Форма № Н-9.02

Міністерство освіти і науки України
Херсонський державний університет
Факультет фізики, математики та інформатики
Кафедра інформатики

Дипломна робота

магістр

на тему: **РОЗРОБКА ЗОВНІШНЬОГО ПРОГРАМНОГО ІНТЕРФЕЙСУ СИСТЕМИ ПОБУДОВИ КОНТУРУ ЗВОРОТНОГО ЗВ'ЯЗКУ ТИПУ «KSU FEEDBACK»**

Виконав: студентка 5 курсу, групи 531
Спеціальності 8.04030201. Інформатика
Клименко Наталія Олегівна
Керівник проф. Співаковський О.В.
Рецензент доц. Круглик В.С.

Херсон - 2013 року

СОДЕРЖАНИЕ

ВСТУПЛЕНИЕ

Наличие обратной связи можно рассматривать как одну из характеристик успешного управления. Контур обратной связи может быть определён как средство для передачи информации о процессах, происходящих внутри системы, будь то университет, предприятие или любая другая организационно-административная система.

Функционирование любой системы как единого целого основывается на обеспечении связей между составными элементами. Обратная связь может стать основой для саморегуляции, развития систем, приспособления их к изменяющимся условиям существования. В случае решения проблемы речь может идти о связях между этапом реализации решения и началом процесса принятия решения – этапом формулирования проблемы. Обратная связь даёт возможность понять, устранено ли препятствие, мешающее достижению целей системы, иными словами, решена ли проблема.

В обобщённом виде контур обратной связи представляет собой абстрактную модель, которая преобразует информацию обратной связи в информацию для управленческих решений, что, в сущности, и составляет содержание процесса управления. Реализация такой абстрактной модели может быть представлена на примере реализации приложения “KSU Feedback”. Данное приложение позволяет с помощью анонимного голосования получить объективность оценки какого-либо процесса или решения. Так как комплекс системы “KSU Feedback” представлен модулями, что соответствует сервис-ориентированной архитектуре, одним из таких модулей является KSU Feedback API. API (application programming interface) - это внешний программный интерфейс— представлен набором готовых классов, методов и структур, предоставляемых приложением либо сервисом для использования во внешних программных продуктах. [47]

Проанализировав некоторые существующие приложения для построения контура обратной связи можно сделать вывод, что зачастую у них либо нет API вообще, либо он не удовлетворяет требованиям конечного пользователя. Подобные ресурсы не позволяют самостоятельно формировать анкеты и создавать собственные шаблоны отчётов, что делает возможности анализа результата менее гибкими. Кроме того, проведение анонимного опроса не предусмотрено совсем. API для веб-проектов довольно широко распространились за последние пару лет. Сегодня API являются незаменимым инструментом каждого веб-разработчика и, более того, всё больше и больше API очень эффективно применяются в маркетинге для многих типов бизнеса. Веб API позволяют разработчикам создавать веб-страницы и веб-приложения, используя данные из нескольких источников сразу. Приложение, которое предоставляет API, существенно выигрывает на фоне приложений своего типа. Подобный анализ обосновывает **актуальность** темы.

Цель дипломной работы - реализовать API системы построения контура обратной связи типа «KSU Feedback».

Для достижения цели дипломной работы необходимо было выполнить следующие задания:

- исследовать современные концепции создания внешнего программного интерфейса web-приложения;
- разработать требования к API системы типа «KSU Feedback»;
- разработать API предоставления доступа к данным системы «KSU Feedback» для возможности использования этой информации в других приложениях или дополнять существующие данные системы новыми объектами.

Объект исследования - системы построения контура обратной связи типа «KSU Feedback».

Предмет - внешний программный интерфейс системы построения контура обратной связи типа «KSU Feedback».

В качестве **методов исследования** были использованы методы: анализа, классификации, моделирования. Метод анализа предполагает условное или реальное разбиение изучаемого объекта на составные части в целях изучения каждой составляющей отдельно. В случаем системы «KSU Feedback» разделение системы на функциональные модули. Метод классификации предполагает разделение составных частей объекта изучения по определенным признакам с целью выстраивания понятийной системы. Метод моделирования представляет собой метод, при котором создается и исследуется копия объекта. Как правило, копия в точности соответствует реальному объекту.

Теоретическая значимость работы состоит в возможности использования выявленных особенностей разработки API веб-приложения для дальнейшего развития теоретической базы по данной теме. Теоретически обоснована и практически доказана возможность использования результатом исследования на примере тестовых сценариев.

Практическая значимость - возможность использования эффективного инструмента для построения контура обратной связи при интеграции с другими программными продуктами на основе системы типа «KSU Feedback».

Практическая значимость дипломной работы определяется также и тем, что ее результаты могут явиться теоретико-методической основой для создания веб-приложений с эффективным механизмом построения API для управления и распространения функциональности, предоставляемой приложением.

Проблематика исследования:

- определение и создание функциональности, которую предоставляет приложение «KSU Feedback» с помощью API;
- API должен позволять абстрагироваться от того, как именно эта функциональность реализована, и отображать концепцию получения

конечного результата, а именно интеграции приложения с другими программными продуктами.

Апробация исследования. Основные теоретические положения исследования опубликованы в статье «The Problem of Architecture Design in a Context of Partially Known Requirements of Complex Web Based Application "KSU Feedback"». Также в статье «Разработка внешнего программного интерфейса системы построения контура обратной связи "KSU Feedback"» представлен обзор диплома, представленного к защите на кафедре информатики факультета физики, математики и информатики Херсонского государственного университета.

Структура работы обусловлена предметом, целью и задачами исследования. Работа состоит из введения, трех разделов и заключения. Введение раскрывает актуальность, определяет степень научной разработки темы, объект, предмет, цель, задачи и методы исследования, раскрывает теоретическую и практическую значимость работы. В первом разделе рассматриваются понятия обратной связи в теории управления и значимость наличия контура обратной связи в управленческой системе. Во втором разделе раскрываются особенности и методы реализации API веб-приложения. Третий раздел посвящен описанию тестирования полученных результатов разработки. В заключение подводятся итоги исследования, формируются окончательные выводы по рассматриваемой теме.

РАЗДЕЛ 1
Концепция контура обратной связи в системах управления

1.1. Обратная связь в управленческой деятельности

Одним из определений понятия управления является следующее: формирование и реализация воздействий, выбранных из множества возможных на основании определённой информации, обеспечивающих желаемое движение (функционирование, поведение) объекта, приводящее к поставленной цели. Такое определение указывает на одну из составляющих частей эффективного управления, а именно информацию, которой оперирует субъект (тот, кто осуществляет управляющее действие) при принятии управленческих решений. Управление как целенаправленный процесс требует анализа, разработки и постановки определённых целей. При воздействии субъекта управления на объект управления должен быть получен результат. Этот результат нужно сопоставить с целью, и чем точнее будет результат, тем выше качество управления.

Управление как целенаправленная деятельность присуща организационным системам. Особенностями организационной системы являются: ориентация на определённые потребности; целенаправленность; субъект управления; иерархическая структура; регламентированное поведение и деятельность членов организации; способность к самообучению и саморазвитию. Под управлением такой системой понимается процесс выработки решения, планирования, организации, коммуникации, мотивации и контроля, необходимый для того чтобы сформулировать и достичь цели.

Чтобы подойти к рассмотрению механизма управления и его роли в процессе развития любой системы, введем понятие *обратной связи.*

В абстрактном представлении под обратной связью понимают воздействие результатов функционирования какой-либо системы (или объекта) на характер этого функционирования.

Одна из важнейших задач управления связана с моделированием возможных оптимальных путей протекания процессов функционирования системы и поиском воздействий, позволяющих поддерживать устойчивое состояние системы, а при возможности и улучшать деятельность системы. При этом наиболее существенным оказывается характер воздействия. Управление сложными объектами всегда связано с целым комплексом мероприятий, важнейшим элементом которого является сам человек. В таком случае организационную систему можно рассматривать как упорядоченную совокупность звеньев управления (должностей, подразделений, служб), находящихся в определенной взаимной связи и соподчинении. Наличие обратной связи позволяет получать информацию о взаимодействии уровней системы и функциональных областей (отделов, подразделений).

В теории управления более конкретное значение обратной связи представлено как реакция системы на воздействие.

Когда элементом системы становится человек, который сам по себе представляет сложнейший комплекс управляющих и управляемых систем, то механизмы управления достигают высокой степеней сложности.

Понятие обратной связи возникло в теории регулирования при создании систем автоматического управления техническими и административными системами. Принцип обратной связи позволяет учитывать новые сведения о состоянии объекта при отклонениях от желаемого состояния и соответственно изменять управляющие воздействия. Этот эффект называется саморегулированием системы. Именно с помощью обратной связи система получает новые сведения о собственном состоянии, фиксирует отклонения и корректирует управляющие воздействие.

Обратная связь, которая за счёт своего влияния усиливает результат функционирования, называется положительной. Наличие положительной обратной связи стабилизирует функционирование системы, делает ее работу более устойчивой. Также особую роль в управлении также играет и обратная

отрицательная связь, именно отрицательная связь способствует конкурентной борьбе, создает движение, развитие, прогресс. Обратная связь позволяет анализировать ошибки с помощью полученной информации.

Обратные связи являются основным фактором в формировании системных свойств, накоплении информации о поведении системы. Управляющая система, которая игнорирует сигналы обратной связи, может привести в тупик деятельность. Ни одна организационная система не может развиваться в условиях полной и жесткой централизации управления, без элементов саморегулирования, без обратной связи. В некоторых случаях принцип обратной связи использовать не удается из-за практической невозможности получить информацию о результатах управления, т.к. необходимо учитывать объективность оценки управления из-за значительной роли человеческого фактора. В таком случае возникает необходимость построения контура обратной связи. Для этого общим методом является проведение опроса. Однако такой метод требует много времени, особенно если есть необходимость в проведении опроса среди большого количества респондентов и при этом обеспечить анонимность ответа.

Традиционные способы обеспечения обратной связи уступают более совершенным веб-приложениям для обеспечения процесса опроса. При обратной связи, которая представлена в качестве средства для достижения определенной цели управления, появляется возможность решить проблему, а не просто критиковать её. Обратная связь работает лучше, когда становится постоянным процессом для получения актуальной информации об управлении системой, но для этого нужен инструмент, обеспечивающий потребности субъекта управления при этом абстрагировавшись от сферы деятельности.

Обратная связь – основа поддержания и развития. Важная роль обратной связи состоит в том, что, сообщая органу управления информацию о

реальном состоянии объекта, она позволяет осуществлять регулирование в условиях неполной информации о возмущающих воздействиях.

