Содержание

[Введение 4](#_Toc323915097)

[1 Описание предметной области 6](#_Toc323915098)

[1.1 Проблемы и задачи экспертных систем 6](#_Toc323915099)

[1.2 Особенности разработки ЭС 7](#_Toc323915100)

[1.2.1 Структура ЭС 8](#_Toc323915101)

[1.2.2 Представление знаний и логический вывод 9](#_Toc323915102)

[1.2.3 Состав команды разработчиков ЭС 11](#_Toc323915103)

[1.2.4 Этапы разработки ЭС 11](#_Toc323915104)

[1.3 Обзор платформы Drools 13](#_Toc323915105)

[1.3.1 Машина вывода 13](#_Toc323915106)

[1.3.2 Инструменты для разработки ЭС 14](#_Toc323915107)

[1.4 Проектирование оптических систем 15](#_Toc323915108)

[1.5 Классификация оптических систем 18](#_Toc323915109)

[1.5.1 Общая классификация ОС 18](#_Toc323915110)

[1.5.2 Классификация объективов по техническим характеристикам 18](#_Toc323915111)

[1.5.3 Классификация объективов по обобщенным характеристикам 19](#_Toc323915112)

[1.6 Функциональные типы оптических элементов 21](#_Toc323915113)

[1.7 Типы оптических поверхностей 22](#_Toc323915114)

[1.8 Обозначение оптических элементов 23](#_Toc323915115)

[1.9 Обзор аналогов 24](#_Toc323915116)

[1.10 Постановка задачи 25](#_Toc323915117)

[1.10.1 Нефункциональные требования 25](#_Toc323915118)

[1.10.2 Функциональные требования 25](#_Toc323915119)

[2 Проектирование 27](#_Toc323915120)

[2.1 Системная архитектура 27](#_Toc323915121)

[2.1.1 Выбор инструментов разработки клиента 27](#_Toc323915122)

[2.1.2 Выбор инструментов разработки сервера 28](#_Toc323915123)

[2.2 Программная архитектура 29](#_Toc323915124)

[2.2.1 Компоненты системы 30](#_Toc323915125)

[2.3 Архитектура данных 32](#_Toc323915126)

[2.3.1 Модели фактов 32](#_Toc323915127)

[3 Реализация и тестирование 33](#_Toc323915128)

[3.1 Технологические особенности платформы 34](#_Toc323915129)

[3.2 Технологические особенности реализации 34](#_Toc323915130)

[3.2.1 Алгоритм структурного синтеза ОС 34](#_Toc323915131)

[3.2.2 Правила классификации ОС 35](#_Toc323915132)

[3.2.3 Правила отбора ОЭ 35](#_Toc323915133)

[3.2.4 Правила генерации структурных схем 35](#_Toc323915134)

[3.3 Описание интерфейса пользователя 35](#_Toc323915135)

[3.4 Отладка алгоритмов 35](#_Toc323915136)

[3.5 Тестирование 36](#_Toc323915137)

[3.5.1 Модульное тестирование 36](#_Toc323915138)

[3.5.2 Тестирование интерфейса 36](#_Toc323915139)

[3.5.3 Тестирование базы знаний 36](#_Toc323915140)

Введение

Экспертные системы (ЭС) были разработаны как научно-исследовательские инструментальные средства в 1960-х годах. Первое коммерческое внедрение произошло в 1980-х годах и с того времени они получили широкое распространение. Их предназначением является тиражирование опыта и знаний опытных экспертов в различных предметных областях, позволяющее устранить нехватку специалистов-экспертов, которые смог ли бы отвечать на многочисленные вопросы в своей области знаний. ЭС являются одним из успешных направлений искусственного интеллекта, применяемое в различных процессах принятия решений и решения задач в разных сферах социальной и технической деятельности человека.

