Введение

Образование необходимо всем, только иногда не хватает времени, денег или желания для дополнительного обучения. Тем, кто не хочет или не может учиться, помочь сложно, но если просто не хватает времени, то можно воспользоваться такими проверенными способами, как заочное или вечернее образование. А скоро к этому списку добавится еще один - дистанционное обучение через Internet. Казалось бы, тут может пригодиться модель заочного обучения, однако сеть имеет такие возможности, которые не в состоянии реализовать традиционная система образования. Впрочем, технологии, которая наиболее полно реализовывала бы возможности Internet, пока еще нет.

Обучение через Internet имеет два важных преимущества перед другими методами: это интерактивность и моделирование. Подключение к образовательному серверу в режиме реального времени позволяет вести живой диалог между учителем и учеником, а вычислительная мощность ПК, образующих сеть, помогает легче усваивать знания. Для объединения этих функций лучше всего подходит Java-технология. Основное ее достоинство в том, что приложения могут работать практически на любых платформах, максимально использовать сеть и мультимедийные возможности современных платформ.

Этими обстоятельствами решили воспользоваться разработчики представленной телекоммуникационной системы моделирования. Предлагаемая к рассмотрению телекоммуникационная система является альтернативой имитационным стендам, используемым в схемотехнических дисциплинах для освоения основ построения узлов вычислительной техники. Система обеспечивает моделирование цифровых схем при работе в среде Microsoft Internet Explorer 4.0 и выше. Поддерживает следующие основные функции: создание моделей, формирование схем с извлечением элементов из палитры функциональных узлов, формирование временных диаграмм, отражающих динамические процессы в моделируемых схемах, сохранение схем на сервере с поддержкой механизмов авторизации доступов. Система представляет собой интеграцию гипертекстовой и графической информации, а также программного кода на языке Java. К достоинствам данной системы можно отнести следующее: простота инсталляции и администрирования, русскоязычность интерфейса, простота овладения возможностями системы, неограниченные возможности по внедрению в систему компонентов помощи пользователю и интерактивных обучающих курсов.

1. Расширенная постановка задачи

Оценить возможность применения новейших информационных технологий в образовательном процессе ВУЗа. Разработать телекоммуникационную систему предназначенную для поддержки обучения основам схемотехники и контроля знаний по данной дисциплине.

Система должна обеспечивать следующие возможности:

- возможность ознакомления пользователей с теоретическими материалами, реализованными в гипертекстовом виде;
- реализация возможностей применения полученных теоретических знаний на практике – проектирование мелких цифровых устройств из предоставляемого набора элементов системы;
- моделирование работы созданных схем цифровых устройств;
- представление результатов моделирования в наглядном виде – в виде временных диаграмм динамических процессов;
- возможность тестирования и оценки полученных знаний;
- реализация эффективного обмена схемотехническими решениями между пользователями и преподавателем.

Система должна удовлетворять следующим требованиям:

- устойчивое функционирование в телекоммуникационной среде ВУЗа;
- поддержка функционирования в различных операционных системах (Windows 95/98/NT, Unix);
- простота администрирования и сопровождения;
- интуитивно-понятный интерфейс и простота овладения возможностями системы;
- минимальные требования к клиентскому программному обеспечению;
- максимальная интеграция пользователя в информационную среду ВУЗа.

2. Функциональная структура системы

Рассмотрим подробнее структуру системы. Она представляет собой комплекс взаимодействующих компонентов.

В данной системе серверу отводится роль хранилища компонентов программы, обучающих курсов, а также через него осуществляется доступ к базе накопленных схемотехнических решений (чтение, обновление, пополнение). Также сервер является носителем элементной базы, которая легко может пополняться. Это возможно на уровне языка Java. Особенности языка Java позволяют реализовывать и внедрять в систему классы новых элементов без перекомпилляции всей системы. Поэтому создание текстового описания элемента, как это принято во многих системах автоматизированного проектирования, не требуется. И новый элемент, внедряемый в систему, представляет собой класс, в котором заложены и функциональные характеристики элемента и методы его отображения на экране. Это свойство позволяет ускорить некоторые функциональные процессы программы, например, моделирование работы созданной схемы. Также, благодаря этому свойству, можно разрабатывать и внедрять в систему элементы довольно сложной структуры, например, целые функциональные узлы (счетчики, регистры, микропроцессоры и т.д.), не прибегая к громоздкому текстовому описанию функциональных особенностей этих компонентов.

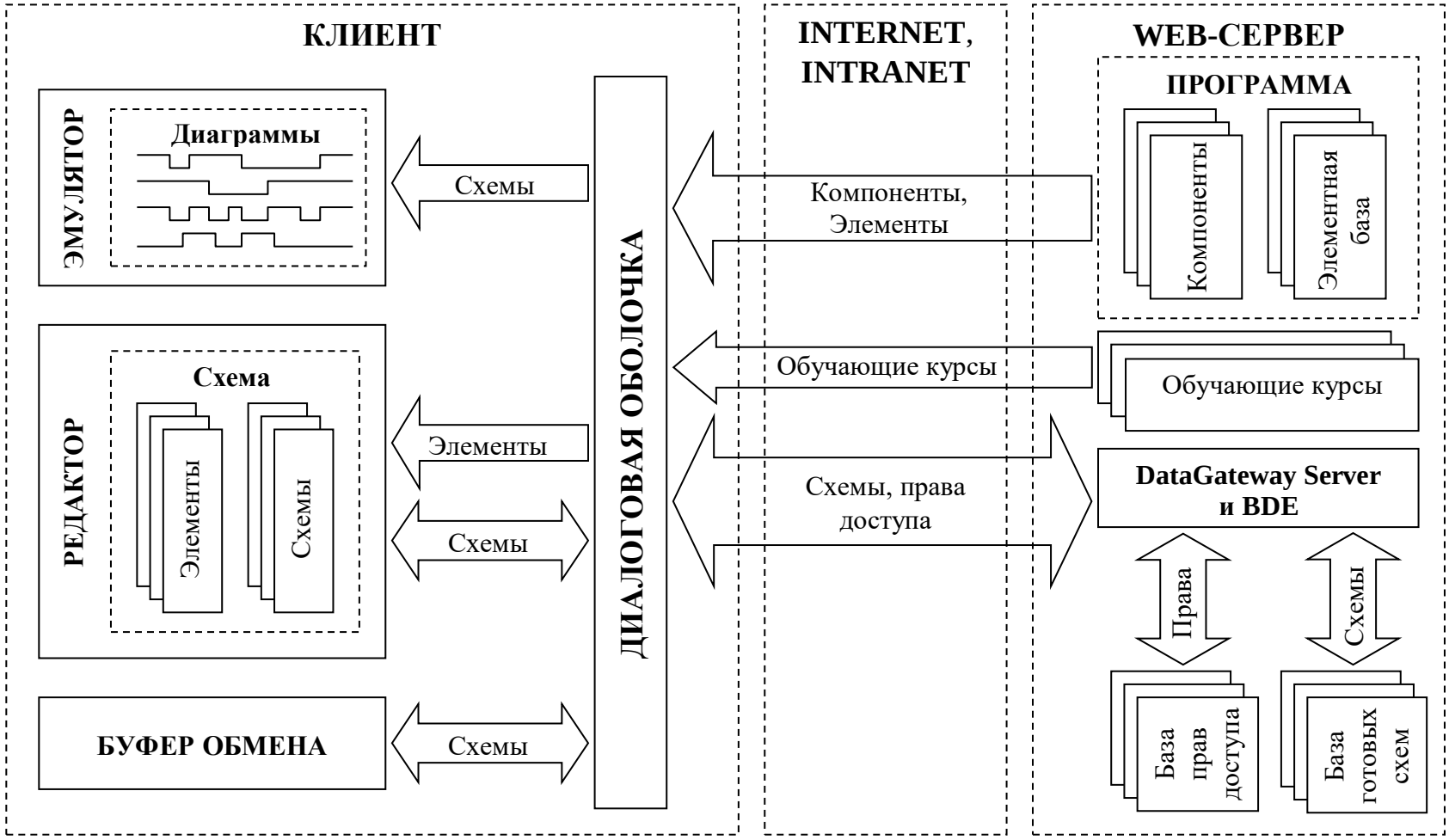
Все основные действия: разработка и редактирование схем, моделирование их работы – отводится клиентской машине. Для этого в системе реализован редактор, предоставляющий необходимые возможности по редактированию и построению схем. И эмулятор работы схемы – наглядно отображающий результаты моделирования. На данный момент результатом моделирования работы построенной схемы является набор временных диаграмм. Состояния, отображаемые на временных диаграммах, снимаются в любых точках схемы, предварительно отмеченных специальными маркерами. Данный прием довольно стандартен для современных систем автоматизированного проектирования электронных компонентов. На временных диаграммах возможно отследить состояние моделируемого узла в любой квант времени. Отображение результатов моделирования в виде временных диаграмм является не единственным способом представления полученных данных. Пользователь также имеет возможность наблюдать состояния всех контактов элементов в нужный квант времени моделирования, используя определенные опции программы.

Связь между клиентской машиной и сервером осуществляется через Internet, Intranet сеть, используя навигатор Microsoft Internet Explorer 4.x или выше.

Особо значащим звеном системы является «мост», связывающий клиентскую машину с Web-сервером. Через него осуществляется доступ к базе данных готовых схемотехнических решений и к базе данных, в которой содержится информация о правах доступа пользователей. По этому мосту осуществляется движение информации в обоих направлениях, как в базу данных сервера, так и из базы на клиентскую машину. Можно заметить, что от производительности данного звена зависит надежность работы всей системы. Иначе, убрав его, использование всей системы будет практически нецелесообразно. Поэтому значительная часть времени была уделена разработке именно этой части системы. Решено было использовать для связи с выше перечисленными базами данных технологию фирмы Borland – Borland Database Engine. Обоснование выбора использования технологии связи клиента с базами данных на сервере приведено в соответствующей главе. Данная технология позволяет осуществлять доступ ко многим широко распространенным типам баз данных, а также подключаться к серверам баз данных других производителей. Использование данного подхода позволило базу готовых схемотехнических решений и базу прав доступа пользователей представлять в виде файлов данных формата FoxPro. Связь с Borland Database Engine, установленной на сервере, и клиентской машиной осуществляется с помощью DataGateway Server посредством стандартного JDBC интерфейса (набора классов производства той же Borland Int). Перечисленные компоненты (Borland Database Engine, Borland DataGateway Server и JDBC-классы) поставляются в пакете Borland DataGateway и легко устанавливаютя на любую машину. Таким образом решается проблема доступа к базам данных в разработанной системе.

С основными функциональными компонентами системы и связями между ними можно ознакомиться на схеме, представленной ниже:

ФУНКЦИОНАЛЬНАЯ СТРУКТУРА СИСТЕМЫ



3. Исследование технологий доступа к базам данных в сетях Internet/Intranet. Выбор технологии и ее реализация в системе.

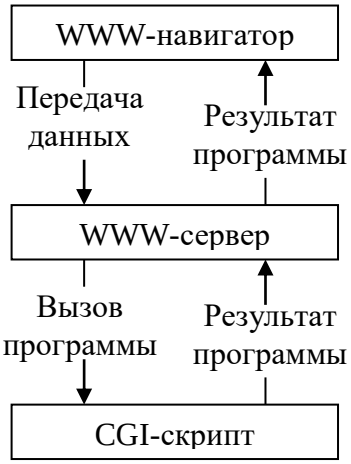
3.1. Средства доступа к базам данных на стороне сервера

3.1.1. CGI

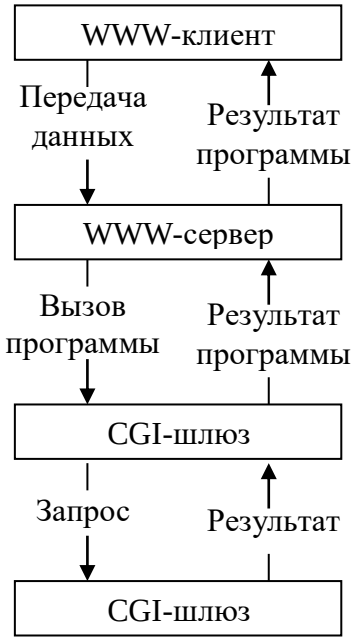
Common Gateway Interface - это спецификация интерфейса взаимодействия Web-сервера с внешними прикладными программами. Главное назначение CGI - обеспечение единообразного потока данных между сервером и работающим на нем приложением. CGI определяет:

1. порядок взаимодействия сервера с прикладной программой, в котором сервер выступает инициирующей стороной;
2. механизм реального обмена данными и управляющими командами в этом взаимодействии, что не определено в протоколе HTTP. Такие понятия, как метод доступа, переменные заголовка, MIME, типы данных, заимствованы из HTTP и делают спецификацию прозрачной для тех, кто знаком с самим протоколом.

Обычно гипертекстовые документы, возвращаемые по запросу клиента WWW сервером, содержат статические данные. CGI обеспечивает средства создания динамических Web-страниц на основе данных, полученных от пользователя. Программы, написанные в соответствии со спецификацией CGI, называются CGI-скриптами или шлюзами. Шлюз - это CGI-скрипт, который используется для обмена данными с другими информационными ресурсами Internet или приложениями-демонами такими, как, например, система управления базами данных. Обычная CGI-программа запускается Web-сервером для выполнения некоторой работы, возвращает результаты серверу и завершает свое выполнение.



Шлюз выполняется точно также, только, фактически, он инициирует взаимодействие в качестве клиента с третьей программой. Если эта третья программа является сервером БД, то шлюз становится клиентом СУБД, который посылает запрос по определенному порту соединения с системой управления базами данных, а после получения ответа пересылает его WWW-серверу.



Обмен данными по спецификации CGI реализуется обычно через переменные окружения и стандартный ввод/вывод. Выбор механизма передачи параметров определяется методом доступа, который указывается в форме в атрибуте METHOD. Если используется метод GET, то передача параметров происходит с помощью переменных окружения, которые сервер создает при запуске внешней программы. Через них передается приложению как служебная информация (версия программного обеспечения, доменное имя сервера и др.), так сами данные (в переменной QUERY_STRING). При методе POST для передачи используется стандартный ввод. А в переменных окружения фиксируется тип и длина передаваемой информации (CONTENT_TYPE и CONTENT_LENGTH).

Стандартный вывод используется скриптом для возврата данных серверу. При этом вывод состоит из заголовка и собственно данных. Результат работы скрипта может передаваться клиенту без каких-либо преобразований со стороны сервера, если скрипт обеспечивает построение полного HTTP-заголовка, в противном случае сервер модифицирует заголовок в соответствии со спецификацией HTTP. Обязательным для скриптов при генерировании документов «на лету», когда реального документа в

файловой системе сервера не остается является только HTTP-заголовок *Content-type*, в котором указывается тип возвращаемого документа для правильной интерпретации браузером. Обычно в Content-type указывают текстовые типы text/plain и text/html. При использовании такого вида скриптов следует учитывать, что не все серверы и клиенты отрабатывают так, как представляется разработчику скрипта. Так, при указании Content-type: text/html, некоторые клиенты не реализуют сканирования полученного текста на предмет наличия в нем встроенной графики.

