High Level Architecture

COMPETITION

**Программно-аппаратный комплекс Competition**

**конкурентного анализа сегмента рынка**

Версия 1.1.

(Modified 25 December 2012)

Содержание

[1. Введение 3](#_Toc344292762)

[1.1 Цель документа 3](#_Toc344292763)

[1.2 Ссылки 3](#_Toc344292764)

[1.3 Определения и сокращения 3](#_Toc344292765)

[1.4 Краткий обзор 3](#_Toc344292766)

[2. Внутренняя структура приложения 5](#_Toc344292767)

[2.1 Варианты использования (Use cases) 5](#_Toc344292768)

[2.2 Модульная декомпозиция (Subsystem Decomposition) 8](#_Toc344292769)

[2.3 Функциональная декомпозиция 9](#_Toc344292770)

[3. Диаграммы классов (Class Diagrams) 10](#_Toc344292772)

[4. Программно-аппаратный комплекс 13](#_Toc344292773)

[4.1 Функциональная модель конкурентного анализа 13](#_Toc344292774)

[4.2 Описание интерфейсов пользователя 15](#_Toc344292775)

[4.3 Дополнение к диаграмме классов 16](#_Toc344292776)

[4.4 Диаграмма последовательности 17](#_Toc344292777)

[4.5 Высокоуровневая архитектура программно-аппаратного комплекса 18](#_Toc344292778)

# Введение

## Цель документа

Документ описывает архитектурные решения, принятые при разработке программно-аппаратного комплекса Competition.

Программно-аппаратный комплекс Competition предназначен для анализа конкуренции высокотехнологичного сектора экономики (на примере сегмента отечественной авиационной техники), поиска путей повышения конкурентоспособности предприятий за счет применения математических моделей и алгоритмов.

## Ссылки

1. Демо-прототип (<http://competitive-analysis.ru/>)

## Определения и сокращения

Этот раздел содержит список терминов и сокращений, используемых в документе, с их кратким описанием.

**Программно-аппаратный комплекс** - набор технических и программных средств, работающих совместно для выполнения одной или нескольких сходных задач.

**Framework -** программный каркас. Отличается от библиотеки (library) тем, что выполняет код написанный для него, а не исполняется сам.

**Домен -** это семантическое понятие. Домен можно рассматривать как подмножество значений некоторого типа данных имеющих определенный смысл.

**GUI -** графический интерфейс пользователя.

**UML** - Язык UML представляет собой общецелевой язык визуального моделирования, который разработан для спецификации, визуализации, проектирования и документирования компонентов программного обеспечения, бизнес-процессов и других систем.

**Use Case** - диаграмма которая описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования

**СУБД (Система управления базами данных)** - совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

**БД (База данных)** - представленная в объективной форме совокупность самостоятельных материалов (статей, расчётов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины (ЭВМ).

## Краткий обзор

Competition – инструмент, автоматизирующий работу специалиста в области конкурентного анализа.

С приложением работает пять групп экторов, делящихся на внутренних, в роли которых выступают сотрудники, поддерживающие разрабатываемую систему и внешних пользователей и специалистов.

**Внутренние:**

1. **Пользователь**. В роли Пользователя, как правило, выступает эксперт в какой-либо области или сегменте рынка. До тех пор, пока он не зарегистрируется в системе и не приобретет статус «Зарегистрированный пользователь», он обладает крайне ограниченным спектром возможностей. По сути, все их возможные действия ограничиваются генерацией тестовой модели и ряда простейших операций над ней. Тестовая модель создается в ходе проектирования демонстрационного прототипа и представляет собой сегмент рынка авиапромышленности. Пользователь может ознакомиться с ней, изменить графическое расположение сил и агентов (без возможности редактировать связи между ними, а также их параметры), повернуть ее, масштабировать. Масштабирование модели является одной из наиболее важных графических составляющих модели, так как в случае расчета крупного сегмента рынка мы можем иметь дело с большой иерархической вложенностью (фрактализацией), большим количеством сил, агентов и связей между ними. В таком случае возможность приблизить и отдалить какой-либо сегмент модели является крайне необходимой. Также Пользователь может пройти процедуру авторизации или выхода из системы.
2. **Зарегистрированный пользователь**. Зарегистрированный пользователь, по сути, является тем же самым Пользователем, описанным выше, однако имеет расширенный спектр возможностей, таких как: генерация собственной модели из исходного набора агентов и параметров, изменение основного сегмента модели, ее фрактализация, возможность редактировать исходные параметры агентов (внесение собственных данных в модель), а также ряд функций по работе с моделью (выгрузка модели в файл, печать результатов конкурентного анализа, сохранение и загрузка модели из БД). При этом роль Зарегистрированного пользователя содержит в себе все возможности Пользователя.

**Внешние:**

1. **Специалист**. В роли Специалиста выступает эксперт в определенной области (или компания-поставщик/производитель). Он является основным накопителем и поставщиком информации, вносимой в систему, которая в дальнейшем проверяется модератором. Если в роли Специалиста оказывается компания-заказчик моделирования, то его действия связаны с добавлением этой компании, ее основных конкурентов, поставщиков (если они не были внесены ранее) и т.д. в разработанную систему, с одной единственной целью – в дальнейшем самому же воспользоваться указанными данными для конкурентного анализа в интересующем его сегменте рынка. Специалист обладает так же и более широкими возможностями по добавлению страны/рынка или сегмента рынка в том случае, если ранее данная информация внесена не была.
2. **Модератор**. Его действия связаны с проверкой достоверности информации, вносимой специалистом. Если информация оказывается подлинной (Специалистом предоставлены соответствующие документы, которые подтверждают наличие того или иного агента, или же подлинность установлена самим модератором), то она вносится в систему (например, добавляется новый Основной Игрок рынка); в противном случае добавление информации отклоняется, а Специалист получает письмо с причинами отказа о внесении. Так же, Модератор следит за актуальностью информации об агентах и принимает соответствующие решения и действия связанные либо с редактированием агента (если о нем изменилась информация, скажем, он перестал быть Основным Игроком на рынке), либо с его удалением из системы (в случае, если, скажем, какая-то компания обанкротилась).
3. **Администратор**. Основная роль Администратора – добавление (назначение) Модераторов, а также контроль их деятельности. Помимо этого Администратор рассматривает и обрабатывает ряд запросов от Специалиста, например, запрос на добавление неучтенного ранее параметра какому-либо агенту, или добавление ранее отсутствующего в системе сегмента рынка или же исполняет их по собственной инициативе.