Так, например, обратная связь с клиентом – инструмент маркетинговой стратегии. «Неправильное» отношение к жалобам рождает круг:

- жалобы клиентов не воспринимают в компании, клиенты уходят
- оставшиеся клиенты понимают, что жаловаться бесполезно
- не получая обратной связи, сервис компании ухудшается
- потребители всё еще пользуются услугами компании в основном за счет низких цен
- персонал видит проблемы и тоже уходит, что еще сильнее ухудшает сервис
- количество клиентов сокращается и как следствие круг замыкается.

Обратную связь важно получать не только от клиентов компании, но и от ее сотрудников. Анкетирование (или иной метод) позволяет измерять вовлеченность сотрудников – важнейший фактор успеха компании. Вот какую иерархию вовлеченности сотрудников выстраивают Джон Флеминг и Джим Асплунд в книге «Управление качеством услуг: Метод HumanSigma» (Рис. 1.1.). [6] Подобная схема даёт представление значимости отзыва сотрудников об управленческом решении. Контур обратной связи позволяет эффективно настроить иерархическую связь в системе управления, но при этом формировать объективную оценку принимаемых решений.

Закон обратной связи: без наличия обратной связи между взаимосвязанными и взаимодействующими элементами, частями или системами невозможна организация эффективного управления ими.

Следствие 1. Обратная связь играет решающую роль в выполнении менеджментом двух основных функций: поддержании и развитии.

Следствие 2. Беспрепятственное протекание обратной связи – залог развития. Ограничения на обратную связь ведут к бюрократизации и отрыву от реальности.

Следствие 3. Получение обратной связи от клиентов и сотрудников компании позволяет управлять их вовлеченностью – основой рыночного успеха компании.[5]

1.2. Содержание и механизм построения контура обратной связи по средствам информационных технологий

Контур управления – это информационная система руководителя, которая агрегирует оперативные данные и предоставляет их удобную визуализацию с использованием информационных технологии. Понятие контура управления происходит из теории автоматического управления. Он определяет и описывает границы регулирования работы объекта управления. Контуры управления различаются по типам, определяемым наличием или отсутствием обратной связи. Если в контуре управления нет качественной обратной связи, субъект управления не может оценить результаты своего воздействия на объект управления.

При управлении технологическими процессами наиболее распространено регулирование с наличием обратной связи (замкнутый контур), а при управлении процессами производства обратная связь чаще всего выражена опосредованно. Тем не менее, она поддается формализации и эффективно влияет на качество управления.

В упрощенном виде на рисунке 1.2. представлена схема системы управление с наличием обратной связи. Входная информация действует на управляемый процесс и в соответствии с передаточной функцией, характерной для данного объекта и определяющей соотношение между входными и выходными данными, превращается в выходную информацию или же после анализа в управленческое решение. Входная информация при помощи канала обратной связи подается на вход, корректирует входящие

данные и в виде управляющего сигнала воздействует, но уже по-новому, на объект. Возникшая таким образом связь образует замкнутый контур. Механизмы обратной связи обеспечивают организационные системы данными для заинтересованных сторон, как для субъекта управления, так и объекта (руководитель и подчинённые).

Реальные системы управления могут включать в себя несколько контуров обратной связи, что позволяет при необходимости идентифицировать и по возможности устранять любые изменения, препятствующие достижению целей проекта. Например, проект может столкнуться с непредвиденными обстоятельствами, которые не были изначально учтены при разработке системы контроля. В этом случае в системе управления должно быть введено столько контуров, сколько типов показателей необходимо учитывать при управлении процессом, например, по входным показателям, показателям самого процесса и показателям плана (система управления по числу типов показателей).

Методологическая функция принципа обратной связи определяется рядом его существенных признаков.

Во-первых, обратная связь не исчерпывается наличием причинно-следственных связей или обратным воздействием вообще. Она приобретает характер специального обратного воздействия, а именно управляющего обратного воздействия и обуславливается передачей осведомительной и контролирующей информации. Иначе говоря, это принцип управления, необходимым условием которого является информационное взаимодействие между управляемой и управляющей подсистемами. Функция этого признака обратной связи и состоит в том, чтобы руководители любого ранга в своей управленческой деятельности обеспечили, организовали, наладили своевременную передачу осведомительной и контролирующей информации. Таким образом, появляется понятие целевых групп. Руководителей и подчинённых системы (или же пользователей какого-либо ресурса) можно разделить на группы по правам доступа, параметрам функционирования в

системе, значимости результата. Подобное разделение целевой аудитории респондентов позволяет контролировать параметры, по которым будет проанализирован конечный результат обратной связи.

“Во-вторых, существенным признаком принципа контура обратной связи является то, что это принцип организации, поскольку его действие обусловлено определенной внутренней структурой и достаточно высоким уровнем организации системы управления, а также потому, что этот принцип способствует повышению уровня организации систем управления.

Вместе с тем - это и важнейшее требование к управленческой деятельности руководителей различного уровня. Это значит, что система построения обратного контура должна предоставлять уровни ответственности при создании, редактировании и анализе данных обратной связи.

В-третьих, принцип обратной связи является внутренней основой и необходимым условием развития и совершенствования сложных самоуправляющихся систем. Иначе говоря, это один из принципов сохранения или сохранения и развития системы управления. Таким образом, данные, полученные с помощью системы построения контура обратной связи, должны быть доступны для текущего анализа, но при этом должны быть сохранены для возможности повторного обращения к статистическим данным и последующего применения в управлении” [4].

Принцип обратной связи обеспечивает также выполнение функции согласования, координации действий с другими системами управления - и в этом также заключается его важное значение для управленческой деятельности руководителей различного уровня доступа к данным. Взаимодействие с другими системами позволяем распространять при необходимости полученные результаты с учётом уровня доступа к данным или же использовать дополнительный функционал сторонних приложений.

При построении контура обратной связи важным этапом является выбор технологий и методов реализации. “Для такого выбора необходимо очертить

проблему поиска методов построения контура обратной связи, провести анализ существующих решений, выявить их недостатки и достоинства.

При поиске мы использовали следующие критерии:

1. объективность ответа респондента;

2. контролируемая целевая группа;

3. определение предметной области и критериев оценки;

4. относительно малая времязатратность и ресурсоемкость проведения мониторинга;

5.возможность анализа полученных данных с возможностью повторного применения в дальнейшей обработке результатов.”[11, с.40].

Выбор метода построения обратной связи

Выбор метода построения обратной связи зависит от специфики конкретного проекта, технической (в смысле компьютерной) оснащенности организации, принятой в ней технологической дисциплины и т.д. Однако в любом случае при выборе метода следует учитывать три основных фактора:

1. Размер проекта. Если проект достаточно прост и состоит примерно из десяти задач, руководитель проекта, как правило, способен отслеживать состояние всех работ «вручную». Если же проект содержит большое количество целей, задач и направлений деятельности, то целесообразно использовать соответствующие инструментальные средства.
2. Доступность инструментальных средств для построения контура обратной связи. Если в реализации проекта участвует достаточно большое число исполнителей, и (или) они разнесены территориально, одним из наиболее эффективных способов сбора информации о состоянии работ является электронная почта либо средства Web. Если имеющиеся инструментальные средства не поддерживают работу с электронной почтой или Web, информацию по проекту придется, скорее всего, вводить вручную. Таким образом приоритетным становится получение результатов в режиме online и минимизация человеческого фактора при обработке результатов.

3. Уровень детализации, который необходим при отслеживании состояния работ проекта. Как правило, требуемый уровень зависит от сложности проекта и его текущего состояния. Так что необходим достаточно высокий уровень детализации, возможностей анализа и отчётности. Кроме того, более детальный анализ обычно проводится на завершающих стадиях обратной связи, однако, получение текущих данных также может быть необходимо на этапе формирования статистических обобщений.

Оценка результатов обратной связи

Если в результате обратной связи было выявлено отклонение реального состояния дел от исходного плана, то в некоторых случаях это может потребовать разработки нового плана для оставшейся части проекта. Чтобы сделать это с наименьшими издержками, целесообразно придерживаться следующей методики:

1. Провести анализ по уже существующим результатам (опросам, анкетам).
2. Для получения новых данных сформировать структуру для проведения следующего цикла обратной связи.
3. Распределить ответственность и уровни доступа ответственных за проведение мероприятий обратной связи и анализ результатов, а также сформировать целевые группы.

Современные инструменты построения контура обратной связи

К настоящему времени количество таких продуктов измеряется десятками, а то и сотнями. Имеющиеся на рынке программного обеспечения продукты различаются набором предоставляемых функций, уровнем поддержки пользователя, надежностью и, соответственно, стоимостью. Существует два подхода к классификации таких продуктов: по цене (неявно предполагается, что она отражает уровень продукта) и по набору реализуемых функций.

По второму критерию средства построения контура обратной связи также разделяют на две группы (которые чаще всего соответствуют ценовому

делению): на профессиональные и настольные. Считается, что профессиональные системы реализуют более сложные алгоритмы планирования и анализа, и для их освоения требуются более глубокие знания в области менеджмента и анализа данных.

Вместе с тем, подобное разделение инструментальных средств становится с каждым годом все более условным, поскольку даже наиболее простые из них обеспечивают вполне приемлемое качество планирования, обеспечивают функционал, состоящий из обширного количества ресурсов для построения контура обратной связи, используют различные виды ресурсов, поддерживают групповую работу над созданием средств проведения опросов, анкетирования и многое другое. Выявить отличия в реализации отдельных функций часто удается лишь при детальном изучении и тестировании системы.

К достаточно устоявшемуся, «базовому» набору функций, реализованному на сегодняшний день практически во всех системах, можно отнести следующие:

- создание/редактирование анкет;
- организация хранения данных;
- частичное редактирование стилей для представления данных;
- организация доступа.

Несмотря на обилие различных сервисных функций, предоставляемых системами построения обратной связи, следует иметь в виду одно важное обстоятельство. Сам по себе метод опроса не дает оптимальный вариант реализации контура обратной связи. Полученные с его помощью результаты могут считаться рациональными и объективными если процесс получения обратной связи был получен с учётом требований безопасности (анонимности). Поэтому любой инструмент построения контура обратной связи следует рассматривать как средство информационной поддержки в процессе принятия решения менеджером или руководством.

Необходимо также помнить, что как бы ни был хорош построенный план, эффект от него окажется нулевым, если в распоряжении пользователя не будет механизма анализа результатов и возможности своевременного внесения адекватных изменений.

Процессы развития складываются из множества контуров управления и самоуправления. Каждый такой контур представляет собой целенаправленный информационно-управленческий процесс.

Воздействие внешней среды вызывает отклонение какого-либо ожидаемого параметра, возникает информация, обратная связь, что, в конечном итоге, и формирует замкнутые контуры и функциональные системы.