При применение спецификации CGI для обмена данными с внешними прикладными программами можно выделить следующие преимущества:

- Прозрачность использования;
- «Языковая» независимость - CGI-программы могут быть написаны на любом языке программирования или командном языке, имеющим средства работы со строками;
- Процессная изолированность - при запуске CGI-програмы на сервере порождается отдельный процесс и ошибочный CGI-скрипт не может сломать Web-сервер или получить доступ к закрытой информации;
- Открытость стандарта - CGI интерфейс применим на каждом Web-сервере;
- Архитектурная независимость - CGI не зависит от особенностей реализации архитектуры сервера (однопоточности, многопоточности и т.д.);

Но CGI имеет также и существенные недостатки. Главная проблема заключается в затратах на выполнение CGI-приложений: поскольку на сервере для каждого очередного запроса порождается новый процесс, который завершается после его выполнения, то это приводит к невысокому быстродействию CGI-скрипта и снижает эффективность работы сервера. При использовании CGI-программ для доступа к базам данных из-за неподдержки непрерывного соединения Web-сервера и соответствующей СУБД очень сложно произвести процесс «ведения» пользователя базой данных, так как каждый раз при генерации очередного запроса требуется новое подключение. Но в то же время закрытие соединения после обработки каждого запроса сильно осложняет деятельность хакеров, так как при отсутствии постоянного подключения к БД проникнуть в нее гораздо сложнее. Другое достоинство этого «недостатка» состоит в том, что связь с Web-сервером устанавливается только на короткий промежуток времени, в результате чего он не перегружается и может выполнять другие задачи.

CGI также ограничен по способности функционирования - спецификация предусматривает только простую «ответную» роль скрипта при генерации результата на запрос пользователя. CGI-программы не имеют взаимосвязей с установлением аутентификации пользователя и проверки его входных данных.

3.1.2. API

В ответ на ограничения и недостатки спецификации CGI была разработана спецификация прикладных модулей API, встроенных в сервер. Данное расширение Web-сервера запускается как динамическая библиотека и выполняет обработку каждого вызова сервера по отдельной структуре памяти, что значительно проще, чем создание отдельного процесса для каждого клиентского запроса. Наиболее известны два API-интерфейса - NSAPI компании Netscape и ISAPI компании Microsoft. Свободно распространяемый популярный Unix-сервер Apache также имеет модуль PHP, реализующий данный интерфейс. Приложения, работающие через API, соединяются с сервером значительно быстрее, чем CGI-программы, так как API выполняется в основном процессе сервера и постоянно находится в состоянии ожидания запросов, поэтому время на запуск программы и порождения нового процесса не требуется. API-интерфейс предоставляет и большую функциональность, чем CGI - можно написать дополнительные процедуры, осуществляющие контроль доступа к файлам, получающие доступ к log-файлам сервера и связывающиеся с другими этапами обработки запроса сервером.

Тем не менее спецификация API не имеет преимуществ CGI-интерфейса и поставщики API-модулей тоже сталкиваются с целым рядом проблем:

- «Языковая» зависимость - прикладные программы могут быть написаны только на языках, поддерживаемых в данном API (обычно это C/C++);Perl, наиболее популярный язык для CGI-скриптов, как правило, не используется в существующих поставляемых API-модулях.
- Неизолированность процесса - так как приложения выполняются в адресном пространстве сервера, то ошибочные программы могут «уронить» сервер или какое-либо приложение. Таким образом вполне возможно (намеренно или нет) сломать систему безопасности сервера.

- Ограниченность применения - написанные программы в соответствии с данным API могут использоваться только на данном сервере.
- Архитектурная зависимость - API-приложения зависимы от архитектуры сервера: если сервер поддерживает однопоточность, то многопотоковые приложения не получают никакого преимущества в быстродействии при выполнении. Также при изменении производителем архитектуры сервера, модуль API обычно тоже подвергается изменениям, и прикладные программы соответственно тоже требуют переделки или даже могут быть написаны заново.

3.1.3. FastCGI

Интерфейс FastCGI сочетает в себе наилучшие аспекты спецификаций CGI и API. Взаимодействие в соответствии с FastCGI происходит сходным образом с CGI. FastCGI-приложения запускаются отдельными изолированными процессами. Отличие состоит в том, что эти процессы являются постоянно работающими и после выполнения запроса не завершаются, а ожидают новых запросов. Вместо использования переменных окружения операционной системы и стандартных потоков ввода/вывода протокол FastCGI объединяет информацию среды, стандартный ввод, вывод и сообщения об ошибках в единственное дуплексное соединение. Это позволяет FastCGI-программам выполняться на удаленных машинах, используя TCP-соединения между Web-сервером и FastCGI-модулем.

Таким образом, преимущества FastCGI состоят в следующем:

- Быстродействие - благодаря постоянному функционированию FastCGI-процессов обеспечивается обслуживание одним процессом многих запросов, что решает задачу и связанные с ней проблемы порождения нового процесса на отдельный клиентский запрос.
- Простота применения и легкость миграции из CGI.
- «Языковая» независимость - как и CGI, FastCGI-приложения могут быть написаны на любых языках программирования или командных языках.
- Изолированность процессов – «неисправные» FastCGI-программы не могут разрушить ядро сервера или какие-

либо другие приложения, а также получить секретную служебную информацию.

- Совместимость - FastCGI поддерживается во всех открытых продуктах, включая коммерческие серверы Netscape и Microsoft, NCSA сервер и свободно распространяемый Apache.
- Архитектурная независимость - FastCGI интерфейс не зависит от особенностей реализации серверной архитектуры и прикладные программы могут быть как одно-, так и многопоточными.
- Распределенность - FastCGI обеспечивает возможность выполнять приложения удаленно, что используется для распределенной загрузки и управления внешними Web-сайтами.

3.2. Доступ к базам данных на стороне клиента Java-технология

Аналитики предсказывают в ближайшие несколько лет бурный рост рынка Intranet. На приложения для этих сетей будет истрачено в четыре раза больше средств, чем для приложения Internet. По оценкам Forrester Research, в 1999 году суммарные затраты на Intranet и Internet превысят 9 млрд. долларов.

Однако чтобы заставить технологию Intranet работать в полную силу, нужен язык программирования, с помощью которого компании могли бы приспособить под свои конкретные задачи приложения для внутрикорпоративных сетей. Главное, что этот язык должен иметь средства доступа к базам данных. «Бурный рост Internet и Intranet рождает необходимость в протоколе баз данных, способном обеспечить клиент-серверные СУБД характерными для сегодняшних приложений в архитектуре клиент-сервер функциональными возможностями и удобством использования», - сказал Боб Перро, вице-президент по разработкам Visigenic.

На эту роль есть серьезный претендент. Язык Java, разработанный подразделением Sun Microsystems, компанией JavaSoft, обладает всеми необходимыми качествами, чтобы стать (если этого до сих пор не случилось) своеобразным эсперанто для этого сектора рынка. Благодаря своему интерфейсу Database Connectivity API, Java позволяет обращаться к базам данных на естественном языке.

Первая версия Java не имела вообще никаких средств работы с базами данных, что давало повод к справедливым сомнениям в

необходимости языка вообще. Однако вскоре JavaSoft опубликовала предварительный стандарт интерфейса прикладного программирования для баз данных под названием Java Database Connectivity (JDBC). В окончательную версию языка, Java 1.1, включена поддержка спецификации JDBC.

«Современные решения для Internet и Web предполагают работу с приложениями, которые поддерживают статичные тексты и графику, - поясняет Перро. - Java обогащает Internet интерактивными приложениями, но до сих пор этот язык не имел встроенных средств взаимодействия с базами данных. Теперь же, при помощи новых продуктов, через Internet можно обращаться к хранящейся в базах данных корпоративной информации. Доступ контролируется непосредственно логикой приложений Java, таким образом, он больше не зависит от статичного представления данных в формате HTML, как это было в пору медленных серверов CGI».

JDBC представляет собой стандартный интерфейс доступа к базам данных SQL. Компания обнародовала версию 1.0 JDBC API в июне 1996 года. Версия 1.1, распространяемая сейчас, входит в комплект Java Developer's Kit 1.1. Программное обеспечение JDBC API предоставляет программистам на Java единый интерфейс доступа к многочисленным реляционным базам данных и общую платформу для разработки высокоуровневых приложений и интерфейсов. Ведущие производители баз данных, средств доступа и инструментов уже взяли JDBC на вооружение и применяют его при разработке своих продуктов. Многие предлагают драйверы JDBC для организации взаимодействия Java с разнообразными СУБД.

Подключение базы данных к Intranet сулит массу преимуществ. Эти преимущества столь велики, что некоторым менеджерам по разработке они видятся просто безграничными - конечно, в пределах отведенного бюджета. Первое же приложение, разработанное одним из таких неофитов менее чем за месяц, открыло членам его группы доступ к необходимой им для работы основной базе данных.

Тех же, кто хочет получить дополнительное подтверждение технологических и финансовых преимуществ модели Intranet по сравнению с традиционной клиент-серверной, можно отослать к опыту многочисленных администраторов, имевших дело даже с достаточно удачными проектами систем в архитектуре клиент-сервер. По их мнению, разработка таковых слишком часто оканчивается плачевно. Опубликованные результаты исследований свидетельствуют, что более 50% проектов разработки клиент-серверных систем либо не увенчались успехом, либо были закрыты

до окончания работ, либо оказались попросту заброшены. Часто произведенное в рамках таких проектов ПО не имело важных функциональных возможностей. Нередко случалось и так, что бизнес-процессы успевали настолько измениться за время разработки, что созданный продукт оказывался совершенно бесполезным.

Разумеется, опыт применения приложений на базе Intranet еще слишком мал для серьезных статистических исследований, однако уже сейчас можно привести некоторые соображения в пользу приложений на базе Java. Так, например, Ян Кемпбелл, директор по сотрудничеству и Intranet в International Data Corp. (IDC) провел исследование предварительных данных об окупаемости (Returns on Investment, ROI) для шести ведущих корпораций. Подводя итоги работы, он сказал: «Предварительные данные анализа ROI для внутрикорпортивных сетей Netscape показывают, что стандартная окупаемость составляет 1000% - намного больше, чем при вложениях в любую другую технологию». Кроме того, затраты окупаются всего за 6-12 недель, а значит, проекты Intranet менее рискованны.

Офер Бен-Шакар, председатель совета директоров и исполнительный директор по технологиям компании NetDynamics, выделил три основных достоинства JDBC. Во-первых, JDBC - это не что иное, как Java; другие решения для взаимодействия с базой данных предполагают привлечение механизмов C++. Во-вторых, JDBC переносим между базами данных; для перехода, например, с SQL Server на Informix требуется немного исправить (или не исправлять вовсе) код приложения. И, наконец, в-третьих, все основные производители баз данных уже поддерживают или в скором времени будут поддерживать JDBC.

Даже поверхностный взгляд на генеалогию JDBC натыкается на API. В 1969 году Е. Ф. Кодд опубликовал свою модель данных, указав на связь между хранимой информацией, или базой данных, и теорией множеств. В 70-х и начале 80-х годов Кодд с группой исследователей, к которым вскоре присоединились IBM, Oracle и другие компании, развили эти представления и разработали первую систему управления реляционной базой данных (СУРБД).

К сожалению, большинство разработчиков «продвигали» собственные разновидности языка SQL, или Structured Query Language, для выборки и обработки данных из реляционной базы данных. Это естественным образом привело к тому, что пользователи и администраторы баз данных оказались привязаны к продуктам, которые случилось купить их компаниям. Предвидя грядущий хаос, производители во главе с ANSI и впоследствии ISO

стандартизировали в 1986 году SQL. Спустя шесть лет, в 1992 году, стандарт был пересмотрен.

Увы, но даже в новой редакции стандарт ANSI слишком абстрактен для коммерческого применения. Он определяет 13 основных предложений SQL, в том числе Select, Update и Create Table, и не описывает способа взаимодействия СУБД и приложений. Для стандартизации интерфейса производители сформировали группу SQL Access Group (SAG) и определили интерфейс Call Level Interface (CLI), благодаря которому приложения могут обращаться к базам данных без использования SQL. Этот интерфейс предоставляет библиотеку функций, которые могут быть собраны и вызваны в реальном времени, в результате приложение имеет возможность использовать и обновлять данные, хранимые в базе.

В 1992 году в результате работы этих групп наконец-то появилось то, что позднее стало популярной, нашедшей широкое применение технологией Open Database Connectivity (ODBC) API компании Microsoft. В своей реализации Microsoft применила спецификации CLI группы SAG, разделив их на два компонента: диспетчера ODBC и драйвера ODBC.

Первый предлагал единый интерфейс всем приложениям, нуждающимся в доступе к базе данных. Это достигалось благодаря полному, достаточно сложному набору функций, с помощью которых программы могли выполнять все задачи, связанные с СУБД - запросы, добавления и обновления данных, исполнение хранимых процедур, а также обращение к источнику данных с просьбой представить описание самого себя. Интерфейс администратора ODBC оставался одинаковым независимо от того, с какой СУБД, приложение взаимодействовало.

Другой же компонент, драйвер ODBC, напротив, зависел от СУБД. Диспетчер использовал драйверы для преобразования запросов на обслуживание от приложений в запросы на языке базы данных. В последних версиях ODBC-продуктов архитектура достигла логического завершения в виде единственного универсального драйвера, работающего на клиентских машинах. Он взаимодействует с диспетчером на сервере, а тот в свою очередь использует специфический драйвер СУРБД. Это усовершенствование стало значительным шагом вперед, поскольку конфигурация драйвера на клиенте была весьма сложной и трудоемкой задачей. Благодаря новой архитектуре достаточно проделать всю работу один раз на сервере, а не выполнять ее снова и снова на каждом клиенте.

JDBC во многом опирается на опыт разработчиков ODBC. Прикладной интерфейс реализован как набор классов Java и

позволяет программисту на Java подавать запросы SQL, а также обрабатывать результаты их выполнения внутри апплета или приложения Java. Как и в ODBC, разработчики реализуют JDBC API посредством диспетчера драйверов, а он в свою очередь поддерживает многочисленные драйверы, осуществляющие связь с различными базами данных. Драйверы JDBC можно либо полностью написать на Java, чтобы загружать как часть апплета, либо разработать естественными средствами, чтобы построить мост к существующим библиотекам доступа к базе данных.

В сущности первым коммерческим продуктом на базе JDBC был простой драйвер моста от JDBC к ODBC, разработанный совместно компаниями JavaSoft и InterSolv. В самом термине «драйвер моста» содержится ссылка на метод, которым вызовы JDBC на сервере передаются - практически без изменений - стандартному драйверу ODBC. Благодаря наличию моста, JDBC может использовать те средства взаимодействия с базой данных, которые предоставляют имеющиеся драйверы ODBC. Структура JDBC позволяет эффективного реализовывать JDBC поверх ODBC, поэтому мост между JDBC и ODBC - наилучший способ использования ODBC из Java.

Хотя предметом нашей статьи является взаимодействие JDBC с реляционными базами данных, нельзя не отметить, что JDBC API не ограничен только одним конкретным типом хранилищ постоянных данных. Если наделить реализацию драйвера JDBC необходимыми функциональными возможностями, этот интерфейс можно будет использовать для связи с такими несхожими источниками информации, как аудио, видео и другие мультимедийные типы, а также нереляционными базами данных.

Важнейшим событием в области JDBC в этом году должно стать появление высокоуровневых компонентов со встроенным JDBC API. Конкурирующие инструменты программирования содержат компоненты, предлагающие аналогичные функциональные возможности. Например, Visual Basic компании Microsoft предоставляет Data Access Objects и Remote Data Objects как в управляющих формах OLE, так и API. При создании не очень сложных приложений программисты используют управляющие формы OLE, которые можно перетащить из панели инструментов на отображаемую форму; те же, кто хочет создать приложения с более изощренными средствами манипулирования данными, пользуются формами API.