Роль Администратора содержит в себе все возможности Модератора и Специалиста. Роль Модератора содержит в себе все возможности Специалиста.

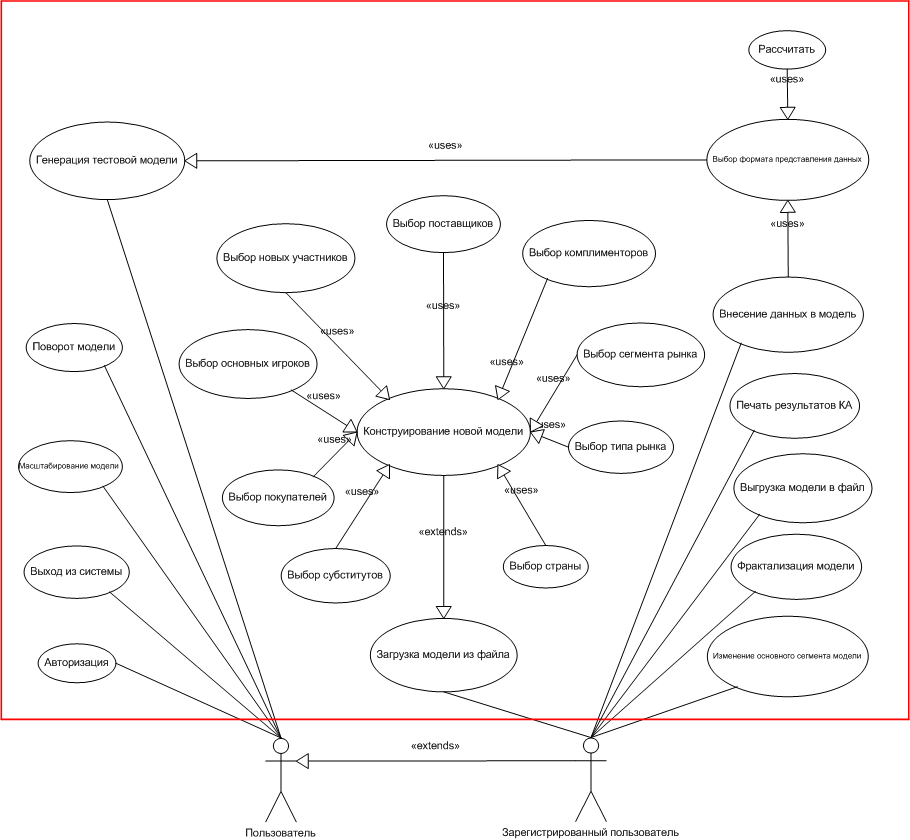
В качестве языка программирования для реализации проекта был выбран скриптовый объектно-ориентированный язык программирования PHP. С целью ускорения хода разработки использовался PHP-Framework CakePHP. Перечислим причины, по которым был выбран именно этот Framework:

* + - MVC архитектура;
    - Возможность бесплатного использования и открытый код ядра;
    - Гибкая система кэширования;
    - Генерация программного кода в соответствии со схемой БД;
    - Простое внедрение AJAX-технологии;
    - Активное сообщество и служба online-поддержки;
    - Возможность локализации;
    - Отлаженные механизмы, отвечающие за безопасность и сохранение данных.

