Информатизация образования

ПЕТРОВ Олег Михайлович – *доктор технических наук, профессор, зав. кафедрой Московской государственной академии приборостроения и информатики (МГАПИ)*

ЖУКОВ Дмитрий Олегович *- кандидат физико-математических наук, доцент, директор ЦНИТ МГАПИ*

ЗОТОВ Дмитрий Дмитриевич – *инженер-программист ЦНИТ МГАПИ*

АРХИТЕКТУРА WEB-СИСТЕМ ОБУЧЕНИЯ

Излишне говорить о том, насколько актуальной стала проблема построения надежной системы дистанционного обучения, отвечающей всем современным требованиям и стандартам. Почти каждый вуз сегодня имеет несколько аналогов подобных систем в «зачаточном» состоянии, но, как правило, ни одна из них не доживает до «Альфа-версии», то есть до стадии готового продукта. В тоже время множество отечественных и зарубежных фирм пытаются разрабатывать системы дистанционного обучения и позиционировать их как коммерческий продукт. Немногие из них действительно успешны, и причин тому несколько. Главная – это неполное, или вернее неточное понимание фирмой-разработчиком потребностей конкретного вуза. У каждого учебного заведения есть ряд собственных требований к подобной системе, при этом они достаточно схожи, но не настолько, чтобы можно было создать один продукт для всех, соответствующий единому стандарту. Даже если такой стандарт в последствии будет создан, он еще долгие годы будет дополняться и расширяться.

В общем случае судьбу такой коммерческой системы можно проследить следующим образом. Первоначально кафедра или деканат заказывает некий продукт, который будет отвечать тем или иным требованиям. Как правило, этот продукт «вырастает» из студенческой разработки. Фирма выполняет заказ, программа внедряется. Параллельно на другой кафедре внедряется подобная программа другого изготовителя. Проходит несколько лет, и когда для успешного взаимодействия разных подразделений вуза требуется объединить эти несколько программных продуктов в одну систему, выясняется, что их форматы данных не совместимы, и затраты на интеграцию двух существующих систем значительно превысят затраты на написание новой. Таким образом, мы возвращаемся в самое начало пути.

Вторая проблема – это огромная пропасть между двумя группами конечных пользователей системы – студентами и преподавателями. Дело в том, что большинство преподавателей не владеют достаточными навыками в области разработки и эксплуатации компьютерных систем, чтобы подготовить свой предмет для представления в электронном виде. Речь идет не о текстовом файле, а о специализированном мультимедийном формате, способном содержать текст, графику, звук, анимацию и т. п. Преподавателю это и не нужно. А студенты, в свою очередь, в большинстве своем являются «продвинутыми» пользователями, поэтому освоение системы дистанционного обучения происходит у них интуитивно. Вследствие этого, может получиться, что студенты опередят преподавательский состав в освоении системы, что может вызвать негативную реакцию как у тех, так и у других.

Разработчики системы должны дать в руки преподавателям такой инструмент, который позволил бы им полностью контролировать ее работу, не прибегая при этом к специальным областям знаний, таким как программирование, web-разработка, администрирование интернет-серверов и т. д.

Ну и, конечно, не нужно забывать о таких аспектах, как скорость работы, устойчивость и безопасность, которыми порой пренебрегают в угоду дружественному интерфейсу или сокращению затрат.

Исходя из выше сказанного, можно сформулировать основные аспекты, которые нужно учитывать при разработке системы дистанционного обучения.

Система дистанционного обучения должна быть построена таким образом, чтобы быть доступной пользователям в любое время из любого места. Уровень требований к стабильности и надежности передачи и хранения данных является критичным, поскольку по результатам работы студента с системой может быть выставлена оценка. Система должна обладать определенным уровнем гибкости, для того чтобы каждый пользователь (и вуз, и студент, и преподаватель) мог настроить ее «под себя». Также нужно учитывать тот факт, что у каждого учебного заведения могут быть несколько различные представления о структуре и наполнении системы, а, следовательно, нужно создать интерфейс для возможности написания дополнительных модулей сторонними разработчиками [1, 2].