Все драйверы JDBC можно разделить на четыре категории: мост от JDBC к ODBC; драйвер частично на Java с оригинальным

API; драйвер полностью на Java с сетевым протоколом; и, наконец, драйвер на Java с оригинальным протоколом.

Мост от JDBC к ODBC обеспечивает доступ с помощью драйверов ODBC. Стоит заметить, что для этого нужно загрузить некие двоичные коды ODBC и в большинстве случаев клиентскую программу базы данных на каждую клиентскую машину, где используется этот драйвер. Таким образом, мост более всего подходит для корпоративной сети или сервера приложений, написанного на Java, с трехзвенной архитектурой.

Драйвер частично на Java с оригинальным API преобразует вызовы Java в клиентский API для Oracle, Sybase, Informix, DB2 и других СУБД. Подобно всем драйверам моста, драйверы этого типа предполагают загрузку некоего двоичного кода на каждой клиентской машине.

Драйвер полностью на Java с сетевым протоколом преобразует вызовы JDBC в независящий от СУБД сетевой протокол, который далее на сервере преобразуется в протокол СУБД. Это сетевое серверное промежуточное программное обеспечение связывает все клиенты Java с различными базами данных; конкретный протокол СУБД зависит от производителя. Как правило, такой драйвер является наиболее гибким вариантом JDBC. Производители, реализующие это решение, способны, пожалуй, предлагать и продукты для Intranet. Чтобы такие продукты поддерживали и доступ через Internet, поставщики должны предусмотреть поддержку защиты, возможность доступа через брандмауэры и т. д. Некоторые, например, IBM, Sybase и Informix, включают драйверы JDBC в промежуточное программное обеспечение для своих баз данных. В драйверах такого рода часто используют драйверы ODBC на сервере для реального взаимодействия с СУБД.

Драйвер полностью на Java с оригинальным протоколом непосредственно преобразует вызовы JDBC в сетевой протокол, который данная СУБД использует. Этот подход позволяет направлять вызовы с клиентов непосредственно на сервер СУБД. Такое решение имеет практический смысл, если конфигурация клиентов контролируется достаточно жестко, как это и бывает в типичных приложениях для Intranet. Поскольку большинство протоколов для баз данных нестандартны, драйверы этого типа должны поставлять сами производители.

Логически взаимосвязанные классы Java объединены в пакеты; пакет JDBC называется java.sql. В пакете классы сгруппированы в интерфейсы. Каждый интерфейс предоставляет набор переменных и методов, которые апплеты и приложения Java могут вызывать. Совокупность переменных представляет собой

состояние, или текущие условия, либо данного логического пакета, либо интерфейса; методы не определяют диапазон поведения и действия, которые пакет или интерфейс могут выполнить. Пакет java.sql имеет следующие интерфейсы: Driver, Connection, Database-MetaData, Statement, PreparedStatement, CallableStatement, ResultSet и ResultSetMetaData. Кроме того, java.sql включает в себя классы Date, Time и Timestamp, DriverManager, DriverPropertyInfo и Types, а также исключительные ситуации DataTruncation, SQLException и SQLWarning.

Интерфейс Driver включают оба компонента: драйвер и диспетчер драйверов. При активизации драйвера его метод getConnection связывается с базой данных и обращается к объекту соединения. Последний может участвовать только в одном сеансе с базой данных, поэтому, если программа хочет получить доступ, например, к двум базам данных Informix, каждая из них должна иметь свой объект соединения. Тот же, кто планирует обращаться из программы к нескольким базам, должен будет загрузить несколько драйверов или создать несколько объектов соединения.

Интерфейс Connection представляет сеанс с конкретной базой данных и является главным компонентом пакета JDBC. Интерфейс предназначен для подачи и возвращения результатов команд SQL. Методы соединения commit, rollback и setAutoCommit управляют транзакциями, а объекты CallableStatement, создаваемые методом CreateStatement интерфейса Connection, - хранимыми процедурами. Интерфейс Connection можно использовать и для предоставления советов базе данных с целью оптимизации запросов.

Для получения информации интерфейс Connection использует метод getMetaData, который возвращает объект DatabaseMetaData. Он предоставляет информацию с описанием таблиц базы данных, поддерживаемую ею грамматику SQL, ее хранимые процедуры, возможности соединения и т. д.

Интерфейс DatabaseMetaData запрашивает у базы данных информацию о ней самой. Вы можете с успехом писать программы, «не знающие» практически ничего о базе данных, к которой обращаются. Интерфейс DatabaseMetaData позволяет приложениям извлекать необходимую информацию в реальном времени (при условии, что пользователь имеет соответствующие права на выполнение заданий), благодаря чему можно получать мелкие, но важные подробности, например, какой символ используется в качестве разделителя и как СУБД обрабатывает значение Null при сортировке.

Три интерфейса - Statement, PreparedStatement и CallableStatement - предоставляют приложению или апплету

средства, с помощью которых собственно приложение и передает команды SQL базе данных. Statement and PreparedStatement очень похожи: они различаются только тем, сколько раз используется SQL: один или несколько. CallableStatement применяется для исполнения набора команд SQL, хранимых в базе данных.

Как Statement, так и PreparedStatement применяются, когда требуется передать базе данных команды для исполнения. При этом база данных создает многочисленные различные планы для удовлетворения нового запроса. (В конце концов, SQL - это логический язык, предоставляющий множество способов выборки данных из физического хранилища.) После создания планов оптимизатор базы оценивает их и выбирает наиболее эффективный, который далее компилируется в двоичное представление. Приложение же получает указатель или метку этой двоичной версии. Существенное различие между Statement и PreparedStatement в том, что Statement не сохраняет этой метки, а PreparedStatement сохраняет, так что проделанная работа не пропадает втуне, и ее результаты могут быть использованы повторно. Предусмотрительный разработчик непременно воспользуется этим немаловажным преимуществом.

Как, впрочем, и многопоточными возможностями Java. Поскольку JDBC работает по последовательному принципу (т. е. очередной команде программы приходится ждать, пока не выполнится другая), сокращение задержки при обработке длинных задач (например, выполнение запросов SQL) за счет создания нескольких потоков может оказаться весьма кстати.

Интерфейс CallableStatement выполняет хранимые процедуры SQL. Эта возможность, зачастую незамечаемая досужими обозревателями, крайне важна, поскольку в большинстве организаций активные команды SQL (Insert, Update, Delete) запрещены для прямого использования. JDBC (как и весь Java) - средство объектно-ориентированное, что означает простоту задания параметров и получения результатов. Таким образом, исполнение хранимых процедур (которые возвращают набор из нескольких результатов) в случае JDBC гораздо проще, чем в случае многих других сред, например Visual Basic.

Наборы из нескольких результатов часто используют при разработке стратегии выборки данных, отвечающих объектно-ориентированной структуре (некоторые атрибуты объекта могут быть реализованы как совокупности самих объектов, например объект СТУДЕНТ может иметь атрибут, содержащий совокупность курсов, на которые некий студент записался или окончил). Кроме того, такие множества применяются для разработки стратегий

выборки данных, имеющих целью минимизацию числа запросов за счет прогнозирования дополнительной потребности в данных (например, в поиске всех заказов данного клиента после нахождения самого клиента).

Интерфейс ResultSet представляет собой курсор - иными словами, совокупность строк, выбранных базой данных по запросу. Поскольку апплеты или приложения, как правило, обрабатывают одновременно только одну строку (или ограниченное число строк, которые выдаются в виде таблицы или решетки), курсор представляет собой средство «удержать» все множество выданных строк, а также и указатель на множество. Когда один из объектов интерфейса Statement исполняет запрос, возвращаемым значением будет ResultSet. Только захватив этот результат, приложение получает возможность действительно предоставить пользователю информацию.

Некоторые технологии промежуточного программного обеспечения, такие как оригинальные драйверы или ODBC-драйверы, поддерживают курсоры с полной прокруткой, т. е. апплет может перемещать указатель текущей строки в любом направлении. Тем не менее JDBC поддерживает лишь прокрутку вперед. Апплеты, которым необходима возможность перемещаться по множеству результатов в обоих направлениях, необходимо моделирование обратного направления. Это достигается путем повторного обращения к базе данных и прохода вперед посредством вызова следующего метода ResultSet до того места, которое требуется просмотреть. Это серьезный недостаток. Хотелось бы, чтобы он был преодолен как можно скорее.

Интерфейс ResultSetMetaData позволяет получить информацию о типах и свойствах столбцов в ResultSet. Эта возможность особенно важна при построении динамических систем, таких как среда разработки или инструментарий для написания запросов, когда база данных и ее структуры неизвестны заранее.

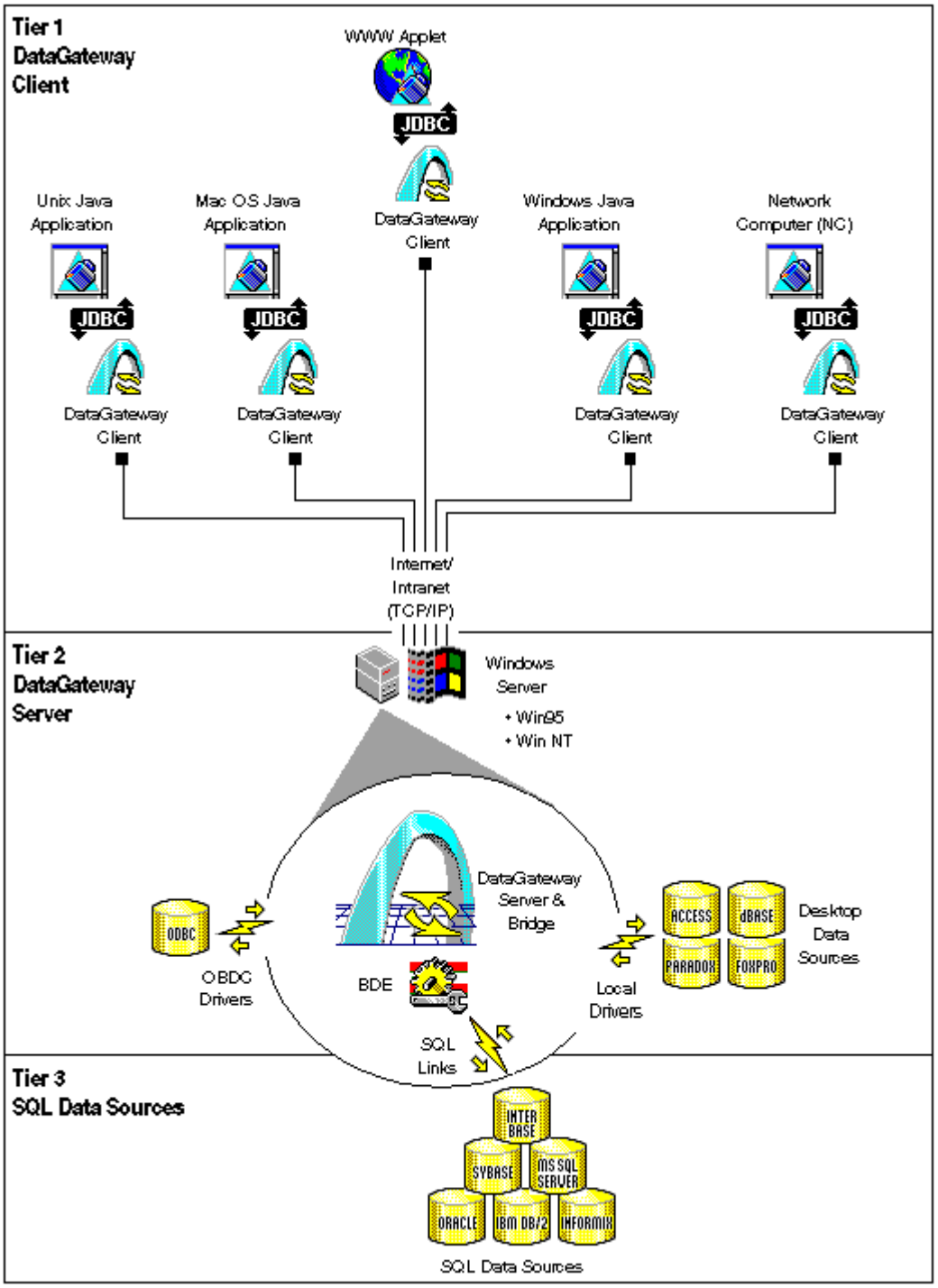
Язык, претендующий на звание языка общего назначения, должен обладать функциональными возможностями для связи с базами данных и манипуляции с извлеченными из них данными. Java в этом смысле не исключение. Технология JDBC прошла сложный путь до хорошо реализованного, удобного в применении средства менее чем за год. На первый взгляд такой темп слишком высок, но только на первый взгляд, так как его диктует эпоха Internet.

JDBC, бесспорно, есть в чем совершенствоваться. Однако если учесть скорость, с которой теперь на рынок поставляются работоспособные продукты, и в буквальном смысле жадность, с

которой пользователи «расхватывают» их, у JDBC прекрасные перспективы.

3.3. Организация обмена данными в телекоммуникационной системе схемотехнического моделирования.

В данной телекоммуникационной системе использован интерфейс JDBC фирмы Borland, функциональная структура которого представлена ниже:



База данных телекоммуникационной системы схемотехнического моделирования представлена двумя таблицами. Одна из которых содержит информацию о зарегистрированных пользователях системы, а в другой накапливаются схемотехнические решения пользователей. Использование интерфейса JDBC фирмы Borland и DataGateway сервера дает значительные преимущества для работы пользователей с базой данных, расположенной на сервере. Данный интерфейс позволяет эффективно использовать SQL запросы к базе данных. Еще одним свойством данного интерфейса является постоянное поддержание связи с сервером баз данных до ее принудительного разрыва, т.е. нет необходимости при каждом SQL запросе устанавливать связь с сервером баз данных заново. Данное свойство может быть расценено как недостаток, который может послужить несанкционированному доступу к базе данных, но, взвешивая все за и против, данная особенность должна быть все таки отнесена к достоинствам данного интерфейса, так как снижается время доступа к данным на сервере и уменьшается нагрузка на сервер.

Ниже перечислены описания атрибутов таблиц, составляющих базу данных:

Таблица USERS. Содержит множество зарегистрированных пользователей.

Поле	***Тип***	***Комментарий***
Login	Varchar(30)	Учетное имя пользователя
Passwrd	Varchar(30)	Пароль пользователя
Rights	Varchar(20)	Права пользователя (admin, user)
Entries	Integer(5)	Количество выделенных строк в таблице SCHEMES для данного пользователя
Fullname	Varchar(60)	Полное имя или описание пользователя

Таблица SCHEMES. Содержит множество схемотехнических решений.