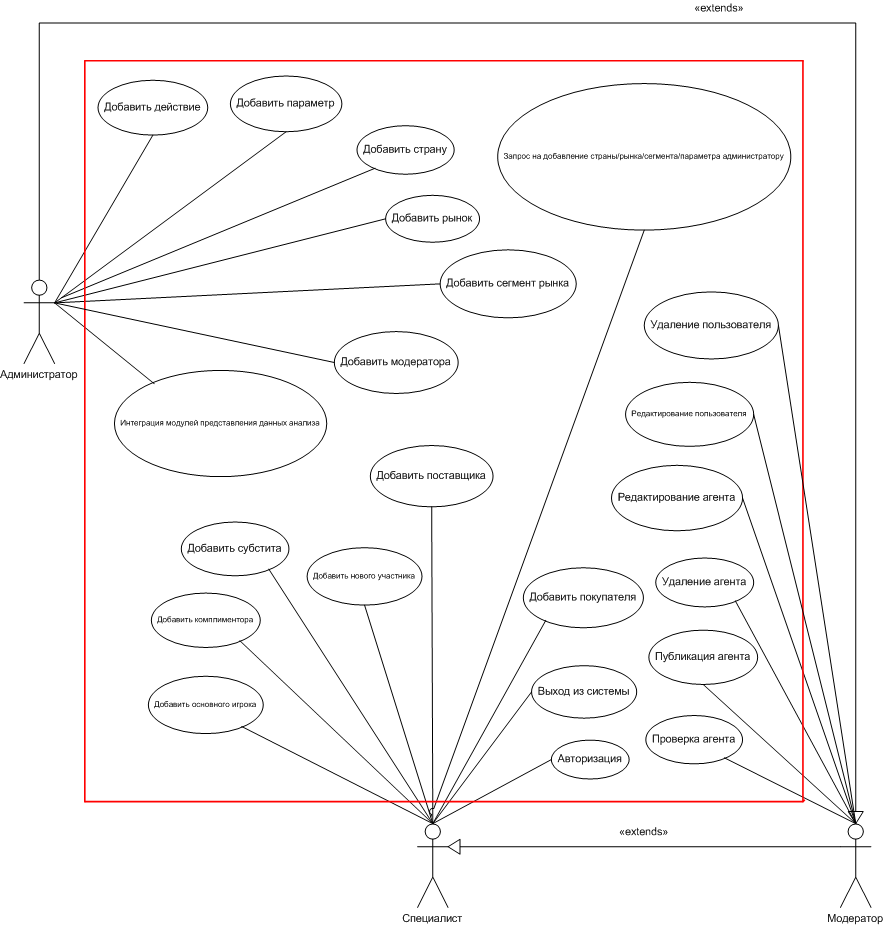
# Внутренняя структура приложения

## Варианты использования (Use cases)

Перейдем к подробному рассмотрению взаимодействия нашей системы с экторами, опираясь на диаграммы прецедентов внешней (рис. 1) и внутренней (рис. 2) её частей:



*Рис. 1. Диаграмма вариантов использования внешней части системы*

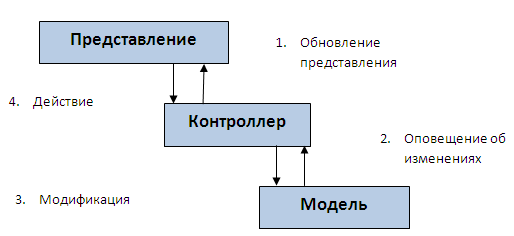


*Рис. 2. Диаграмма вариантов использования внутренней части системы*

На основании представленных диаграмм были разработаны спецификации внешних требований к проектируемой системе. На следующем шаге были построены функциональная и модульная модели проектируемого приложения.

## Модульная декомпозиция (Subsystem Decomposition)

MVC (Model View Controller/Модель - Вид (Представление) - Контроллер) – это проверенная и надежная схема, дающая возможность быстро разработать модульные и расширяемые приложения. Разбивка приложения на отдельные блоки (виды, модели и контроллеры) позволяет вести разработку прямо «на ходу». Новые возможности легко могут быть добавлены, а старые изменены. Модульная структура приложения позволяет разработчику и дизайнеру работать одновременно. Кроме того, существует возможность быстрого прототипирования. Разделение на модули позволяет изменять часть программы, не затрагивая остальные модули. Стандартная схема архитектуры «Модель-Вид-Контроллер» изображена на рис. 3:

****

*Рис. 3. Схема MVC*

Рассмотрим данную схему подробнее. Шаблон MVC содержит в себе три основных модуля:

* *Модель* – отвечает за предоставление данных и методов работы с ними, а также реагирует на запросы, изменяя свое состояние. Модель отображает внутреннее устройство системы.
* *Вид (Представление)* **–** отвечает за отображение информации, поступающей из системы или в систему.
* *Контроллер –* обеспечивает связь между пользователем и системой. Он получает данные от пользователя (через Вид) и передает их в Модель, и наоборот, получает сообщения от Модели и передает их в Вид.

Еще один вариант схемы MVC предлагают разработчики CakePHP (рис. 4).

****

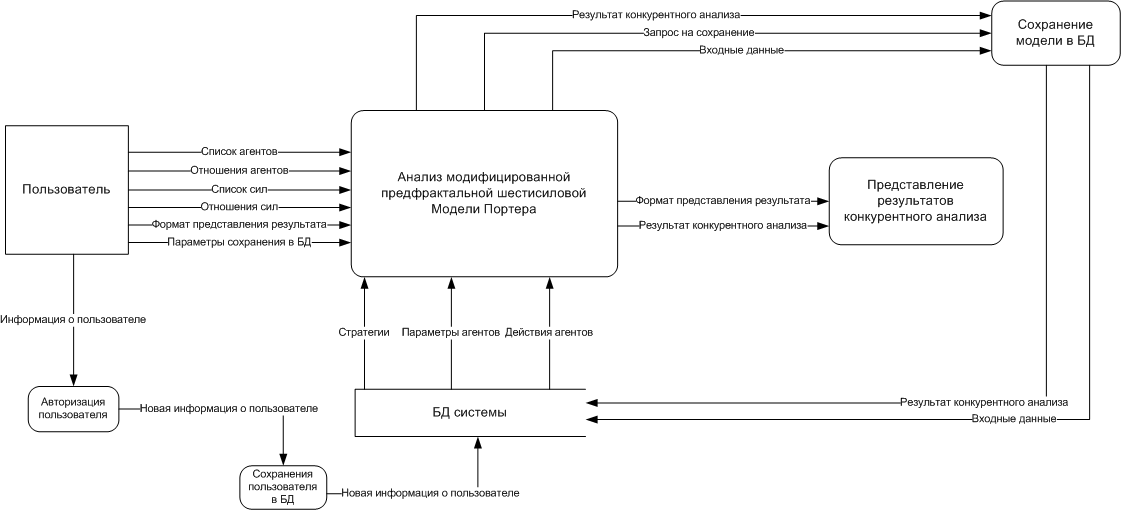
*Рис. 4. Схема MVC на примере фрэймворка CakePHP*

На этой схеме отображен процесс взаимодействия клиента с системой, состоящий из 6 последовательных действий:

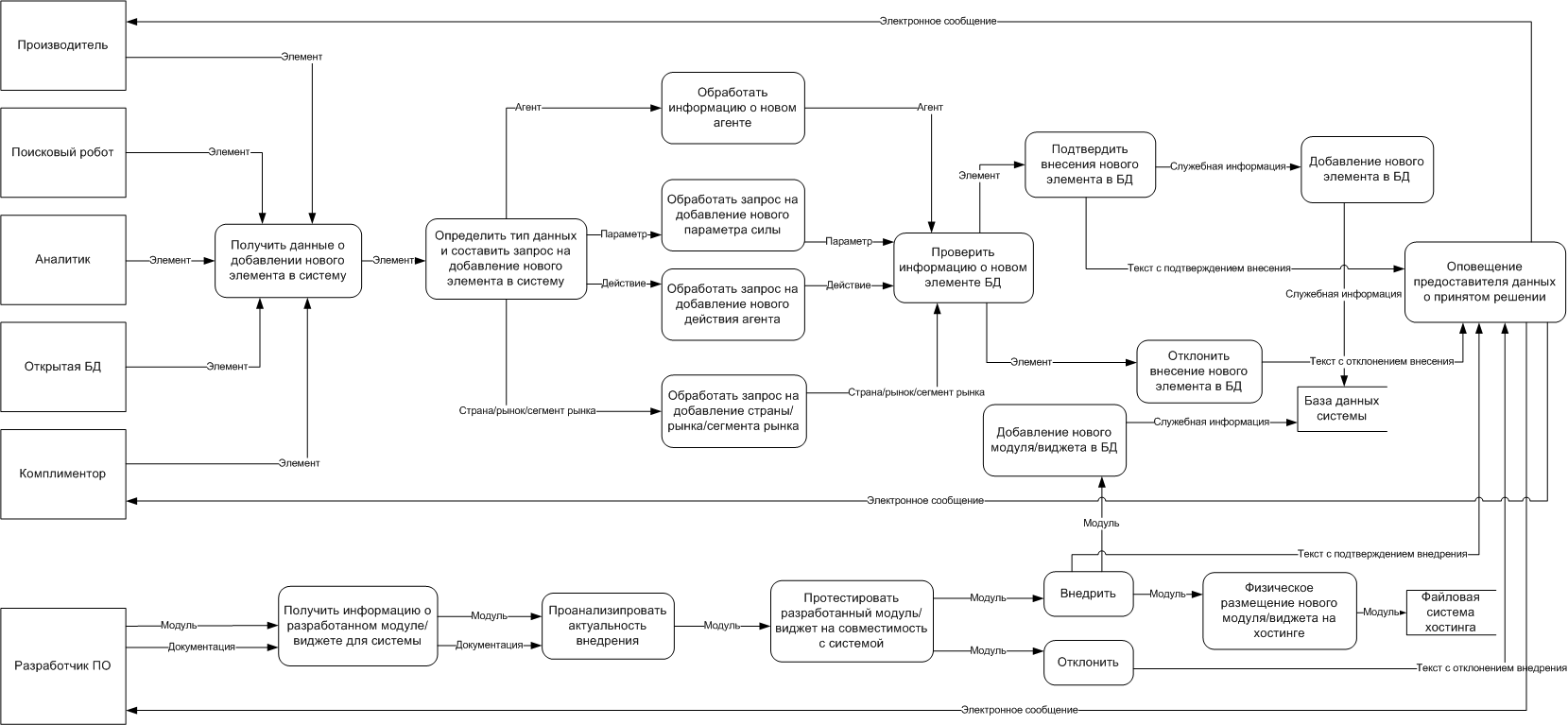
1. Клиент активирует ссылку, после чего браузер делает запрос к серверу.
2. Диспетчер проверяет URL запроса и перенаправляет запрос соответствующему контроллеру для дальнейшей обработки.
3. Контроллер действует согласно своей логике и обращается к модели (как правило к таблице БД, для извлечения, добавления, удаления или проверки информации)
4. Модель отвечает на запрос контроллера, обмениваясь с ним данными.
5. После обработки данных контроллер передает их Виду. Вид, получая данные, готовит их к отображению браузером.
6. Используя данные, полученные от контроллера, Вид передает клиенту полностью отформатированный ответ.

## Функциональная декомпозиция

## В данном разделе мы воспользуемся методологией графического структурного анализа для описания внешних по отношению к системе источников и адресатов данных, логических функций, потоков данных и хранилищ данных, к которым осуществляется доступ. Построение диаграмм потоков данных существенно поможет в проектировании программно-аппаратного комплекса, а также наглядно проиллюстрирует внешнее (рис. 5) и внутреннее (рис. 6) устройство информационной системы. Начнем с рассмотрения диаграммы внешнего устройства потоков данных web-системы. Проведем функциональную декомпозицию на примере диаграмм потоков данных.



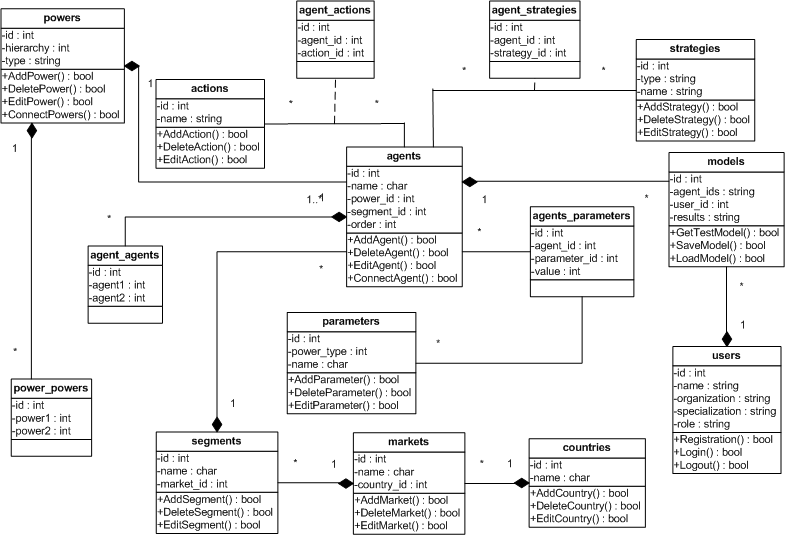
*Рис. 5. DFD-диаграмма: внешнее устройство потоков данных системы*



*Рис. 6. DFD-диаграмма: внутреннее устройство потоков данных системы*

# Диаграммы классов (Class Diagrams)

Диаграмма классов (class diagram) служит для представления структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы. При помощи диаграммы классов (рис. 7) была построена схема базы данных системы (рис. 8).