1.3. Система «KSU FEEDBACK» как средство реализации контура обратной связи

Зачастую большинство решений руководители организационных систем вынуждены принимать исходя из учетной модели, не смотря на ее очевидную ущербность и подверженность сознательному или бессознательному манипулированию со стороны учетных работников. Вместо того, чтобы анализировать реальные потоки данных, руководители вынуждены ограничиваться только учетными потоками, сформированным по правилам, не учитывающим все многообразие ситуаций.

Для оптимизации и особенно автоматизации управления необходимо разрабатывать формализованные модели.

Так как построение контура обратной связи в обобщённом виде представлено абстрактной моделью взаимодействия субъекта и объекта системы управления, то появляется необходимость реализации данной модели на конкретном примере с учётом требований к функциональности.

Проведя анализ существующих способов построения контура обратной связи можно сделать вывод, что зачастую есть выбор использовать стандартный метод опроса. Однако такой метод является не эффективным в

силу своей громоздкости. Обычно метод представлен следующим образом: создаётся анкета с некоторым множеством вопросов, которые соответствуют некоторой предметной области; определяется группа респондентов для проведения опроса; определяется группа руководителей процесса опроса; проведение опроса; анализ полученных данных (очень часто в таких опросах подведение итогов опроса производиться вручную).

Другой, более эффективный метод – это создание электронного контур обратной связи, служащий повышению эффективности процессов сбора и анализа данных. Использование электронных систем могут усовершенствовать работу обратной связи.

Такой электронной система будет описана на примере системы построения контура обратной связи «KSU Feedback».

В пределах системы «KSU Feedback» контур обратной связи рассматривается как мониторинг состояния системы, для которой осуществляется построения контура обратной связи, в целях контроля за происходящими изменениями.

“Проект, который получил название "KSU Feedback" (первоначальная версия*http://feedback.ksu.ks.ua/*), был реализован в виде веб-приложения на базе фреймворка Django. Суть этого сервиса заключается в проведении анонимного или обычного голосования по четко определенным критериям среди строго определенного множества респондентов.

Действительно, с помощью возможности анонимного голосования достигается объективность оценки. Также возможно удаленное голосование из любого удобного места, что уменьшает влияние заинтересованных лиц на ответ респондента. Благодаря системе одноразовых уникальных ключей организаторы голосования могут сами определить группу людей, которые могут принимать участие в оценивании.”[11, с.41]

«KSU Feedback» был разработан для обеспечения объективных средств для построения цепи обратной связи. Другими словами, это система для сбора и анализа данных, взятых из анкетирования анонимных респондентов. Одной

из главных целей здесь является достижение полной анонимности целевых групп, и четкое определение эти целевые группы. Общая схема системы «KSU Feedback» представлена на рисунке 1.3.

Система «KSU Feedback» оперирует следующими понятиями:

Анкета представляет собой набор вопросов различных типов и вариантов ответов (если тип вопроса это предполагает).

Опрос – это объект который использует набор вопросов из определенной анкеты и по которому может проходить голосование. У опроса есть два управляемых состояния: Открыт – голосование допустимо; Закрыт – голосование запрещено.

Отчет служит для представления полученных результатов в удобном виде. Может содержать текст, диаграммы, картинки, таблицы и т.д.

Директория – иерархический элемент, который служит для древовидной организации данных. Она может быть нескольких типов:

Каталог – предназначен для удобной организации данных; может включать в себя отчеты, опросы, диаграммы или другие каталоги;

Отдел – предназначен для отображения структурных единиц иерархии клиентской организации;

Организация – элемент, который предназначен для отображения структурных единиц иерархии более высоких уровней.

С директорией связано еще несколько понятий:

Домашний каталог – мета-информация о пользователе, которая используется для привязки его к иерархической структуре.

Организация – корневой каталог всех пользователей, которые работают в одной и той же организации. Служит для разделения клиентских данных. Так, например, для всех пользователей Херсонского государственного университета корневым каталогом будет директория с названием "Херсонский государственный университет".

Группа – множество пользователей одной организации, чьи домашние каталоги совпадают.

Ключ – уникальная последовательность символов, которая позволяет открыть доступ к процессу голосования по определенному опросу.

Система «KSU Feedback» представлена совокупностью программных модулей, что соответствует сервис-ориентированной архитектуре. Каждый модуль соответствует определённой функциональности системы. Распределение ответственности между модулями программного кода даёт возможность гибкого изменения и внедрения требований в соответствии с функциональностью, которая необходима пользователю для построения контура обратной связи, а также тестирования отдельных частей при реализации других.

РАЗДЕЛ 2
Разработка внешнего программного интерфейса системы «KSU Feedback»

2.1. API как средство интеграции приложений

Внешний программный интерфейс (иногда интерфейс прикладного программирования) (англ. *Application programming interface, API*) — набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах.

API определяет функциональность, которую предоставляет программа (модуль, библиотека), при этом API позволяет абстрагироваться от того, как именно эта функциональность реализована.[47]

Вешний программный интерфейс - система унифицированных связей, предназначенных для обмена данными между компонентами системы. API задает набор необходимых процедур, их параметров и способов обращения, а также предоставляет функциональность, которую некоторый программный компонент предоставляет другим программным компонентам.

Обратная связь - это одна из характеристик управления. Обратная связь в сложных системах может рассматриваться как передача информации о протекании процесса, на основе которой вырабатывается то или иное управляющее воздействие. В этом случае обратную связь называют информационной. При наличии обратной связи результат предыдущего действия влияет на последующее течение процесса, а самоуправляемая система вследствие наличия в ней прямых и обратных связей, то есть управляющей и корректирующей информации, функционирует по замкнутому циклу. Этот достаточно устойчивый и хорошо

структурированный цикл лежит в основе управленческою цикла и называется контуром управления. Контур управления - это простейшая абстрактная модель управления. Она рассматривает управление как информационный процесс и показывает движение информации в системе. Понятие контура управления лежит в основе циклического характера управления.

Цикличность управления связана с постоянным взаимодействием управляющей и управляемой подсистем по каналам прямой и обратной информационной связи друг с другом, что типично для сложных функциональных систем. Преобразование информации обратной связи в информацию управленческих решений, в сущности, и составляет содержание процесса управления.

Система с обратной связью, несмотря на свою простоту, используется практически в любой системе управления. Основная проблема состоит в ее правильной настройке, чтобы она не мешала, а помогала в реализации проекта.

API может рассматриваться как абстракция над функциональностью и реализацией компонентов.

“Другими словами, API ссылается на набор функций, встроенных в приложения, которые могут быть использованы другими приложениями (или сам по себе, как мы увидим позже), чтобы взаимодействовать с приложением. API является отличным способом надежно и безопасно раскрыть функциональность приложения для внешних приложений.Все функциональные возможности, доступные внешним приложениям, ограничиваются предоставляемым API.”[30].

API системы “KSUFEEDBACK” представляет собой инструмент, с помощью которого информация, накопленная системой web-аналитики, становится доступной для использования на стороне другого сайта. Этот механизм представляет огромные возможности для программистов и администраторов сайта. Например, можно перенаправить посетителя на

нужную страницу в зависимости от его статуса и прав предоставив ему таким образом уникальную информацию.

Механизм API реализован в виде набора функций, которые могут быть использованы на сайте для получения информации с сервера аналитики. Для использования функций API необходимо вставить на страницу сайта код виджета, который доступен владельцу сайта после регистрации в системе “KSU Feedback”. Таким образомAPI даёт возможность наладить связи различным приложениям для эффективного общения друг с другом. API хорошо подходит для распространения приложения с помощью других сайтов.

“В веб-разработке, как правило, используется определенный набор HTTP-запросов, а также определение структуры HTTP-ответов, для выражения которых используют XML или JSON форматы. WebAPI является практически синонимом для веб-службы, хотя в последнее время осуществлен переход от SOAP к REST типу коммуникации. Веб-интерфейсы, обеспечивающие сочетание нескольких сервисов в новых приложениях, известны как гибридные.” [30]

2.2. Варианты реализации внешнего программного интерфейса для веб-приложений

Несмотря на многочисленные трудности, связанные с реализацией программных систем в среде всемирной паутины World WideWeb, создание решений на платформе Веб вот уже более десяти лет образует наиболее перспективное и динамично развивающееся направление современной индустрии разработки приложений.

Приложения подобные Gmail [6], Google Maps [7], Flickr [8] оставили в прошлом привычную для разработчиков модель «тонкого» клиента, и заставили в полной мере работать стек технологий DHTML (JavaScript [5] /DOM [3] /CSS [4]), заложенный в стандартной архитектуре современного

Веб-клиента. Вместе с тем нельзя не заметить, что создание таких приложений является очень трудоемкой, сложной в техническом плане.

REST (Representationalstatetransfer) – это стиль архитектуры программного обеспечения для распределенных систем, таких как WorldWideWeb, который, как правило, используется для построения веб-служб. Термин REST был введен в 2000 году Роем Филдингом, одним из авторов HTTP-протокола. Системы, поддерживающие REST, называются RESTful-системами. В общем случае REST является очень простым интерфейсом управления информацией без использования каких-то дополнительных внутренних прослоек. Каждая единица информации однозначно определяется глобальным идентификатором, таким как URL. Каждая URL в свою очередь имеет строго заданный формат.

Отсутствие дополнительных внутренних прослоек означает передачу данных в том же виде, что и сами данные. Т.е. мы не заворачиваем данные в XML, как это делает SOAP и XML-RPC, не используем AMF, как это делает Flash и т.д. Просто отдаем сами данные. Но это может повлечь за собой проблемы связанные с безопасностью передачи данных. [48]

Каждая единица информации однозначно определяется URL – это значит, что URL по сути является первичным ключом для единицы данных. Т.е. например третья книга с книжной полки будет иметь вид /book/3, а 35 страница в этой книге — /book/3/page/35. Отсюда и получается строго заданный формат. Причем совершенно не имеет значения, в каком формате находятся данные по адресу /book/3/page/35 – это может быть и HTML, и отсканированная копия в виде jpeg-файла, и документ Microsoft Word.

Как происходит управление информацией сервиса – это целиком и полностью основывается на протоколе передачи данных. Наиболее распространенный протокол конечно же HTTP. Для HTTP действие над данными задается с помощью методов: GET (получить), PUT (добавить, заменить), POST (добавить, изменить, удалить), DELETE (удалить). Таким образом, действия CRUD (Create-Read-Updtae-Delete) могут выполняться как

со всеми 4-мя методами, так и только с помощью GET и POST. Но подобны подход ограничивает допустимые методы для оперирования данными.