Поле	***Тип***	***Комментарий***
Name	Varchar(254)	Название схемы или краткое ее описание
Scheme	Blob	Исходный текст схемы
Public	Boolean	Общедоступная схема или нет
Author	Varchar(30)	Учетное имя автора схемы
Dt	Date	Дата последней модификации
Comment	Blob	Сопроводительный комментарий к схеме

4. Моделирование вычислительных устройств и систем. Реализация моделирования в рассматриваемой системе

4.1. Методы моделирования вычислительных устройств и систем

4.1.1. Анализ вычислительных устройств методом моделирования

По международному стандарту ISO 2382/1-84 ***моделирование*** (simulation) – это представление различных характеристик поведения физической или абстрактной системы с помощью другой системы. Таким образом, модель – это физическая или абстрактная (математическая) система, с помощью которой мы можем получить характеристики другой, в общем случае еще не существующей системы. Математические модели представляют собой функции, выражающие значения характеристик систем через их параметры и определяются целью анализа, например, модели функционирования, быстродействия и т.д. К математическим моделям предъявляются требования универсальности, адекватности, точности и экономичности.

Степень универсальности модели характеризует полноту отображения в модели свойств реального объекта.

Точность модели оценивается степенью совпадения характеристик реального объекта и значений характеристик, полученных с помощью модели.

Адекватность модели – способность отображать заданные свойства объекта не ниже заданной величины.

Экономичность модели характеризуется затратами вычислительных ресурсов: машинного времени и памяти на ее реализацию.

Требования высоких универсальности, точности и адекватности, с одной стороны, и высокой экономичности, с другой, являются противоречивыми.

В процессе автоматизированного проектирования моделирование проектируемого устройства выполняется многократно и на различных уровнях детализации описания.

Для исследования средств вычислительной техники применяются следующие методы:

- аналитические, заключающиеся в составлении математической модели вычислительного устройства (например, в виде системы уравнений);

- статистические, заключающиеся в составлении математико-статистической модели устройства или системы;
- описательные, обеспечивающие исследования на содержательном уровне;
- экспериментальные, основанные на натурных экспериментах с устройством или его физической моделью;
- имитационные, заключающиеся в проведении экспериментов с имитационной моделью системы или устройства в виде программы.

Формализация процесса обработки информации и определение аналитических зависимостей характеристик системы или вычислительного процесса от их параметров, т.е. построение математической модели функционирования системы или устройства, имеют ограничения с точки зрения полноты описания работы реального устройства.

В рассматриваемой системе использован метод имитационного моделирования вычислительного устройства. Традиционно модели цифровых вычислительных устройств строились в форме алгоритмических описаний их функционирования на основе представлений об исследуемом объекте как о системе с дискретными событиями. Вне зависимости от языковых средств при создании алгоритмических моделей используется функциональный подход к описанию цифровых устройств, при котором модель представляется в виде совокупности развивающихся во времени взаимодействующих вычислительных процессов.

Метод имитационного моделирования выделяется своей эффективностью и универсальностью при решении задач синтеза и анализа цифровых вычислительных устройств. В широком смысле имитационное моделирование может быть определено как процесс представления динамической системы моделью для получения информации об этой системе путем проведения экспериментов над ее моделью. Применение этого метода полезно в том случае, когда прямое экспериментирование с исследуемым устройством нецелесообразно или невозможно.

Суть этого метода в том, что весь процесс функционирования вычислительного устройства рассматривается как совокупность процессов функционирования его составных частей.

Устройство разбивается на части, которые позволяют достаточно просто формально описать поведение каждой из них, и затем описываются взаимосвязи между отдельными частями

устройства. Удобнее всего при этом разбивать вычислительное устройство на такие составные части, которые соответствовали бы отдельным и блокам устройства, тогда модель будет состоять из частей, имитирующих поведение устройств и связей в реальном объекте, и в силу этого имитировать поведение самого объекта моделирования.

Используемые для построения имитационных моделей вычислительных устройств и систем алгоритмические языки благодаря гибкости и доступности позволяет отражать в моделях многие детали и свойства структур и функций вычислительных устройств, которые опускаются или утрачиваются в математически строгих моделях. Свойственные имитационным моделям реалистичность, адекватность, простота основаны на использовании для их построения всех имеющихся теоретических представлений об объекте исследования.

4.1.2. Характеристика имитационного моделирования

Моделирование вычислительных устройств производится на следующих уровнях: системном, регистровом, логическом и электрическом.

Целью моделирования на системном уровне является определение эксплуатационных характеристик проектируемого устройства в целом (быстродействия, надежности и т.д.). Модель представляет собой описание структуры и порядка функционирования устройства в виде программы на одном из алгоритмических языков или языков имитационного моделирования. Использование в этом случае специальных языков имитационного моделирования предпочтительнее в силу наличия в них специальных лингвистических средств управления процессом моделирования, работы с псевдослучайными последовательностями, сбора статистических характеристик, организации параллельных процессов в системе и т.д. Кроме этого языки имитационного моделирования обеспечивают значительную автоматизацию при программировании имитационной модели.

На регистровом уровне моделирования исследуются следующие вопросы:

- определение временных параметров программ (скорость выполнения команд, цепочек команд, программ);
- определение частоты использования аппаратных средств;
- определение степени загрузки трактов передачи данных и распределение этой загрузки по времени.

Модель системы на уровне регистровых передач может быть использована как инструментальное средство для отладки математического обеспечения проектируемой системы.

На этапе моделирования логических элементов исследуется логико-временное поведение устройства, при этом решаются следующие основные задачи:

- проверка правильности логической структуры;
- сравнение характеристик различных вариантов логических схем;
- выявление состязаний и риска сбоя;
- оценка средств контроля и диагностики.

В моделировании на уровне анализа электронных схем предметом рассмотрения являются элементы типа транзисторов, резисторов и др. По характеристикам элементов и связям между ними вычисляются значения токов и напряжений в схеме, параметры переходных процессов в элементах.

В моделирующих программах с помощью математических моделей имитируется процесс ввода, обработки и вывода данных проектируемого устройства. Процедуры получения математической модели объекта и имитации его функционирования на этой модели называются имитационным моделированием.

При имитационном моделировании прежде всего возникает необходимость в описании объекта проектирования. Это функционально-структурное описание обычно выполняется первоначально на естественном языке, включает в себя ряд формул и содержит значительную по объему графическую информацию. Подобное описание не формализовано, поэтому в практике имитационного моделирования особое место занимают языки моделирования, позволяющие формализовать запись описания произвольной структуры вычислительного устройства и алгоритма его функционирования.

4.2. Моделирование работы схемы в рассматриваемой системе

4.2.1. Формирование функционально-структурного описания моделей элементов

4.2.2. Описание алгоритма моделирования схем

В основу моделирования работы схемы положен метод имитационного моделирования. Перед тем как описать алгоритм моделирования работы созданной схемы в рассматриваемой телекоммуникационной системе необходимо в общих чертах представить структуры, использующиеся в процессе моделирования. Каждый элемент схемы (кроме элементов оформления) имеет набор контактов, через которые он подсоединен к другим элементам (в том числе проводникам). Контакты элементов могут быть двух типов: либо входные, либо выходные. Каждый контакт элемента к моменту начала моделирования имеет массив, размерность которого равна времени моделирования, измеряемому в квантах времени. В данном массиве накапливается информация о состоянии контакта в определенный квант времени. Контакт в определенный квант времени может находиться в одном из трех состояний: «0», «1» и неопределенное состояние «X». Моделирование работы схемы основано на сценариях работы каждого элемента в определенный квант времени. Сценарию каждого элемента передается в качестве параметра текущий квант времени. В сценарии могут выполняться следующие действия:

1. Можно узнать состояния любых контактов в любой квант времени.
2. Затем можно произвести необходимые преобразования.
3. И установить состояния любых контактов в любой квант времени.

В преобразованиях участвуют значения изменяемых параметров элемента и любые другие константные или переменные величины установленные разработчиком класса данного элемента.

Итак, алгоритм моделирования работы схемы состоит из трех этапов: начало моделирования, собственно моделирование и конец моделирования.

Первый этап:

К началу моделирования все элементы находятся в «куче». Ни одному элементу не известно какие его контакты связаны с контактами проводников или с контактами других элементов. Для моделирования необходимо установить связи между контактами элементов. Проводники и элементы оформления в процессе моделирования не несут никакой функциональной нагрузки. Поэтому на данном этапе производится формирование отдельных векторов элементов: вектора проводников и вектора функциональных элементов. Остальные элементы вообще не играют никакой роли в моделировании. Вектор функциональных элементов используется на втором этапе моделирования, а вектор проводников используется только на этом этапе для установления связей между

всеми элементами. Совокупность всех проводников образует несвязанный граф, вершинами которого являются контакты проводников, а дугами - сами проводники. Поэтому поиск связей между контактами элементов сводится к поиску путей от одного контакта ко всем с ним связанным в выше описанном графе. В данном графе проводники могут образовывать циклы, которые перед началом поиска путей необходимо разорвать, т.е. необходимо построить дерево этого графа. Построение дерева производится по следующему алгоритму:

1. Все вершины представляются вектором V={1,2,3,…,p}, где p - количество вершин в графе.
2. Граф представляется матрицей E={$a_{1,i}$,$b_{2,i}$}, где i=1,2,3,…,q; q - количество ребер в графе, а $a_{1,i}$ и $b_{2,i}$ - начало и конец i-го ребра соответственно.
3. На i-ом шаге производится сравнение V[E(1,i)] и V[E(2,i)]. Если они не равны, то i-ое ребро относится к дереву и V[E(1,i)]=V[E(2,i)], в противном случае осуществляется следующий шаг.
4. Процесс заканчивается, когда к дереву отнесено p-1 ребро.

После применения вышеописанного алгоритма из вектора проводников удаляются все проводники не относящиеся к дереву графа. Затем начинается формирование связей между элементами. Для этого берется каждый контакт элемента из вектора элементов и осуществляется проход от данного контакта к концам всех путей, ведущих от него. Если на конце пути есть контакт (или контакты) другого (или других) элемента (или элементов), то они заносятся в вектор присоединенных контактов. Следует заметить, что, если контакт является входным, то вектор присоединенных к нему контактов содержит только выходные контакты, и наоборот. На завершающей стадии первого этапа в каждом контакте отводится массив размерностью равной времени моделирования. Каждому элементу этого массива присваивается значение состояния «X». И, наконец, вызывается функция инициализации каждого элемента, которая может выполнять некоторые действия, необходимые перед началом собственно моделирования.

Второй этап:

На этом этапе происходит моделирование работы схемы, результатом которого является заполнение массивов состояний всех контактов всех элементов. В каждый i-й квант времени (i=0,1,…,n; n - длина интервала моделирования в квантах времени) выполняются следующие действия:

1. Начинается сканирование по вектору функциональных элементов.

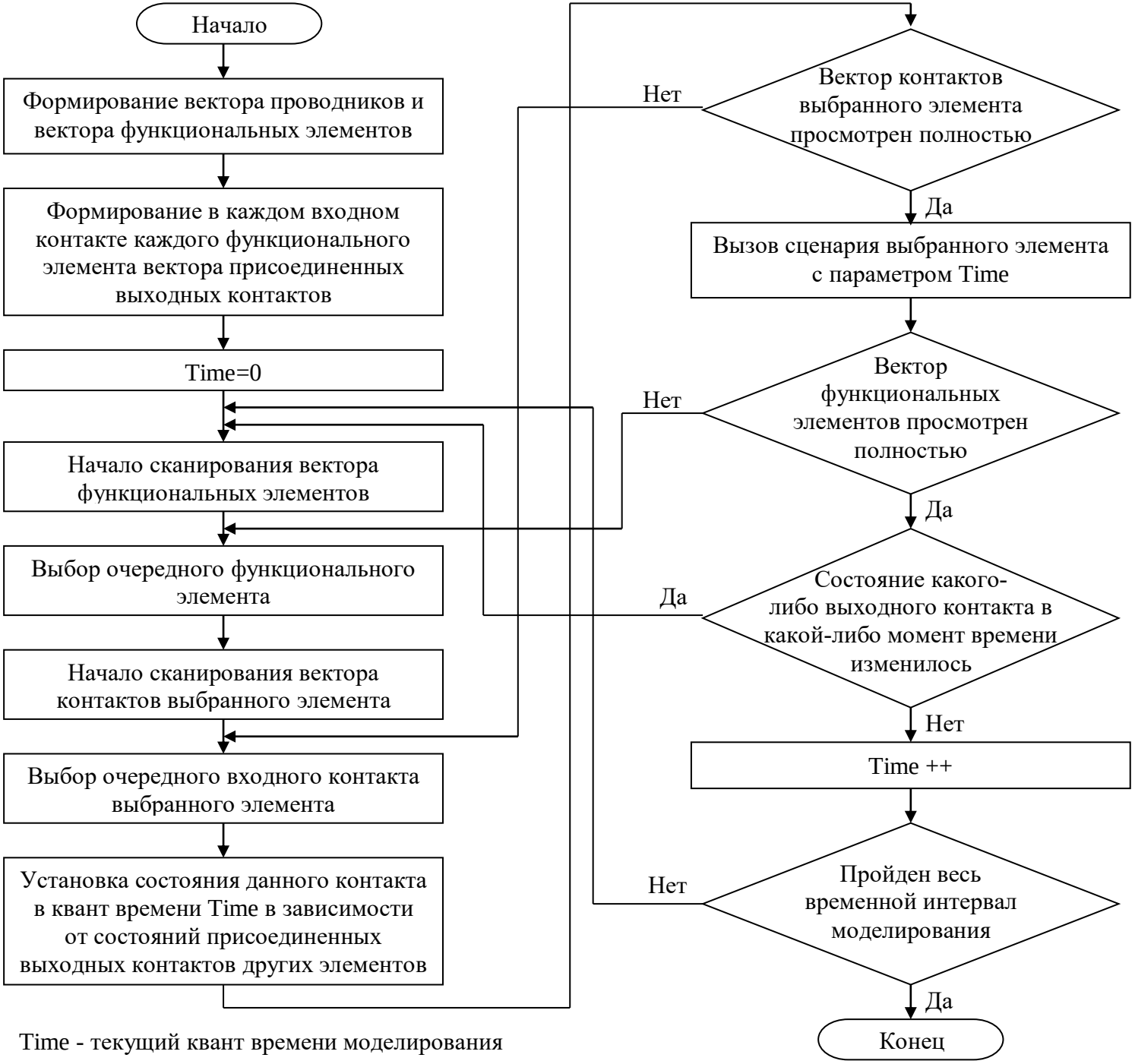
2. Выбирается очередной элемент.
3. Начинается сканирование по вектору контактов выбранного элемента.
4. Выбирается очередной входной контакт этого элемента.
5. В данный квант времени выбранному контакту присваивается состояние «1», если все присоединенные к нему выходные контакты других элементов находятся в состоянии «1», аналогично, если все присоединенные к нему выходные контакты других элементов находятся в состоянии «0», то выбранный контакт в текущий квант времени устанавливается в «0». Во все остальных случаях выбранному входному контакту присваивается состояние «X».
6. Выбор входных контактов продолжается до тех пор, пока не просмотрен весь вектор контактов элемента.
7. После установления состояний всех входных контактов в данный квант времени вызывается сценарий работы элемента.
8. Если просмотрен еще не весь вектор функциональных элементов, то осуществляется переход к пункту 2.
9. Если в результате выше перечисленных действий изменилось состояние хотя бы одного выходного контакта какого-либо элемента в любой квант времени, то выполняется переход к пункту 1.

После того, как выше перечисленные действия выполнены в каждый квант времени второй этап моделирования заканчивается.