*Рис. 7. Диаграмма классов*

1. **Класс powers.** Содержит атрибуты hierarchy – степень фрактализации и type – тип силы и добавляемых агентов (напр., «Новые игроки», «Поставщики» и т.д.), а также стандартные операторы, доступные эктору «Пользователь» – добавить\_силу(), удалить\_силу(), отредактировать\_силу(), связать\_силы(). Одной силе соответствует несколько агентов.
2. **Класс powers\_powers.** Вспомогательный класс для powers. Содержит атрибуты power1 и power2, обозначающие связь между силой1 и силой2.
3. **Класс agents.** Основной класс проектируемого программно-аппаратного комплекса. Так или иначе, с ним связаны все остальные классы системы. Содержит атрибуты name – имя объекта, order – порядковый номер агента в силе, а также стандартные операторы, доступные эктору «Модератор» – добавить\_агента(), удалить\_агента (), отредактировать\_агента() и «Пользователь» – связать\_агентов(). Одному агенту соответствует несколько действий, параметров и связей, а также единственная стратегия и единственный сегмент рынка.
4. **Класс actions.** Содержит атрибут name – название действия, по которому вызывается соответствующая функция, участвующая в получении результата конкурентного анализа сегмента рынка, а также стандартные операторы, доступные эктору «Администратор» – добавить\_действие (), удалить\_действие(), отредактировать\_действие().
5. **Класс agent\_actions.** Вспомогательный класс для agents и actions. Содержит атрибуты agent\_id и action\_id, обозначающие привязку нескольких возможных действий к одному агенту.
6. **Класс agent\_agents.** Вспомогательный класс для agents. Содержит атрибуты agent1 agent2, обозначающие связь между агентом1 и агентом2.
7. **Класс strategies.** Содержит атрибуты type – тип стратегии и name – имя стратегии (например, стратегия «Лидер»), а также стандартные операторы, доступные эктору «Администратор», а именно: добваить\_стратегию(), удалить\_стратегию(), отредактировать\_стратегию(). Выбранная агентом стратегия, определяет его поведение (действия) на рынке. В каждый определенный момент времени одному агенту может соответствовать только одна стратегия.
8. **Класс agent\_strategies.** Вспомогательный класс для agents и strategies. Содержит атрибуты agent\_id и strategy\_id, обозначающие привязку стратегии к агенту.
9. **Класс parameters.** Содержит атрибуты power\_type – тип силы (поскольку тип агента зависит от типа силы, в которую он помещен) и name – название параметра, а также стандартные операторы, доступные эктору «Администратор» – добавить\_параметр(), удалить\_параметр(), отредактировать\_параметр().
10. **Класс agent\_parameters.** Вспомогательный класс для agents и parameters. Содержит атрибуты agent\_id и parameter\_id, обозначающие привязку параметра к агенту, а также значение параметра – value.
11. **Класс segments.** Класс, помогающий осуществлять фильтрацию агентов по определенному сегменту рынка (в нашем случае это БПЛА). Содержит атрибут name – название сегмента рынка, а также операторы, доступные эктору «Администратор», а именно: добваить\_сегмент(), удалить\_сегмент(), отредактировать\_сегмент().Связан с классом markets. Одному сегменту соответствует несколько агентов.
12. **Класс markets.** Класс, помогающий осуществлять фильтрацию агентов по определенному типу рынка (в нашем случае это Авиастроение). Одному рынку может соответствовать несколько сегментов рынка. Содержит атрибут name – название типа рынка, а также операторы, доступные эктору «Администратор», а именно: добваить\_рынок(), удалить\_рынок(), отредактировать\_рынок ().Связан с классом countries.
13. **Класс countries.** Класс, помогающий осуществлять фильтрацию агентов по стране (в нашем случае это Российская Федерация). Одной стране может соответствовать несколько типов рынка. Содержит атрибут name – название типа рынка, а также операторы, доступные эктору «Администратор», а именно: добваить\_страну(), удалить\_страну(), отредактировать\_страну ().