“REST — это не стандарт и не спецификация, а архитектурный стиль, выстроенный на существующих, хорошо известных и контролируемых консорциумом W3C стандартах, таких, как HTTP, URI (UniformResourceIdentifier), XML и RDF (ResourceDescriptionFormat). В REST-сервисах акцент сделан на доступ к ресурсам, а не на исполнение удаленных сервисов; в этом их кардинальное отличие от SOAP-сервисов. Если SOAP-клиенты запрашивают выполнение действия на сервере, то REST-клиенты попросту требуют сам ресурс. Например, вместо того чтобы запрашивать удаленное исполнение функции для нахождения нужного формуляра заказа, пользователь просто запрашивает этот формуляр, примерно так же, как статичнуюWeb-страницу.”[52].

С точки зрения теории это вроде бы несущественная разница, но на практике она огромная, особенно в отношении инфраструктуры, необходимой для поддержания каждого из этих подходов. Технически REST-сервисы можно реализовать при помощи статических HTML-страниц, что попросту неприменимо к SOAP-сервисам.

И хотя может случиться так, что REST-запрос вызовет к исполнению удаленный код, создающий возвращаемый XML-отклик, однако данный факт в REST замаскирован лучше, чем у его SOAP-конкурента, который обязательно требует указания той функции, которая будет выполнена. SOAP-сервисам, как правило (хотя и не обязательно), необходим сервер приложений, такой, как WebLogic компании BEA Systems, WebSpherекомпании IBM или Tomcat, для разбора XML-запросов, требуемых по протоколу SOAP, и исполнения соответствующего программного кода. У REST-сервисов таких требований нет, так как желаемый ресурс явно указывается в строке URI. С REST-запросом может быть связана некая бизнес-логика, которая потребует доступа к источнику данных, или же условная логика, основанная на параметрах запроса, хотя это вовсе не

обязательно и определяется исключительно внутрикорпоративными стандартами организации, проектными практиками и ограничениями на длину URI.

Популярность REST неудивительна, учитывая тот факт, что встраивание REST-сервисов в хостируемый сайт — относительно простой процесс по сравнению с усилиями, требуемыми для развертывания инфраструктуры и кода с целью поддержания SOAP-коммуникаций. Еще в 2003 г. Джефф Бар, пропагандист Web-сервисов из Amazon.com, рассказал, что его компания обрабатывает больше REST-, чем SOAP-запросов. [7]

Следует также учесть, что почти все новостные каналы RSS и ATOM по сути своей являются REST-сервисами. Собственно, любой URI-запрос, предполагающий обычный XML-ответ (POX — Plain-Old XML), технически остается REST-сервисом, значит, REST-сервисы почти наверняка прямо сейчас работают в вашей организации. Проверьте и убедитесь в этом сами, мы подождем.

За и против

Пропагандисты REST-сервисов утверждают, что они лучше масштабируются и, следовательно, лучше справляются с большими объемами обслуживания. SOAP-сервисы гораздо сильнее загружают вычислительные ресурсы, поскольку требуют разбора XML-кода и упорядочивания объектов, — а для оптимизации управления и ускорения этих операций необходимо специальное программное и аппаратное обеспечение. Тогда как REST-сервисы попросту являются HTTP-запросами, а значит, легко контролируются штатными средствами выравнивания нагрузки обычных устройств управления трафиком. Мониторинг использования REST-сервисов тоже значительно проще и зачастую менее дорогостоящий, поскольку мониторинг на базе URI уже стал установившейся технологией; предлагаются как соответствующие коммерческие продукты, так и решения с открытым исходным кодом.

“REST-сервисы проще реализовать, поскольку они базируются на хорошо известных Web-протоколах и не требуют от разработчика изучения WSDL, SOAP и массы других WS-спецификаций, используемых для управления и обеспечения безопасности SOAP-сервисов. Наипростейший REST-сервис можно реализовать за несколько минут — статичный XML-файл, возвращаемый Web-сервисом, это технически и есть REST-сервис, ведь XML-данные запрашивались через HTTP.”[53].

Однако это вряд ли может стать оптимальным способом построения серьёзной сервисной инфраструктуры такой как система «KSU Feedback», но для статичных или редко меняющихся ресурсов такая возможность весьма привлекательна.

Поскольку технология REST базируется на HTTP, то она несет на себе отпечаток ненадежности этого протокола и невозможности сохранения состояния (его stateless-характера). Здесь-то и пригодились бы сложные WS-спецификации. Безопасный, надежный обмен сообщениями и поддержка транзакций — вот лишь некоторые задачи, решаемые в масштабе SOAP (этого нет в REST).

REST также не обеспечивает стандартизованного механизма доступа к ресурсам, следовательно, разработчику самому придется определять такие стандарты и обеспечивать их исполнение, а также информировать о способе доступа сервисов к ресурсам своих партнеров, разработчиков и пользователей. Причем нужно будет своевременно обновлять этот механизм коммуникаций в случае любых изменений сервиса. В случае SOAP это выполняет WSDL, действуя как динамический посредник между сервисом и его клиентами.

2.3. Сервисы, предоставляющие API

Примеры сервисов, предоставляющих API

Почти любой сервис, достигающий определенных размеров и набравший потенциал, открывает свой API для сторонних приложений. Наиболее известные сервисы с API:

Twitter — предоставляет широчайшие возможности для сторонних приложений, начиная от выдачи информации о конкретном аккаунте и заканчивая поиском по собственной базе и аутентификацией. На базе TwitterAPI создано огромное количество сторонних сайтов и приложений. Кроме того, многие сервисы принимают Twitter-аутентификацию как свою собственную;

Feedburner- его API позволяет получать информацию о RSS канале выбранного блога, а также предоставляет некоторые возможности по управлению его настройками;

ВКонтакте — имеет хорошо проработанный интерфейс взаимодействия со сторонними приложениями, предоставляя последним возможности по получению информации о выбранном пользователе, доступу к функционалу его странички и многое другое. ВКонтактеAPI изначально было ориентировано только на разработку Flash-приложений, но сейчас он стал доступен и "обычным" веб-сайтам.

API-ориентированное веб-приложение - это веб-приложение, в котором весь или большую часть функционала реализуется через вызовы API. Например, если вы должны авторизоваться, то вы отправляете свои данные через функцию API. Результатом выполнения этой функции будет либо авторизация, либо сообщение об ошибке. Другой характерной особенностью API-ориентированного веб-приложения является то, что API не зависит от состояния пользователя. Это ограничение на самом деле хорошее - это «заставляет» разработчика создавать API, который не зависит от текущего состояния пользователя, а представляет собой чистую функциональность, что, в свою очередь, облегчает тестирование, так как текущее состояние пользователя не нужно имитировать.

Одним из преимуществ создания API-ориентированных веб-приложений, является то, что это поможет построить функциональность, которая может быть использована любым устройством, будь то браузер, мобильный телефон, планшет или даже desktop приложение. Все, что нужно сделать, это создать API, таким образом, чтобы все эти устройства могли взаимодействовать с ним. Таким образом можно создать централизованное приложение, которое может принимать вызовы с любого устройства, которое есть у пользователя. На рис.2.1. отображена схема API-ориентированного веб-приложения.

2.4. Организация работы API веб-службы SOAP

Чтобы извлечь наибольшую пользу из веб-сервисов и SOAP, нужно иметь четкое представление о принципах и технологии, на которых они основаны. Веб-сервис – доступный по сети интерфейс для функциональности приложения, построенный с использованием стандартных Интернет технологий. Другими словами, если приложение может получить доступ к функционалу через сеть, используя комбинацию протоколов, таких как HTTP, XML, SMTP, то оно является веб-сервисом. На рисунке 1.2 изображена схема действия веб-сервиса.

Принципы работы любого API достаточно унифицированы и схожи друг с другом. По сути, API представляет собой набор функций, которое может вызывать сторонне приложение и которые, условно, можно разделить на две большие группы:

Возвращающие — стороннее приложение запрашивает какой-либо метод (с конкретными параметрами) сервиса-сервера и в ответ получает запрашиваемую информацию в заранее определенном формате;

Изменяющие — приложение-клиент вызывает какую-либо функцию сайта-сервера (опять-таки с указанными параметрами), которая изменяет определенные настройки на нем, либо вводит новую информацию.

Главное, что следует помнить — вызов, совершенный через GET не должен менять состояние сервера. Это в свою очередь значит, что ваши запросы могут кэшироваться любым промежуточным прокси (снижение нагрузки). Таким образом Вы, как разработчик сервера, не должны публиковать GET методы, которые меняют данные в вашей базе данных. Это нарушает философию RESTful, особенно второй пункт, описанный выше. Ваши GET вызовы не должны даже оставлять записей в access.log или обновлять данные типа “Lastloggedin”. Если вы меняете данные в базе, это обязательно должны быть методы POST/PUT.

В общем и целом, всю работу с TwitterAPI можно свести к следующей схеме:

- Отправляем GET-запрос спараметрам (если есть) по адресу нужной нам функции;
- Получаем XML или jSON данные (в зависимости от выбранного формата);Если есть необходимость — парсим эти данные.

“Веб-сервисы и SOAP призваны решить проблемы кросс-платформенного взаимодействия приложений.

В настоящее время используемые технологии удаленного вызова методов (DCOM, CORBA/IIOP и RMI) довольно сложны в настройке и организации взаимодействия. Это влечет за собой проблемы в эксплуатации и функционировании распределенных систем (проблемы безопасности, транспорт через брандмауэры и т.д.). Существующие проблемы успешно решены созданием SOAP(SimpleObjectAccessProtocol), простого протокола, основанного на XML, для обмена сообщениями в распределенных средах (WWW). Он предназначен для создания веб-сервисов и удаленного вызова методов. SOAP можно использовать с разными транспортными протоколами, включая HTTP, SMTP и т.д.”[49].

Веб-сервисы - это функциональность и данные, предоставляемые для использования внешними приложениями, которые работают с сервисами

посредством стандартных протоколов и форматов данных. Веб-сервисы полностью независимы от языка и платформы реализации.

Будучи синтаксисом для описания данных, XML управляется определением (при помощи DTD или схем) и позволяет программно управлять информацией. Это означает, что большая часть предполагаемой работы может быть получена из B2B-взаимодействий. Теги могут быть согласованы, интерфейсы определены, а обработка данных стандартизована. Web-сервисы – это компонентные программы многократного пользования, которые используют XML как стандартную расширяемую сетевую структуру, с целью облегчения этого вида межкомпьютерного взаимодействия.

“Web-сервисы предоставляют интерфейсы для передачи компонентных данных и бизнес-логики по HTTP. Огромное количество данных располагается сразу за скриптами, выполняемыми на стороне сервера, в традиционных репозиториях, ожидая доступа к данным из Web-броузеров и клиентских приложений.”[44].Web-сервисы обещают восстановить ценные свойства корпоративных приложений, неиспользуемые в настоящее время во многих областях предприятия.