Одной из таких сфер применения является поддержка систем автоматизированного проектирования (САПР). Здесь экспертные системы позволяют усилить автоматизацию традиционных САПР за счёт встраивания правил проектирования и инженерных знаний об оптимизации процессов проектирования, обеспечивающих решения различных задач, как по автоматизация повторяющихся задач, не требующих “творческого” мышления так и задач требующих мультидисциплинарных знаний.

Данная работа посвящена разработке продукционной экспертной системы нацеленной на автоматизацию одного из этапов проектирования оптических систем (ОС) - структурного синтеза. Так же работа ограничена только одним классом оптических систем - фотообъективами (в дальнейшем приставка “фото” будет опущена).

Работа является продолжением реализации теории композиции М.М. Русинова [[1]](https://docs.google.com/document/d/1K8JLzBVCg_ty5BBj8g5v_8m960awBC4h4GGWNQLUmSc/edit" \l "bookmark=id.baf7d18h49zc) и её развития в работах И.Л.Лившиц [[2]](https://docs.google.com/document/d/1K8JLzBVCg_ty5BBj8g5v_8m960awBC4h4GGWNQLUmSc/edit" \l "bookmark=id.6ijojpo8hpth)[[3]](https://docs.google.com/document/d/1K8JLzBVCg_ty5BBj8g5v_8m960awBC4h4GGWNQLUmSc/edit#bookmark=id.hoj24fgdtps8). Потенциал данной теории и сегодня остается раскрытым не полностью, что подтверждается интересом со стороны международного сообщества к данной тематике. Основными задачами теории композиции ОС является классификация элементов в оптической системе и анализ их применимости в тех или иных случаях. Элементы оптической системы по своему назначению разделяются на базовые (B), коррекционные (C), светосильные (T) и широкоугольные (Y). Получение описания последовательности элементов для достижения конкретных оптических характеристик называется структурным синтезом и является первым этапом автоматизированного проектирования ОС.

Задача структурного синтеза оптической системы не имеет детерминированного алгоритма решения, так как под одни и те же техническим требованиям может подходить большое количество как похожих, так и абсолютно различных структурных схем. Поэтому специалист-оптик может полагаться только на свой опыт и знания в проектировании ОС в выборе оптимальной структурной схемы. И чем он опытнее, тем более оптимальная будет выбрана схема.

Теория Русинова и Лившиц, описывает огромный опыт и знания в проектировании ОС, который способствует его формализации и использовании в качестве основы для разработки экспертной системы автоматизирующей процесс выбора структурной схемы ОС.

# 

# Описание предметной области

## Проблемы и задачи экспертных систем

Цель любой экспертной системы (ЭС) состоит в решении задач без прямого участия эксперта в предметной области. Принципиально отличие ЭС от других программ является то что она не просто ассистирует человеку, выполняя часть работы, а консультирует в какой-либо конкретной предметной области. ЭС не призваны заменить собою эксперта, в него непосредственной деятельности, а на наоборот расширяют возможную сферу применения знаний авторитетных специалистов, за счет аккумулирования и тиражирования опыта и знаний высококвалифицированных специалистов, позволяя пользоваться этими знаниями пользователям, не являющимися специалистами в данной предметной области. Кроме того экспертные системы имеют некоторые преимущества над человеком-экспертом:

* *Доступность*: ЭС может работать там, где есть компьютер,
* *Стоимость*: цена для отдельного пользователя гораздо ниже,
* *Опасность*: может быть использована в условиях не пригодных для деятельности человека,
* *Срок эксплуатации*: зависит только от оборудования,
* *Скорость*: может предоставлять быстрый ответ, либо в режиме реального времени для критичных приложений,
* *Достоверность*: может предоставлять мнение, отличное от мнения человека-эксперта.

Но и имеют ограничения, которые не присущи эксперту:

* *Отсутствие глубоких знаний*. ЭС не имеет понятия о реальных причинах или следствиях в системе, в основном это потому что легче запрограммировать поверхностные знания, основанные на эмпирических и эвристических знаниях. А проектирование ЭС основанной на базовых понятиях и поведении некоторых базовых объектов потребует намного больше усилий, а в результате система будет слишком сложной для поддержки.
* *Незнание границ неопределенности*. Человек-эксперт знает границы своих знаний и может изменить свою рекомендацию, когда проблема требует знаний за пределами знаний эксперта. Когда как ЭС предложит рекомендацию даже в ситуации, когда данных не хватает для предоставления реального решения.