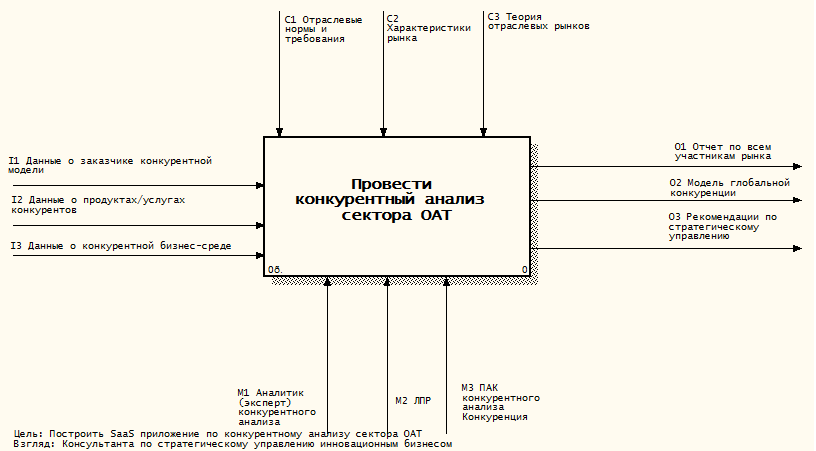


*Рис. 8. Схема базы данных системы*

# Программно-аппаратный комплекс

## Функциональная модель конкурентного анализа

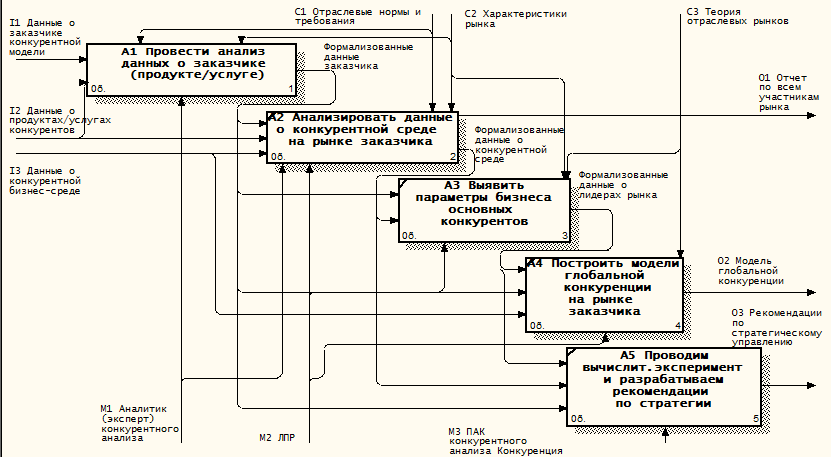
В данном разделе речь идет о разработке функциональной модели конкурентного анализа для описания бизнес-процессов использования программно-аппаратного комплекса топ-менеджерами предприятий - производителей объектов боевой авиации. Для этого была выбрана методология IDEF0, позволяющая сделать акцент на взаимосвязи процессов. Бизнес-процесс представляется в детализированном виде, позволяющем эффективно взаимодействовать системным аналитикам, программным архитекторам, специалистам в предметной области и другим занятым в проекте лицам. На верхнем уровне модели (рис. 9, диаграмма A0) определяются основные входы (Input, I1-I3) и выходы (Output, O1-O3), управляющие воздействия (Control, C1-C3) и исполнители (M1-M3).



*Рис. 9. Функциональная модель конкурентного анализа сектора ОАТ в виде IDEF0-диаграммы (A-0)*

Декомпозиция диаграммы A0 отображает основные этапы конкурентного анализа (рис. 10) в интересах заказчика – ЛПР от производителя ЛА:

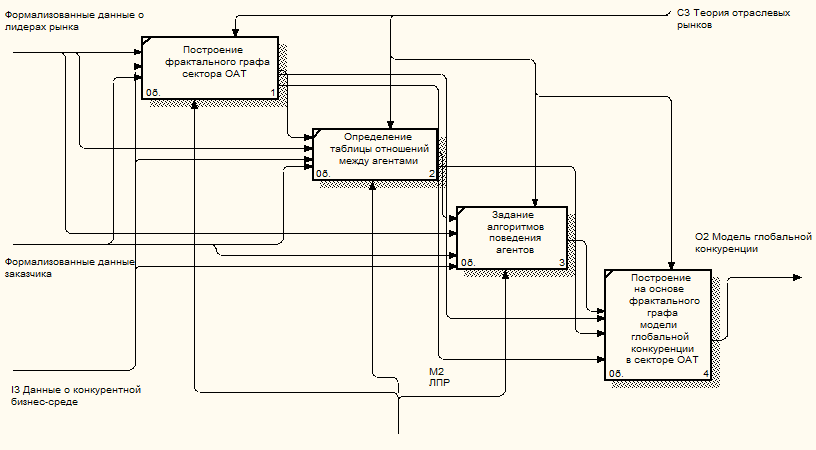
* проведение анализа данных о заказчике и его продукте;
* анализ данных конкурентной среды рынка или сегмента рынка заказчика;
* выявление параметров бизнеса основных конкурентов для использования в агентной модели;
* построение модели глобальной конкуренции на рынке заказчика;
* проведение вычислительного эксперимента с целью выработки рекомендаций по формированию конкурентной стратегии заказчика.



*Рис. 10. Декомпозиция процесса конкурентного анализа (A0)*

Рассмотрим декомпозицию четвертого и основного из перечисленных выше этапов конкурентного анализа (A4). Для построения модели глобальной конкуренции на рынке заказчика (рис. 11) требуется:

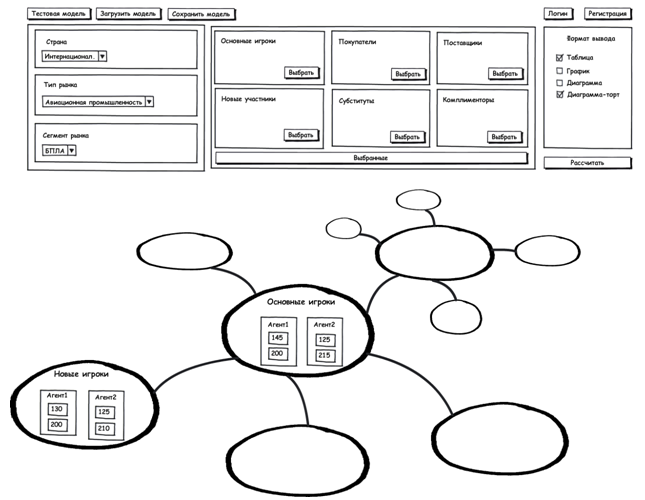
* построить фрактальный нагруженный граф для сегмента или рынка заказчика;
* определить все виды отношений между агентами и зафиксировать в виде таблицы;
* задать и детализировать алгоритмы поведения агентов рынка;
* построить на основании предыдущих этапов модель глобальной конкуренции.



*Рис. 11. Декомпозиция процесса построения модели глобальной конкуренции на рынке заказчика*

## Описание интерфейсов пользователя

Важным этапом проектирования и дальнейшей реализации web-системы является тщательно продуманная информационная архитектура, а также подготовка и составление макетов пользовательского интерфейса (GUI). На рис. 12 представлен основной макет интерфейса – главная страница с рабочей областью.