Очевидно, что подобная система может быть построена только на клиент-серверной архитектуре. Поскольку клиент и сервер находятся на разных машинах (в условиях сети Интернет это естественно), возникает один из основных вопросов – вопрос передачи данных. Есть два пути решения этой проблемы: первый – написание клиентского и серверного приложений. Серверное приложение запускается и постоянно работает на стороне сервера, а пользователь каждый раз запускает клиентскую программу со своего рабочего места, соединяется с сервером и работает в режиме on-line. (Существует модель, когда студент получает задание по сети, а выполняет его off-line, но такую модель мы не рассматриваем, поскольку она не отвечает требованиям безопасности и, к тому же, не дает преподавателю возможности контролировать учебный процесс.) Главным недостатком такого подхода является необходимость установки и настройки клиентской программы на компьютер, с которого планируется подключаться к системе. Это делает весьма

сложной или даже невозможной работу из интернет-кафе или с чужого компьютера. Еще один недостаток – привязанность программы к конкретной платформе или операционной системе, что тоже снижает ее универсальность. К достоинствам можно отнести защищенность передачи данных и огромные функциональные возможности языков, на которых пишутся подобные приложения [3].

Вторая возможная логическая модель системы дистанционного обучения – полностью серверно-ориентированная программа, то есть web-приложение. В этом случае абсолютно вся информация хранится на сервере, а клиент соединяется с системой через интернет-обозреватель. Таким образом, снимается ряд проблем, связанных с совместимостью платформ и операционных систем, поскольку большинство обозревателей поддерживают одни и те же стандарты. Конечно, различия в интерпретации кода браузерами MS Internet Explorer, Mozilla Fire Fox для Linux и Opera для MacOS существуют, но адаптация кода под несколько похожих систем значительно менее трудоемкий процесс, чем написание нескольких клиентских приложений для разных платформ.

К недостаткам подобного подхода можно отнести более сложные методы защиты данных и некоторые ограничения функциональности, которые, правда, не очень влияют на конечный результат.

Кстати, говоря о платформах, справедливо будет заметить, что большинство серверов в сети Интернет работают на системах Linux и FreeBSD, а Windows широко распространен в качестве настольной ОС, поэтому кросс-платформенность является одной из наиболее актуальных проблем. Ориентироваться, конечно, лучше на системы семейства Unix (Linux, FreeBSD, OpenBSD и т. п.), поскольку они являются «серверным стандартом». Но и Windows постепенно начинает занимать позиции в мире серверных технологий. Исходя из этого, необходимо выбрать такой язык разработки, который обеспечивал бы одинаково стабильную работу со всеми операционными системами.

Таким языком является PHP. Это гипертекстовый процессор, который генерирует на выходе страницы формата HTML. Функциональность этого языка настолько широка, что трудно представить задачи в рамках системы дистанционного обучения, с которыми он бы не справился.

Затронем еще один вопрос – размещение данных. Очевидно, что в такой сложной и многофункциональной системе данных будет много, и на первых этапах разработки трудно спрогнозировать их конечную структуру и объем. Одно ясно сразу – нам необходимо использовать СУБД. Разработка собственной СУБД – дело долгое и неблагодарное, к тому же это не всегда оправдано при создании коммерческих систем. В нашем случае СУБД должна отвечать следующим параметрам: быть кросс-платформенной, иметь возможность удаленного доступа и взаимодействовать с языком PHP [4].

В общем виде архитектура подобной системы будет выглядеть так (рис.1):

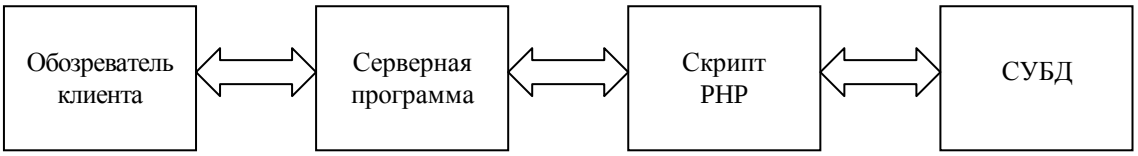

Рис.1. Принципиальная схема взаимодействий в интернет-системе обучения.

Но это всего лишь принципиальная схема работы, в которой каждый из блоков может быть реализован различными способами. Серверную программу, обозреватель клиента и СУБД мы рассматривать не станем, поскольку эти части системы заменяемы, и в рамках данной работы мы не разрабатываем их, а только реализуем взаимодействие между ними посредством PHP скриптов.

