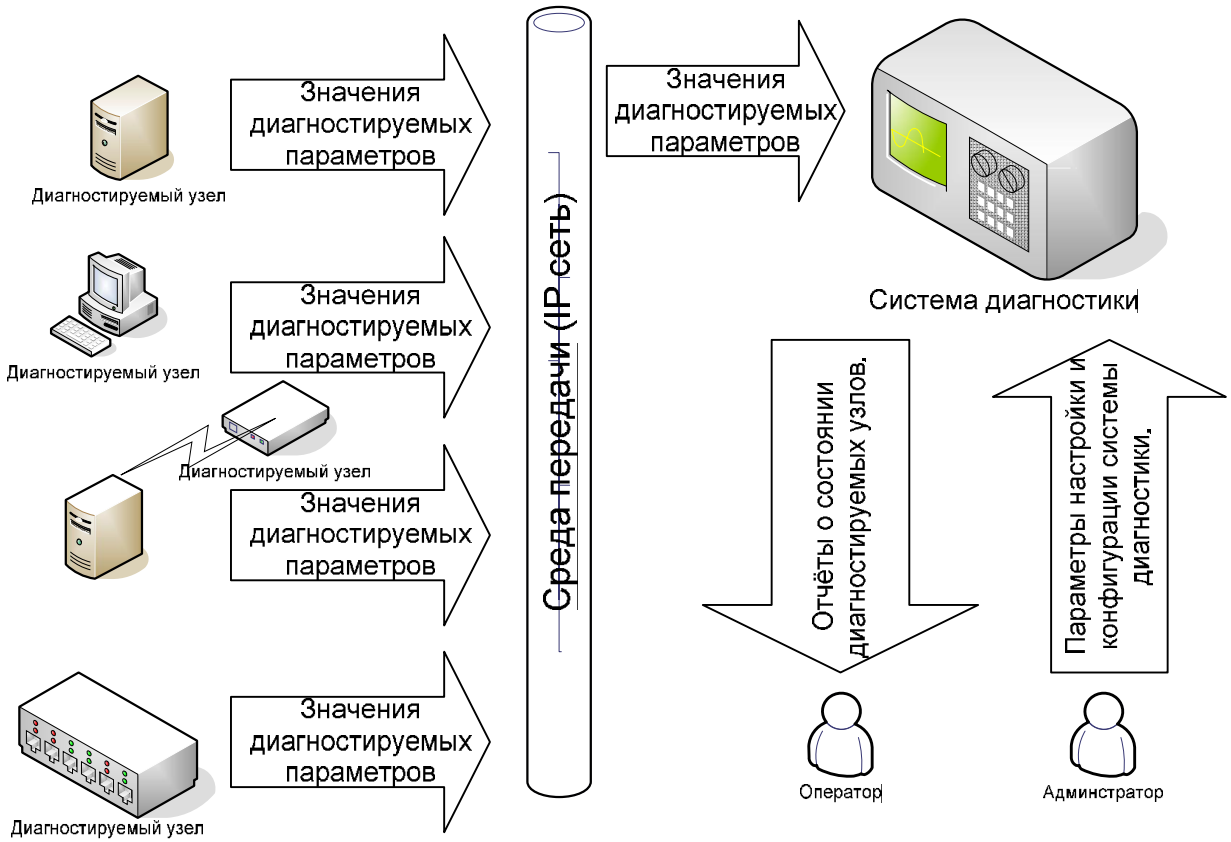

Рис. П1.1 Схема предметной области

На

Рис. **П1.2** приведена топология СПО «Система диагностики гетерогенных компьютеризированных систем». СПО состоит из следующих частей:

- система диагностики:
  - серверы диагностики;
  - АРМ администратора;
  - АРМы операторов.
- диагностируемые узлы (единица диагностирования):
  - серверы;
  - АРМ;
  - сетевое оборудование;
  - периферийные устройства, подсоединенные к АРМ или серверу.

Система диагностики - Сервера диагностики

Основное назначение серверов диагностики заключается в сборе и обработке информации, собираемой от диагностируемых узлов. Предоставление WEB-интерфейса для выполнения задач администрирования системы и задач, связанных с диагностированием узлов.

Система диагностики - АРМ администратора

Часть СПО «Система диагностики гетерогенных компьютеризированных систем», предназначенная для выполнения задач по администрированию серверов диагностики.

Система диагностики – АРМы операторов

Часть СПО «Система диагностики гетерогенных компьютеризированных систем», предназначенная для выполнения задач по диагностике и мониторингу узлов.

Диагностируемые узлы

Под диагностируемыми узлами понимаются некоторые компьютеризированные модули, зарегистрированные в системе с поддержкой протокола системы диагностики. Диагностируемые узлы можно подразделить на основные виды:

- серверы;
- АРМ;
- сетевое оборудование (диагностирование осуществляется с использованием протокола SNMP);
- периферийные устройства, подсоединенные к АРМ или серверу (подсоединение осуществляется не в IP сеть).