Третий этап:

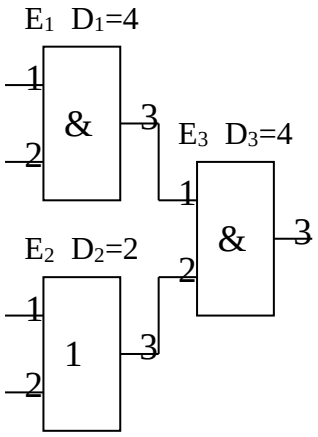
На этом этапе осуществляется удаление вектора проводников и вектора функциональных элементов. Моделирование работы схемы завершается.

Блок-схема алгоритма моделирования работы схемы



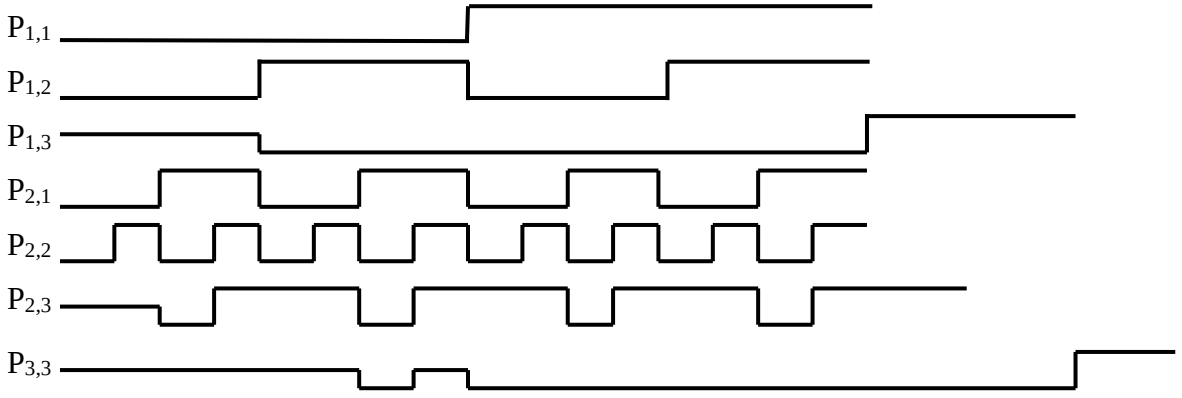
Time - текущий квант времени моделирования

Пример накопления и отображения результатов моделирования:



E_i – элемент i;
D_i – задержка на k-ом элементе;
T_i – i-й квант времени моделирования;
$P_{i,j}$ – j-й контакт i-го элемента.

	T_1	T_2	T_3 …																			
$P_{1,1}$	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1						
$P_{1,2}$	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1						
$P_{1,3}$	x	x	x	x	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1		
$P_{2,1}$	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1						
$P_{2,2}$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1						
$P_{2,3}$	x	x	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1				
$P_{3,1}$	x	x	x	x	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1		
$P_{3,2}$	x	x	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1				



5. Поддержка обучения

5.1. Дистанционное образование и новая информационно-образовательная технология (НИОТ)

5.1.1. Общая характеристика дистанционного образования (выдержки из документа: «Концепция создания и развития единой системы дистанционного образования в России», утвержденного решением совета ИДО МЭСИ 29 апреля 1998 года)

Дистанционное обучение (ДО) – это универсальная гуманистическая форма обучения, базирующаяся на использовании широкого спектра традиционных, новых информационных и телекоммуникационных технологий, и технических средств, которые создают условия для обучаемого свободного выбора образовательных дисциплин, соответствующих стандартам, диалогового обмена с преподавателем, при этом процесс обучения не зависит от расположения обучаемого в пространстве и во времени.

Дистанционное образование – это система, в которой реализуется процесс дистанционного обучения для достижения и подтверждения обучаемым определенного образовательного ценза, который становится основой его дальнейшей творческой и (или) трудовой деятельности.

Информационно-образовательная среда ДО представляет собой системно организованную совокупность средств передачи данных, информационных ресурсов, протоколов взаимодействия, аппаратно-программного и организационно-методического обеспечения, и ориентируется на удовлетворение образовательных потребностей пользователей.

Дистанционное обучение от традиционных форм обучения отличают следующие характерные черты:

- Гибкость. Возможность заниматься в удобное для себя время, в удобном месте и темпе. Нерегламентированный отрезок времени для освоения дисциплины.
- Модульность. Возможность из набора независимых учебных курсов - модулей формировать учебный план, отвечающий индивидуальным или групповым потребностям.

- Параллельность. Параллельное с профессиональной деятельностью обучение, т.е. без отрыва от производства.
- Охват. Одновременное обращение ко многим источникам учебной информации (электронным библиотекам, банкам данных, базам знаний и т.д.) большого количества обучающихся. Общение через сети связи друг с другом и с преподавателями.
- Экономичность. Эффективное использование учебных площадей, технических средств, транспортных средств, концентрированное и унифицированное представление учебной информации и мультидоступ к ней снижает затраты на подготовку специалистов.
- Технологичность. Использование в образовательном процессе новейших достижений информационных и телекоммуникационных технологий, способствующих продвижению человека в мировое постиндустриальное информационное пространство.
- Социальное равноправие. Равные возможности получения образования независимо от места проживания, состояния здоровья, элитарности и материальной обеспеченности обучаемого.
- Интернациональность. Экспорт и импорт мировых достижений на рынке образовательных услуг.
- Новая роль преподавателя. ДО расширяет и обновляет роль преподавателя, который должен координировать познавательный процесс, постоянно усовершенствовать преподаваемые им курсы, повышать творческую активность и квалификацию в соответствии с нововведениями и инновациями.
- Позитивное влияние оказывает ДО и на студента, повышая его творческий и интеллектуальный потенциала за счет самоорганизации, стремления к знаниям, умения взаимодействовать с компьютерной техникой и самостоятельно принимать ответственные решения.
- Качество ДО не уступает качеству очной формы получения образования, а улучшается за счет привлечения выдающегося кадрового профессорско-преподавательского состава и использования в учебном процессе наилучших учебно-методических изданий и контролирующих тестов по тем или иным дисциплинам.

5.1.2. Характеристика новой информационно-образовательной технологии (НИОТ)

НИОТ - это процесс получения знаний, построенный не на общении с преподавателем, а на использовании новейших методик и приемов, основанных на использовании компьютерных, аудио- и видео технических средств. Суть НИОТ заключается в создании образовательной технологии XXI века, где студент самостоятельно изучает с помощью компьютера у себя дома или на работе, при поддержке наставника – тьютора полный набор электронных учебников по выбранной специальности со всеми предметами, входящими в государственный образовательный стандарт данной профессии.

НИОТ является основной частью системы дистанционного обучения и открывает возможности:

- приобретения высшего образования различными категориями людей без территориальных и временных ограничений;
- изучения учебных курсов в режиме и темпе, удобном для учащихся;
- широкого использования экстерната;
- предоставления образовательных услуг лицам с ограниченными возможностями посещения вуза в очном режиме (инвалидность, работа, учеба в другом учебном заведении);
- приобретения навыков работы с современными компьютерными технологиями.

Переход к НИОТ открывает огромные возможности для творческого развития каждого студента и стимулирует творческий потенциал преподавательского корпуса. Эта технология и методология гарантируют индивидуальный подход и учет специфики, менталитета, психологии, уровня образованности, культуры каждого специалиста, подготовленного с ее помощью. Имеются также дополнительные преимущества, первостепенного значения при оценке важности государственной поддержки этого нового вида образования. Одно из этих преимуществ заключается в том, что НИОТ в варианте дистанционного обучения может быть использованы для обучения инвалидов и людей с ограниченными возможностями.

НИОТ позволяют визуализировать все тексты как письменные, так и устные, для обучения глухонемых. Существует возможность озвучивания всех текстов для обучения слепых и слабо видящих. Имеются также хорошо известные на Западе и начинающие получать ограниченное распространение в России

технологии, допускающие управление компьютерными системами пользователями, страдающими заболеваниями опорно-двигательного аппарата. Таким образом, использование НИОТ в дистанционном обучении позволяет обучать людей, которые в силу ряда обстоятельств или заболеваний ранее были обречены на пребывание в маргинальных слоях общества.

НИОТ позволяет создать систему надомного обучения:

- обеспечивающую возможность получить образование в сжатые сроки;
- не требующую отрыва обучающегося от основной работы;
- не снижающую качества обучения, построенного на государственных образовательных стандартах;
- эффективную на любом расстоянии от образовательного центра.

Современные Internet-технологии решают ряд проблем, стоящих перед дистанционным образованием, таких как надежная и эффективная передача текстовой, аудио- и видеоинформации, что составляет основу любого дистанционного курса лекций. Имеющиеся технологии видеоконференций создают иллюзию непосредственного общения учителя и ученика, обеспечивая консультации и устную сдачу экзамена или даже защиту квалификационной работы.

Для создания условий полноценной учебы, однако, недостаточно всего вышеперечисленного. Необходимо развитие определенных практических навыков. При традиционном подходе в учебные курсы вводятся лабораторные работы, где учащиеся на конкретном оборудовании приобретают такие навыки.

Все чаще при изучении классических дисциплин (например, физики) и общетехнических дисциплин (схемотехники) вместо работы с физическими объектами используется компьютерное имитационное моделирование, что, в принципе, вполне приемлемо для дистанционного образования.

Стоимость разработки имитационной модели объекта зависит от степени упрощения, абстракции. Чем более абстрактен объект, тем проще его моделирование. Чем ближе объект к реальному, тем больше его особенностей нужно учитывать в модели, тем выше стоимость модели. В классических дисциплинах иллюстрируются фундаментальные закономерности, которые, как правило, достаточно просты и хорошо исследованы. В этом случае стоимость моделирования невысока. То же, хотя и в меньшей степени, можно сказать и об общетехнических дисциплинах.

Отметим еще одну существенную особенность вышеупомянутых дисциплин – количество специальностей, для которых они обязательны, велико. Следовательно, каждой такой

учебной имитационной моделью воспользуется большое количество учащихся, так что удельная стоимость в расчете на одного учащегося будет небольшой.

Ранее отмечалось, что для дистанционного образования наиболее подходит Internet, в том числе и развитостью своих технологий. Одной из наиболее распространенных в настоящее время технологий является гипертекст, т.е. текст, включающий в себя ссылки на другой текст, аудио- и видеоинформацию и т.п., в том числе и на программы, написанные на языке Java. Для просмотра гипертекста используются специальные программы просмотра - браузеры, причем наиболее распространенные среди них MS Internet Explorer и Netscape Communicator старших версий обеспечивают выполнение Java-апплетов.

Разработка клиентского программного обеспечения в виде Java-апплетов позволяет его использовать на практически любой допустимой вычислительной технике, от небольших ПК до рабочих станций и мейнфреймов.

5.2. Поддержка обучения в телекоммуникационой системе схемотехнического моделирования

5.2.1. Взаимодействие преподавателя и обучаемого. Реализация и перспективы

5.2.2. Тестирование созданной схемы

В представленной системе предусмотрена возможность тестирования создаваемых схем на соответствие заданию. Если пользователь желает проверить свои возможности и знания в проектировании каких-либо узлов вычислительной техники, то он может выбрать себе задание, выполнить его и оценить работоспособность созданной схемы. Все задания делятся на группы (счетчики, регистры, триггеры и др.), поэтому пользователь легко может ориентироваться при выборе себе задания. Задание представляется в виде текста. В задании могут указываться элементы, которые обязательно должны присутствовать в создаваемой схеме и пользователь должен четко следовать указаниям. После того, как схема создана и пользователь уверен в ее правильности (или не уверен) можно перейти в режим тестирования.

В этом режиме пользователь должен указать, какому заданию соответствует спроектированная им схема и произвести тестирование. В результате тестирования система оценивает адекватность работы схемы на основе тестовых последовательностей, указанных в задании, но скрытых от пользователя. Пользователю сообщается о том, сколько тестов схема прошла всего и сколько из них пройдено успешно. Также после тестирования пользователь имеет возможность просмотреть временные диаграммы работы схемы в тестовом режиме.

Тестирование подготовленной схемы основано на наблюдениях за действиями человека проверяющего правильность созданной схемы на учебном схемотехническом стенде. Пользователь производит некоторые управляющие воздействия (манипулирует переключателями, генерирует импульсы и т.д.), а затем наблюдает за состоянием каких-либо индикаторов, которые отражают работу схемы. Тестирование схемы в данной системе основано на подобных принципах. В задании пользователю указывается о том, какие генераторы и маркеры должны обязательно присутствовать в создаваемой схеме. Через указанные генераторы передаются управляющие тестовые последовательности их состояний, а с помощью указанных маркеров осуществляется оценка работы схемы. В этом и состоит аналогия с деятельностью человека в подобной ситуации. Из скрытой от пользователя части задания извлекаются тестовые последовательности для всех необходимых генераторов схемы. Эти последовательности разбиты на интервалы. На каждом интервале заданный генератор может вести себя как угодно. Количество интервалов для каждого генератора, используемого при тестировании должно быть одинаково. В результате выполнения заданных управляющих воздействий каждым генератором на текущем интервале на маркерах соответственно тоже происходят какие-либо изменения (или не происходят). Поэтому в задании указывается, какое состояние должен принять каждый маркер на текущем интервале после применения генераторами управляющих воздействий. На основании этого и оценивается адекватность схемы после моделирования ее работы.

Итак, тестирование проходит в два этапа: моделирование работы схемы по специфической программе и оценка результатов моделирования.

Первый этап:

Происходит настройка управляющих воздействий для каждого необходимого генератора в схеме и выполняется процесс моделирования работы схемы.

Второй этап:

Рассматриваются полученные последовательности состояний маркеров в схеме. На каждом тестовом интервале полученная последовательность состояний сравнивается с эталонной и делается заключение об адекватности. Сумма пройденных тестовых интервалов и не пройденных представляется пользователю. А также представляются временные диаграммы работы схемы, на основании которых пользователь может сделать вывод о том, на какие тестовые управляющие воздействия схема реагирует неверно.

6. Тестирование функционирования системы

Этап тестирования – это один из самых трудоемких этапов при создании программ. Этот этап может повлечь затраты, составляющие половину общих затрат на создание программы.

В процессе тестирования используются данные, характерные для системы в рабочем состоянии, т.е. данные для тестирования нельзя выбирать случайным образом.

План проведения испытаний должен быть составлен заранее, а большую часть тестовых данных следует определить на этапе проектирования системы.

В процессе тестирования для определения правильности программы используются различные критерии, наиболее важными из которых являются следующие:

1. каждый оператор должен выполняться хотя бы один раз (что равносильно прохождению каждой ветви программы хотя бы один раз);
2. каждое условие принимает значение «ложь» и «истина» хотя бы один раз;
3. каждый путь в программе должен быть испытан хотя бы один раз;

Существует две стратегии тестирования:

1. метод «черного ящика»;
2. метод «белого ящика»;

При тестировании методом черного ящика программа рассматривается как «черный ящик», т.е. не учитываются знания о внутренней структуре. Обычно это тестирование по входу-выходу, т.е. тестирование по данным. При таком подходе обнаружение всех ошибок в программе является критерием исчерпывающего входного тестирования, т.е. в качестве тестовых наборов использовать все возможные наборы входных данных. Построение исчерпывающего тестового набора вручную, при большом количестве тестовых наборов, практически невозможно.

Стратегия тестирования методом «белого ящика» - тестирование, управляемое логикой программы, позволяет исследовать внутреннюю структуру программы.