****

*Рис. 12. Основной макет интерфейса: главная страница с рабочей областью*

Рассмотрим макет подробнее, остановившись на каждом из спроектированных блоков. Верхнее меню для удобства разбито на две части:

* первая отвечает за работу с модифицированной моделью М. Портера («Тестовая модель», «Загрузить модель», «Сохранить модель»);
* вторая же отвечает за взаимодействием с пользователем («Логин», «Регистрация»; после входа в систему меняются на «Личный кабинет» и «Выход»).

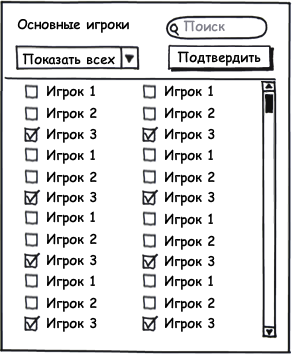
Большой блок настроек и фильтров, следующий сразу за основным меню, разбит на три части:

* В первой части происходит фильтрация агентов по местоположению, типу рынка и сегменту рынка. В случае выбора страны «Интернационал» (что соответствует международному конкурентному рынку) предлагается возможность выбора как всех стран, так и только некоторых из них. Данный блок позволяет пользователю сузить круг возможных агентов до отдельного сегмента рынка. Такая возможность является необходимой, потому как наша web-система должна будет располагать очень большим количеством возможных для выбора агентов и возможностями детализации и масштабирования.
* Во второй части происходит отбор агентов для проведения конкурентного анализа. При добавлении того или иного агента к силе, происходит автоматическая фильтрация по типу силы (например, если мы работаем с силой «Новые участники», то среди агентов также будут только «Новые участники»). К каждой силе можно привязать несколько разных агентов.
* В третьей части выбирается формат представления данных, полученных в результате конкурентного анализа.

Далее разворачивается сама модифицированная модель М. Портера, соответствующая выбранным силам и привязанным к ним агентам. Следующим этапом является связь между силами и агентами, после чего возможно проведение конкурентного анализа в заданном сегменте рынка и вывод результатов в выбранном формате.

На данном макете блок настроек и фильтров представлен в развернутом виде, однако при реализации дизайна средствами HTML, CSS и JavaScript он будет свернут в один единый блок, навигация по которому будет производиться с помощью перехода между вкладками, вынесенными в подменю.

На рис. 13 показано вспомогательное всплывающее окно, разворачивающееся по нажатию на кнопку «Выбрать».

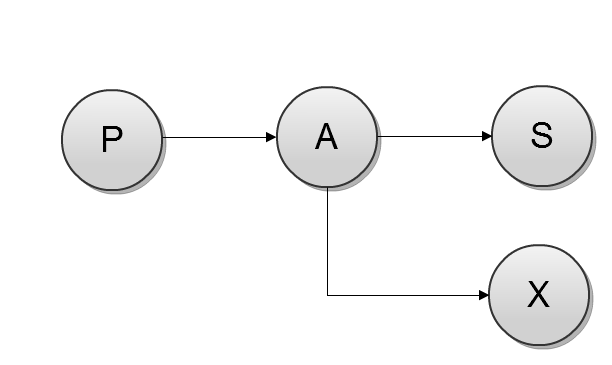


*Рис. 13. Окно выбора агентов*

В конкретном случае речь идет о вкладке «Основные игроки». Нашему рассмотрению представлен полный список позиций (в данном случае – игроков), предварительно внесенных в базу данных системы, с возможностью фильтрации по поисковой строке на основе полнотекстового поиска с использованием ajax-технологий. Также, помимо отображения всех позиций, предусмотрено отображение только отмеченных или, напротив еще *не* отмеченных позиций. Кнопка «Подтвердить» сохраняет выбранные позиции для последующего проведения конкурентного анализа.

## Дополнение к диаграмме классов

Подробнее остановимся на диаграмме классов, т.к. при её проектировании был разработан переход от математических моделей, описывающих взаимодействие элементов системы (агентов) к взаимодействию объектов различных классов в структуре программно-аппаратного комплекса.



*Рис. 14. Основные сущности при переходе от математических моделей к программному приложению*

Взаимодействие и порядок инициации основных сущностей изображен на рис. 14. В первую очередь задается набор элементов системы (силы в расширенной модели М. Портера): где *k* - тип силы (основные конкуренты, поставщики, покупатели и т.д.), *i -* уровень иерархии, и связи между ними (класс power\_powers (PP)). Затем в соответствие к каждому элементу систем (силе) создается агент , где индексы *k* и *i* соответствуют индексам силы, *j* – порядковый номер агента в множестве подобных. Далее для каждого агента задается набор влияний (класс agent\_agents (AA)). Классы PP и AA по сути представляют собой двумерные матрицы (рис. 15).

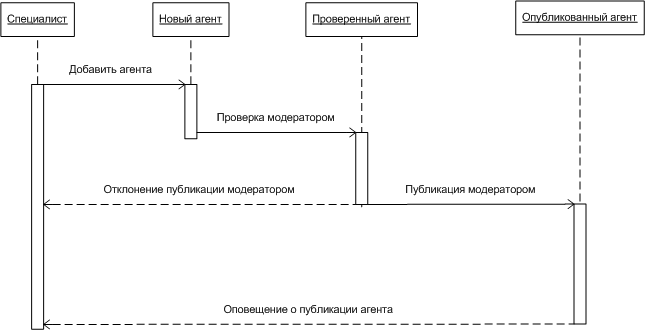
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  | - | 1 | 0 | … | 1 |
|  | 1 | - | 0 | … | 0 |
|  | 0 | 0 | - | … | 1 |
|  | … | … | … | - | … |
|  | 1 | 0 | 1 | … | - |

*Рис. 15. Матрица взаимных влияний агентов*

Набор стратегий задан строго – четырьмя отдельными строками в БД. Действия задаются по параметру name (рис. 7). При выгрузке данных из БД происходит сравнение по данному параметру, и вызывается соответствующая функция. Все связи задаются через вспомогательные таблицы. Каждому агенту с их помощью ставится в соответствие набор действий, а так же стратегий (рис. 7).

