ОБЩИЕ ПРИНЦИПЫ ПОСТРОЕНИЯ ПРОГРАМНЫХ КОМПЛЕКСОВ ТЕСТИРОВАНИЯ ПРИ РЕАЛИЗАЦИИ ДИСТАНЦИОННЫХ ОБРАЗОВАТЕЛЬНЫХ ТЕХНОЛОГИЙ.

Басов В.А., Васьковский А.Н.
Коломенский филиал НАЧОУ ВПО Современная гуманитарная академия
Коломна, Россия

Последнее время в практику работы многих ВУЗов входит использование «информационно-коммуникационных (дистанционных) образовательных технологий - ДОТ». Главным достоинством ДОТ – является преодоление расстояния между преподавателем или источником учебной информации и учеником. Специфика работы ВУЗа при этом состоит в четкой организации информационных ресурсов и принципов взаимодействия с ними. Создание програмно-аппаратных комплексов тестирования при реализации ДОТ становится одной из важнейших задач, по обеспечению качества обучения.

В настоящее время существует множество программных продуктов, позволяющих проводить тестирование знаний учащихся. Одни из них обладают большим набором тестовых шаблонов и компонентов, другие имеют развитые сетевые средства и позволяют взаимодействовать с различными базами данных. Основной их недостаток – сложность расширения функциональности и набора тестовых компонентов, которые нельзя «собрать» из стандартных элементов. Например, отображаемые апплетами 3D модели (рис 1).

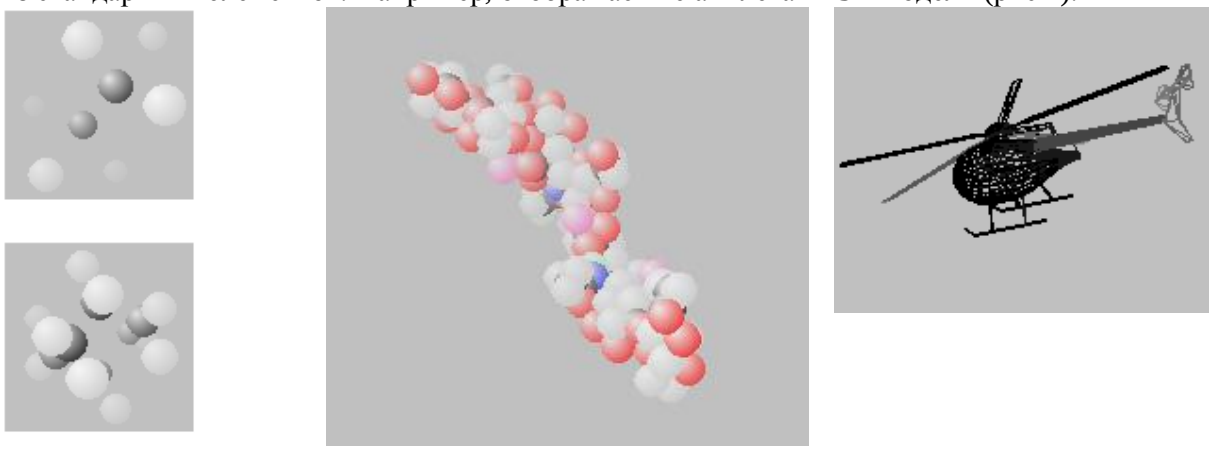

Рисунок 1. Demo апплеты из Java Developer Kit

Это значительно ограничивает возможности создания тестов с элементами сложного моделирования предметной области (особенно в сфере естественных наук: физике, химии, биологии).

Опыт эксплуатации информационных систем дистанционного образования в Коломенском филиале НАЧОУ ВПО Современная Гуманитарная Академия, позволил сформулировать основные принципы построения современного комплекса программ тестирования знаний учащихся.

1.Интегрированность компонент и расширяемость их функциональности

Задача: Обеспечить расширяемость создаваемой образовательной системы, путем применения плагинов, созданных сторонними разработчиками. Подобные плагины позволят внедрить в систему компоненты различных видов: эмуляторы различного типа, сложные динамические трёхмерные модели предметной области и т.д. Платформа должна предоставлять разработчикам плагинов инструментарий для обращения к хранилищам данных и средства сетевого взаимодействия, позволяющие абстрагироваться от конкретных реализаций данных технологий и сосредоточиться на моделировании предметной области.