Таким образом, исчерпывающему входному тестированию может быть поставлено в соответствие тестирование – исчерпывающее, по всем возможным маршрутам. Т.е. имеется или графическое, или текстовое представление программы или

алгоритма и необходимо проверить все возможные маршруты от начала прохождения данных до получения результата.
Этот метод имеет 2 недостатка:

1. число не повторяющих друг друга маршрутов велико;
2. хотя стратегия «белого ящика» позволяет проверить все маршруты, сама программа все же может содержать ошибки.

Поэтому для тестирования программ обычно используют комбинацию этих двух методов.

Тестирование системы проводилось как методом «черного ящика», так и методом «белого ящика». Тестовый набор входных данных при тестировании методом «белого ящика» был получен путем анализа логики программы. Для проведения теста были разработаны схемы различных вычислительных устройств (регистров, счетчиков, триггеров и др.), пользуясь только возможностями системы. Проведено моделирование их работы при различных параметрах системы. Были разработаны примеры тестовых заданий и построены схемы устройств, соответствующих заданию. Проведено их тестирование. Таким образом была осуществлена проверка функций по всем возможным маршрутам выполнения.

Тестирование методом «черного ящика» проводилось путем проверки работоспособности системы в различных операционных системах с использованием различных Internet/Intranet навигаторов: MS Internet Explorer 4.x для Windows 95/98/NT, Netscape Communicator 4.5 для Windows 95/98/NT и Netscape Communicator 4.07 для Unix.

7. Описание применения

7.1. Инсталляция системы

7.1.1. Аппаратные и программные требования к серверу

486 или лучший процессор;
Windows 95, Windows NT 3.51 или более высокие версии;
Поддержка протокола TCP/IP;
Web-сервер;
BDE 4.0 (Borland Database Engine) или более высокая версия;
Borland DataGateway Server 1.0 или более высокая версия;
1 МБ свободного пространства на диске;

7.1.2. Аппаратные и программные требования к клиентской машине

486 или лучший процессор;
Windows 95, Windows NT 3.51 или более высокие версии;
Поддержка протокола TCP/IP;
Установленный навигатор Microsoft Internet Explorer 4.x;

7.1.3. Инсталляция системы

При установке системы необходимо выполнить следующие шаги. Особые случаи когда на машине уже присутствуют некоторые из ниже перечисляемых компонентов здесь не рассматриваются:

1. Установить на сервер пакет Borland DataGateway, который содержит Borland Database Engine и Borland DataGateway Server. При установке пакета необходимо следовать указаниям инсталлятора.
2. Установить на Web-сервер систему Net schematics. Все директории и файлы необходимо разместить в одной директории на Web-сервере и разрешить доступ к этой директории из-вне, установив виртуальный каталог или путь для доступа к файлу index.html.
3. Каталог tables, являющийся частью системы необходимо расположить в отдельной директории, к которой нет доступа через сеть. Используя утилиту BDE Administrator, необходимо

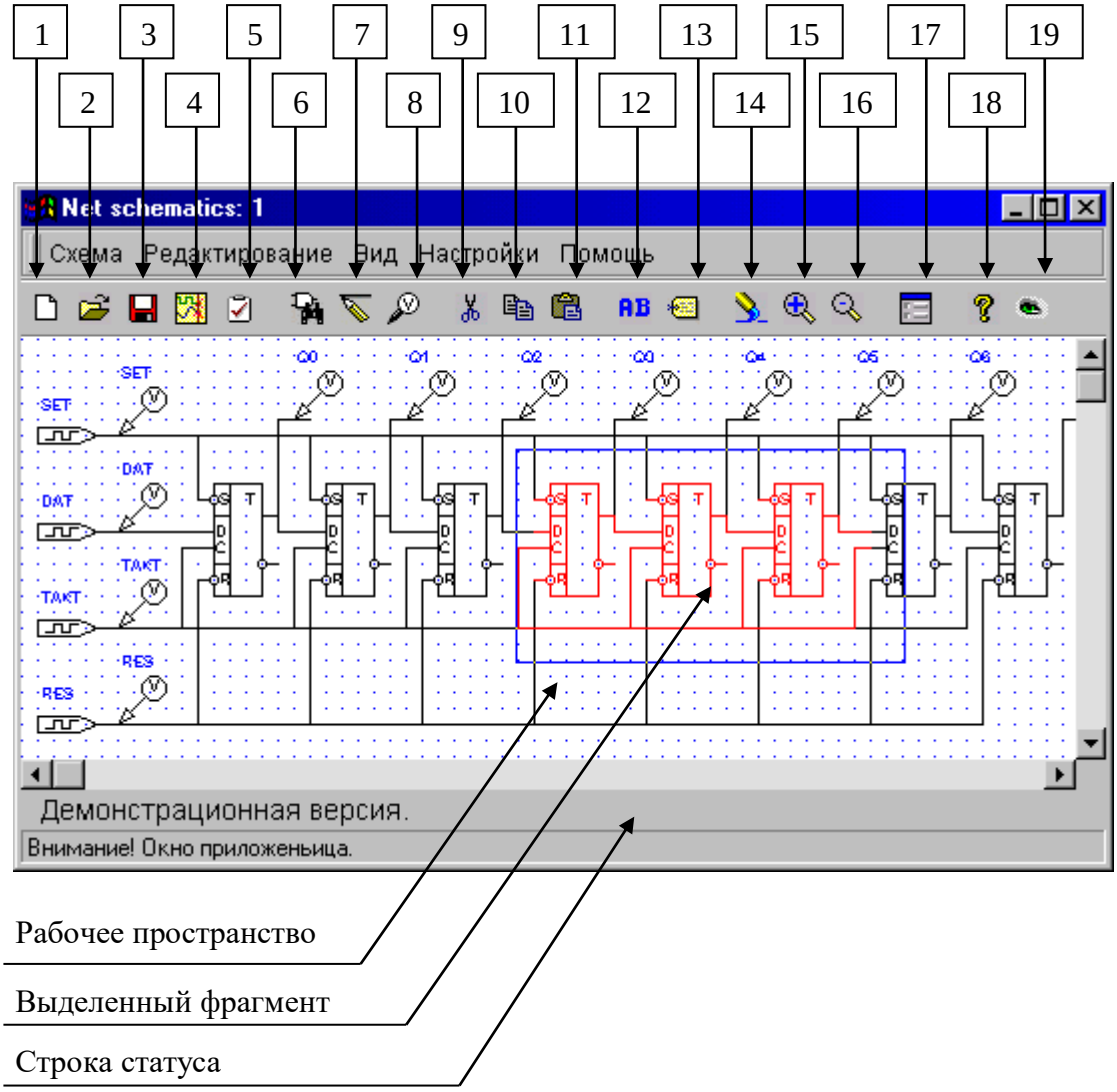
зарегистрировать базу данных под псевдонимом SCHEMATICS и указать в нем использование стандартного драйвера доступа к базам данных DBASE или FOXPRO. В переменной PATH драйвера необходимо указать путь к таблицам, расположенным в каталоге tables.

4. Необходимо запустить DataGateway Server. Пока он будет в рабочем состоянии, пользователи системы Net schematics будут иметь возможность доступа к базам данных системы. Если же DataGateway Server не будет запущен, то доступ пользователей к базам данных системы будет невозможен.
5. Теперь можно запускать систему Net schematics, используя навигатор Microsoft Internet Explorer 4.x.

7.2. Инструкция пользователю

При запуске очередного окна телекоммуникационной системы схемотехнического моделирования первое, что предстает взору – это диалог идентификации пользователя и организации связи с базой данных системы (см. описание пункта меню «Настройки/Связь»). После чего открывается главное окно системы. Данное окно представляет собой интерфейс, через который осуществляется формирование схем моделей вычислительных устройств. Здесь представлены основные возможности по редактированию (вставка элементов из элементной базы телекоммуникационной системы, их компоновка и размещение, удаление элементов и участков схемы, размножение и т.д.). Окно снабжено меню, через которое осуществляются необходимые действия, и панелью функциональных кнопок, позволяющих выполнять часто используемые функции не обращаясь к соответствующим пунктам меню.

Окно редактора моделей схем вычислительных устройств



Соответствие функциональных кнопок пунктам меню:

1. Схема/Новая
2. Схема/Открыть…
3. Схема/Сохранить как …
4. Схема/Моделировать работу
5. Схема/Выполнить тест
6. Редактирование/Вставить элемент
7. Редактирование/Проводник
8. Редактирование/Вставить маркер
9. Редактирование/Вырезать
10. Редактирование/Скопировать
11. Редактирование/Вклеить
12. Редактирование/Изменить метку
13. Редактирование/Изменить параметры
14. Вид/Перерисовать
15. Вид/Увеличить
16. Вид/Уменьшить
17. Настройки/Настройки
18. Помощь/Помощь
19. Помощь/О программе…

МЕНЮ СХЕМА:

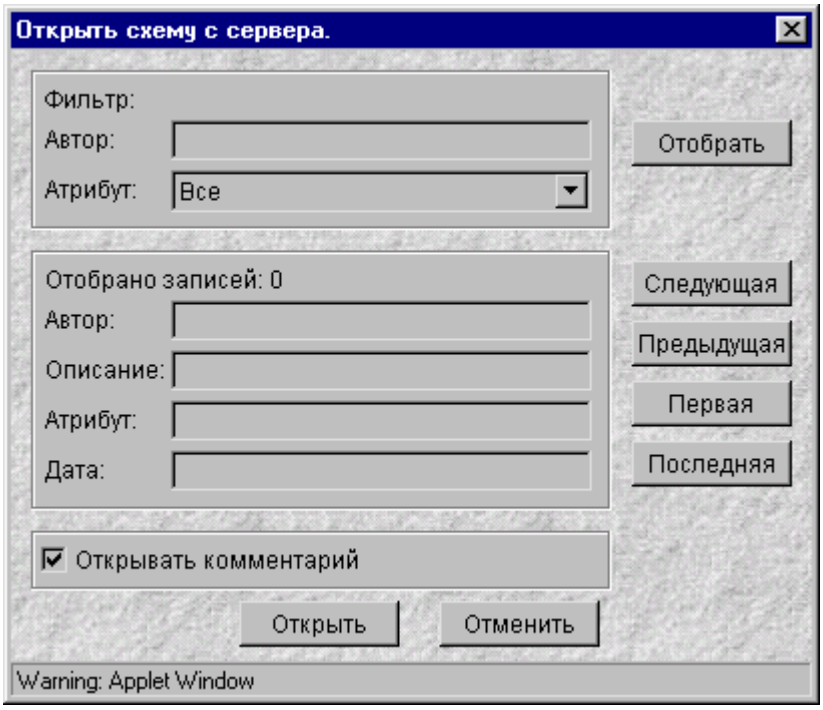
Новая: Очистка рабочего пространства и подготовка к созданию новой схемы. Не забудьте сохранить уже созданную схему, если она есть.

Открыть…: Данный пункт меню предоставляет пользователю диалог открытия файла схемы с диска локальной машины или любого подключенного сетевого диска. При открытии файла схемы производится попытка открытия файла комментария, который должен иметь такое же имя, как и файл схемы, но с расширением *.cmt. Если при выборе этого пункта меню вы увидите сообщение «Доступ запрещен», то это означает, что ваш навигатор не позволяет вам обращаться к файлам на диске. Если вы пользуетесь IE 4.x и выше, то устранить данный запрет можно следующим образом:

1. Выбрать пункт меню View навигатора, в котором в свою очередь выбрать пукт Internet Options...;
2. В появившемся диалоге выбрать закладку Security;
3. После этого необходимо выбрать зону, из которой загружена данная телекоммуникационная система;
4. Из набора Set the security level for this zone выбрать радиокнопку Custom и нажать кнопку Settings;
5. В появившемся диалоге найти пункт Java и установить Java permissions в Custom, после чего нажать кнопку Java Custom Settings...;
6. В появившемся диалоге выбрать закладку Edit Permissions;
7. В разделе Unsigned Content - Run Unsigned Content - Additional Unsigned Permissions - Access to all Files установить Enable;
8. В разделе Unsigned Content - Run Unsigned Content - Additional Unsigned Permissions - Dialogs установить Enable;
9. После этого во всех выше упомянутых диалогах нажать кнопки «Ok», «Apply» и т.д.;
10. В завершении необходимо перезапустить систему путем нажатия Ctrl+F5.

Сохранить как…: Данный пункт меню предоставляет пользователю диалог сохранения файла схемы на диск локальной машины или любой подключенный сетевой диск. При сохранении файла схемы производится сохранение файла комментария, который сохраняется под тем же именем, как и файл схемы, но с расширением *.cmt. Если при выборе этого пункта меню вы увидите сообщение «Доступ запрещен», то необходимо предпринять те же действия, что описаны в инструкции к пункту меню «Открыть…».

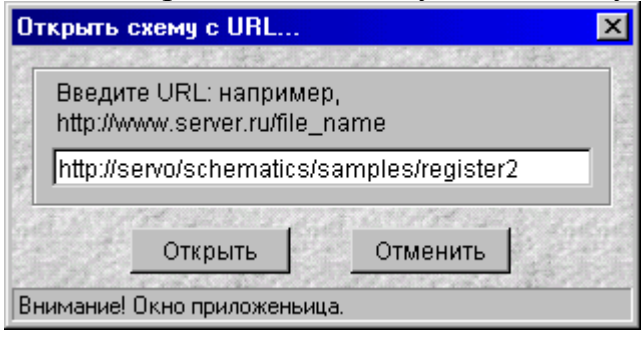
Загрузить из базы: Пользователю предоставляется возможность загрузить из подключенной базы уже созданную схему. Для этого служит диалог, представленный ниже:



С помощью фильтра можно отобрать либо все схемы, либо схемы, принадлежащие одному автору, либо только публичные или не публичные. Также можно воспользоваться комбинацией этих параметров. После отбора схем необходимо выбрать требуемую и открыть ее. Если же вместо этого диалога вы увидите сообщение «Связь с базой не установлена», то необходимо воспользоваться пунктом меню «Настройки/Связь» для установления связи с базой данных телекоммуникационной системы.

Сохранить в базу: Пользователю предоставляется возможность сохранить в подключенную базу созданную схему. Для этого служит диалог, аналогичный представленному выше. Пользователь имеет возможность сохранить схему в базе данных, если он обладает соответствующими правами, либо обновить уже существующую схему в базе. Если же вместо этого диалога вы увидите сообщение «Связь с базой не установлена», то необходимо воспользоваться пунктом меню «Настройки/Связь» для установления связи с базой данных телекоммуникационной системы.

