Подсистема обработки характеристик пуска-выбега позволяет выделять участки переходных характеристик, осуществляет их аппроксимацию, преобразование единиц измерений, представление в виде графиков, сохранение рассчитанных характеристик в файлах специального формата для дальнейшей обработки, вычисление ИЗП, сравнительный анализ характеристик, полученных в разное время.

Подсистема обработки временных реализаций вибросигналов предоставляет широкий спектр возможностей по их преобразованию и исследованию. Интегрирование однократное и двойное позволяет перейти от единиц виброускорения к единицам виброскорости и виброперемещения. Реализованы разнообразные виды цифровой обработки сигналов: низкочастотная, высокочастотная и полосовая цифровая фильтрация (рекурсивная и методом частотного окна), спектральный и полосовой частотный анализ, кепстральный анализ, выделение огибающей сигнала и определение её спектра, сглаживание сигнала с использованием разностей, удаление низкочастотных дрейфов, вычисление пик-фактора, построение разностных спектров, вейвлет-анализ. Для более полного исследования вибросигналов, полученных на механизмах с вращательным движением, имеется возможность определить точное значение амплитуды спектральной составляющей, частота которой не кратна частотному разрешению спектрального анализа, и выполнить порядковый частотный анализ. Также реализована статистическая обработка. Все результаты представляются в графическом и численном виде. По указанию оператора выбираются параметры, значения которых записываются в специальные файлы и используются в дальнейшем в качестве ИЗП в системе диагностирования.

АРМ оценки технического состояния и диагностирования является элементом следующего уровня иерархии. Исходные данные для его функционирования получаются на более низких уровнях. Адаптация данной системы производится под конкретный технический объект на протяжении определенного времени эксплуатации. За этот период: формулируется множество дефектов, требующих обнаружения; определяется множество ИЗП, характеризующих техническое состояние объекта; устанавливаются базовые значения ИЗП; для каждого типа дефекта устанавливается подмножество ИЗП, по изменению значений которых можно идентифицировать проявление дефекта; формулируются решающие функции для обнаружения дефектов; определяется множество рекомендаций по устранению выявленных дефектов; создается подсистема выбора набора этих рекомендаций для конкретно возникающих ситуаций. В дальнейшем система диагностирования функционирует по разработанным правилам и продолжается ее совершенствование. При достаточном уровне развития ее функционирование может быть организовано в режиме реального времени, а информация непосредственно выдаваться оперативному персоналу.

Данный комплекс реализован в различных конфигурациях и эксплуатируется на ряде предприятий Беларуси.

О НОВОМ ВАРИАНТЕ СИСТЕМЫ ТЕСТИРОВАНИЯ — ЧТО И КАК

Ванюков С. В., Теут А. А.

Брестский государственный университет им. А.С. Пушкина, г. Брест

Своя собственная система тестирования знаний учащихся есть сейчас практически в любой школе, колледже, вузе. Большинство из них написаны учениками и преподавателями, поэтому их файловые форматы, возможности и даже принципы взаимодействия весьма различаются. Большинство из них разработаны для строго определённого типа тестов.

Начиная разработку системы тестирования TENMA, нам была поставлена задача создать, по возможности, универсальный инструмент, который можно будет легко расширить и приспособить для любой системы тестирования. В качестве инструмента раз-

работки был выбран язык C# на платформе Visual Studio .NET 2.0. Предоставляемые ею средства для сетевого программирования, построения классов и шифрования передаваемых данных оказались подходящими для решения поставленной перед нами задачи.

Основное внимание при проектировании мы собираемся уделить открытости системы и простоте её обновления.

Версия 1.0, которая находится сейчас в разработке, позволит проводить тестирование в режиме «вопрос-ответ» для различных типов тестов. В частности, она должна позволить:

– строить древовидно организованные структуры связей, благодаря которым разработчик тестов сможет моделировать динамически организованный диалог «опрашивающий-опрашиваемый»; и

– обеспечить поддержку импорта тестов формата TQF, который применяется в настоящее время в системе теоретического тестирования на математическом факультете БрГУ.

– в дальнейшем обеспечить тестирование практических задач. Мы полагаем, что такая возможность будет реализована в следующих версиях данной системы. Принцип же работы системы мы планируем оставить таким же, как и у тестирования теоретического.

Пакет состоит из двух частей — клиентской и серверной. Клиентская часть инсталлируется на все компьютеры, за которыми производят тестирование. Серверная часть устанавливается только на один, с которого и осуществляется администрирование. Пакет содержит следующие программы:

1. Клиентская часть

– TENMAH Client — клиентская часть, на которой и осуществляется тестирование.