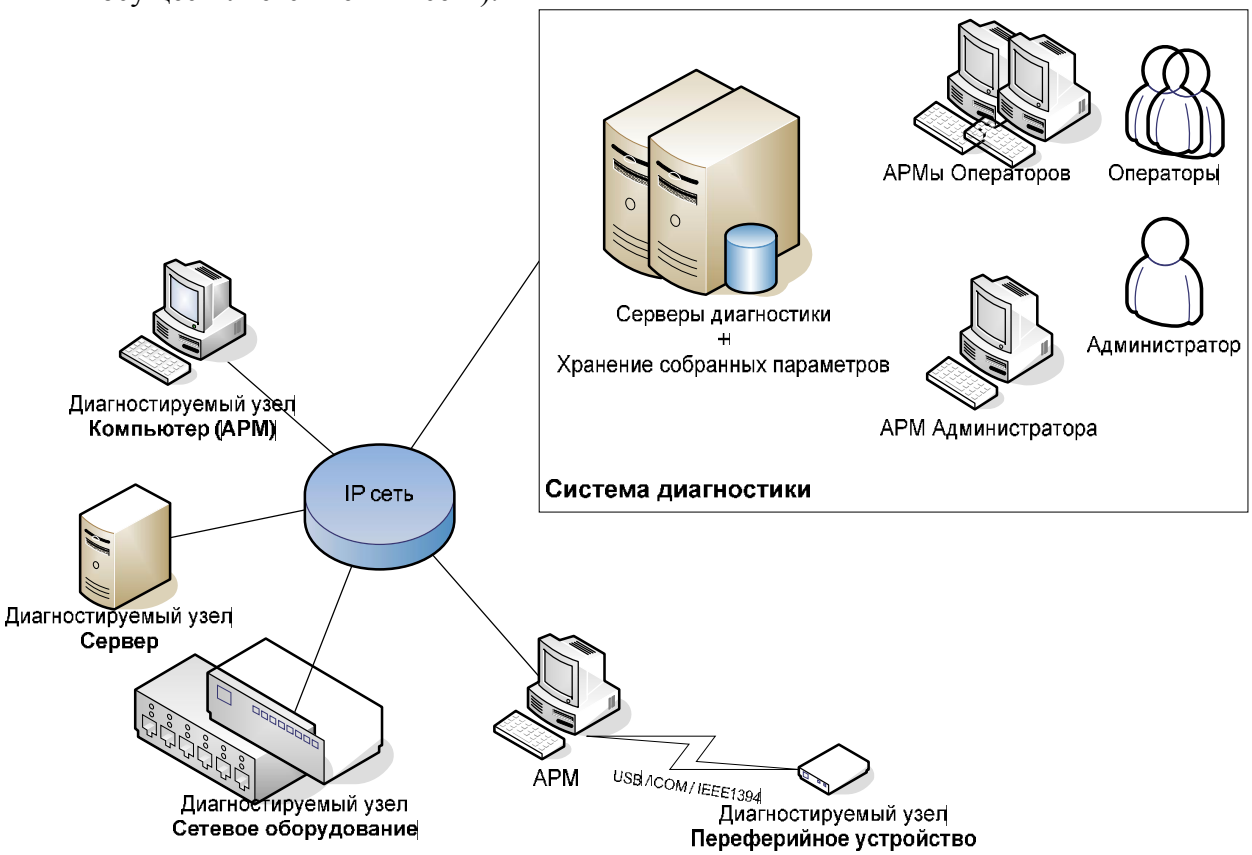

Рис. П1.2 Топология СПО «Система диагностики гетерогенных компьютеризированных систем».

Основания для разработки

Разработка ведется в рамках выполнения лабораторных работ по курсу «Технология программирования» на основании утверждённого учебного плана.

Назначение разработки

СПО «Система диагностики гетерогенных компьютеризированных систем» предназначена для сбора информации о состоянии зарегистрированных узлов и визуализации информации о состоянии узлов для поддержки принятия решений операторами.

Требования к системе

1. Разрабатываемое программное обеспечение должно обеспечивать функционирование системы в режиме 24/7/365.
2. Время восстановления системы после сбоя не должно превышать 3-х часов.
3. Система диагностики должна обеспечивать хранение значений диагностируемых параметров.
4. Система должна обеспечивать децентрализацию сбора диагностической информации с помощью формирования вынесенных локальных серверов сбора информации, которые, в свою очередь, передают информацию в центральный сервер диагностирования.
5. Система должна обеспечивать визуализацию состояний диагностируемых узлов в табличном и графическом виде.
6. Система должна поддерживать возможность «горячего» переконфигурирования системы.

Требования к функциональным характеристикам

1. Время отклика системы на запрос пользователя не должно превышать 10 секунд.
2. Время отклика системы на запрос «Добавить значение диагностируемого параметра в систему» не должно превышать 1 секунды.
3. Обеспечить заданные временные характеристики для 20 одновременно подключённых диагностируемых узлов.

Требования по реализации

1. СПО должно предоставлять удобные интерфейсы для выполнения функций администрирования, поддерживать запуск и конфигурирование из командной строки.
2. Система должна обеспечивать возможность запуска всех компонентов как сервисов ОС.
3. Интерфейсы управления и интерфейсы операторов должны быть реализованы как WEB-интерфейсы.
4. Пароли учётных записей должны подвергаться хешированию.
5. Узел должен обладать уникальным идентификатором в пределах системы диагностики, который должен храниться в системе диагностики.
6. Система должна предоставлять как минимум два интерфейса – интерфейс пользователя и администратора.
7. Для хранения данных о пользователях, узлах, группах диагностируемых параметров и их значений использовать СУБД Microsoft SQL Server 2005.

Функциональные требования к системе с точки зрения пользователя

СПО должна обеспечивать реализацию следующих функций.

1. Система должна обеспечивать регистрацию пользователей.
2. Система должна обеспечивать аутентификацию пользователей.
3. Система должна обеспечивать разделение пользователей на две роли:
   - администратор;
   - оператор.
4. Система должна предоставлять **администратору** следующие функции:
   - регистрация новых пользователей;
   - изменение паролей пользователей;
   - изменение информации о пользователе;
   - добавление новых узлов;

- изменение информации об узле;
- формирование своего списка диагностируемых параметров для узлов;
- удаление узлов;
- просмотр списка узлов.

5. Система должна предоставлять **оператору** следующие функции:
   - просмотр информации о диагностируемых узлах;
   - формирование отчёта отчётов об изменении диагностируемых параметров узла за выбранный период;
   - просмотр списка узлов.

Входные параметры системы

1. Система должна содержать следующую информацию о пользователях:
   - учётное имя пользователя в системе (необходимо для идентификации);
   - пароль в хешированном виде по алгоритму SHA256 (стандарт FIPS PUB 180-3);
   - текстовое описание (носит поясняющий характер);
   - тип пользователя, который может быть следующим: администратор, оператор;
   - имя пользователя;
   - фамилия пользователя;
   - время добавления пользователя в систему.
2. Система должна содержать следующую информацию о диагностируемом параметре:
   - идентификатор (уникальный в пределах диагностируемого узла);
   - текстовое описание (носит поясняющий характер);
   - тип значения параметра, который может быть: целым, вещественным, логическим;
   - верхний допустимый предел (в соответствии с типом значения параметра);
   - нижний допустимый предел (в соответствии с типом значения параметра).
3. Система должна содержать следующую информацию о типе диагностируемого узла:
   - идентификатор (уникальный в пределах системы);
   - текстовое описание (носит поясняющий характер);
   - список диагностируемых параметров.
4. Система должна содержать следующую информацию о диагностируемом узле:
   - идентификатор (уникальный в пределах системы);
   - текстовое описание (носит поясняющий характер);
   - сетевой адрес;
   - тип узла, может быть выбран из списка зарегистрированных в системе типов узлов.
5. Считанные параметры диагностируемых узлов, которые должны включать следующую информацию:
   - идентификатор узла;
   - набор пар вида «идентификатор параметра – значение». Значение может быть следующих типов: целое, вещественное, логическое;
   - время определения значения параметра.

Выходные параметры системы

Графическое отображение значений параметров узла (узлов) в виде графиков и/или диаграмм в основном интерфейсе оператора. Интервал отображения должен задаваться пользователем. Графики и/или диаграммы должны отражать тенденцию изменения значения диагностируемого параметра в зависимости от времени.

Требования к составу и параметрам технических средств

Минимальные требования к программно-аппаратному обеспечению для компьютеров без хранения значений параметров:

- тактовая частота не менее 1 ГГц;
- оперативная память не менее 1 Гб;
- ОС версии не ниже Windows Server 2003;
- свободное пространство на жестком диске не менее 10 Гб для ОС;
- свободное пространство на жестком диске не менее 200 Мб для СПО;
- Ethernet адаптер стандарта 1000BASE-T.

Минимальные требования к программно-аппаратному обеспечению для компьютеров с хранением значений параметров:

- тактовая частота не менее 2 ГГц;
- оперативная память не менее 2 Гб;
- ОС версии не ниже Microsoft Windows Server 2003;
- СУБД версии не ниже Microsoft SQL Server 2005 Standard Edition;
- свободное пространство на жестком диске не менее 10 Гб для ОС;
- свободное пространство на жестком диске не менее 200 Мб для СПО;
- пространство на жестком диске не менее 200 Гб для файлов базы данных;
- Ethernet адаптер стандарта 1000BASE-T.

Минимальные требования к программно-аппаратному обеспечению для компьютеров АРМов:

- тактовая частота не менее 1 ГГц;
- оперативная память не менее 512 Мб;
- ОС с браузером Opera 9.65 или совместимым с ним;
- Ethernet адаптер стандарта 1000BASE-T.