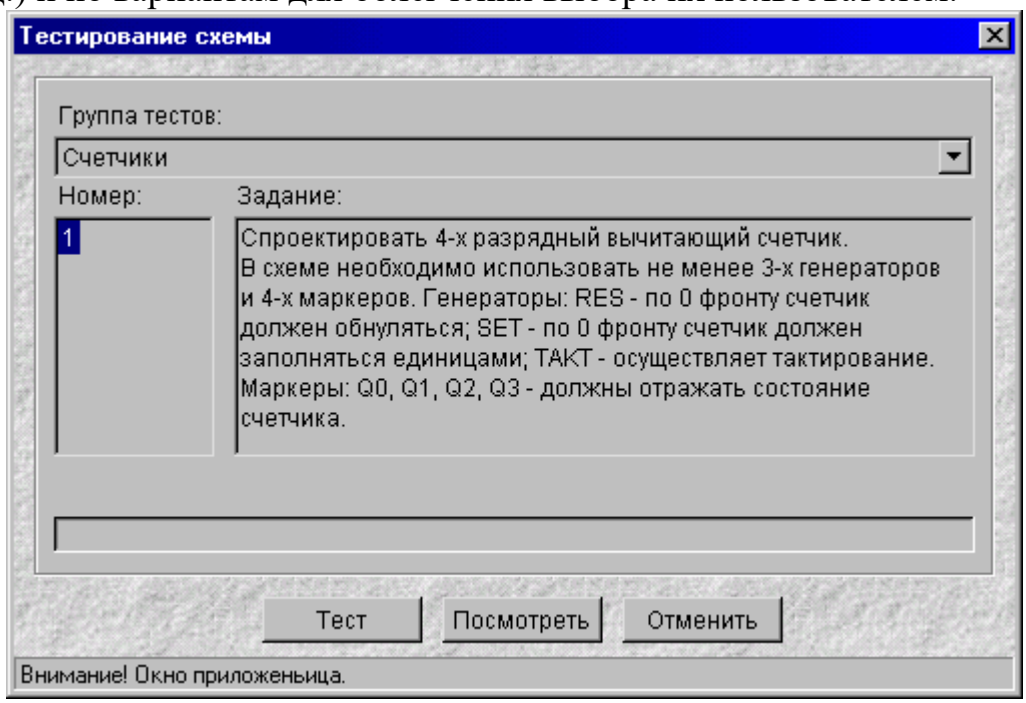
Открыть с URL: Если вам известен URL нахождения созданной схемы, то вы можете открыть ее используя данный пункт меню.



Все что требуется от пользователя, это ввести URL нахождения файла схемы и имя соответствующего файла. Если такой файл действительно существует, то он будет открыт. В противном случае пользователь получит сообщение о неверном URL. При открытии файла схемы также предпринимается попытка открытия файла комментария, который должен называться так же, как и файл схемы, но с расширением *.cmt.

Моделировать работу: Выбором данного пункта меню пользователь инициирует процесс моделирования работы схемы. Время моделирования работы существенно зависит от аппаратных возможностей машины (процессор, объем оперативной памяти и др.), а также от заданного временного интервала моделирования. Интервал может быть любым числом (о том как установить интервал моделирования см. описания пункта меню «Настройки/Настройки»). Поэтому, устанавливая интервал, необходимо руководствоваться аппаратными возможностями машины.

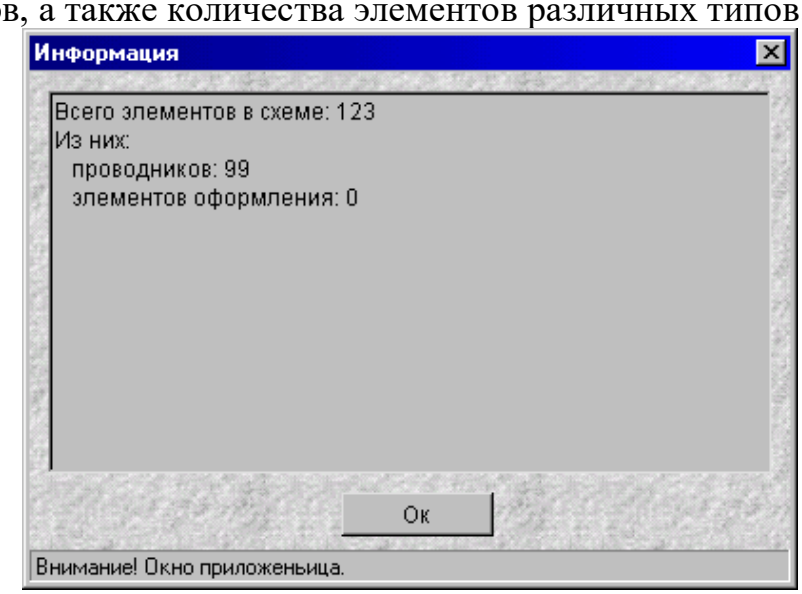
Выполнить тест: При выборе данного пункта меню пользователю представляется диалог, в котором можно выбрать задание на построение схемы какого-либо устройства. Задания структурированы по категориям (регистры, счетчики, триггеры и т.д.) и по вариантам для облегчения выбора их пользователем.



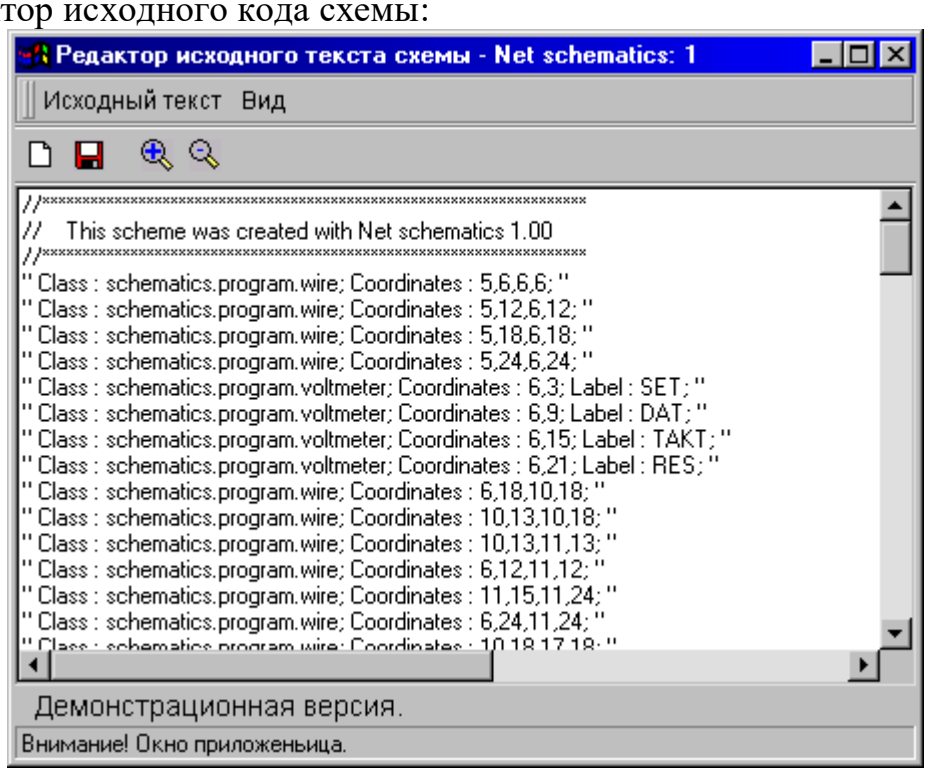
После выбора задания и ознакомления с ним, пользователь может выйти из данного диалога и построить схему соответствующую заданию. После того, как он будет уверен в ее работоспособности, необходимо снова выбрать данный пункт меню и нажатием кнопки «Тест» провести тестирование созданной схемы. Результаты

тестирования сообщаются пользователю. Ему предъявляется процентное соотношение общего количества проведенных над схемой тестов и количества правильно выполненных. Кроме того в данном диалоге присутствует кнопка «Посмотреть», нажатием на которую можно посмотреть временные диаграммы моделирования работы схемы. На данных диаграммах можно увидеть предпринятые тестовые управляющие воздействия и реакцию на них созданной схемы.

Информация: Данный пункт меню позволяет просмотреть пользователю информацию о схеме, т.е. общее количество элементов, а также количества элементов различных типов.



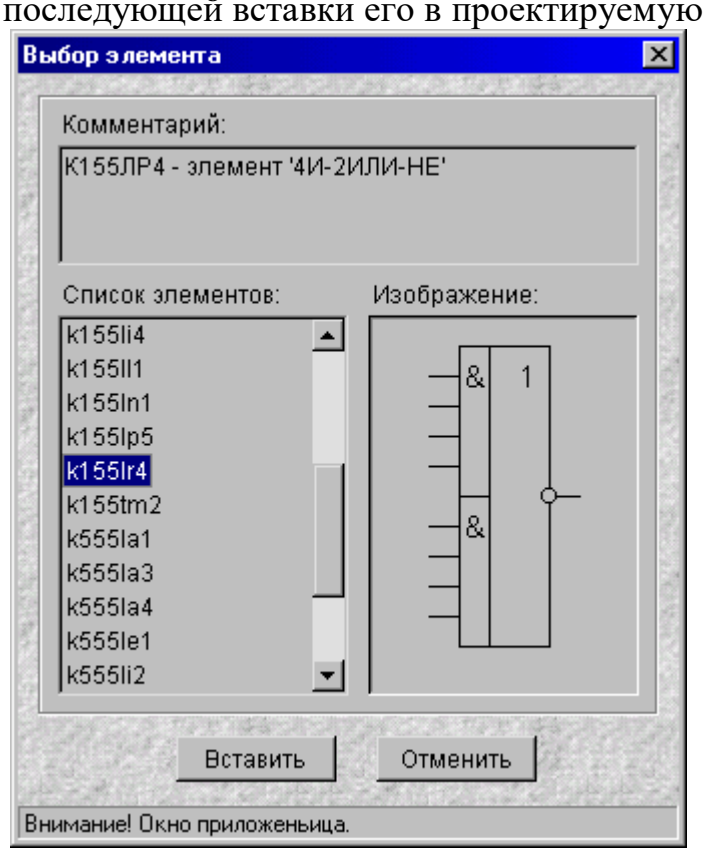
Исходный текст: При выборе этого пункта меню открывается редактор исходного кода схемы:



Исходный код схемы представляется в виде текста созданного по определенным правилам. Поэтому редактор исходного кода схемы представляет собой текстовый редактор с обычными функциями редактирования текста (ввод текста, удаление, перемещение, размножение). Данный редактор позволяет выделять текст и забирать его в системный буфер обмена, что удобно в случае перемещения кода схемы в какие-либо другие приложения.
Выход: Завершение работы и закрытие текущего окна телекоммуникационной системы схемотехнического моделирования.

МЕНЮ РЕДАКТИРОВАНИЕ:

Вставить элемент: Данный пункт меню позволяет войти в диалог выбора элемента из элементной базы телекоммуникационной системы для последующей вставки его в проектируемую схему:



Для вставки необходимо выбрать необходимый элемент из списка элементов и нажать кнопку «Вставить». Узнать некоторую информацию о выбранном элементе можно, обратив внимание на поле «Комментарий». Также для обеспечения наглядности в данном диалоге присутствует поле «Изображение», где представляется схемотехническое изображение выбранного элемента.
Проводник: Выбор данного пункта меню переводит программу в режим изображения проводников. Курсор из обычного превращается в крестообразный, что символизирует о переходе в режим формирования проводников между элементами. После того

как вы изобразили проводник или группу проводников, выйти из данного режима можно повторным выбором данного пункта меню.

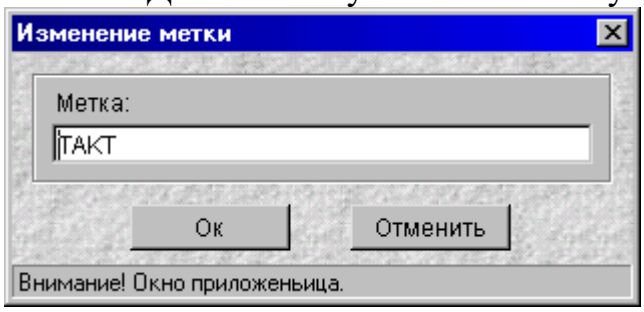
Вставить маркер: Пункт меню позволяет вставить маркер в точку, где необходимо наблюдать работу схемы. Желательно каждому маркеру присваивать уникальную метку, чтобы после моделирования работы созданной схемы вы могли удобно наблюдать временные диаграммы работы схемы.

Вырезать: Выбор данного пункта позволяет вырезать выделенный фрагмент схемы в буфер обмена. Буфер обмена позволяет эффективно редактировать схему, размножать повторяющиеся элементы и фрагменты схемы.

Скопировать: Выбор данного пункта позволяет скопировать выделенный фрагмент в буфер обмена, не удаляя его с рабочего пространства.

Вклеить: Активизирование данного пункта меню позволяет вклеить ранее вырезанный или скопированный фрагмент в необходимое место на схеме.

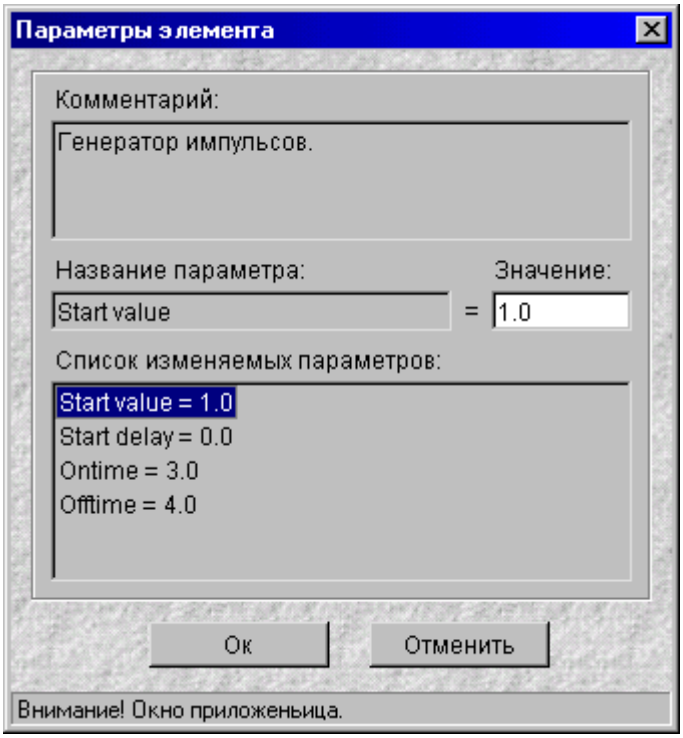
Изменить метку: Каждому элементу схемы можно присвоить любую символьную метку, которая будет отображаться на схеме поверх него. Назначение меток обычно используется для вставляемых маркеров. Это позволяет правильно идентифицировать маркеры на временных диаграммах работы схемы, так как каждой диаграмме присваивается метка соответствующего ей маркера. Выбор данного пункта позволяет изменить метку некоторого выделенного элемента. Для этого служит соответствующий диалог.



Если пользователь не выделил ни одного или выделил более одного элемента, то программа выдаст соответствующее предупреждение.

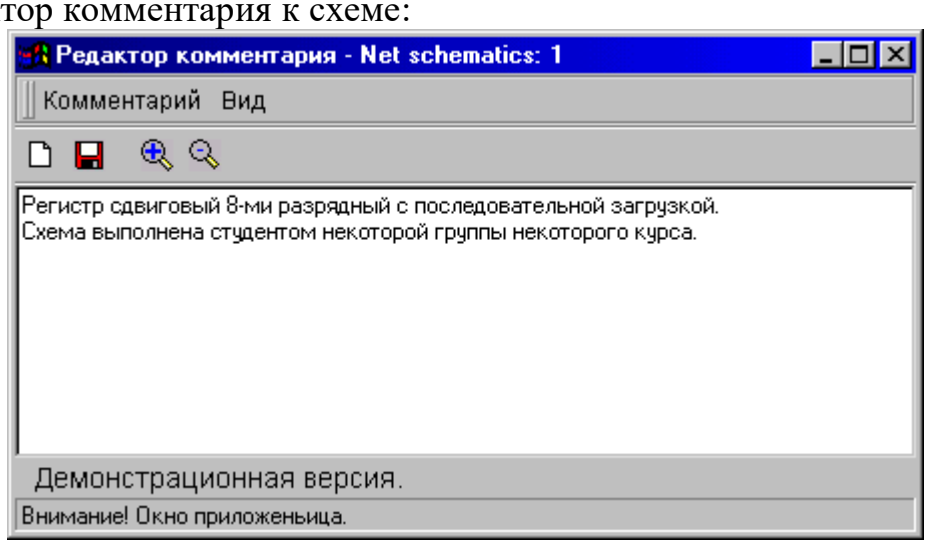
Изменить параметры: Выбор данного пункта позволяет изменить настраиваемые параметры выделенного элемента. Диалог представленный ниже и служит этим целям. Для изменения некоторого параметра необходимо выбрать этот параметр из списка параметров и в поле «Значение» ввести новое значение этого параметра. После ввода значения параметра необходимо нажать клавишу Enter, чтобы измененное значение зафиксировалось в списке параметров. Все параметры представляются вещественными числами, но некоторые параметры элементов могут принимать

только целочисленные значения. Поэтому в некоторых случаях введя вещественное число, оно будет округлено к ближайшему целому.



