УДК 004.04

ВЫБОР АРХИТЕКТУРЫ СИСТЕМЫ УПРАВЛЕНИЯ УЧЕБНЫМ ПРОЦЕССОМ

Иванющенко Н.С., Привалов М.В.
Донецкий национальный технический университет,
кафедра автоматизированных систем управления
E-mail: nikita@layar.com

В статье производится выбор архитектуры для системы управления качеством учебного процесса. Рассматривается несколько подобных систем, формируется список требований к системе. Предложены несколько моделей архитектуры информационных систем, произведён выбор архитектуры для реализации системы. Оцениваются основные параметры системы, предложены методы для интеграции с существующими системами.

Общая постановка проблемы

На данный момент системы управления качеством учебного процесса весьма востребованы, что обусловлено ужесточившимися требованиями к образованию в целом и переходом к рыночной конкуренции между ВУЗами. Поэтому весьма логичным является вопрос построения системы для организации интернет-доступа к данным учебного процесса. Так как подобная работа на базе университета проводится впервые, то целью становится также разработка общей архитектуры системы.

Анализ существующих разработок

Подобные системы разрабатываются для каждого учебного заведения индивидуально, с учётом особенностей учебного процесса и дополнительных функций и являются закрытыми. Кроме того, развёртывание готовой системы поверх существующего множества различных АСУ университета, успешно функционирующих в

течение длительного времени является весьма проблематичным и затратным.

Поэтому, использование уже готовой системы не представляется возможным.

Постановка задачи исследования

В данной работе необходимо разработать архитектуру системы управления качеством учебного процесса с учётом требований надёжности, отказоустойчивости, защиты хранимых данных.

Было рассмотрено 2 подхода к разработке программного обеспечения: сервис-ориентированная и мультиагентная архитектуры.

Мультиагентная архитектура – это система, образованная несколькими взаимодействующими интеллектуальными агентами [4]. Мультиагентные системы могут быть использованы для решения таких проблем, которые сложно или невозможно решить с помощью одного агента или монолитной системы. Примерами таких задач являются онлайн-торговля, ликвидация чрезвычайных ситуаций и моделирование социальных структур. Данная архитектура была отвергнута в силу своей децентрализованности – нет агентов, управляющих всей системой.

С учётом весьма широкого круга задач и разнородности данных, наиболее логичной архитектурой для данной системы является сервис-ориентированная архитектура [5], характеризующаяся модульным подходом к разработке программного обеспечения, основанном на использовании сервисов (служб) со стандартизированными интерфейсами.

В основе SOA лежат принципы многократного использования функциональных элементов информационных технологий, ликвидации дублирования функциональности в программном обеспечении, унификации типовых операционных процессов, обеспечения перевода операционной модели компании на централизованные процессы и функциональную организацию на

основе промышленной платформы интеграции.

Компоненты программы могут быть распределены по разным узлам сети, и предлагаются как независимые, слабо связанные, заменяемые сервисы-приложения. Программные комплексы, разработанные в соответствии с SOA, часто реализуются как набор веб-сервисов, интегрированных при помощи известных стандартных протоколов.

Компоненты такой системы могут быть распределены по разным узлам сети, и предлагаются как независимые, слабо связанные, заменяемые сервисы-приложения.

Для крупных информационных систем, уровня предприятия и выше, использование подобной архитектуры предпочтительно по следующим причинам:

- сокращение издержек при разработке приложений, за счёт упорядочивания процесса разработки;
- расширение возможностей повторного использования кода;
- независимость от используемых платформ, инструментов, языков разработки;
- повышение масштабируемости создаваемых систем;
- улучшение управляемости создаваемых систем.

В данный момент университет обладает достаточно большой автоматизированной системой управления, но лишь для внутреннего использования. Базы данных являются распределёнными и содержат практически все необходимые данные. АСУ включает такие системы как «Деканат», «Библиотека», «Отдел кадров», «Абитуриент». АСУ «Деканат» расположена на достаточно большой площади и в данный момент прямого взаимодействия между узлами нет, лишь главный узел имеет доступ ко всем дочерним, а каждый деканат имеет доступ только к узлу, обслуживающему данный факультет. Это создаёт лишние трудности при внесении и обновлении данных.

С учётом использования выбранной сервис-ориентированной архитектуры, нынешняя структура системы различных АСУ достаточно хорошо подходит для построения системы.

Для доступа к каждой БД будет реализован сервис, занимающийся обменом данных с БД или её главным узлом (как в случае с БД АСУ «Деканат»), в том числе, обновлением и добавлением данных. Каждый из этих сервисов будет иметь доступ к данным только заданной автоматизированной системы управления. Также, будут реализованы дополнительные базы данных и дополнительные сервисы, необходимые для реализации всех функций системы, как то: размещение учебных материалов, регистрация пользователей, выдача прав, размещение объявлений. Главный сервер, являющийся веб-сервером, будет заниматься обработкой данных, предоставляемых каждым из сервисов и предоставлением окончательного результата работы системы.

Данная архитектура, в силу распределения обязанностей между сервисами, достаточно легко справится с подобной нагрузкой.