Сценарии функционирования системы

Регистрирование нового узла

Действие должно выполняться пользователем с типом учётной записи «Администратор» посредством интерфейса администратора и состоять из следующих шагов.

1. Администратор вводит логин и пароль. Если аутентификация пройдена успешно, то осуществляется переход к следующему шагу, в противном случае система должна предложить пользователю повторить этот шаг.
2. Администратор выбирает функцию «Добавить узел» и формирует список диагностируемых параметров.
3. Администратор заполняет все информационные поля.
4. Система проводит валидацию введённых данных, проверяя заполнение необходимых полей и соответствие введенной информации требуемым шаблонам.
5. Администратор подтверждает ввод данных. В случае ввода некорректных данных система должна выдавать пользователю сообщение об ошибке с указанием положения некорректных данных и предложить повторить шаг 3.

Удаление узла

Действие должно выполняться пользователем с типом учётной записи «Администратор» посредством интерфейса администратора и состоять из следующих шагов.

1. Администратор вводит логин и пароль. Если аутентификация пройдена успешно, то осуществляется переход к следующему шагу, в противном случае система должна предложить пользователю повторить этот шаг.

2. Администратор выбирает функцию «Удалить узел».
3. Система выдает сообщение о подтверждении удаления узла и возможных последствиях.
4. Администратор подтверждает удаление, система удаляет выбранный узел.

Регистрирование нового пользователя

Действие должно выполняться пользователем с типом учётной записи «Администратор» посредством интерфейса администратора и состоять из следующих шагов.

1. Администратор вводит логин и пароль. Если аутентификация пройдена успешно, то осуществляется переход к следующему шагу, в противном случае система должна предложить пользователю повторить этот шаг.
2. Администратор выбирает функцию «Добавить пользователя».
3. Администратор заполняет все информационные поля.
4. Система выполняет проверку ввода всех необходимых данных.
5. Администратор подтверждает добавление пользователя. В случае неудачного выполнения проверки на шаге 4 система выдает сообщение об ошибке с указанием положения некорректных данных.

Отправка значений диагностируемых параметров

Действие выполняется диагностируемым узлом и состоит из следующих шагов.

1. Диагностируемый узел автоматически считывает значения диагностируемых параметров.
2. Диагностируемый узел формирует пакет для отправки по сети.
3. Диагностируемый узел отправляет пакет.

Удаление пользователя

Действие должно выполняться пользователем с типом учётной записи «Администратор» посредством интерфейса администратора и состоять из следующих шагов.

1. Администратор вводит логин и пароль. Если аутентификация пройдена успешно, то осуществляется переход к следующему шагу, в противном случае система должна предложить пользователю повторить этот шаг.
2. Администратор выбирает функцию «Удалить пользователя».
3. Система выдает сообщение о подтверждении удаления пользователя и возможных последствиях.
4. Администратор подтверждает удаление, система удаляет выбранного пользователя.

Замечание: необходимо предусмотреть невозможность удаления администратора, который в данный момент выполняет операции администрирования.

Приём значений диагностируемых параметров

Описывает действия предпринимаемые сервером диагностики при приёме параметров и состоит из следующих шагов.

1. Сервер диагностики принимает пакет из сети.
2. Сервер диагностики выполняет операцию валидации над принятым пакетом, проверяя корректность переданных данных. Если данные некорректны, пакет игнорируется.
3. Сервер диагностики извлекает значения диагностируемых параметров из принятого пакета.
4. Сервер диагностики записывает полученные значения в хранилище.

Требования к надежности

Для повышения надёжности необходимо предусмотреть возможность установки дублирующего сервера для сервера, обеспечивающего работу пользователей через web-

интерфейс, поскольку он является наиболее уязвимым и важным звеном в архитектуре системы.

Администратор баз данных должен обеспечить периодическое создание резервных копий базы данных (1 раз в сутки).

Для предотвращения сбоев в работе СУБД при сбое в подаче электропитания, необходимо обеспечить подключение серверов к устройствам бесперебойного электропитания, которые обеспечат не менее 30 минут автономной работы.

Требования к документации

Документация должна включать:

- руководство по развертыванию системы;
- руководство по использованию системы для администратора;
- руководство по использованию системы для оператора.