Механизмы реализации:
- Java плагины. Java обеспечивает простой удобный и надёжный механизм загрузки плагинов и работы с ними, используя Reflection API. Java апплеты являются стандартной кроссплатформенной технологией, позволяющей внедрить в HTML документ сложные активные компоненты.
- .Net плагины. Также поддерживает возможность загрузки плагинов и Reflection API. Позволяет использовать в браузере ActiveX компоненты. Технология уступает Java в вопросах безопасности и переносимости приложения.

2. Хранение данных. Абстракция хранилища данных.
Задача: Абстрагировать программные модули от конкретной реализации технологии хранения данных (локальные и сетевые файлы, полученные по различным протоколам, базы данных различного типа). Предоставить системному администратору возможность самому выбирать тип хранилища, максимально приспособленный под его цели и задачи.
Механизмы реализации:
- сетевые платформы, обеспечивающие средства унифицированного доступа к базам данных. Технологии Java и .Net предоставляют средства абстракции хранилищ данных, позволяющие в рамках одного программного интерфейса переносить сложные объектные данные как в файловые хранилища (сериализация), так и в базы данных (объектно-реляционое отражение). Но в данном случае лучше реализовать собственную технологию в рамках образовательной платформы, предоставляющую более простой и удобный интерфейс для разработчиков учебных продуктов.

3. Сетевое взаимодействие.
Задача: Обеспечить взаимодействие компонент образовательной системы. Предоставить разработчикам плагинов возможность абстрагироваться от конкретных технологий и протоколов сетевого взаимодействия, разрабатывать приложение так, как если бы оно было локальным.
Механизмы реализации:
- использование технологии работы с удаленными объектами
1. CORBA – самый старый механизм работы с удаленными объектами. Поддерживается взаимодействие с программами на C++, Java, .Net.
2. RMI – более простой механизм работы с удаленными объектами, реализованный в Java с помощью Reflection API.
3. .Net Remoting – аналогичная технология в рамках .Net. Позволяет внедрять дополнительные звенья для шифрования данных.
Недостатки данных технологий: сложность реализации и необходимость запуска отдельного сервера для обеспечения работы с удаленными объектами и возможные конфликты с антивирусным программным обеспечением. Альтернативой является реализация отдельной технологии подобного типа, обладающей простым интерфейсом, позволяющей внедрять промежуточные протоколы, и работающей через HTTP или HTTPS, что снимает проблемы с брандмауэрами.

4. Синхронизация хранилищ данных.
Задача: Обеспечить возможность автономной работы локальных версий образовательной среды и синхронизацию данных в подобных версиях с центральной системой. Это позволит создать несколько промежуточных звеньев, работающих автономно, что важно при отсутствии постоянного сетевого подключения.
Механизмы реализации:
Реализуется при создании хранилища данных, его внутренними механизмами репликации.

5. Кросплатформенность.

Задача: Образовательная система должна иметь возможность запускаться на различных типах устройств, работающих под управлением различных операционных систем, предоставляя системному администратору возможность самому выбирать оптимальную для его нужд платформу.
Механизмы реализации:
- Java – кросплатформенная среда, поддерживаемая большинством мобильных устройств и операционных систем на уровне двоичного кода. Для большинства операционных систем Java платформа реализована несколькими независимыми разработчиками: Sun JDK, Open JDK, GCJ, Jikes и др.

- .Net – поддерживается некоторыми мобильными устройствами. Переносимость на другие операционные системы неполная, на уровне исходников, многие библиотеки не являются кросплатформенными и защищены от переноса патентами Microsoft.

5. Свободное программное обеспечение.

Образовательная платформа должна поставляться с открытыми исходными кодами для того, чтобы разработчики плагинов могли ознакомиться с её реализацией, и быть доступной для редистрибьюции.

6. Использование облачных вычислений в ДОТ.

Задача: Обеспечить взаимодействие между различными образовательными системами через Интернет, при оптимальном распределение нагрузки между локальными и удаленными серверами.
Механизмы реализации: Облачные (рассеянные) вычисления (англ. ***cloud computing***, также используется термин ***Облачная (рассеянная) обработка данных***) - технология обработки данных, в которой компьютерные ресурсы и мощности предоставляется пользователю как Интернет-сервис. Пользователь имеет доступ к собственным данным, но не может управлять и не должен заботиться об инфраструктуре, операционной системе и собственно программном обеспечении, с которым он работает.
Примеры реализации:
Sun Cloud — сервис облачных вычислений, предоставляемый корпорацией Sun Microsystems. Он использует открытые технологии, такие как Solaris 10, Sun Grid Engine и платформа Java. [http://ru.wikipedia.org/wiki/Sun_Cloud]
Amazon Elastic Compute Cloud (Amazon EC2) — Веб-сервис, который предоставляет вычислительные мощности в облаке. Сервис входит в инфраструктуру Amazon Web Services [http://ru.wikipedia.org/wiki/Amazon_EC2]