2. Серверная часть

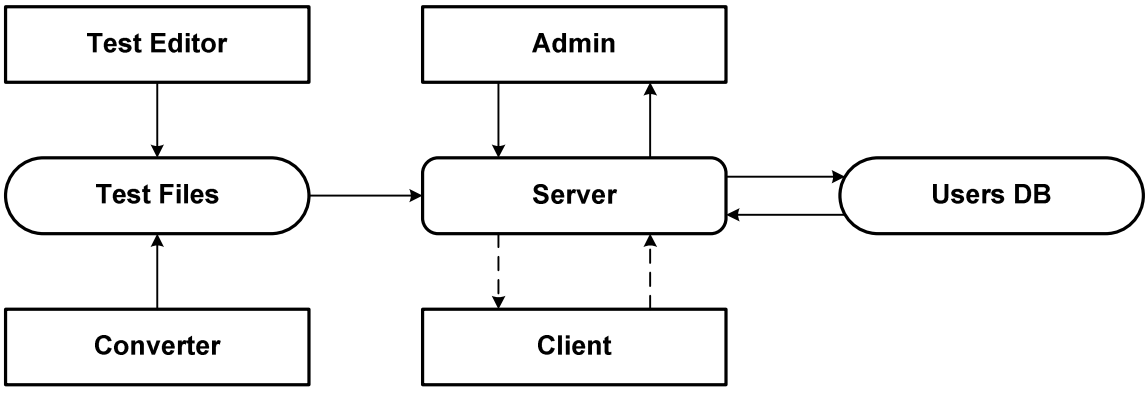
– TENMAH Server — сервер тестирования, оформленный в виде службы Windows. Отвечает за обмен данными и сохранение результатов. Для доступа к настройкам и информации сервера используется TENMAH Admin;

– TENMAH Admin — оболочка для доступа к серверу. В текущей версии доступна только с того же компьютера, на котором запущен сервер;

– TENMAH Converter — небольшая утилита для преобразования из TQF в TNM — формат тестов TENMAH;

– TENMAH Test Editor — программа для создания и редактирования тестов.

Для работы тестирующего комплекса потребуется работающая сеть, ОС Windows не ниже 98, установленный .NET Framework 2.0 (он включен в дистрибутив и устанавливается автоматически) и пакет Microsoft Office 97 или выше (для доступа к базам данных). Никаких специализированных программ (локальные сервера, СУБД) не требуется. Более того, нет даже жёстких требований к головной машине сети — сервером может стать любая машина, подключённая к сети. Структура взаимодействия компонентов программы будет выглядеть следующим образом:



Непрерывными линиями обозначены взаимодействия на локальном компьютере, прерывистыми — сетевые.

Всё управление сервером осуществляется через оболочку TENMAH Admin. Данные о группах и тестируемых загружаются из базы Users DB. Данные самих тестов берутся из файлов Test Files.

Соединение с клиентом осуществляется посредством технологии .Net Remoting. Клиенту отправляется только форма вопросов, оформленная в виде блока HTML кода, все сопутствующие изображения также передаются вместе с ним. Ответ клиента посылается обратно на сервер и уже там обрабатывается. Результаты сохраняются в Users DB.

Формат теста TNM основан на языке XML и позволяет создавать весьма сложные тесты с различными типами ответа на один и тот же вопрос, ветвлениями вопросов и ступенчатыми ответами. Кроме того, архитектура XML, включающая в себя средства XLST, позволяет легко и удобно обновлять старые версии.

Достаточно ли всего этого, чтобы система тестирования была полноценной и универсальной? Для этого попытаемся определить основные требования, которые обычно предъявляются к программе, тестирующей теоретические знания или навыки программирования? Мы будем рассматривать только аналогичные TENMAH системы простого сетевого тестирования — когда имеется несколько компьютеров, объединенных между собой в сеть, на которых различные пользователи запускают клиентские части программы и проходят на них тестирование, данные о результатах которого можно впоследствии сохранить в некоторой общей базе данных. С точки зрения программирования, такими требованиями будут:

1. ***Удобство администрирования***. Серверная часть должна представлять администратору все необходимые возможности для отслеживания процесса тестирования, но при этом (желательно) работать и без его участия. В TENMAH это реализовано посредством разделения серверной и администрирующей частей.

2. ***Совместимость***. Программа должна поддерживать уже существующие форматы тестов или поставляться с конвертером. Крайне желательно, чтобы отчёт о результатах тестов был представлялся в известном и удобном формат. Как уже было упомянуто, в программу включены возможности для поддержки из существующих форматов. В будущем их можно легко расширить, добавив распознавание и других популярных форматов.

3. ***Полнота***. Всё необходимое для работы программы должно быть включено в её дистрибутив, чтобы не создавать проблем с инсталляцией. Все необходимые библиотеки .NET Framework включаются в дистрибутив и инсталлируются автоматически. Серверная часть программы является полноценным локальным сервером, и сама осуществляет все необходимые операции.