Приложение 2

ДИАГРАММА ПРЕЦЕДЕНТОВ

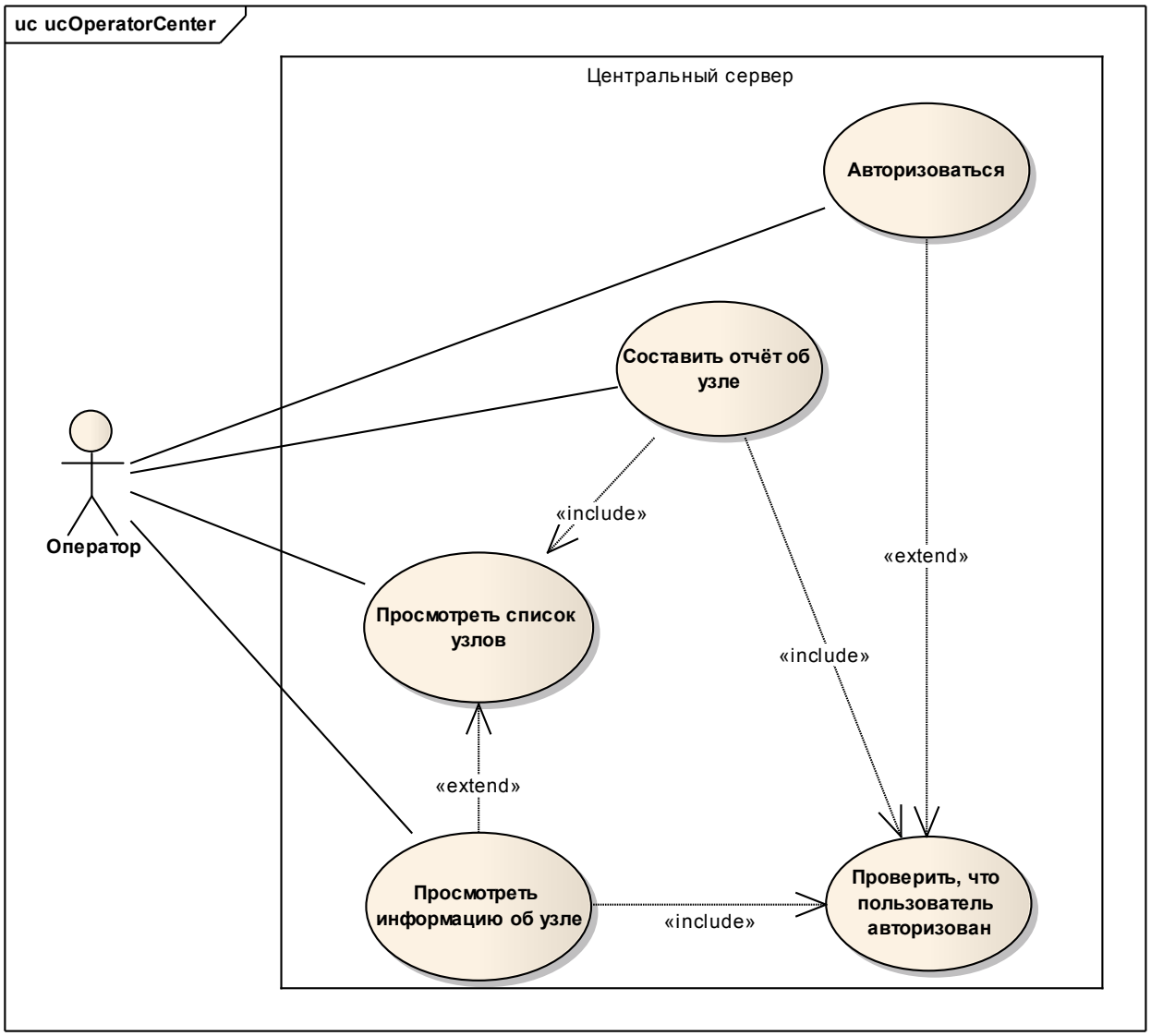

Рис. П2.3 Диаграмма прецедентов с точки зрения оператора

Приложение 3

ДИАГРАММА ПОТОКОВ ДАННЫХ

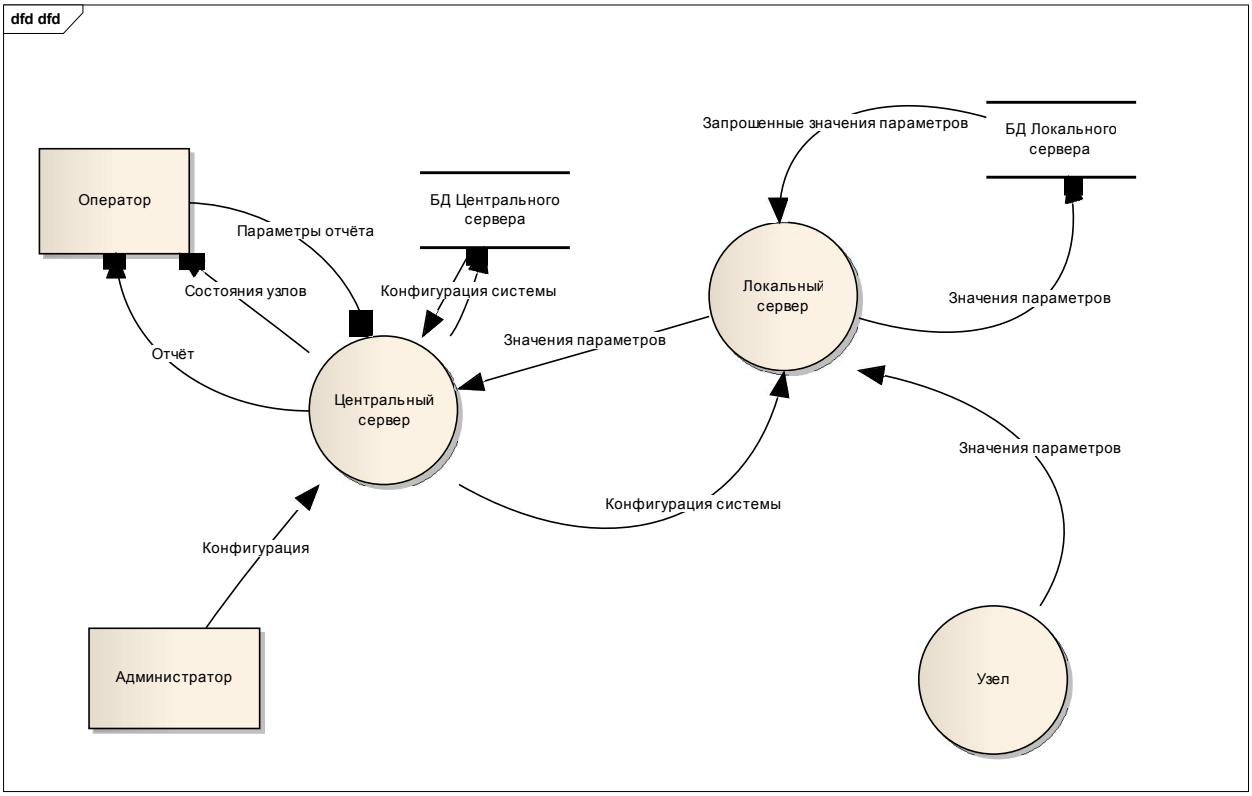

Рис. П3.1 Диаграмма потоков данных для СПО

Приложение 4

ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТИ ДЕЙСТВИЙ

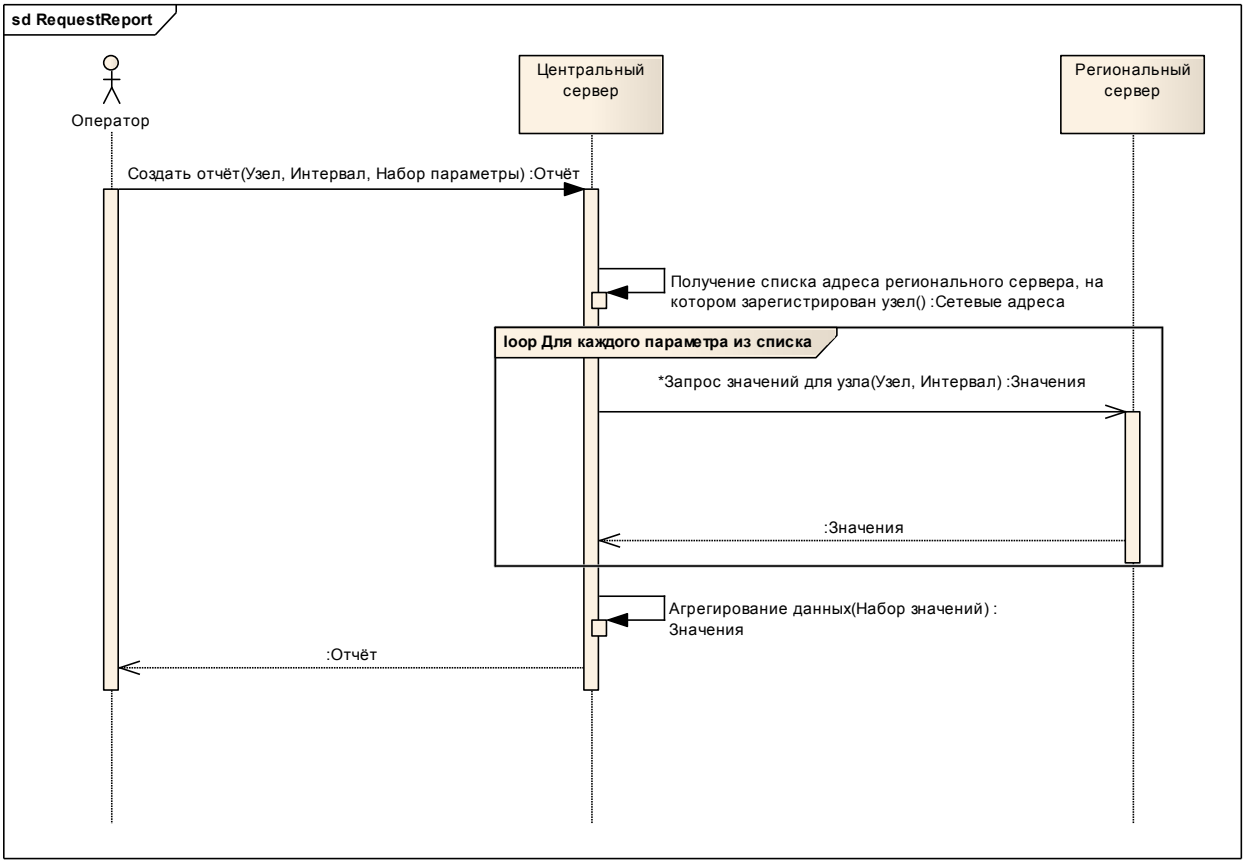

Рис. П4.1 Диаграмма последовательности действий: «Создание отчета о состоянии узла»

Приложение 5

ПРОЕКТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Выбор средства разработки интерфейса

Для разработки пользовательского Web-интерфейса в рамках технологии .NET использовались средства ASP.NET.