XML играет ключевую роль в интеграции Web-резидентных данных в корпоративные приложения, а также в координировании бизнес-логики, соединяющей составные части. Определенные бизнес-задачи и услуги (включая логику автоматизации, бизнес-логику, логику упорядочения компонентов, логику формирования транзакций и т.д.) могут быть включены в XML-документ и интегрированы в существующие бизнес-среды. Это позволяет предприятиям усилить существующие ценные свойства и процессы, а также представить эту информацию в виде Web-сервисов, упрощая деловые операции и поддерживая цепное взаимодействие во всей сети WWW.

Поскольку XML удобен для чтения и основан на тексте, он идеально подходит в качестве транспортной структуры для слабосвязанныхWeb-

сервисов. Основным моментом является то, что автоматизированные транзакции увеличивают производительность, уменьшают затраты и улучшают услуги. Присутствие стандартов в сети делают возможными автоматизированные транзакции, приводя к росту производительности.

SOAP – это технология, которая произошла от раннего XML-стандарта (XML-RPC) и, в некотором смысле, стремится к появляющемуся стандарту ebXML (electronicbusinessXML). ebXML – это незавершенная технология, направленная на обеспечение всестороннего определения общедоступных бизнес-сообщений между торговыми партнерами. SOAP является менее обширным и менее сложным в выполнении.

Web-сервисы отделяют объекты от связывающей их платформы. То есть Web-сервисы облегчают взаимодействие между платформно-независимыми объектами, способными обращаться к данным из любой точки Web. Осуществляя удаление от частных платформ, Web-сервисы полагаются скорее на слабые, чем сильные связи между Web-компонентами. Системы, которые полагаются на специфичные для одной платформы объекты, называются сильно связанными системами, т.к. они опираются на однозначный, но хрупкий интерфейс. Если какая-либо часть соединения между объектом-приложением и серверным объектом разорвана, или если вызов некорректен, это может привести к непредсказуемым результатам. [31].

Система «KSUFeedback»является слабосвязанной системой засчёт разделения на модули и предоставление программных интерфейсов. Это обеспечивает стабильность работы и адекватную реакцию при обнаружении ошибок и срабатывании исключений.

Структура Web-сервисов состоит из цикла “публиковать-находить-связывать”, посредством которого поставщики сервисов делают данные, содержимое или услуги доступными для зарегистрированных запрашивающих сторон, потребляющих ресурсы, локализуя сервисы и соединяясь с ними. Запрашивающие приложения настраиваются на Web-

сервисы при помощи WSDL (WebServicesDefinitionLanguage – язык описания Web-сервисов), который предоставляет низкоуровневую техническую информацию о желаемом сервисе, допускает обращение приложений к информации XMLSchema для кодировки данных и гарантирует, что правильные операции будут осуществлены по правильным протоколам.

Однако WSDL не только является языком описания интерфейсов, он также включает конструкции, позволяющие описывать информацию об адресе и протоколе публикуемого Web-сервиса. Интересно то, что WSDL описывает абстрактный интерфейс для Web-сервиса, одновременно позволяя (в мучительных подробностях) связывать Web-сервис с определенным транспортным механизмом, таким как HTTP. Абстрагируя интерфейс, WSDL функционирует как технология многократно используемыхWeb-сервисов. Связывая Web-сервис с определенным транспортным механизмом, WSDL делает абстракцию конкретной.

SOAP позволяет создавать приложения, удаленно вызывая методы объектов.SOAP устраняет требование того, что две системы должны запускаться на одной платформе или быть написаны на одном языке программирования. Вместо вызова методов по определенному бинарному протоколу, пакет SOAP использует XML, основанный на тексте синтаксис для осуществления вызова методов. Вся информация между запрашивающим приложением и принимающим объектом пересылается в виде данных, заключенных между тегами, в XML-потоке по HTTP. С точки зрения Web-сервисов SOAP может быть реализован в качестве как клиента, так и сервера.

Для предоставления функциональности системы «KSUFeedback»сторонним приложениям необходимо было определить XMLSchemaдля объектов с которыми будет работать API. Это нужно для дальнейшего работы с сообщениями SOAPClient и преобразованием ответов сервера в формате XMLдля дальнейшего отображения на стороне пользователя.

Механизм взаимодействия клиента и сервера

1. Клиентское приложение создает экземпляр объекта SOAPClient

2. SOAPClient читает файлы описания методов веб-сервиса (WSDL и WebServicesMetaLanguage - WSML). Эти файлы могут храниться и на клиенте.

3. Клиентское приложение, используя возможности позднего связывания методов объекта SOAPClient, вызывает метод сервиса. SOAPClient формирует пакет запроса (SOAPEnvelope) и отправляет на сервер. Возможно использование любого транспортного протокола, но, как правило, используется HTTP.

4. Пакет принимает серверное приложение Listener, создает объект SOAPServer и передает ему пакет запроса

5. SOAPServer читает описание веб-сервиса, загружает описание и пакет запроса в XMLDOM деревья

6. SOAPServer вызывает метод объекта/приложения, реализующего сервис

7. Результаты выполнения метода или описание ошибки конвертируются объектом SOAPServer в пакет ответа и отправляются клиенту

8. Объект SOAPClient проводит разбор принятого пакета и возвращает клиентскому приложению результаты работы сервиса или описание возникшей ошибки.

WSDL файл это документ в формате XML, описывающий методы, предоставляемые веб-сервисом. Также параметры методов, их типы, названия и местонахождение Listener`a сервиса. SOAPToolkitвизард автоматически генерирует этот документ.

SOAPEnvelope (Пакет) - XML документ, который содержит в себе запрос/ответ на выполнение метода. Удобнее всего рассматривать его как почтовый конверт, в который вложена информация. Тэг Envelope должен быть корневым элементом пакета. Элемент Header не обязателен, а Body должен присутствовать и быть прямым потомком элемента Envelope. В случае ошибки выполнения метода сервер формирует пакет, содержащий в

тэге Body элемент Fault, который содержит подробное описание ошибки. Если вы пользуетесь высокоуровневыми интерфейсами SOAPClient, SOAPServer, то вам не придется вдаваться в тонкости формата пакета, но, при желании, можно воспользоваться низкоуровневыми интерфейсами или же вообще создать пакет "руками".

Объектная модель SOAPToolkit дает возможность работать с объектами низкоуровневогоAPI:

- SoapConnector - Обеспечивает работу с транспортным протоколом для обмена SOAP пакетами
- SoapConnectorFactory - Обеспечивает метод создания коннектора для транспортного протокола, указанного в WSDL файле (тэг)
- SoapReader - Читает SOAP сообщения и строит XMLDOM деревья
- SoapSerializer - Содержит методы создания SOAP сообщения
- IsoapTypeMapper, SoapTypeMapperFactory - Интерфейсы, позволяющие работать со сложными типами данных.[27]

Используя объекты высокоуровневогоAPI можно передавать данные только простых типов (int, srting, float …), но спецификация SOAP 1.1 допускает работу с более сложными типами данных, например с массивами, структурами, списками и их комбинациями. Для работы с такими типами приходится использовать интерфейсы IsoapTypeMapper и SoapTypeMapperFactory.

Клиент SOAP – это программа, которая создает XML-документ, содержащий информацию, необходимую для удаленного вызова метода в распределенной системе. В дополнение к тому, что клиенты SOAP пополняют разнообразие приложений, клиенты SOAP могут служить Web-серверными или основанными на серверной технологии приложениями.

Сообщения и запросы от клиентов SOAP обычно пересылаются по HTTP. В результате документы SOAP способны обойти почти любой брандмауэр (firewall), допуская обмен информацией между разными платформами.

Сервер SOAP – это специальный код, который слушает SOAP сообщения и действует как распределитель и интерпретатор SOAP документов. Внешние Web-сервисы могут взаимодействовать с приложениями-серверами, основанными на технологии J2EE, которые обрабатывают запросы SOAP от множества клиентов.

Серверы SOAP гарантируют, что документы, полученные по HTTP соединению, преобразованы к языку, который понятен объекту и всем окружающим. Поскольку все коммуникации осуществляются в форме XML, объекты, написанные на одном языке (скажем, Java), могут связываться через SOAP с объектами, написанными на другом языке (например, C++). Работа SOAP сервера заключается в том, чтобы удостовериться, что конечные пункты понимают обслуживающий их SOAP.[27]

SOAP является расширяемым, потому что SOAP клиенты, серверы и сам протокол могут развиваться, не разрушая уже существующие приложения.

Формат сообщений. Все вышеперечисленное выполняется в контексте стандартизованного формата сообщений. Основная часть сообщения имеет MIME тип “text/xml” и содержит SOAP конверт. Этот конверт является XML документом. Конверт содержит заголовок (опционально) и тело (обязательно). Тело конверта всегда предназначено конечному получателю сообщения, тогда как записи в заголовке могут быть адресованы узлам, выполняющим промежуточную обработку. К телу также могут быть присоединены бинарные или какие-либо другие вложения.

SOAP предоставляет клиенту возможность определить, какой из промежуточных узлов обрабатывает ту или иную запись заголовка. Поскольку заголовки ортогональны основному содержимому SOAP сообщения, они полезны для добавления к сообщению той информации, которая не влияет на обработку его тела.

Заголовки, к примеру, могут быть использованы для предоставления цифровой подписи к запросу, находящемуся в теле сообщения. В этом случае сервер аутентификации или авторизации может обработать запись заголовка

(независящего от тела), разбирая информацию для проверки правильности подписи. Как только проверка правильности прошла успешно, остальная часть конверта будет передана на SOAP сервер, который обработает тело сообщения. Более детальное рассмотрение конверта SOAP поможет выяснить расположение и цель заголовка и элементов тела.