4. ***Универсальность***. Программа должна поддерживать различные типы вопросов и ответов. Например, вопрос может содержать картинки и текстовое форматирование, ответ может быть только одним из предложенных, набором из нескольких предложенных вариантов или вводиться с клавиатуры. Однако помимо обычного теста по принципу «вопрос-ответ» возможны и другие виды тестов — например, на написание программы, которая, получая из командной строки входные данные, выводит на экран результат вычислений. Получив эти результаты, мы можем сравнивать их с эталонными и определить, насколько точно решена задача. Действующий формат тестов TNM обеспечивает самые широкие возможности в плане построения вопросов. В будущих будет добавлен и модуль для практического тестирования, работающий по тому же принципу.

5. ***Лёгкость апробирования***. Особенностью любой сетевой системы тестирования является то, что с ней одновременно работает несколько человек. Произвести её испытания в условиях, близких к «рабочим», весьма непросто — не у всякого разработчика найдётся под рукой необходимое количество компьютеров. Даже представляя готовый продукт заказчику, нам при обычной схеме требуется как минимум два компьютера. Именно поэтому крайне желательна независимость программы от внутреннего устройст-

ва сети. В идеальном случае и сервер, и клиент (а, возможно, даже несколько клиентов) должны запускаться на одном и том же компьютере и взаимодействовать точно так же, как если бы они были запущены на различных. Благодаря технологии .NET Remoting, клиент и сервер можно запускать в том числе и на одном и том же компьютере.

6. ***Защита данных***. И клиентская, и серверная часть должны быть устойчивы к различного рода атакам. Клиентская часть требует защиты в первую очередь для внутренних данных, серверная — от атак прямым перебором паролей и «отказ в обслуживании» (DDOS атаки). Даже без применения криптографии сама структура программы делает невозможным большинство методов взлома. Узнать правильный ответ, работая с клиентом, невозможно в принципе: сравнение и анализ ответов производится только на сервере. Перехват трафика и попытка «обмануть» сервер поддельными пакетами возможны только в том случае, если известна структура запроса, то есть, фактически, если на руках у злоумышленника будут полные исходники клиентской и серверной части.

7. ***Мобильность***. Сеть любого университета или даже школы — сложная и хорошо отлаженная система. Программа должна быть проста в установке, настройке и апробировании. Кроме того, она не должна требовать существенного изменения текущих настроек сети. В нашем случае мобильность реализована за счёт использования технологии .NET Remoting, которая целиком абстрагирует разработчика от специфики данной сети.

8. ***Высокая отказоустойчивость***. Программа должна чётко и грамотно реагировать на сбои в работе сети, потерю пакетов, внезапный разрыв связи, уметь взаимодействовать с различными системами защиты (антивирусами, файрволлами, системами защиты на самих узлах связи). В случае ошибки и администратор, и пользователи должны чётко знать, что произошло и как устранить возникшую проблему. Благодаря системе исключений C# и отказу от технологии сетевого программирования через сокеты, система тестирования будет весьма устойчива. Так, в случае обрыва связи она может просто дождаться, когда связь восстановится и дать пользователю возможность продолжить тестирование.

ПРИМЕНЕНИЕ СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ НА ОСНОВЕ ПРЕЦЕДЕНТОВ В СИСТЕМАХ ДИАГНОСТИКИ ТЕХНИЧЕСКОГО СОСТОЯНИЯ МЕХАНИЗМОВ С ВРАЩАТЕЛЬНЫМ ДВИЖЕНИЕМ

Гончарова С. А.

Белорусский государственный университет информатики и радиоэлектроники, г. Минск

В настоящее время системы поддержки принятия решений (СППР) используются во многих сферах жизнедеятельности человека: в медицине для постановки диагноза, в промышленности для диагностирования технического состояния оборудования, в экономике для принятия управленческих решений и даже в философии для построения высказываний. Одним из способов организации системы поддержки принятия решений является построение системы на основе прецедентов.

Вывод на основе прецедентов – это метод, основанный на использовании знаний о ситуациях и случаях из предыдущего опыта (прецедентов). Прецедент в СППР представляет собой описание случая в виде набора информативно значимых признаков и их значений, совокупность действий, предпринятых в данных условиях (набор рекомендаций), а также возможно сохранение признака ошибочности решения проблемы, для исключения подобных ситуаций в будущем. Как правило, такие методы рассуждений включают в себя четыре основных этапа, образующие так называемый цикл рассуждения на основе прецедентов или CBR-цикл (Case-Based Reasoning) [1]:

- извлечение наиболее соответствующего (подобного) прецедента (или прецедентов) для сложившейся ситуации из библиотеки прецедентов (БП);