Выбор стратегии авторизации

В качестве стратегии авторизации выбрана авторизация на основе ролей. Для авторизации используется пара логин-пароль. Для осуществления авторизации не используются встроенные средства ASP.NET.

Авторизация в компонентах пользовательского интерфейса

После авторизации пользователь попадает на домашнюю страницу согласно его роли. Пользователь имеет доступ к функциям системы, соответствующим его роли. При попытке доступа к любой странице проверяется разрешение на просмотр страницы согласно роли пользователя.

Пользовательские интерфейсы и пользовательские процессы

Для реализации независимости в ресурсах при создании пользовательского интерфейса выбрана концепция MVC и использован MVC-каркас на основе ASP.NET. Таким образом, происходит разделение данных, их представления и пользовательских процессов.

Образцы прототипов пользовательского интерфейса

На рисунке П5.1 показан прототип интерфейса для авторизации пользователя, созданный в соответствии с ТЗ. Страница «Авторизация» предназначена для получения доступа к системе. Страница предполагает ввод логина и пароля. Все поля текстовые.

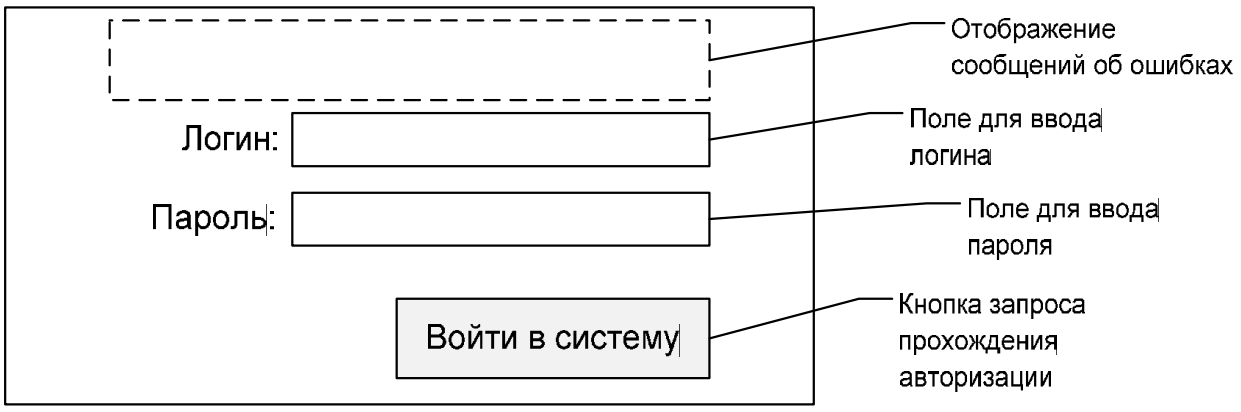

Рис. П5.4 Прототип страницы предназначенной для авторизации

На рисунке П5.2 показан прототип интерфейса для добавления нового пользователя в систему, созданный в соответствии с ТЗ. Страница предполагает ввод следующих полей: логин, пароль, подтверждение пароля, имя, фамилия и тип пользователя. Все поля текстовые, если не указано другое.

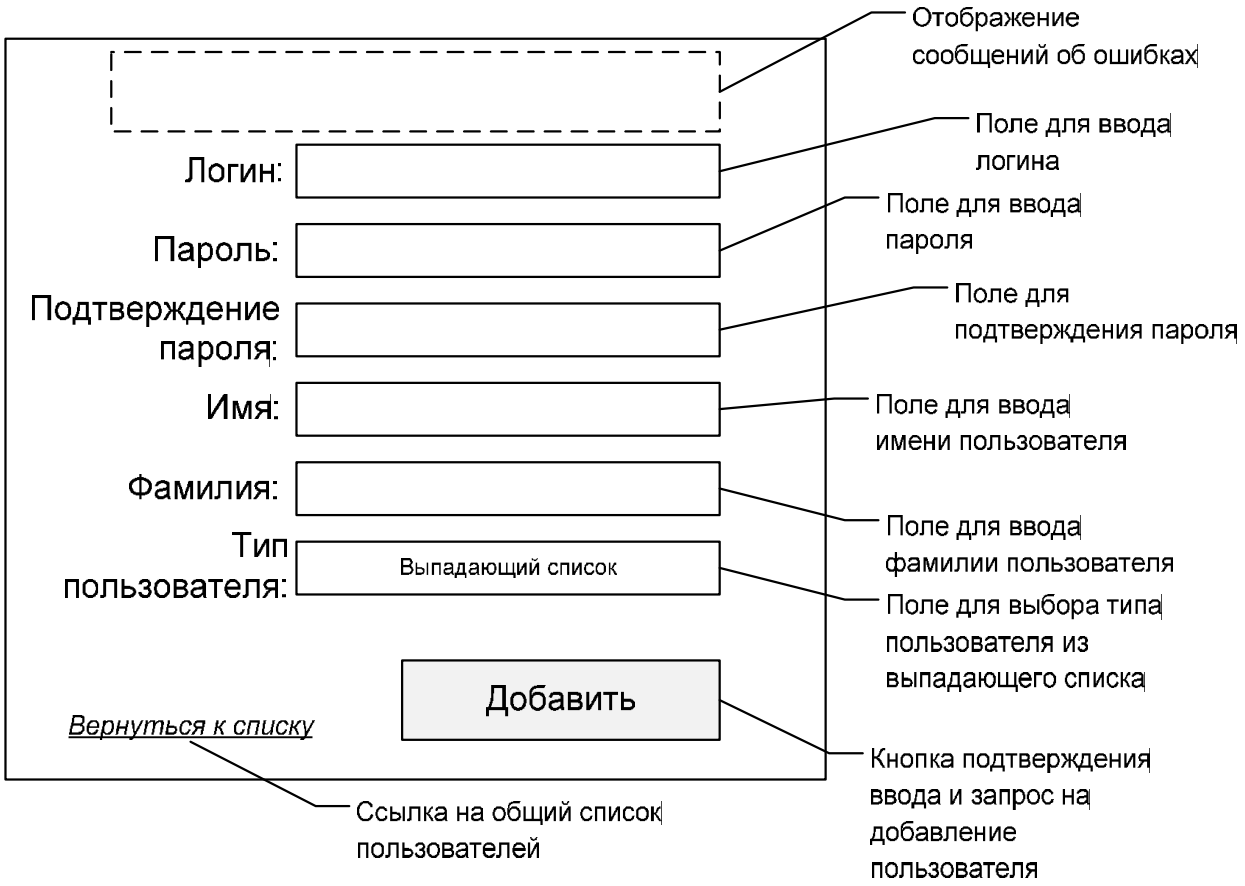

Рис. П5.5 Прототип страницы предназначенной для добавления нового пользователя в систему

Организация помощи пользователю

- Встроенная справочная система не предусмотрена в соответствии с требованиями заказчика.
- Организация всплывающих подсказок. При попадании фокуса ввода в поле должно выводиться облако-подсказка о формате ожидаемых данных.
- Организация защиты от некорректного ввода входных данных. Если в полях формы обнаружены ошибки, то информация об этом отображается в поле «Отображение сообщений об ошибках».

