Исполнимость спецификаций предполагает формализованное построение языка спецификации и наличие «исполнителя» − соответствующей программной поддержки генерации загрузочных кодов либо интерпретации спецификаций.

Инструментально средства разработки могут строиться как совокупность информационного обеспечения (это библиотеки описаний проектов и ЯВУ-описаний проектов) и программного обеспечения. Последнее включает подсистемы первичной обработки описаний и генерации исполнимых спецификаций. Кроме этого, система должна обеспечивать необходимую информационную поддержку с учетом иерархичности проектов, степени их детализации и версий описания.

Опыт использования указанных подходов для построения средств обучения разработке и проектированию программ, обучения имитационному моделированию применительно к системам проектирования со входным языком VHDL для производственных систем приведен в [5−8].

ЛИТЕРАТУРА

1. Орлов, С.А. Технологии разработки программного обеспечения / С.А. Орлов. − СПб.: Питер, 2004. – 527 с.

2. Липаев, В.В. Программная инженерия. Методологические основы: учеб. / В.В. Липаев; Гос. ун-т − Высшая школа экономики. − М.: ТЕИС, 2006. − 608 с.

3. Маклаков, С.В. Создание информационных систем с AllFusion Modeling Suite / С.В. Маклаков. − М: ДИАЛОГ-МИФИ, 2005. − 432 с.

4. Мяцяшек, Л.А. Анализ требований и проектирование систем. Разработка информационных систем с использованием UML / Л.А. Мяцяшек, пер. с англ. − М: Издательский дом «Вильямс», 2002. – 432 с.

5. Муравьев, Г.Л. Автоматизация обучения алгоритмизации и программированию / Г.Л. Муравьев, С.В. Мухов // Вести ИСЗ. − 2004. − № 3. − С. 24−29.

6. Муравьев, Г.Л. Компьютерная генерация спецификаций сетевых архитектур заданной сложности / Г.Л. Муравьев, А.Н. Никонюк, В.И. Хвещук // Технологии информатизации и управления: сб. науч. ст. 2-й междунар. научно-практ. конф. (TIM-2011), Минск, 2011. − С. 50−53.

7. Муравьев, Г.Л. Построение моделей по описаниям, согласованным с процессным способом моделирования / Г.Л. Муравьев, В.И. Хвещук // Современные информационные компьютерные технологии mcIT-2008: сб. научн. ст., Гродно: ГрГУ, 2008. − Ч. 2. − С. 235−238.

8. Мухов, С.В. Типизация моделей для обработки баз данных как основа надежности программного продукта / С.В. Мухов, Г.Л. Муравьев, Ю.П. Ашаев // Информационные системы и технологии IST’2010: материалы 6 Междунар. конф., Минск, 2010. − С. 360−364.

**Г. Л. МУРАВЬЕВ, А. С. РЫЩУК**
БрГТУ (г. Брест, Беларусь)

## О ПОСТРОЕНИИ WEB-СИСТЕМ ОПЕРАТИВНОЙ ОБРАБОТКИ ИНФОРМАЦИИ

Существует большой класс систем обработки информации в оперативном режиме (СОО). Например, системы класса OLTP для он-лайн обработки транзакций. Они характеризуются наличием независимых потоков запросов, отличающихся случайным поведением. Их организация должна обеспечивать нужное качество обслуживания в условиях непредсказуемого колебания загрузки вычислительных средств от слабой до «пиковой». Среди них можно выделить вид, предназначенный для оперативного управления (диспетчирования) ограниченным набором обслуживающих ресурсов по запросам внешних пользователей. Например, системы поддержки диспетчеров разнообразных экстренных служб.

Их существенные особенности:

- массовость и территориальная распределенность пользователей;
- повышенные требования к скорости обслуживания, качеству решений;
- непрерывный контроль за состоянием обслуживающих ресурсов (специалистов, бригад, машин, механизмов и т. п.);
- относительная алгоритмическая простота, наличие типовых решений, ограниченность типов запросов;
- относительно невысокая квалификация персонала и пользователей.

Требования к системам:

- работа в реальном масштабе времени, учет приоритетов запросов;
- масштабируемость, адаптация к резкому изменению характеристик задач, росту числа пользователей;
- рационализация труда диспетчеров, обеспечение принятия корректных решений, документированность процессов управления;
- поддержка разнообразных средств доступа, включая традиционные средства интернета, мобильные средства;

- расширяемость как наращивание функций, замена платформы, увеличения вычислительных мощностей.