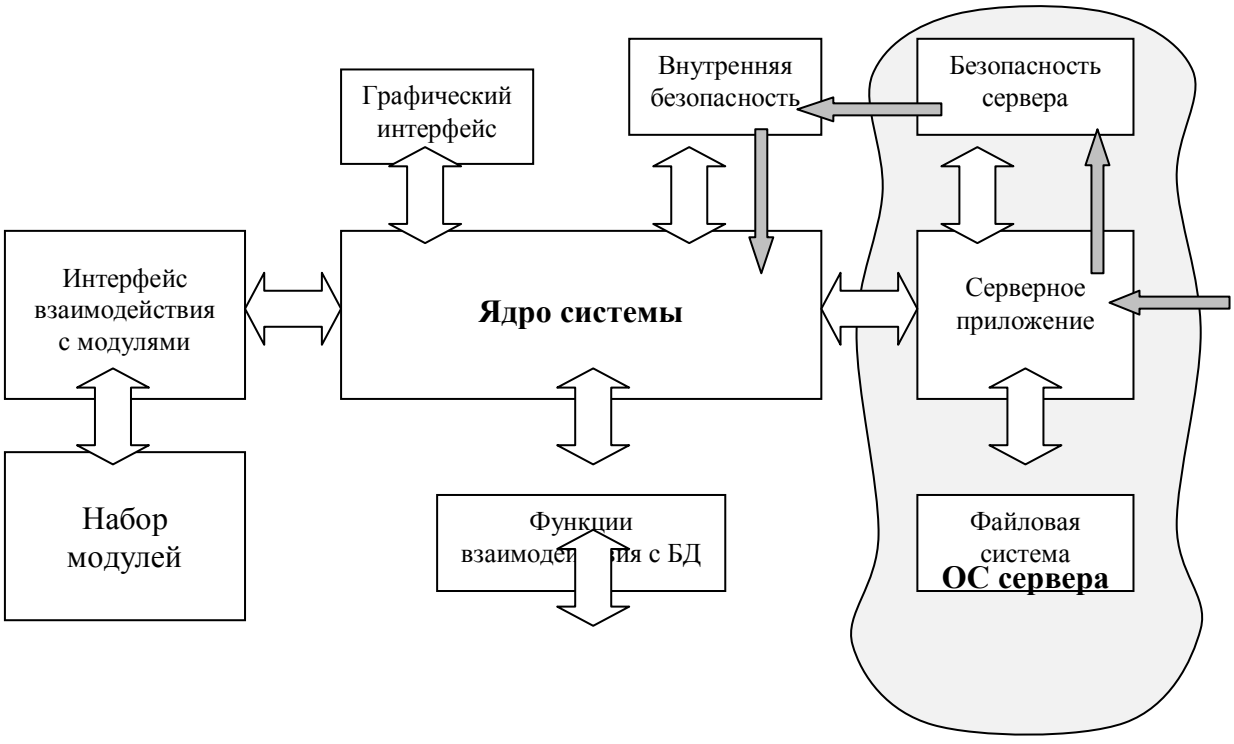

Рис.2. Общие принци ения.

Блок «ядро системы» представляет собой набор функций, решающих базовые алгоритмы системы, а также интерфейсы для взаимодействия с другими блоками. Для наглядного представления материалов системы, а также элементов управления служит модуль графического интерфейса, который генерирует HTML страницы (рис.2). Страницы формируются на основе шаблонов, что позволяет изменить интерфейс или язык системы, не меняя программного кода.

Нужно заметить, что в работе интерфейсной части системы широко используется язык JavaScript. Это вызвано необходимостью выполнять некоторые действия на стороне клиента (открывать, закрывать и обновлять окна, работать с элементами управления), чего PHP сделать не может, поскольку является серверным языком.

Как уже говорилось выше, одним из приоритетных направлений разработки является безопасность системы. Рассмотрим поподробнее некоторые моменты этой проблемы [5].

Каждый пользователь, регистрируясь в системе, получает логин и пароль на доступ. Данные пользователя хранятся в базе данных в зашифрованном виде. При входе в систему пользователь вводит логин, пароль и получает доступ к системе, согласно своим правам на ее использование. Как показано на схеме (темная стрелка), соединение устанавливается через серверную программу, и контроль безопасности осуществляется средствами сервера. Но существует несколько способов обойти этот контроль. Один из таких способов – послать ложный запрос. Дело в том, что все данные между страницами системы передаются методом POST. В целях увеличения быстродействия системы «легальность» того или иного пользователя проверяется только единожды, при входе в систему. Для того чтобы послать запрос, к примеру, на удаление пользователя, генерируется форма со специальным набором данных, которая передается в соответствующий скрипт. Содержимое этой формы можно просмотреть из любого интернет-обозревателя. Теоретически можно создать идентичную форму на другом сайте, и результат выполнения ее запроса будет соответствовать результату легального запроса из системы. (Надо заметить, что для того чтобы создать копию какой-либо из форм системы, необходимо хотя бы один раз войти в систему легально, но теоретически получение структуры форм недобросовестным пользователем возможно.) Если при каждом действии или при открытии каждой новой страницы проверять логин и пароль, нагрузка на СУБД возрастет настолько, что это может привести к системным сбоям. Возникает вопрос: как же защитить данные системы, не перегружая ее [6]?