Основными архитектурными элементами программного комплекса системы тестирования при этом будут являться:
-***Основной сервер (ОС)*** – хранит данные о студентах, для каждого студента: учебный план, данные об успеваемости и результаты прохождения занятий.
-***Сервер (облако) учебного продукта (сервиса) (СУП)*** – осуществляет процесс прохождения занятия. Отправляет на основной сервер результаты прохождения занятия.
-***Сервер мультимедиа контента (СМК)*** – хранит тяжелый мультимедиа контент, который проблематично загружать через Интернет в режиме реального времени, используя низкоскоростные каналы связи.
-***Клиент (браузер)*** – отображает учебные продукты, обеспечивает взаимодействие с серверами, для учебного менеджера обеспечивает доступ к данным об успеваемости, для разработчиков учебных продуктов доступ – доступ к редактору учебных продуктов.

-Инструментарий разработчика (ИР) – шаблоны к интегрированным средствам разработки приложений, снабженные необходимыми библиотеками и документацией для создания плагинов и шаблонов учебных продуктов.

Реализуется следующая схема процесса прохождения тестирования:

Формат:
Инициатор [=> Сервер.] Процесс (входящее сообщение) {Действия} [=> Инициатор (возвращаемое сервером сообщение)]

1. ***Клиент*** => ***Основной сервер***. *Вход в систему* (логин, пароль){
    1.1. Авторизация (логин, пароль)
    1.2. Загрузка учебного плана, данных об успеваемости, расписания занятий
        Для каждого задания учебного плана определяется (либо хранится в учебном плане)
        - информация о сервере, предоставляющем сервис задания для данного студента и
        - информация о сервере, хранящем мультимедиа данные для данного задания (для данного студента).
    1.3. Передача учебного плана студенту в ответ на запрос
        (или сообщение об ошибке авторизации)
} => ***Клиент*** (учебный план)

2. ***Клиент***. Выбор занятия (учебный план){
    На основе полученного учебного плана студент выбирает задание,
    и обращается к серверу выбранного занятия, передавая ему необходимые параметры:
    - идентификатор сессии (в начале 0)
    - статус сессии
    - идентификатор занятия
    - имя основного сервера, хранящего данные об успеваемости и учебном плане
    - имя сервера, с которого планируется загружать мультимедиа данные
}

3. ***Клиент*** => ***Сервер выбранного занятия.*** *Цикл обработки сообщений*
        (идентификатор сессии, статус сессии и дополнительные параметры){
    3.1.1 Для первого запроса: Инициализация сессии, загрузка плагина сервиса
    3.1.2 Для последующих: Получение параметров сессии и загруженного сервиса
    3.2 Вызов функции обработки запросов данного сервиса
    3.3 Если сессия завершена - отправляет на основной сервер результаты занятия и высвобождает ресурсы.
    3.3 Передача клиенту следующего документа
            (следующий вопрос, или результат, или сообщение)
} => ***Клиент*** (документ)

4. **Клиент**. *Загрузка документа* (полученный документ){
    Клиент получив от сервера документ, отображает его,
    если для просмотра документа необходимы мультимедиа данные
    или дополнительные плагины, обращается к мультимедиа серверу.
}

5. ***Клиент*** => ***Мультимедиа сервер.*** *Загрузка мультимедиа* (имя файла){
    Загрузка запрашиваемого файла и отправка его клиенту.

} => ***Клиент*** (файл)

6. ***Клиент.*** *Прохождение занятия* (документ, мультимедиа, плагины учебных продуктов){

Студент проходит занятие. По мере прохождения занятия клиент отправляет на сервер сообщения, содержащие:

- идентификатор сессии (содержится в документе)
- статус сессии (при инициации изменения статуса)
- другие параметры, специфичные для каждого сервиса (формируется документами и апплетами)

Далее переходит к п.3

В конце обучения отображается страница с результатами и клиент либо завершает работу с программой, либо переходит к п.1.2

}

Выполнение указанных принципов позволит построить современный комплекс тестирования, для использования на всех этапах реализации ДОТ.