Описание бизнес-процессов – формализация деятельности отдельных элементов организации, начиная от организации в целом и ее отдельных ключевых структурных подразделений и заканчивая отельными сотрудниками. При этом определяется, какие конкретно функции выполняет каждый элемент, кто несет ответственность за его деятельность, каким стандартам она соответствует, какими параметрами определяется ее успешность. Если бизнес-процессы правильно описаны, не должно возникать никаких вопросов относительно требований к каждому внутреннему бизнес-процессу организации. Первым шагом к достижению контроля над организацией являются знание и понимание всего происходящего в ней, включая самые простейшие процессы.

Приведём бизнес-процессы системы обучения, касающиеся успеваемости студентов (рис. 1).

Из бизнес-процессов, описанных на диаграмме, наша система управления качеством учебного процесса затрагивает два: контроль успеваемости и контроль рейтинга и посещаемости.

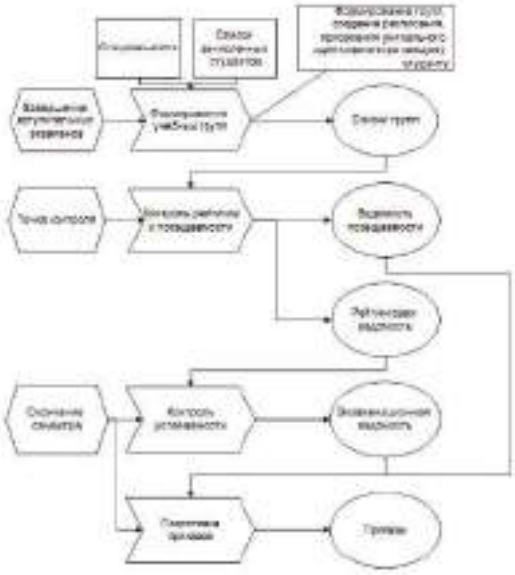

Рисунок 1 – Бизнес-процессы системы обучения, касающиеся успеваемости студентов

Проектирование системы с учётом выбранной архитектуры

Низкая связанность - важный архитектурный принцип при разработке СОА систем. Использование этого принципа позволяет связывать различные компоненты информационной системы во время ее функционирования с помощью, так называемого, позднего связывания (late binding). Благодаря этой особенности также значительно облегчается внесение изменений в функциональность сервисов, поскольку это совершенно не затрагивает другие сервисы, также значительно упрощается пошаговое создание корпоративной системы из-за отсутствия барьеров реализации функциональности сервиса за несколько итераций.

Возможность динамически подключать новые сервисы, также как и поиск этих сервисов клиентами является также одним из краеугольных принципов системы, построенной на основе service-oriented architecture.

Итак, основными компонентами (рисунок 2) являются сервисная шина предприятия (ESB), СОА реестр (SOA Registry), workflow engine, сервисный брокер (service broker), СОА супервизор (SOA supervisor). Все они играют собственную роль в системе, при этом взаимодействуя друг с другом.

В сервис-ориентированной архитектуре все различные части программного обеспечения общаются друг с другом, как правило, посылаю друг другу достаточно много сообщений. Эти сообщения должны быть доставлены быстро и доставка должна быть гарантирована. Для передачи сообщений в SOA как правило используют сервисную шину предприятия (Enterprise Service Bus – ESB).

Каждый бизнес имеет свой workflow, являющимся случайным в каждом конкретном случае или формально описанным, сложившийся как бы само собой или возникший в результате тщательного анализа и автоматизации. Workflow engine - это программный продукт, позволяющий соединить весь бизнес процесс в корпоративной информационной системе от начала до его завершения, система для воспроизведения потока работ по имеющейся модели. Ядро веб-портала, обеспечивающие обработку данных и является workflow

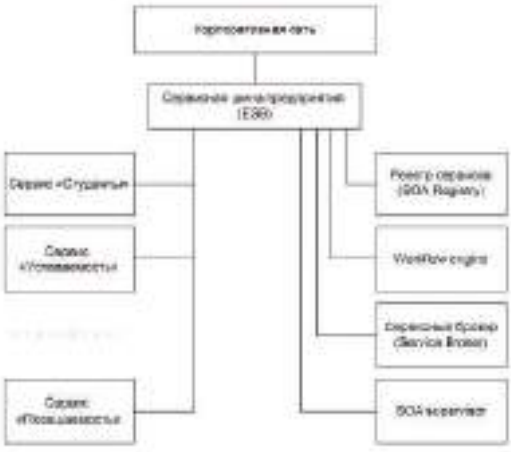

Рисунок 2 – Основные компоненты SOA

engine. Так как workflow engine не является обособленной частью системы, то необходимость в супервизоре отпадает.

Сервис-брокер производит регистрацию, учет и ведение слежения зависимых друг от друга сервисов. Клиент (Service Requester) обращается к сервис-брокеру за описанием необходимого ему веб-сервиса. После того как он получает ответ (функции и параметры, определенные для данного сервиса на WSDL), клиент начинает общаться с провайдером (Service Provider), обмениваясь с последним сообщениями SOAP.