Приложение 6

ТЕСТИРОВАНИЕ WEB-ПРИЛОЖЕНИЯ

1. Базовое тестирование

Для базового тестирования использовалась интегрированная в Microsoft Visual Studio 2008 среда тестирования.

1.1. Модульное тестирование

Будем тестировать класс NodeController с использованием разбиения методов класса на категории по функциональности [1]. Категория объединяет в себе методы класса, имеющие близкую по смыслу функциональность. Методы класса можно разбить на две категории по функциональности:

- методы класса, используемые для заполнения шаблонов отображений;
- методы класса, используемые для манипуляции с данными: добавление, удаление и модификация диагностируемых узлов.

Пример оформления результатов модульного тестирования приведен в табл. П6.1.

Таблица П6.1. Результаты модульного тестирования

Номер теста	W-1
Название теста	DeleteTest
Тестируемый метод	NodeController.Delete
Описание теста	Проверка удаления сервера с индексом 0 из списка региональных серверов
Степень важности ошибки	Фатальная
Ожидаемый результат	Сервер с индексом 0 удаляется из списка региональных серверов и пользователь перенаправляется на домашнюю страницу
Результат теста	Тест пройден

Результаты автоматического тестирования сохраняются средой Microsoft Visual Studio в отдельный файл в формате XML. Для пользователя результаты тестирования представляются в виде:

Test Run: Administrator@MSTU-17A7F40456 2009-06-10 22:15:47
Test Name DeleteTest
Result Passed
Duration: 00:00:00.0056613
Computer Name: MSTU-17A7F40456
Start Time 6/10/2009 10:16:25 PM
End Time 6/10/2009 10:16:26 PM

Выводы по результатам модульного тестирования. Все тесты для класса NodeController пройдены успешно. Возможно использование класса NodeController,.

Текст тестового драйвера следующий:

```
[TestMethod()]
[HostType("ASP.NET")]
     // Корневая папка Web-сервера
[AspNetDevelopmentServerHost("..\\DS.MasterServer.Web", "/")]
[UrlToTest("http://localhost:61634/")]
public void DeleteTest()
{
        // За обработку запросов удаления диагностируемых
        // узлов отвечает контроллер класса NodeController
    NodeController target = new NodeController();
        // Номер удаляемого узла
    int id = 0;
    ActionResult expected = RedirectToAction("Index");
        // Выполняем запрос на удаление
    ActionResult actual = target.Delete(id);
        // Критерий прохождения теста.
        // Первый параметр – ожидаемый результат
        // Второй параметр – реальный результат
        // Ожидается, что если после успешного удаления узла
        // пользователь будет перенаправлен на домашнюю страницу
    Assert.AreEqual(expected, actual);
}
```

1.2. Интеграционное тестирование

Из диаграмм классов, взаимодействия и сотрудничества можно выделить набор методов, посредством которых осуществляется взаимодействие объектов заданных классов между собой.

Будем тестировать взаимодействие контроллера администратора AdminController с моделью пользователя User.

Таблица П6.2. Результаты интеграционного тестирования

Номер теста	W-2
Название теста	IndexTest
Названия взаимодействующих классов	AdminController, User
Описание теста	Проверяется, что пользователю, который не является администратором, отказывается в доступе к домашней странице администратора
Степень важности ошибки	Фатальная
Ожидаемый результат	Пользователь будет перенаправлен на страницу «Доступ запрещен»
Результат теста	Тест пройден

Выводы по результатам интеграционного тестирования. Взаимодействие классов AdminController и User соответствует разработанным диаграммам взаимодействия и сотрудничества (все тесты пройдены успешно).

Приведем исходный текст тестового драйвера:

```
[TestMethod()]
[HostType("ASP.NET")]
    // Корневая папка Web-сервера
[AspNetDevelopmentServerHost("..\\DS.MasterServer.Web", "/")]
[UrlToTest("http://localhost:61634/")]
```

```
public void IndexWhenNotLoginedTest()
{
        // Создаем контроллер
    AdminController target = new AdminController();
        // Результат работы контроллера
    ActionResult actual;
        // Получение страницы
    actual = target.Index();
        // Критерий прохождения теста.
        // Первый параметр – ожидаемый результат
        // Второй параметр – реальный результат
        // Ожидается, что если авторизации не было, то
        // возвращается страница «Доступ запрещен»
    Assert.AreEqual(View("Forbidden"), actual);
}
```

2. Тестирование пользовательских функций

2.1. Тестирование функциональных возможностей

На данном этапе проверяется соответствие реализованного приложения техническому заданию. Набор тестов получают из диаграммы прецедентов, выделяя для каждого прецедента классы эквивалентности.

Классом эквивалентности называется совокупность входных параметров, приводящих систему в одно и то же состояние.

Наиболее простой способ выделения таких классов можно сделать на основании описанных сценариев: основной и каждый альтернативный сценарий образуют классы эквивалентности. Пример выделения классов эквивалентности для прецедентов авторизации и регистрации пользователей показан в табл. П6.3.

Таблица П6.3. Классы эквивалентности для прецедентов авторизации и регистрации пользователей

№ п/п	Прецедент	Описание класса эквивалентности	Состояние системы	Пример входных данных для данного класса эквивалентности
1	Авторизация пользователя	Известные системе пары «логин-пароль» (основной сценарий)	Аутентификация пользователя прошла успешно	Логин: admin Пароль: 123
2	Авторизация пользователя	Неизвестные системе пары «логин-пароль» (альтернативный сценарий)	Пользователь не аутентифицирован. Система выдает сообщение об ошибке аутентификации	Логин: admin Пароль: 1234 Логин: adm Пароль: 123 Логин: adm Пароль: 1234
3	Регистрация пользователя	Неизвестный системе логин с непустым паролем (основной сценарий)	Пользователь успешно добавлен	Логин: admin1 Пароль: 1234
4	Регистрация пользователя	Известный системе логин (основной сценарий)	Пользователь не добавлен. Система выдает сообщение об ошибке: «Пользователь с таким логином уже зарегистрирован»	Логин: admin

5	Регистрация пользователя	Неизвестный системе логин с пустым паролем (альтернативный сценарий)	Пользователь не добавлен. Система выдает сообщение об ошибке: «Пароль не может быть пустым»	Логин: admin1 Пароль: (не задан)

Затем для каждого класса эквивалентности каждого прецедента выполнено тестирование, т.е. минимально необходимое количество тестов равно числу выделенных классов эквивалентности. Пример оформления результатов тестирования представлен в табл. П6.4.