Анализ конверта SOAP

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Header>
<t:Transactionxmlns:t="some-URI">
SOAP-ENV:mustUnderstand="1" 5
</t:Transaction>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
<m:GetPollBySyrveyIDxmlns:m="some-URI">
<ID>700</ID>
</m:GetPollBySyrveyID>
</SOAP-ENV:Body>
</SOAP-Envelope>
```

В этом примере запрос GetPollBySyrveyID (получить анкету по идентификатору опроса)посылается службе идентификатор опроса на Web. Запрос принимает строковый параметр, название функции, и возвращает объект Анкета в формате ответаSOAP.

Конверт SOAP – это верхний элемент XML документа, который представляет сообщение SOAP. Пространства имен XML используются для отделения идентификаторов SOAP от специфичных идентификаторов приложения. Пространства имен XML используются в основном для того, чтобы квалифицировать элементы в сообщении или отнести их к определенной области. Для понимания пространств имен SOAP необходимо

быть знакомым со спецификацией пространств имен XML. Пространство имен можно представить как идентификатор некоторой окрестности элементов, который помогает уникально идентифицировать элементы SOAP, ассоциируя их с определенным местоположением.

Тело сообщения SOAP в примере содержит полезную нагрузку XML, которая, предположим, выполняет RPC (RemoteProcedureCalling – вызов удаленной процедуры). SOAP – это не только модульная, но также и довольно скрытая модель организации пакетов.

Здесь ничего явно не указывает на то, что был начат RPC. Все, что мы видим в теле, – это пара XML элементов, один из которых квалифицирован пространством имен. Задачей SOAP сервера является понимание семантики документа и осуществление правильных действий. В действительности, сервер предоставляет структуру для работы с полезной нагрузкой XML (XMLpayload) “значимым” способом. Слово “значимый” в данном случае подразумевает, что сервер осуществляет вызов удаленной процедуры в некоторой вспомогательной базе данных, чтобы получить объект с помощью параметра, который содержится в теле сообщения.

Сообщения SOAP по существу являются односторонними пересылками информации от отправителя к получателю, однако SOAP сообщения часто объединяются для того, чтобы улучшить механизмы запроса и ответа. Для осуществления RPC в SOAP необходимо соблюдать некоторые соглашения. Прежде всего, сообщения запроса и ответа должны быть закодированы как структуры. Для каждого входного параметра операции должен существовать элемент с таким же именем, как у параметра. И для каждого выходного параметра должен существовать элемент с соответствующим именем.

Далее следует укороченный вид SOAP-RPC сообщения, представленного ранее. Показаны только тела сообщений конвертов запроса и ответа SOAP.

Request (Запрос)

<SOAP-ENV:Body>

<m:GetPollBySyrveyIDxmlns:m="some-URI">

<ID>700</ID>

</m: GetPollBySyrveyID>

</SOAP-ENV:Body>

Response (Ответ)

<SOAP-ENV:Body>

<m:GetPollBySurveyIDResponsexmlns:m="some-URI">

<price>22.50</price>

</m:GetPollBySurveyIDResponse>

</SOAP-ENV:Body>

ЗапросвызываетметодGetPollBySurveyID. Ответ определяет операцию GetPollBySurveyIDResponse. Общее соглашение SOAP требует, чтобы в конце операции запроса (Request) добавлялось слово Response для создания структуры ответа (Response). Данная выходная структура содержит элемент с названием ID (идентификатор), который возвращает результат вызова метода, возможно, в виде объекта с его свойствами (полями).

Важно отметить, что нигде в конверте SOAP не выражены явно типы данных, поэтому мы действительно не знаем тип ID или тип результирующего параметра объекта(анкеты), только лишь смотря на сообщение SOAP. Клиентские приложения определяют типы данных либо открыто посредством кодирования, либо частным образом через согласованные контакты с сервером. В обоих случаях эти определения не включены явно в SOAP сообщение.

И наконец, для осуществления RPC необходим низкоуровневый протокол, такой как HTTP. Хотя спецификация SOAP 1.0 предписывает использование HTTP, как транспортного протокола, SOAP 1.1 (а также спецификация “Сообщение SOAP с вложениями”) разрешает использовать FTP, SMTP и даже (возможно) TCP/IP сокеты. Все правила сериализации и кодирования SOAP относятся также и к RPC-параметрам.

РАЗДЕЛ 3
Тестирование API

3.1. Тестирование приложений, предоставляющих API

Внешний программный интерфейс (API) позволяет программному обеспечению общаться с другими приложениями. Как и любое другое программное обеспечение, интерфейсы должны быть протестированы.Возникает вопрос, как можно разработать тесты без воздействия через графический интерфейс? Ответом может стать использование программных интерфейсов, которые позволяют получить прямой доступ к той функциональности, которую надо протестировать.

SoapUIPro (или QualityLogic) – кросс-платформенное решение для тестирования веб-сервисов и API, с помощью SoapUIPro можно быстро создать и запустить автоматизированные функциональные, регрессионные и тесты безопасности.

Тестирование API существенно отличается от других видов тестирования так как GUI редко участвует и есть некоторые трудности, в том числе и в том чтобы просто убедиться, что система готова к тестированию. Необходимо проверить функциональность, соответствие требованиям и безопасности.

Во время тестированияприложения интересными задачами для тестировщиков являются:

1. Обеспечение тестирования различными параметрами API вызовов таким образом, чтобы проверить функциональность и безопасность. Это включает в себя изучение граничных условий.

2. Создание комбинаций значений параметров для вызовов с двумя или более параметрами.

3. Определение содержания API вызовов. Это может включать создание внешних условий окружающей среды (файлы, периферийные устройства и т. д.), а также внутренних хранимых данных, которые влияют на API.

4. Последовательность API вызовов для проверки функциональности.

Порядок тестов должен соответствовать плану тестирования.

Каждый тест должен быть максимально автономным и изолированным от зависимостей насколько это возможно. Основным ориентиром является выявление наиболее распространенных параметров и условий, которые используются при вызове API методов.

API тестирование отличается от других видов тестирования тем, что GUI редко участвует в тестирование API. Необходимо инициализировать тестовое состояние: создание условий, при которых API будет использоваться.

Тестовые сценарии для тестирования API основаны на входных данных:

- Возвращаемое значение на основе данных условиях

Можно отправлять различные комбинации запросов и проверять ответ на соответствие ожидаемому результату.

- Возвращаемое значение не определено

Поведение API в системе должно быть проверено когда нет возвращаемого значения.

- Запуск некоторых других API / событий / исключений

Методы API, если вызывают некоторое событие или исключение, то должны быть отслежены соответствующими обработчиками. Набор тестов должен проверять вызовы API.

- Обновление данных

Обновление данных будет иметь некоторое воздействие на систему, поэтому оно должно быть провалидировано.

- Изменение определенных ресурсов

Если API вызов изменяет некоторые ресурсы, например, делает обновление базы данных, то это должен быть подтвержден доступк соответствующим ресурсам.

Сравнение API тестирования с JUnit тестами (Модульное тестирование).

1. API тестирование не является модульным тестированием. Модульным тестированием зачастую занимаются разработчиков, тестированием API –

QAкоманда. Тестирование API относится к тестированию так называемого черного ящика, когда как модульное тестирование, по существу относится к тестированию белого ящика.

2. API-тестирование и юнит-тестирование ориентированы на уровень кода. Есть несколько инструментов с открытым исходным кодом, доступных для тестирования API: Webinject, JUnit, XMLUnit, HttpUnit, ANT т.д.

3. Процесс API тестирования заключается в проверке методов.

4. Модульное тестирование обеспечивает команда разработчиков, чтобы каждый модуль предоставлял свои Unit тесты перед сборкой программного продукта.

5. Другое ключевое различие между API и модульным тестированием заключается в разработке теста. Юнит-тесты обычно предназначены для проверки каждого блока кода в отдельности. Unit тестирование часто не учитывает уровень системы взаимодействия различных элементов. API тестирование предназначено для уровня функциональности системы, так как она будет использоваться конечным пользователем. Это означает, что API тестирование должно быть гораздо шире, чем юнит-тестирование.