Выход из подобной ситуации может быть найден следующим способом: в систему добавляется модуль, получивший название «внутренняя безопасность системы». Суть его работы заключается в том, что перед выполнением любого запроса он проверяет, не отсылаются ли в систему пустые поля и, главное, с какого сайта выполнен запрос. Если адрес сайта, с которого выполняется запрос, соответствует адресу системы, то значит, форма не является копией. Поскольку все данные в формы заносятся автоматически, то пустых значений быть не может, соответственно, если значения пустые, то значит, пользователь не вошел в систему, а пытается напрямую попасть на ту или иную страницу. В этом случае система отказывает ему в доступе и информирует администратора системы по электронной почте.

У подобного решения есть один недостаток: при некоторых условиях передача данных задерживается, и тогда система отказывает пользователю в доступе, даже если его запрос был легален. Но и эта проблема решаема. Во-первых, такие задержки возникают крайне редко и только при ошибках операционной системы или соединения, а, во-вторых, перед запуском процедуры проверки можно установить флажок, который будет ждать полной загрузки страницы. Поскольку задержки передачи данных измеряются миллисекундами, то подобный способ защиты, в сравнении с проверкой логина и пароля, значительно быстрее и эффективнее.

Еще одним интересным аспектом является интерфейс взаимодействия с модулями. Под модулями понимаются подпрограммы, которые не обязательны для функционирования системы либо которые будут разработаны после ее создания. К таким модулям можно отнести программу обработки статистики, формирования отчетов и т. п. Дополнительные модули должны также быть написаны на языке PHP и иметь стандартизованную логику входных и выходных данных. В остальном, программный код модуля ни чем не ограничен. Интерфейс взаимодействия с модулями считывает содержимое каталога, в который они помещены, подключает файлы к системе и специальной командой выводит в дополнительное меню функции этих модулей. (Каждый модуль должен содержать стандартную функцию вывода в меню.) Таким образом, зная правила написания модулей можно расширять и дополнять систему, не изменяя ее основного кода.

По большому счету, подобная система модулей аналогична plugin-ному принципу построения с одним лишь различием: код PHP не нуждается в компиляции, поэтому файлы хранятся в открытом текстовом виде.

Мы говорили о модульной технологии в привычных рамках, когда модульный интерфейс является частью основной программы, как бы приятным дополнением. Но попробуем взглянуть шире. Что если ядро системы содержит только расширенный и усовершенствованный интерфейс взаимодействия с модулями, а все остальные части системы подключаются как плагины? Конечно, подобный подход несколько усложнит взаимодействие между частями системы, предположим даже, что список модулей не будет считываться автоматически из определенной директории, а будет составлен вручную в некоем конфигурационном файле, в котором также могут быть указаны типы связей между ними. Усложняя первоначальный этап разработки и последующей «сборки» системы, мы открываем и широчайшие горизонты для ее роста и развития. Принцип «полной модульности» позволит каждому вузу собирать систему именно такой, какой она ему нужна, обеспечивая масштабируемость в самом широком смысле слова. Также нельзя забывать и о переносимости, поскольку можно создать набор модулей для работы с различными файловыми и операционными системами [7].

Как уже было сказано в начале статьи, систем дистанционного обучения много, некоторые из них успешно функционируют, некоторые – нет. Но на сегодня основной задачей разработчиков является не написание одной конкретной системы с ее плюсами и минусами, а разработка единого стандарта для подобных систем, что в последствии поможет развить отечественные разработки до конкурентоспособного уровня западных аналогов.

Литература:

1. Голубятников И. В. Основные принципы проектирования и применения мультимедийных обучающих систем. - М.: «Машиностроение», 1999 - 318 с.
2. Барановская Т. П. Архитектура компьютерных сетей и систем. - М.: «Финансы и статистика», 2003 – 256 с.
3. У. Столингс. Структурная организация и архитектура компьютерных систем. - «Вильямс», 2002 – 896 с.
4. Э. Таненбаум, М. Ван Стен. Распределенные системы: принципы и парадигмы. - СПб.: «Питер», 2003 - 880 с.
5. Щербаков А. Ю. Введение в теорию и практику компьютерной безопасности. - «Издатель Молгачева С. В.», 2001 – 352 с.
6. Девянин П. Н. Модели безопасности компьютерных систем. - М.: «Академия», 2005 – 143 с.
7. Хорошевский В. Г. Архитектура вычислительных систем / Учеб. пособие для вузов. - М.: МГТУ имени Н. Э. Баумана, 2005 – 511 с.