Таблица П6.4. Результаты тестирования функциональных возможностей

Номер теста	**Описание теста**	**Входные данные**	**Ожидаемые результаты**	**Реальные результаты**	**Результат теста**
W-3	Авторизация с логином и паролем, известным системе	Логин: admin Пароль: 123	Система переходит на домашнюю страницу пользователя, отображает сообщение с именем пользователя и кнопкой выхода из системы	Система перешла на домашнюю страницу пользователя, вывела имя пользователя в верхнем правом углу экрана рядом с кнопкой «Выйти»	Тест пройден
W-4	Авторизация с логином и паролем, неизвестным системе	Логин: admin Пароль: 123	Сообщение: «Авторизация не выполнена. Проверьте правильность логина и пароля»	Сообщение: «Авторизация не выполнена. Проверьте правильность логина и пароля»	Тест пройден
W-5	Авторизация с логином и паролем, неизвестным системе	Логин: adm Пароль: 1234	Сообщение: «Авторизация не выполнена. Проверьте правильность логина и пароля»	Сообщение: «Авторизация не выполнена. Проверьте правильность логина и пароля»	Тест пройден
W-6	Авторизация с логином и паролем, неизвестным системе	Логин: adm Пароль: 123	Сообщение: «Авторизация не выполнена. Проверьте правильность логина и пароля»	Сообщение: «Авторизация не выполнена. Проверьте правильность логина и пароля»	Тест пройден
W-7	Регистрация пользователя	Логин: admin1 Пароль: (не задан)	Сообщение: «Пароль пользователя не может быть пустым»	Сообщение: «Пароль пользователя не может быть пустым»	Тест пройден
W-8	Регистрация пользователя	Логин: admin1 Пароль: 1234	Сообщение: «Пользователь успешно добавлен» на странице авторизации»	Сообщение: «Пользователь успешно добавлен» на странице авторизации»	Тест пройден

W-9	Регистрация пользователя	Логин: admin	Сообщение: «Пользователь с таким именем уже существует»	Сообщение: «Пользователь с таким именем уже существует»	Тест пройден

2.2. Тестирование практичности

Тестирование проводилось с помощью анкетирования целевой аудитории, состоящей из 50 человек. Потенциальным пользователям предлагалось оценить аспекты практичности по условной пятибалльной шкале, у которой большая оценка соответствует лучшему значению параметра, и заполнить анкету, представленную в табл. П6.5.

Таблица П6.5. Анкета для оценки практичности Web-приложения

№ п/п	Контролируемые характеристики	Информация о характеристике	
		Техническая информация	**Оценка**
1	Наименование задачи	Добавление нового узла	
2	Успешное выполнение задачи	задача не выполнена (0) или задача выполнена (1)	0 или 1
3	Время выполнения задачи	указать время, с	От 0 до 5
4	Число страниц, к которым нужен был доступ для завершения задачи	указать число страниц	От 0 до 5
5	В каких местах пользователь сталкивался с трудностями или где он делал ошибки	указать места, вызвавшие трудности и ошибки сталкивался	От 0 до 5
6	Как пользователь искал помощи в случае неудач	указать способ поиска помощи	От 0 до 5
7	Предоставляет ли интерактивная справка достаточное количество информации	указать, предоставляет ли интерактивная справка достаточное количество информации	От 0 до 5
8	Щелкал ли пользователь по страницам или использовал возможность поиска	указать способ поиска информации	От 0 до 5
9	Как пользователь реагировал на время загрузки отдельных страниц	указать приблизительное среднее время загрузки страниц	От 0 до 5
10	Количество щелчков для выполнения задачи, время между щелчками	указать приблизительное число щелчков (в т.ч. клавишей tab, shift+tab)	От 0 до 5
11	Навигация по сайту	-	От 0 до 5

Результаты статистической обработки анкет представлены в табл. П6.6.

Таблица П6.6. Результаты статистической обработки анкетных данных

№ п/п	Тест	Информация о характеристике		Требуемое значение средней оценки, не менее	Результат теста
		Средняя количественная оценка	Средняя оценка		
Добавление нового узла (основной сценарий)					
W-10	Успешное выполнение задачи добавления нового узла	-	1	1	Тест пройден
W-11	Время выполнения задачи	36,3 с	4,27	4	Тест пройден
W-12	Число страниц, к которым нужен был доступ для завершения задачи	4	4,41	4	Тест пройден
W-13	В каких местах пользователь сталкивался с трудностями или где он делал ошибки	-	4,71	4	Тест пройден
W-14	Как пользователь искал помощи в случае неудач	-	4,19	4	Тест пройден
W-15	Предоставляет ли интерактивная справка достаточное количество информации	-	0	0	Тест пройден
W-16	Щелкал ли пользователь по страницам или использовал возможность поиска	-	0	0	Тест пройден
W-17	Как пользователь реагировал на время загрузки отдельных страниц	560-840 мс (по журналу Web-сервера)	5	4	Тест пройден
W-18	Количество щелчков для выполнения задачи, время между щелчками	8 щелчков, 4,3 сек	4,55	4	Тест пройден
W-19	Навигация по сайту	-	4,12	4	Тест пройден

2.3. Тестирование форм

Для тестирования граничных значений и проверки вводимых данных выделены классы эквивалентности, представленные в табл. П6.7.

Таблица П6.7. Классы эквивалентности для входных данных формы «Добавление нового регионального сервера»

№ п/п	Прецедент	Описание класса эквивалентности	Состояние системы	Пример входных данных для данного класса эквивалентности
1	Добавление нового регионального сервера	Адрес сервера, заданный в формате http://имя_сервера [:номер_порта] (основной сценарий)	Новый сервер успешно добавлен	Адрес сервера: http://localhost:6000
2	Добавление нового регионального сервера	Адрес сервера, не соответствующий формату http://имя_сервера [:номер_порта] (основной сценарий)	Сервер не добавлен. Система выдает сообщение о некорректном адресе сервера	Адрес сервера: афюфужу
3	Добавление нового регионального сервера	Адрес сервера не задан (пустая строка)	Сервер не добавлен. Система выдает сообщение о том, что адрес сервера должен быть задан обязательно	Адрес сервера: (не задан)

Пример оформления результатов тестирования для формы «Добавление нового регионального сервера» в табл. П6.8.

Таблица П6.8. Результаты тестирования формы «Добавление нового регионального сервера»

Номер теста	Описание теста	Входные данные	Ожидаемые результаты	Реальные результаты	Результат теста
Тестирование табуляции					
W-18	Добавление нового сервера	Нажатие на кнопку tab в поле «Имя сервера»	Курсор перейдет в поле «Сетевой адрес»	Курсор перешел в поле «Сетевой адрес»	Тест пройден
W-19	Добавление нового сервера	Нажатие на сочетание клавиш shift+tab в поле «Сетевой адрес»	Курсор перейдет в поле «Имя сервера»	Курсор перешел в поле «Имя сервера»	Тест пройден
Тестирование граничных значений					
W-20	Добавление нового сервера	Поле «Имя сервера»: srv02 Поле «Сетевой адрес»: (не задано) Поле «Описание»: (не задано)	Сообщение: «Поле «Адрес сервера» должно быть заполнено»	Сообщение: «Поле «Адрес сервера» должно быть заполнено»	Тест пройден
Тестирование проверки вводимых данных					
W-21	Добавление нового сер-	Поле «Имя сервера»: srv02	Сообщение: «Поле «Адрес серве-	Сообщение: «Поле «Адрес серве-	Тест пройден

	вера	Поле «Сетевой адрес»: «афюфужу» Поле «Описание»: (не задано)	ра» должно быть заполнено корректным адресом в формате http://имя_сервера [:номер_порта]»	ра» должно быть заполнено корректным адресом в формате http://имя_сервера [:номер_порта]»	
Тестирование обновления формами информации					
W-22	Добавление нового сервера	Поле «Имя сервера»: srv02 Поле «Сетевой адрес»: «http://localhost:6000» Поле «Описание»: (не задано)	Сервер появился в списке серверов на форме «Список региональных серверов»	Сервер появился в списке серверов на форме «Список региональных серверов»	Тест пройден

2.4. Тестирование содержимого страницы

Пример оформления результатов тестирования для страницы «Список региональных серверов» представлен в табл. П6.9.