Таким образом, тестирование API – это специальные команды, вызывающие функции APIсо всеми основными вариантами параметром. Тестирование API имеет свои особенности.

Для нагрузочного тестирования используют JMeter. С помощью тестов для проверки функциональности формируется сценарий нагрузки, которая свойственна для веб-сервисов. Смена параметров позволяет отследить динамику нагрузки при обращении (вызове) пользователем какой-либо функции приложения.

3.2. Разработка тестов для “KSU Feedback” API

Тестовые сценарии должны быть сгруппированы по категориям (граничные случаи, нулевые входы и т. д.).

Стратегия тестирования:

- Создание тестов: тесты пишутся на стандартном языке программирования, который позволяет использовать API-функции.
- Выполнение тестов: понадобятся специальная среда для выполнения тестов.
- Оценка результатов тестов: обычно, ожидаемые результаты определяются в момент создания тестов.

Атрибуты качества:

- Сопровождение и поддержка: высокая. API-функции более стабильны нежели пользовательский интерфейс.
- Проверка: средняя. Тесты пишутся на том же языке программирования и в той же среде, что и модульные/API-тесты, а это означает, что многие разработчики смогут разобраться в коде.
- Целостность и зависимость: средняя. Тестирование реального графического интерфейса пользователя невозможно полностью автоматизировать. Какие-то тесты все равно придется провести вручную.
- Возможность повторного использования: низкая.

Веб-сервисы – достаточно распространенный вид компонент распределенных систем. Фактически это один из интерфейсов удаленного вызова процедур. То есть, из некоторой программы мы можем отправить запрос на выполнение некоторой операции на стороне сервера и получить результат выполнения операции, при этом графический интерфейс не задействован. Подобные компоненты позволяют реализовать интеграцию между различными системами, которые изначально между собой не связаны. Отчасти это делает веб-сервисы достаточно популярными, как результат – они используются в большом количестве приложений, а это в свою очередь влечет необходимость их тестировать, в том числе и автоматически. Для этой задачи есть и специальные средства, наподобие soapUI, но помимо этого возможность тестирования веб-сервисов имеется и в TestComplete

(вTestComplete, начиная с 6-й версии как раз была добавлена возможность тестирования веб-сервисов). [28]

Начальное условие в тестировании API включает в себя создание вариантов, при которых API будет вызван. Возможно, API может быть вызван по причине некоторых событий или в ответ на некоторые исключения.

SOAP предоставляет модель обработки ситуаций, когда при обработке сообщения возникает ошибка. SOAP отдельно рассматривает условия, при которых ошибка возникла, и возможность указания на факт возникновения ошибки узлу, сгенерировавшему ошибку, либо любому другому узлу. Возможность указания на факт возникновения ошибки зависит от используемого механизма передачи сообщений, и одним из аспектов спецификации привязки SOAP к нижележащему протоколу является определение механизма оповещения об ошибках в случае их возникновения.

Анализ угроз. Угрозы веб-сервисов включают в себя неправильное использование сервиса, отказ от обслуживания и несанкционированное использование сервиса. В большинстве случаев неправильное использование сервиса должно предотвращаться самим сервисом; сервис должен проверять пригодность всех входных и выходных данных и выдавать сообщение об ошибке, если данные ввода-вывода выходят за ожидаемые предельные значения. Отказ от обслуживания можно устранить, создавая множественные варианты сервиса на различных машинах, используя запросы выравнивания нагрузки. Несанкционированное использование предотвращается путем надежной аутентификации клиента, запрашивающего услугу. Конкретному веб-сервису может не требоваться аутентификация запросов клиентов или он может требовать очень строгой аутентификации запросов клиентов, в зависимости от характера веб-сервиса.

Отличительной особенностью метода юнит-тестов является то, что нужно строить не модель поведения системы, а модель тестирования использования программных интерфейсов (API). При этом в качестве состояний выделяются

тестируемые функции(методы), а в качестве событий (условий переходов) рассматриваются результаты выполнения функций.

Для автоматизации задач, возникающих при разработке тестов для API, в работе используются существующие подходы преобразования моделей конечных автоматов в XML-формат с последующей генерацией из полученного представления тестовых сценариев и исходного кода тестового приложения.

Данный метод разработки тестов позволяет решить задачу их создания для API, а также автоматизировать некоторые из этапов решения этой задачи.

В работе рассматривается процесс разработки тестов для функционального тестирования программных интерфейсов приложений (API). Эти интерфейсы предназначены для предоставления другим программам функциональности программной системы, в которой они реализованы.

Для проверки соответствия интерфейса спецификации необходимо выполнить следующие виды тестирования:

1. Синтаксическое тестирование отдельных функций интерфейса. Проверка отдельных функций на широком спектре входных данных.

2. Проверка корректности предоставления функциональности отдельных (совокупности) функций интерфейса.

3. Проверка некорректного применения отдельных (совокупности) функций интерфейса.

4. Проверка взаимодействия при интеграции функций интерфейса с другими интерфейсами или программными компонентами.

Результатом применения описываемого метода является приложение, которое тестирует функции программного интерфейса в соответствии со спецификацией. Часть тестов реализована за счёт приложения SOAP UI.

Для разработки тестовых сценариев строится модель, описывающая использование функций тестируемого интерфейса. Модель необходима для формализации процесса разработки тестовых сценариев и тестовых приложений. Применение модели позволяет автоматизировать некоторые

этапы процесса разработки тестов, а также в дальнейшем, при внесения изменений в спецификацию, поддерживать соответствие между спецификацией тестируемого интерфейса и тестовым приложением.

Тестовые сценарии запускаются в автоматическом режиме. Результатом выполнения приложения является вердикт: тест прошел – положительный результат (passed) или тест не прошел – отрицательный результат (failed).При этом увеличивается вероятность обнаружения неисправностей, не только связанных со свойствами отдельных функций, но и с общей функциональностью системы, предоставляемой через интерфейс.

ВЫВОДЫ

Процесс управления начинается с постановки проблемы, определения круга целей и задач, деятельности по их реализации и заканчивается их достижением. После этого на основе обратной связи, информации о результатах управленческой деятельности ставятся новые задачи, и цикл повторяется. Учет современных тенденций управления не изменяет его циклический характер, поскольку и в данном случае действует механизм обратной связи в контуре управления.

Функционирующей системой необходимо управлять, т.е. регулировать ее работу таким образом, чтобы параметры системы приближались к намеченным. Физические системы по природе своей таковы, что их можно изучать в стационарных, установившихся условиях. При рассмотрении же человеческих организаций и социальных систем такой подход неприемлем. В этом случае мы стремимся достигнуть конечную цель и ищем возможность саморегулирования системы, что зависит от характеристик компонентов системы и их взаимосвязей.

Метод управления, основанный на использовании обратной связи, нашел широкое применение как в системах управления техническими объектами, так и в организационно-административных системах. Одним из главных достоинств этого метода является работа элементов систем управления с учётом поступающей информации в достижении упорядоченности, а также обратных связей и роли управления в обеспечении устойчивости системы. Система управления с контуром обратной связи имеет возможность адекватно реагировать на возникающие изменения.

Создание контура обратной связи возможно осуществить в условиях предварительного планирования процесса, выстраивания его алгоритма, прогнозирования его результатов, эффективного управления им и при наличии средств управления — измерительной аппаратуры, позволяющей