## Диаграмма последовательности

Перейдем к рассмотрению последовательности добавления нового агента в проектируемую систему. Для этого мы продемонстрируем процесс взаимодействия и линии жизни четырех объектов, показанных на рис. 16, на примере совершаемых ими действий.



*Рис. 16. Диаграмма последовательностей (добавление агента в систему)*

1. **Специалист.** Деятельность Специалиста не прерывается на протяжении всего времени выполнения действий прочими объектами. Именно он совершает первое действие в последовательности – добавляет агента в систему (информацию о новом агенте), после чего ожидает результата проверки достоверности объекта, тем самым порождая объект «Новый агент».
2. **Новый агент.** Данный объект может соответствовать одному из шести типов сил модифицированной модели М. Портера, а именно: *основные игроки, покупатели, поставщики, новые игроки, субституты* или *комплементоры.* Этот объект содержит в себе информацию о реальном игроке того или иного сегмента рынка (принадлежность к сегменту, тип, имя, набор параметров, официальные документы, подтверждающие его достоверность). Существование Нового агента состоит из трех этапов:

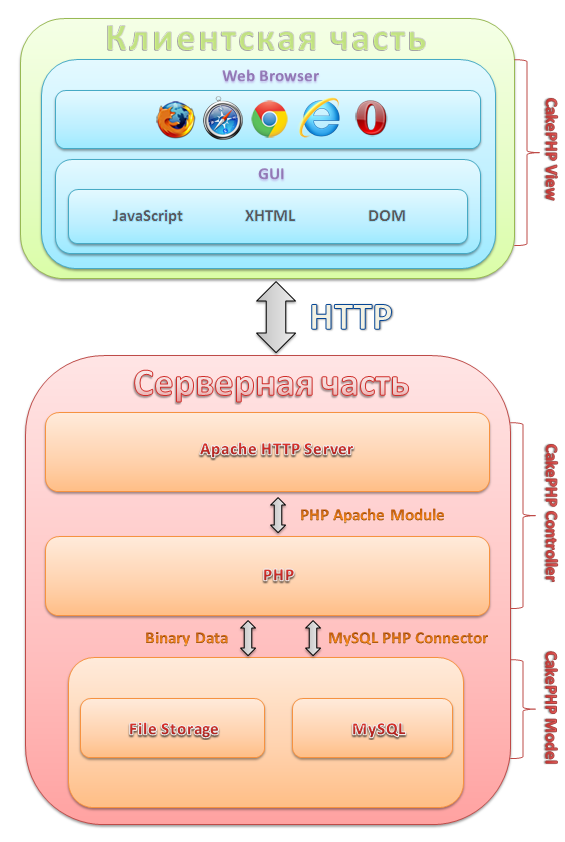
* формирование и добавление информации об агенте в систему;
* получение и обработка информации модератором;
* вынесение решения о публикации или отклонении публикации агента модератором.

Если принимается решение об отклонении публикации, то об этом оповещается Специалист, и процесс завершается. В противном случае, если принимается решение о публикации агента в систему (установлена его достоверность), создается новый объект «Проверенный агент», полностью соответствующий Новому агенту, но с дополнительной пометкой об успешно пройденной проверке.

1. **Проверенный агент.** Время существования Проверенного агента не велико и соответствует тому времени, которое требуется для внесения соответствующей информации в базу данных и сохранении приложений (сканы официальных документов, подтверждающих достоверность агента) в файловое хранилище. После публикации этого объекта модератором, появляется новый объект – «Опубликованный агент».
2. **Опубликованный агент.** Появление данного объекта говорит об успешном прохождении последовательности действий, в ходе которых в систему был добавлен и опубликован новый агент.

## Высокоуровневая архитектура программно-аппаратного комплекса

Высокоуровневая архитектура web-системы представлена на рис. 17.



*Рис. 17. Высокоуровневая архитектура программно-аппаратного комплекса*

Доступ к пользовательскому интерфейсу (клиент) будет осуществляться через окно браузера (Web Browser: MS Internet Explorer, Mozilla Firefox, Opera, Google Chrome, Safari). На наш взгляд, это оптимальное решение, т.к. оно требует от пользователя минимальных затрат, связанных с установкой и обслуживанием ПО. В первое время приложение планируется как бесплатный сервис, однако в дальнейшем возможно введение помесячной абонентской платы за использование. Собственно этим и исчерпываются возможные неудобства клиента. Реализация интерфейса будет организована стандартным образом при помощи XHTML-разметки (англ. Extensible Hypertext Markup Language — расширяемый язык разметки гипертекста), доступ к которой для скриптов (JavaScript) и программ осуществляется посредством DOM (от англ. Document Object Model — «объектная модель документа»).

Общение клиентской части с серверной будет осуществляться при помощи стандартного протокола прикладного уровня передачи данных HTTP (англ. HyperText Transfer Prоtocоl — «протокол передачи гипертекста»)[[1]](#footnote-1).

Серверная часть реализована на основе Apache HTTP Server, к которому средствами PHP Apache Module подключен скриптовый язык программирования PHP (англ. PHP: Hypertext Preprocessor — «PHP: препроцессор гипертекста»). Передача логических данных, обрабатываемых PHP в БД, осуществляется при помощи MySQL PHP Connector. Пользовательский контент передается в файловое хранилище в виде стандартных документов (Binary Data).

В качестве СУБД для сервера БД выбран MySQL. MySQL является решением для малых и средних приложений, что отвечает всем нашим требованиям. Пользовательский контент (файлы с таблицами и диаграммами, изображения, видео и т.д.) будет храниться в файловом хранилище.

1. При коммерческом внедрении данного продукта протокол HTTP планируется заменить его расширенной версией - протоколом HTTPS (Hypertext Transfer Protocol Secure), поддерживающей шифрование. [↑](#footnote-ref-1)