Таблица П6.9. Отчет о тестировании содержимого страницы «Список региональных серверов»

Номер теста	**Описание теста**	**Входные данные**	**Ожидаемые результаты**	**Реальные результата**	**Результат теста**
Тестирование корректного отображения страницы в разных браузерах					
W-25	Просмотр формы «Список региональных серверов»	Операционная система:Win XP Prof 2007 Corp Web-браузер: Internet Explorer 7.0	Заголовок приложения: в левом верхнем углу, имя пользователя и кнопка «Выйти» в правом верхнем углу, кнопка перехода на главную страницу: справа, под именем пользователя, граница формы: выровнена по центру, заголовок формы: в левом верхнем углу формы, таблица серверов: выровнена по ширине формы	Заголовок приложения: в левом верхнем углу, имя пользователя и кнопка «Выйти» в правом верхнем углу, кнопка перехода на главную страницу: справа, под именем пользователя, граница формы: выровнена по центру, заголовок формы: в левом верхнем углу формы, таблица серверов: выровнена по ширине формы	Тест пройден
W-26	Просмотр формы «Список региональных серверов»	Операционная система: Ubuntu 8.04 Web-браузер:	Заголовок приложения: в левом верхнем углу, имя пользо-	Заголовок приложения: в левом верхнем углу, имя пользователя и	Тест пройден

		Mozilla Firefox 3.1	вателя и кнопка «Выйти» в правом верхнем углу, кнопка перехода на главную страницу: справа, под именем пользователя, граница формы: выровнена по центру, заголовок формы: в левом верхнем углу формы, таблица серверов: выровнена по ширине формы	кнопка «Выйти» в правом верхнем углу, кнопка перехода на главную страницу: справа, под именем пользователя, граница формы: выровнена по центру, заголовок формы: в левом верхнем углу формы, таблица серверов: выровнена по ширине формы	
Тестирование соответствия содержимого страницы требованиям ТЗ					
W-27	Просмотр формы «Список региональных серверов»	Операционная система: Win XP Prof 2007 Corp Web-браузер: Internet Explorer 7.0	О каждом сервере представлена следующая информация: имя сервера, адрес сервера, описание сервера, время добавления записи	О каждом сервере представлена следующая информация: имя сервера, адрес сервера, описание сервера, время добавления записи	Тест пройден
Тестирование ссылок на важное содержимое внутри и снаружи узла					
W-28	Редактирование записи о сервере	Нажатие на ссылку «Редактировать»	Переход на форму «Редактирование узла»	Переход на форму «Редактирование узла»	Тест пройден
W-29	Удаление записи о сервере	Нажатие на ссылку «Удалить»	Отобразился запрос на подтверждение удаления, после подтверждения сервер удален из списка серверов	Отобразился запрос на подтверждение удаления, после подтверждения сервер удален из списка серверов	Тест пройден
Тестирование всплывающих подсказок					
W-30	Просмотр формы «Список региональных серверов»	Подведение курсора мыши к ссылкам и полям таблицы	Отображение подсказок с описанием полей и ссылок	Всплывающие подсказки не отображаются	Тест не пройден
Внешний вид формы: строгий, понятный, соблюдена цветовая гамма					
Орфография, грамматика, пунктуация: ошибок не обнаружено					

2.5. Тестирование надежности и доступности

В рамках тестирования надежности и доступности проводилось предельное тестирование. Тестирование проводилось с помощью продукта WebBench компании eTestingLabs [7]. Введем следующие обозначения:

λ, c^{-1} – интенсивность запросов, т.е. число запросов в единицу времени;

T, c – общее время тестирования;

$N = \lambda T$ – общее число посланных запросов;

$N_{отказов}$ – общее число отказов;

$t_0 = 3\,c$ – время восстановления Web-сервера после сбоя, повлекшего отказ в обработке запроса;

$t_1 = \dfrac{T}{N_{отказов}}$ – среднее время бесперебойной работы;

$\theta = \dfrac{t_1}{t_1 + t_0}$ – доступность сервера;

p – вероятность отказа сервера.

Результаты тестирования представлены в табл. *П6.10* .

Таблица П6.10. Пример оформления отчета о тестировании надежности и доступности

Номер теста	Описание теста	Входные данные	Ожидаемые результаты	Реальные результаты	Результат теста
Тестирование работоспособности Web-сервера					
W-41	Нагрузочное тестирование	$\lambda = 10\ c^{-1}$ $T = 1800\ \ c$ $N = 18000$ Тип запроса: запрос случайной страницы	$\theta \geq 99\%$ $p \leq 0{,}1\%$	$t_1 = 450\ c$ $\theta = 99{,}3\%$ $p = 0{,}02\%$	Тест пройден
W-42	Нагрузочное тестирование	$\lambda = 100\ c^{-1}$ $T = 900\ \ c$ $N = 90000$ Тип запроса: запрос случайной страницы	$\theta \geq 99\%$ $p \leq 0{,}1\%$	$t_1 = 112{,}5\ c$ $\theta = 97{,}4\%$ $p = 0{,}01\%$	Тест не пройден

Выводы по результатам тестирования пользовательских функций. Программа требует дальнейшей доработки для устранения выявленных при тестировании ошибок.

СПИСОК ЛИТЕРАТУРЫ

1. Вишневская Т.И., Романова Т.Н. Технология программирования: Мет. указания к лабораторному практикуму. - Ч. 1. – М: Изд-во МГТУ им. Н.Э. Баумана, 2007.
2. Анализ требований и создание архитектуры решений на основе Microsoft .Net. Учебный курс MCSD.-М.: Издательско-торговый дом «Русская редакция», 2004.
3. Байдачный С.С., Маленко Д.А. ASP.NET 2.0: секреты создания Web-приложений. – М.:СОЛОН-ПРЕСС, 2007.
4. Смит Конни, Уильямс Ллойд. Эффективные решения: практическое руководство по созданию гибкого и масштабируемого программного обеспечения. – М.: Издательский дом «Вильямс», 2003.
5. ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы».
6. Л. Тампе. Введение в тестирование программного обеспечения. – Киев: Издательский дом «Вильямс», 2003.
7. Тестирование производительности Web-серверов [Электрон. ресурс] / Д. Намиот, С. Рогов, osp.ru. [1992-2009]. Режим доступа: http://www.osp.ru/os/2002/12/182266/, свободный.

Методическое издание

Татьяна Ивановна Вишневская

Татьяна Николаевна Романова

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

Часть 2

Редактор

Корректор

Компьютерная верстка

Подписано в печать Формат 60x84/16. Бумага офсетная

Печ. л. Усл. Печ. л. Уч.-изд. л.

Тираж экз. Изд. № . Заказ №

Издательство МГТУ им. Н.Э. Баумана,

105005, Москва, 2-я Бауманская ул., 5