вести мониторинг за ходом процесса и при необходимости менять отдельные параметры.

На примере системы построения контура обратной связи «KSU Feedback» в процессе выполнения дипломной работы был показан процесс разработки внешнего программного интерфейса. Описаны технологии, которые были применены при разработке. Также был проведён анализ недостатков и преимуществ описанных способов реализации API, обоснован выбор технологии SOAP.

Изученные технологии разработки веб-сервисов позволили провести процесс разработки эффективнее за счёт применения дополнительных плагинов.

Обратная связь должна позволять получателю проанализировать, что должно быть сделано и что можно сделать иначе, вместо того, чтобы фокусироваться на том, что было сделано плохо. Система «KSU Feedback»как средство построения контура обратной связи даёт возможность получить обобщённые статистические данные и использовать их для формирования отчётов.

Пока обратная связь предоставляет возможность анонимно отреагировать на предмет формирования отзыва и предоставляет сбалансированную информацию с достаточной детализацией, она может быть эффективным инструментом для создания положительной рабочей среды, в которой каждый четко понимает свою роль, владеет инструментами постановки целей и продолжает обучаться, что приводит к улучшению качества работы.

СПИСОК ЛИТЕРАТУРЫ

1. Арсеньев Ю.Н.Принятие решений. Интеллектуальные системы / Ю.Н. Арсеньев, С.И.Шеболаев, Т.Ю. Давыдова.– М.: Юнити-Дана, 2003 – 256с.
2. Беляев А.А. Системология организации – А.А.Беляев, Э.М. Коротков. – М.: ИНФРАМ, 2000 – 342с.
3. Спиваковский А.В. Архитектура и функциональность программного комплекса «KSUFeedback» / А.В. Спиваковский, Д.А. Березовский, С.А. Титенок// Информационные технологии в образовании. – 2010. – 53с.
4. Вертакова Ю.В, Козьева И.А., Кузьбожев Э.Н. Управленческие решения: разработка и выбор / Ю.В.Вертакова, И.А.Козьева, Э.Н. Кузьбожев– М.: КНОРУС, 2005 – 351с.
5. Литвак Б.Г. Экспертные технологии в управлении. – М.: Дело, 2004. – 263с.
6. Платов В.Я. Современные управленческие технологии. – М.: Дело, 2006. – 168с.
7. Bloch, J. Effective Java Programming Language Guide,Addison-Wesley, Boston, MA, 2001.– 462p.
8. Clarke, S. “API Usability and the Cognitive Dimensions Framework”, 2003. – 217p.
9. Ellis, B., Stylos, J. and Myers, B. The Factory Pattern in API Design: A Usability Evaluation. International Conference on Software Engineering, 2007.
10. James Snell Programming Web Services with SOAP / James Snell, Doug Tidwell, PavelKulchenko. – Publisher: O'Reilly MediaReleased: December 2001Pages: 264.
11. Mark MasseREST API Design RulebookDesigning Consistent RESTful Web Service Interfaces / Publisher: O'Reilly Media, October 2011. – 116p.

12. Robert EnglanderJava and SOAP / Publisher: O'Reilly Media, May 2002. – 278p.
13. Jim Webber REST in PracticeHypermedia and Systems Architecture / Jim Webber, SavasParastatidis, Ian Robinson / Publisher: O'Reilly Media, September 2010. – 448p.
14. Stephen T. The Art of Software ArchitectureDesign Methods and Techniques / Publisher: WileyReleased, April 2003. – 336p.
15. James McGovernJava Web Services Architecture / James McGovern, Sameer Tyagi, Michael Stevens, Sunil Mathew / Publisher: Elsevier, May 2003. – 832p.
16. Java Enterprise Best Practices. Publisher: O'Reilly Media, December 2002. – 292p.
17. Kevin MakiceTwitter API: Up and RunningLearn How to Build Applications with the Twitter API / Publisher: O'Reilly Media, March 2009. – 416p.
18. ГенриБекетJava SOAP дляпрофессионаловProfessional Java SOAP Programming. – Издательство: Лори, 2004. – 458с.
19. Eric van der VlistXML SchemaThe W3C's Object-Oriented Descriptions for XML. – Publisher: O'Reilly Media,June 2002. – 400p.
20. Ethan CeramiWeb Services EssentialsDistributed Applications with XML-RPC, SOAP, UDDI & WSDL. – Publisher: O'Reilly Media, February 2002. – 308p.
21. Брайан ТрэвисНазвание: XML и SOAP программирование для серверов. – М.: Русская Редакция, 2001. – 216с.
22. Ken Pugh Interface Oriented DesignWith Patterns. / Publisher: Pragmatic Bookshelf, July 2006. – 240p.
23. Debbie StoneUser Interface Design and Evaluation / Debbie Stone, Debbie Stone, Caroline Jarrett, Mark Woodroffe, ShaileyMinocha. – Publisher: Elsevier / Morgan Kaufmann, April 2005. – 704p.

24. Eric Elliott Programming JavaScript ApplicationsRobust Web Architecture With Node, HTML5, and Modern JS Libraries. / Publisher: O'Reilly Media, February 2013. – 300p.
25. Мухин В.И. Основы теории управления / В.И. Мухин. – М.:Экзамен, 2002. – 256с.
26. SOAP – это просто [Электронный ресурс] – 2003. – Режим доступа: http://www.realcoding.net/articles/soap-eto-prosto.html
27. SOAP и REST[Электронный ресурс]/ Л. Черняк. – 2003. – Режим доступа: http://www.osp.ru/os/2003/09/183376/
28. SOAP UI [Электронный ресурс]– Режим доступа: http://www.soapui.org/Service-Mocking/creating-dynamic-mockservices.html
29. SOAPtesting[Электронный ресурс]– Режим доступа: http://automated-testing.info
30. Правильно написание API сервиса [Электронный ресурс]– Режим доступа:http://habrahabr.ru/qa/21031/
31. Обзор XML-стандартов, часть 1.Базовые XML-стандарты - основа основ[Электронный ресурс]– Режим доступа:http://citforum.ru/internet/xml/standarts/part1.shtml
32. Интеграция приложний в WWW[Электронный ресурс]– Режим доступа:http://www.4stud.info/networking/web-services.html
33. API[Электронный ресурс]– Режим доступа: http://prog-school.ru/products/webapi/
34. Jaxb2:schemagen[Электронный ресурс]– Режим доступа: http://mojo.codehaus.org/jaxb2-maven-plugin/schemagen-mojo.html
35. A-Simple-JavaCC-Parser [Электронный ресурс]– Режим доступа: https://github.com/AaronKalair/A-Simple-JavaCCParser/blob/master/Parser.jj
36. JAXB [Электронный ресурс]– Режим доступа: http://www.vogella.com/articles/JAXB/article.html
37. Learn API testing [Электронный ресурс]– Режим доступа:

http://www.guru99.com/api-testing.html

38. http://programmers.stackexchange.com/questions/187683/a-web-application-as-a-rest-api-client-how-to-handle-resource-identifiers
39. A Web application as a REST API client [Электронныйресурс] – Режимдоступа: http://supervisord.org/api.html
40. JavaCC [Электронный ресурс] – Режим доступа: http://www.engr.mun.ca/~theo/JavaCC-Tutorial/
41. Javacc-maven-plugin[Электронный ресурс] – Режим доступа: http://mojo.codehaus.org/javacc-maven-plugin/
42. ConfluenceXML-RPCandSOAPAPIsplugin[Электронный ресурс] – Режим доступа: https://developer.atlassian.com/display/CONFDEV/
43. SOAP-vs-REST [Электронный ресурс] – Режим доступа: http://www.alliancetek.com/Blog/post/2012/10/12/SOAP-vs-REST.aspx
44. Webservices[Электронный ресурс] – Режим доступа: https://www.e-education.psu.edu/geog583/files/geog583/keynote_chappell.pdf
45. Definision of SOAP and REST[Электронный ресурс] – Режим доступа: http://spf13.com/post/soap-vs-rest
46. Web Services, Part 1: SOAP vs. REST [Электронныйресурс] – Режимдоступа:http://www.ajaxonomy.com/2008/xml/web-services-part-1-soap-vs-rest
47. Википедия [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/
48. Прикладной программный интерфейс[Электронный ресурс] – Режим доступа: http://www.osp.ru/cw/2000/09/3539/
49. Usingthe API[Электронный ресурс] – Режим доступа: https://developers.google.com/books/docs/v1/using
50. OpenLibraryBooks API[Электронный ресурс] – Режим доступа: http://openlibrary.org/dev/docs/api/books
51. DeveloperTools[Электронный ресурс] – Режим доступа:

https://developers.google.com/gadgets/docs/tools

52. REST API Development [Электронный ресурс] – Режим доступа: https://developer.atlassian.com/display/DOCS/REST+API+Development
53. Web services. API. [Электронный ресурс] – Режим доступа:http://research.cs.wisc.edu/htcondor/manual/v7.6/4_5Application_Program.html

ПРИЛОЖЕНИЯ

Приложение А

Рис. 1.1. Иерархия вовлеченности сотрудников Джона Флеминга и Джима Асплунда.

Приложение Б

Рис 1.1 Схема системы управления с наличием обратной связи

Приложение В

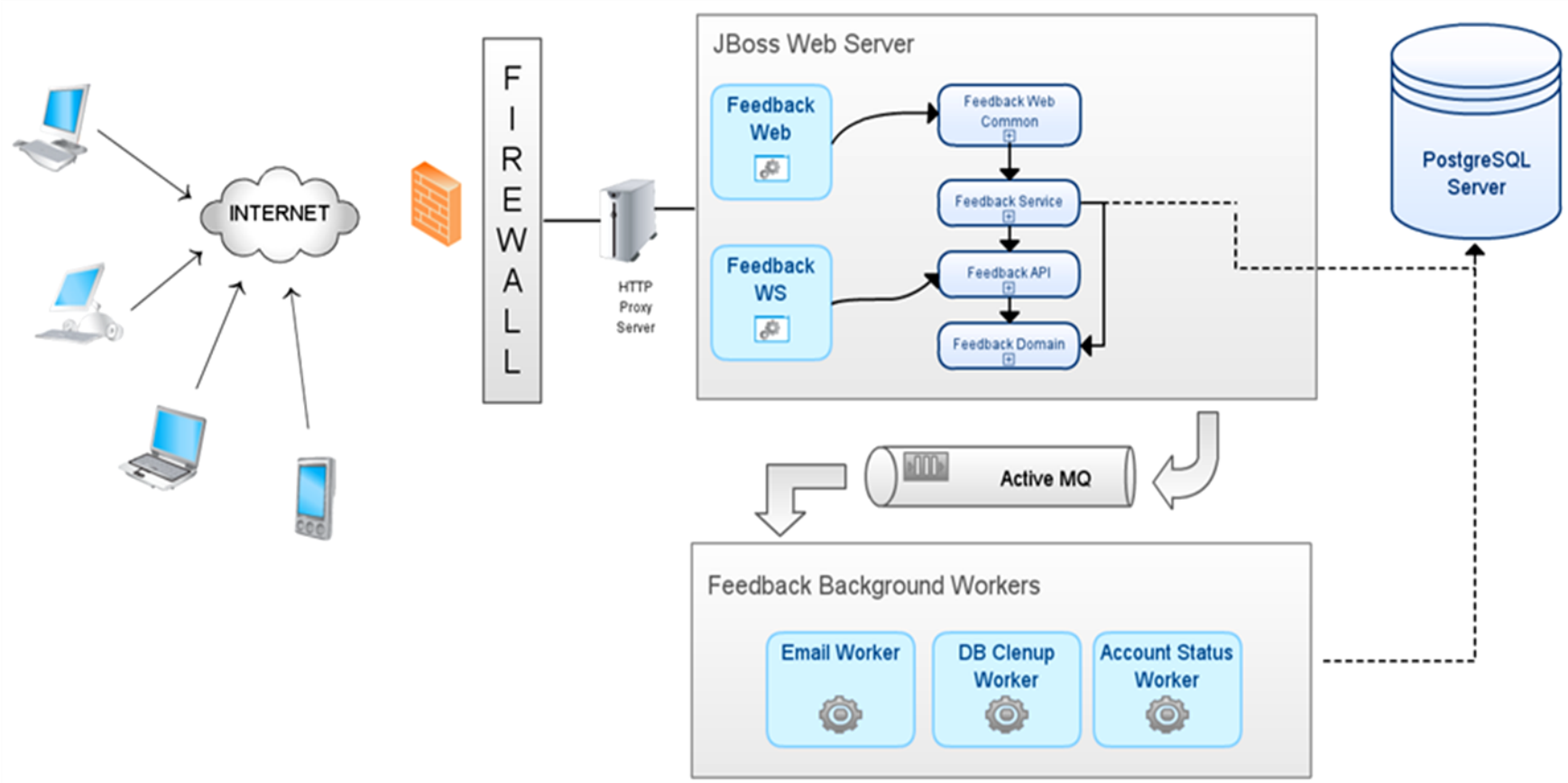

Рис 1.3 Архитектура программного комплекса «KSUFeedback»

Приложение Г

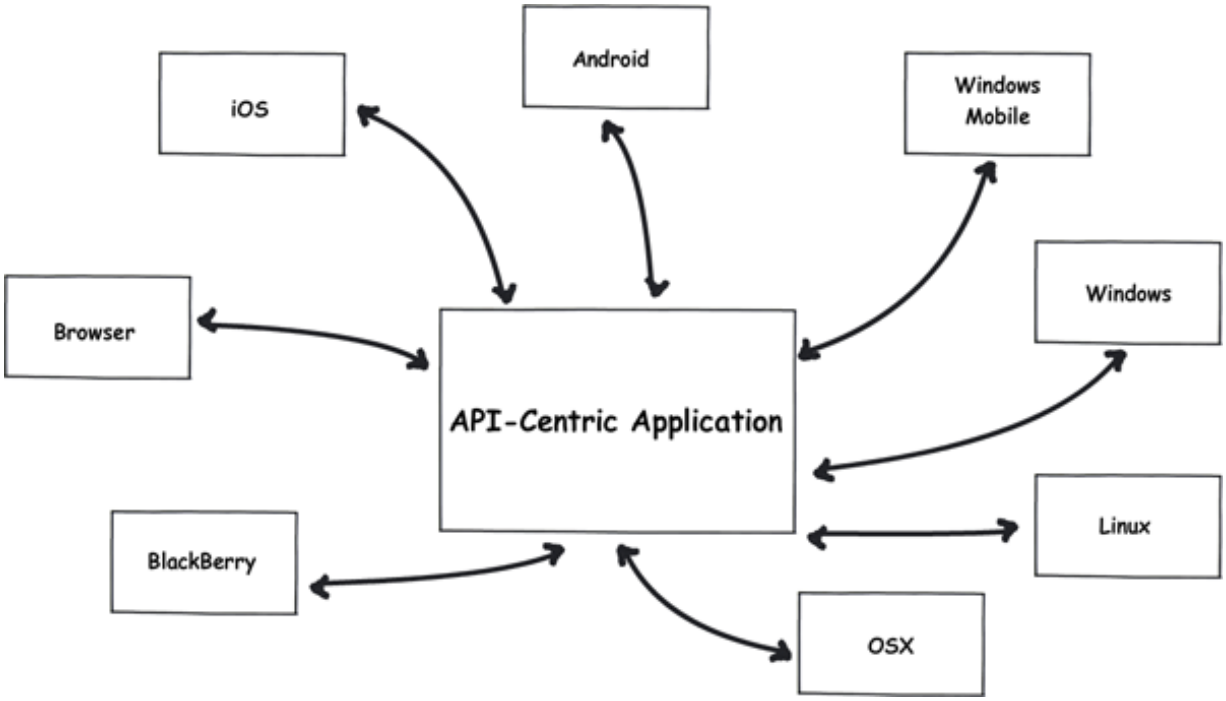

Рис. 2.1.API-ориентированное веб-приложение

Приложение Д

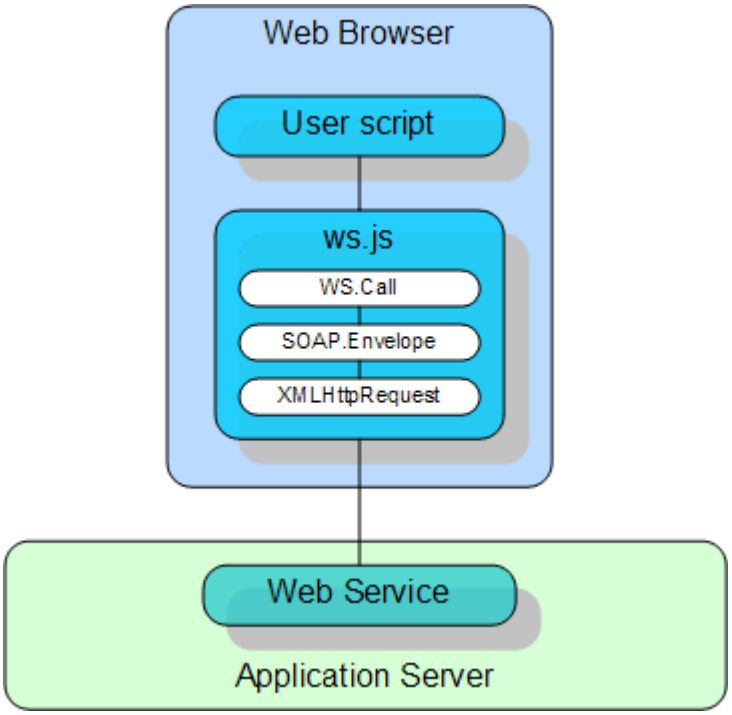

Рис. 2.1 Схема работы веб-сервиса