По окончании изменения параметров необходимо нажать кнопку «Ок». В данном диалоге также присутствует поле «Комментарий», в котором можно наблюдать информацию о данном элементе (его характеристики и др.). Если же элемент не имеет изменяемых параметров, то описываемый диалог не появится, а пользователь будет проинформирован соответствующим сообщением.

Комментарий: При выборе этого пункта меню открывается редактор комментария к схеме:



Редактор комментария к схеме представляет собой текстовый редактор с обычными функциями редактирования текста (ввод текста, удаление, перемещение, размножение). Данный редактор

позволяет выделять текст и забирать его в системный буфер обмена, что удобно в случае перемещения текста в какие-либо другие приложения. Созданный комментарий сохраняется на диск или в базу данных телекоммуникационной системы при сохранении редактируемой схемы.

МЕНЮ ВИД:

Панель инструментов: Пометка данного пункта свидетельствует об отображении в верхней части окна панели функциональных кнопок. Кнопки являются аналогами соответствующих пунктов меню и позволяют облегчить выполнение часто повторяющихся действий. Снятие пометки удаляет панель из окна и все необходимые действия выполняются только через пункты меню.

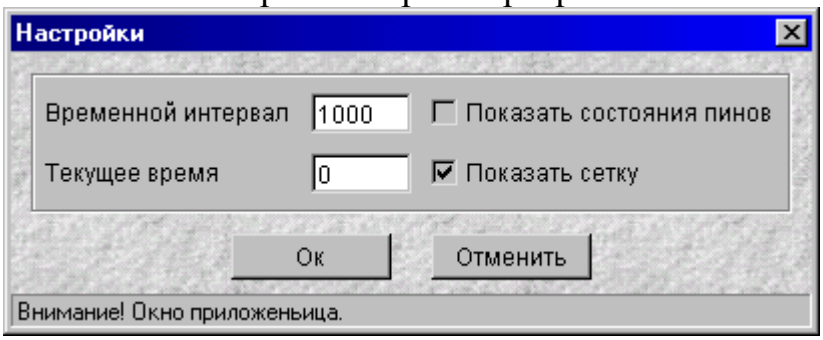
Перерисовать: По выбору данного пункта меню производится перерисовка рабочей области окна программы. И хотя этого никогда не требуется, но данный пункт предусмотрен на всякий случай.

Уменьшить: Уменьшение изображения в рабочем пространстве окна программы. Это позволяет наблюдать более обширные участки разрабатываемой схемы.

Увеличить: Увеличение изображения в рабочем пространстве окна программы. Это позволяет более детально ознакомиться с некоторыми участками схемы.

МЕНЮ НАСТРОЙКИ:

Настройки: Выбор данного пункта меню переводит пользователя в диалог изменения некоторых настроек программы:



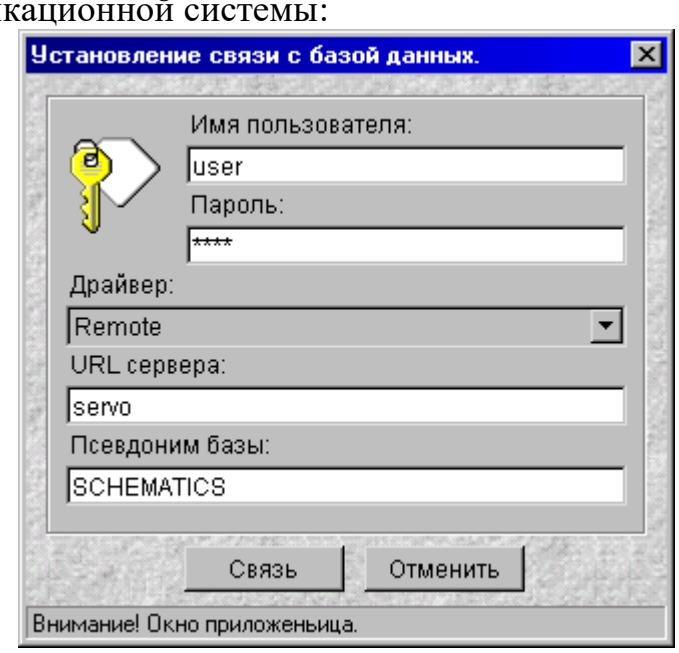
Имеются следующие изменяемые настройки:

- «Временной интервал»: временной интервал эмуляции работы схемы;
- «Текущее время»: в совокупности с пометкой поля «Показать состояния контактов» позволяет отобразить состояния всех контактов элементов в выбранный квант времени;
- «Показать состояния контактов»: пометка данного пункта позволяет отобразить состояния всех контактов элементов в

выбранный квант времени, задаваемый в поле «Текущее время»;

- «Показать сетку»: пометка данного пункта позволяет показать сетку на рабочем пространстве окна программы, которая улучшает ориентацию в рабочем пространстве.

Связь: Выбор данного пункта меню отображает диалог идентификации пользователя и связи с базой данных телекоммуникационной системы:



Имя пользователя и пароль необходимы для идентификации пользователя и предоставления ему соответствующих функциональных возможностей. Пользователь не может назначать себе имя или менять пароль. Регистрация пользователей есть задача администратора системы. В случае если пароль забыт или пользователь не зарегистрирован в базе данных пользователей системы он должен обратиться к администратору. В поле «Драйвер» необходимо указать тип драйвера, обеспечивающего связь с базой данных. Пользователь, запустивший телекоммуникационную систему с Web-сервера должен указать тип драйвера Remote. В поле «URL сервера» необходимо занести имя хоста, с которого запущена система. Имя хоста не должно содержать, ни тип протокола связи, ни каких либо виртуальных каталогов на данном хосте. Если пользователь запустил данную систему с Web-сервера, то в данном поле уже содержится корректное имя хоста. В поле «Псевдоним базы» необходимо внести имя базы данных системы под которым она зарегистрирована на сервере в BDE. Если пользователь не знает псевдоним, он должен обратиться к администратору системы. После заполнения всех полей и нажатия кнопки «Ок» пользователю будет сообщено об

успешности его идентификации и соединения с базой данных или о неуспехе данной операции. В случае неуспеха пользователю будет сообщена причина.

МЕНЮ ПОМОЩЬ:

Помощь: Выбор данного пункта меню позволяет ознакомиться с различной информацией, сопровождающей данную телекоммуникационную систему (инструкции, обучающие курсы и т.д.).

О программе…: Выбор этого пункта позволяет узнать версию системы, условия ее распространения, информацию о разработчиках и др.

8. Экологичность и безопасность проекта

8.1. Введение

Безопасность жизнедеятельности - наука о сохранении здоровья и безопасности человека в среде обитания, призванная выявлять и идентифицировать вредные и опасные факторы, разрабатывая методы и средства защиты человека путем снижения влияния вредных и опасных факторов до приемлемых значений, вырабатывать меры по ликвидации последствий чрезвычайных ситуаций мирного и военного времени.
Охрана труда - система законодательных актов, социально-экономических, организационных, технических, гигиенических и лечебно-профилактических мероприятий и средств, обеспечивающих безопасность, сохранение здоровья и работоспособности человека в процессе труда.

Научно-технический прогресс (**НТП**) внес коренные изменения не только в технологию производства, но и в содержание труда работающих. Современные формы труда характеризуются рядом особенностей. Одной из отличительных способностей **НТП** в современных условиях является тенденция широкого использования персональных электронно-вычислительных машин (**ПЭВМ**) в самых различных сферах человеческой деятельности. Электронно-вычислительная техника стала необходимой частью многочисленных производственных и управленческих процессов. Применение **ПЭВМ** в большей степени изменяет привычный облик многих профессий, в одних случаях устраняя утомляюще-однообразный характер работы, в других - оставляя больше времени для творческого труда, в-третьих - интенсифицируя интеллектуальную деятельность. Однако снижается общая физическая нагрузка, и одной из проблем физиологии труда является решение вопроса борьбы с гиподинамией и монотонией.

Наряду с этим возрастает удельный вес профессий, требующих высокого нервно-психического и нервно-эмоционального напряжения, то есть современные виды труда предъявляют повышенные требования к человеку. Возникла проблема создания такой техники и производственной среды для человека, которая соответствовала бы его анатомо-физиологическим и психологическим особенностям. Эта проблема может быть решена комплексно как с позиций эргономики (улучшений конструкций пультов, терминалов, рабочих мест, средств управления и т.д.), так и с позиций строгой регламентации режимов труда и отдыха, целенаправленной профессиональной ориентацией и т.д.

8.2. Гигиенические требования к видеодисплейным терминалам, ПЭВМ и организация работы

В 1996 году вышли СанПиН 2.2.2.542-96 *«Гигиенические требования к видеодисплейным терминалам, персональным электронно-вычислительным машинам и организации работы»*, где установлены конкретные значения на те или иные параметры ВДТ и ПЭВМ. Выделим основные требования, которые были включены в СанПиН 2.2.2.542-96:

1. Требования к видеодисплейным терминалам и **ПЭВМ**.
   В эти требования включены визуальные эргономические параметры **ВДТ** (параметры безопасности), требования конструкции **ВДТ** (наличие ручек регулировки яркости и контраста) его дизайну (спокойные мягкие тона окраски корпуса, отсутствие бликов), к клавиатуре, средствам индивидуальной защиты (защитный экран).

2. Требования к помещениям для эксплуатации **ВДТ** и **ПЭВМ**.
   В этот раздел входят требования к естественному и искусственному освещению (400 лк), расположению рабочих мест с **ВДТ** и **ПЭВМ**, площади на одно рабочее место (не менее 6 $м^2$), объему (не менее 24 $м^3$), высоте помещения (не менее 4 м), звукоизоляции ограждающих конструкций помещений с **ВДТ** и **ПЭВМ** и т.д.

3. Требования к микроклимату, содержанию аэронов и вредных химических веществ в воздухе помещений эксплуатации **ВДТ** и **ПЭВМ**.
   Таблица: Оптимальные и допустимые параметры температуры и относительной влажности воздуха в помещениях с ВДТ и ПЭВМ:

Оптимальные параметры		Допустимые параметры	
Температура град.С,	Относительная влажность, %	Температура град.С,	Относительная влажность, %
19	62	18	39
20	58	22	31
21	55		

Примечание: скорость движения воздуха - не более 0,1 м/с.

4. Требования к шуму и вибрации.

Уровень шума на рабочем месте не должен превышать 50 дБА. Уровень вибрации зависит от средней геометрической частоты октавных полос, например, при 2 Гц виброскорость равна 79 дБ, 8-63 Гц - 67 дБ.

5. Требования к освещению помещений и рабочих мест с ВДТ и ПЭВМ.
   Освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300-500 лк. Допускается установка светильников местного освещения для подсветки документов. Местное освещение не должно создавать бликов на поверхности экрана и увеличивать освещенность экрана более 300 лк. Также приводится требования к блескости, ослепленности и тд.

6. Требования к организации и оборудованию рабочих мест с ВДТ и ПЭВМ для учащихся высших и средних учебных заведений.
   Здесь указаны требования к размерам стола и стула для сохранения здоровья пользователей ВДТ и ПЭВМ.

7. Требования к организации режима работы с **ВДТ** и **ПЭВМ** студентов высших учебных заведений.
   Установлены ограничения на длительность работы с **ПЭВМ** и **ВДТ** (не более 4 часов), перерывы (15-20 минут). Необходимо проведение упражнений для глаз, физкультминутки (1-2 минуты) для снятия локального утомления, физкультурные паузы и т.д.

8. Требования к организации медицинского обслуживания пользователей ВДТ и **ПЭВМ**.
   Профессиональные пользователи **ВДТ** и **ПЭВМ** должны проходить обязательные предварительные (при поступлении на работу) и периодические медицинские осмотры в порядке в сроки, установленные Минздравмедпромом России и Госкомсанэпиднадзором России.

К сожалению не все требования выполняются на практике. Из-за этого у многих пользователей **ВДТ** и **ПЭВМ** ухудшается здоровье, а прежде всего - зрение.

9. Расчет стоимости разработки

Затраты на разработку системы рассчитываются по формуле:

Зр = Сзп + Сдзп + Сос + Ао + Соп + Рк, (1)

где:

Сзп - затраты на заработную плату;
Сдзп - затраты на дополнительную заработную плату;
Сос - отчисления на социальное страхование;
Ао - амортизационные отчисления;
Соп - стоимость времени на отладку программ;
Рк - косвенные расходы.

Программный продукт разрабатывается программистом со средним окладом 600 рублей в месяц. На разработку затрачено примерно 3 месяца, поэтому затраты на заработную плату составляют:

Сзп = 600 * 3 = 1800 руб.

Дополнительная заработная плата составляет 30% от основной заработной платы:

Сдзп = 1800 * 0.3 = 540 руб.

Отчисления на социальное страхование составляет 37% от основной заработной платы:

Сос = (1800 + 540) * 0.37 = 865.8 руб.

Амортизационные отчисления составляют 20% от стоимости оборудования:

Соб = 3500 руб.

Ао = 3500 *0.2 * 3/12 = 175 руб.

Время, затраченное на отладку программы:

Соп = Сч * Т , (2)

где **Сч** - стоимость одного машинного часа работы ПЭВМ, руб.;

Т - количество времени, затраченное на отладку программы:

Т = 20 * 4 = 80 ч.

т.к. на отладку программы затрачено 20 дней рабочего времени, из них каждый день в среднем затрачивалось по 4 часов машинного времени.

Стоимость одного часа работы на ПЭВМ можно рассчитать по формуле:

Сч = Соб / (Тс * Кр * Кч), (3)

где

Соб - стоимость оборудования;
Тс - срок службы оборудования;
Тс = 5 лет;

Кр - количество рабочих дней в году;
Кр = 240;
Кч - количество часов в смене;
Кч = 8;

Сч = 3500 / (5*240*8) = 0.365 руб.

Стоимость одного часа работы на ПЭВМ составляет 0.365 руб., тогда стоимость машинного времени на отладку программы составляет:

Соп = 0.365*80 = 29.2 руб.

Косвенные расходы составляют 110% от основной заработной платы:

Рк = 1800*1.1 = 1980 руб.

Затраты на разработку составляют:

Зр = 1800 + 540 + 865.8 + 175 + 29.2 + 1980 = 5390 руб.

Плановая прибыль составляет 25% от затрат на разработку:

П = 5390 * 0,25 = 1347.5 руб.

Тогда общая стоимость программы составляет:

К = 5390 + 1347.5 = 6737.5 руб.