При построении таких систем основные задачи сводятся к выбору:

- архитектуры и моделей ее логического представления;
- форматов и механизмов передачи данных;
- способов распределения вычислительной нагрузки;
- механизмов конфигурирования, маршрутизации, безопасности;
- подходов к проектированию интерфейсов;
- выбору средств разработки, обеспечивающих нужные характеристики функционирования.

Преимущественно в их основе лежат модификации клиент-серверной архитектуры, обеспечивающей эффективное многопользовательское использование разделяемых ресурсов. Для поддержки интерактивности и функционирования в режиме реального времени целесообразно использовать комбинации приемов, технологий, например, балансировку вычислительной нагрузки связки клиент-сервер, очереди ожидающих запросов (long polling), протоколы обмена сообщениями браузер-сервер реального времени, полнодуплексной связи «поверх» TCP-соединения и т. п.

Для обеспечения расширяемости, масштабируемости целесообразно построение системы в виде веб-сервиса, спроектированного на основе модели MVC (Модель-Представление-Контроллер) [1]. Это предполагает логическую декомпозицию системы на «слабо связанные» компоненты − модели данных, их представление и процессы взаимодействия с пользователем (аппарат событий). При этом для дополнительного структурирования системы, повышения масштабируемости, а также с учетом требований множественности доступа и производительности при построении серверной части системы предлагается использовать интерфейс REST [2] (RESTful API на базе фреймворка Yii).

Указанный фреймворк, базирующийся на компонентной модели, обеспечивает наиболее высокую скорость среди других PHP-фреймворков. Использует парадигму MVC, модели доступа к данным DAO, ActiveRecord с поддержкой SQL-запросов, обеспечивает независимость от особенностей конкретной базы данных. В качестве модели базы данных выбрана реляционная модель на базе оптимистичного параллельного доступа, в качестве СУРБД − MySQL. Для предотвращения межсайтовового скриптинга, атак типа CSRF (подделка межсайтовых запросов) возможно использование библиотеки HTMLPurifier и других средств фреймворка.

При использовании интерактивных пользовательских интерфейсов веб-приложений в качестве технологии передачи данных целесообразно использовать Ajax, поддерживающий «фоновый» обмен данными между браузерами и веб-сервером. Отсутствие необходимости полной перегрузки веб-страниц при обновлении данных дополнительно снижает трафик и нагрузку на сервер, повышает «реакцию» интерфейса. В качестве формата данных предлагается использовать JSON (JavaScript Object Notation), который оперирует универсальными структурами данных, поддерживаемыми современными языками, отличается «прозрачным» синтаксисом, безызбыточностью, простотой применения [3].

Для реализации клиентской составляющей перспективен фреймворк ExtJs, применяемый для разработки веб-приложений и высоконагруженных пользовательских интерфейсов. Его средства поддерживают технологию MVC на клиентской стороне, обеспечивают кроссбраузерность, использование широко распространенных браузеров. Дополнительный прирост производительности достигается балансировкой и снижением нагрузки на сервер за счет поддержки локальных хранилищ данных библиотеки ExtJs (Ext.data.Store) и переноса трудоемких функций в клиентские приложения. Эти же средства могут использоваться для построения пользовательских интерфейсов, отвечающих требованиям стандарта ISO 9241-11, имитирующих привычный интерфейс в «стандарте» ОС.

Указанные подходы макетировались при разработке систем для служб экстренного реагирования. Так, для службы помощи автовладельцам система строилась как кроссплатформенный REST-веб-сервис, включающий подсистемы оператора, диспетчера, администратора. Информация от клиентов и служб принималась в он-лайн форме и регистрировалась. Запросы пользователей представляли заявки на эвакуацию транспорта и на оказание экстренной сервисной помощи. Службы информировали диспетчера о текущем состоянии и месте нахождения работников, транспортных средств, механизмов. Диспетчирование заключалось в подборе средств, персонала, расстановке приоритетов, контроле исполнения. Проведенные испытания показали удовлетворительность принятых решений.

ЛИТЕРАТУРА

1. Архитектура MVC [Электронный ресурс]. – 2011. – Режим доступа: http://en.wikipedia.org/wiki/Model-view-controller.

2. Интерфейс REST [Электронный ресурс]. – 2012. – Режим доступа: http://en.wikipedia.org/wiki/Representational_state_transfer.

3. Официальная документация формата JSON [Электронный ресурс]. – 2010. – Режим доступа: http://www.json.org.