Описание конкрактов веб-сервисов

По результатам выделения ролей системы мы можем выделить 4 веб-сервиса:

- Base – получение общей информации, поиск пользователей;
- Decanat – получение списков семестров, групп, студентов и их оценок;
- Lector – получение информации о лекторе, списков групп, предметов, семестров и студентов и их оценок;
- Student – получение информации о студент, списков семестров и оценок.

Опишем на языке C# методы каждого из веб-сервисов.

Веб-сервис Base

DataTable GetPersonsByMask(string surnameMask, string nameMask, string fatherMask) – поиск студента или лектора по маске. Возвращает список. Параметры:

- string surnameMask – маска фамилии;
- string nameMask – маска имени;
- string fatherMask – маска отчества.

string GetSubjectName(int subjID) – получение названия предмета по ID (уникальному идентификатору). Возвращает строку. Параметры:

- int subjID – ID предмета.

string GetGroupName(int groupID) – получение имени группы по ID. Возвращает строку. Параметры:

– int groupID – ID группы.

Веб-сервис Decanat

DataTable GetSpecs() – получение списка специальностей. Возвращает список.

DataTable GetGroups(int specID) – получение списка групп по ID специальности. Возвращает список. Параметры:

– int specID – ID специальности.

DataTable GetSemesters(int groupID) – получение списка семестров по ID группы. Возвращает список. Параметры:

– int groupID – ID группы.

DataTable GetStudentsMarks(int groupID, int semesterID, int trainDirectionID) – получение списка студентов по ID группы, ID семестра и ID направления обучения. Возвращает список. Параметры:

– int groupID – ID группы;
– int semesterID – ID семестра;
– int trainDirectionID – ID направления обучения.

Веб-сервис Lector

DataTable GetGroups(int lectorID) – получение списка групп по ID лектора. Возвращает список. Параметры:

– int lectorID – ID лектора.

DataTable GetSubjects(int lectorID, int groupID) – получение списка предметов по ID лектора и ID группы. Возвращает список. Параметры:

– int lectorID – ID лектора;
– int groupID – ID группы.

DataTable GetSemesters(int lectorID, int groupID, int subjectID) – получение списка групп по ID лектора, ID группы и ID предмета. Возвращает список. Параметры:

– int lectorID – ID лектора;

– int groupID – ID группы;
– int subjectID – ID предмета.

DataTable GetStudents(int lectorID, int groupID, int subjectID, int semesterID, int trainDirectionID) – получение списка студентов по ID лектора, ID группы, ID предмета, ID семестра и ID направления обучения. Возвращает список. Параметры:

– int lectorID – ID лектора;
– int groupID – ID группы;
– int subjectID – ID предмета;
– int semesterID – ID семестра;
– int trainDirectionID – ID направления обучения.

DataTable GetInfo(int lectorID) – получение информации о лекторе по ID лектора. Возвращает список. Параметры:

– int lectorID – ID лектора.

Веб-сервис Student

DataTable GetSemesters(int studentID) – получения списка семестров по ID студента. Возвращает список. Параметры:

– int studentID – ID студента.

DataTable GetMarks(int studentID, int semesterID, int trainDirectionID) – получения списка оценок по ID студента, ID семестра и ID направления подготовки. Возвращает список. Параметры:

– int studentID – ID студента;
– int semesterID – ID семестра;
– int trainDirectionID – ID направления подготовки.

DataTable GetInfo(int studentID) – получения информации о студенте по ID студента. Возвращает список. Параметры:

– int studentID – ID студента.

Выводы

В данной работе были рассмотрены системы управления качеством учебного процесса, проанализированы их возможности и сформирован список требований к новой системе. Была выбрана

сервис-ориентированная архитектура, так как она позволяет достаточно просто модифицировать системы под изменившиеся или расширившиеся требования, является гибкой и достаточно простой для внедрения её на существующей базе системы АСУ, так как структура и работа основных БД не изменяется и работоспособность существующих АСУ университета не затрагивется. В дальнейшем данная система будет реализована и внедрена на базе ДонНТУ.

Литература

[1] Самарский Государственный Технический Университет. Информационная система «Управление качеством учебного процесса. Учет успеваемости и посещаемости». – 2009. [Электронный ресурс].URL:http://ivc.samgtu.ru/node/161 (дата обращения 20.03.2010).

[2] University of Colorado at Boulder, CUConnect. – 2008. [Электронный ресурс]. URL:https://cuconnect.colorado.edu/uPortal/index.jsp (дата обращения 17.03.2010).

[3] About Moodle: features. – 2008. [Электронный ресурс]. URL: http://docs.moodle.org/en/Features (дата обращения 15.03.2010).

[4] Многоагентная система. – 2010. [Электронный ресурс]. URL: http://ru.wikipedia.org/wiki/Многоагентная_система (дата обращения 21.03.2010).

[5] Сервис-ориентированная архитектура. – 2010. [Электронный ресурс]. URL: http://ru.wikipedia.org/wiki/Сервис-ориентированная_архитектура (дата обращения 21.03.2010).