Существующие ЭС не могут проводить аналогии, например обобщать имеющиеся знания, для того чтобы рассуждать о новой ситуации так как это может человек.

В настоящее время экспертные системы используется для решения различных типов задач (интерпретация, предсказание, диагностика, планирование, конструирование, контроль, отладка, инструктаж, управление) в самых разнообразных проблемных областях, таких, как финансы,  нефтяная и газовая промышленность, энергетика, транспорт, фармацевтическое производство, космос, металлургия, горное дело, химия, образование, целлюлозно–бумажная промышленность, телекоммуникации и связь и др.

## Особенности разработки ЭС

Разработка ЭС имеет существенные отличия от разработки обычного программного продукта. Опыт создания ЭС показал, что использование при их разработке методологии, принятой в традиционном программировании, либо чрезмерно затягивает процесс создания ЭС, либо вообще приводит к отрицательному результату [[1]](https://docs.google.com/document/d/1K8JLzBVCg_ty5BBj8g5v_8m960awBC4h4GGWNQLUmSc/edit" \l "bookmark=id.klhzy8isybnw).

При разработке ЭС, как правило, используется концепция «быстрого прототипа». Суть этой концепции состоит в том, что разработчики не пытаются сразу построить конечный продукт. На начальном этапе они создают прототип (прототипы) ЭС. Прототипы должны удовлетворять двум противоречивым требованиям: с одной стороны, они должны решать типичные задачи конкретного приложения, а с другой — время и трудоемкость их разработки должны быть весьма незначительны. Для удовлетворения указанным требованиям, как правило, при создании прототипа используются разнообразные средства, ускоряющие процесс проектирования.

Прототип должен продемонстрировать пригодность методов инженерии знаний для данного приложения. В случае успеха эксперт с помощью инженера по знаниям расширяет знания прототипа о проблемной области. При неудаче может потребоваться разработка нового прототипа или разработчики могут прийти к выводу о непригодности методов ЭС для данного приложения. По мере увеличения знаний прототип может достигнуть такого состояния, когда он успешно решает все задачи данного приложения. Преобразование прототипа ЭС в конечный продукт обычно приводит к перепрограммированию ЭС на языках низкого уровня, обеспечивающих как увеличение быстродействия ЭС, так и уменьшение требуемой памяти. Трудоемкость и время создания ЭС в значительной степени зависят от типа используемого инструментария.

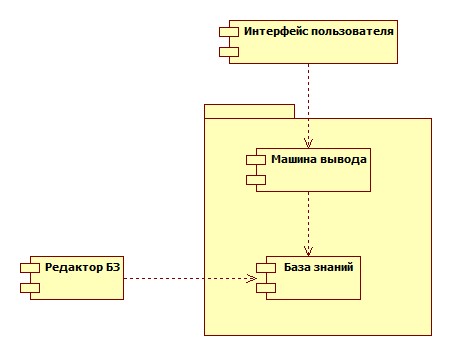
### Структура ЭС

Все экспертные системы имеют сходную структуру. В основе которой лежит разделение знаний, заложенных в системе, и алгоритмов их обработки. ЭС могут иметь сложную, разветвленную  структуру модулей, но для любой ЭС необходимо наличие следующих основных блоков (рисунок 1.1):

*База знаний* (БЗ)  - наиболее ценный компонент ЭС, включающий в себя совокупность знаний о предметной области и способах решения задач. Обычно знания в БЗ записываются в форме, приближенной к естественному языку. Такая форма записи получила название язык представления знаний (ЯПЗ). В различных ЭС могут быть использованные разные ЯПЗ,

*Машина вывода* (МВ) - модуль, выполняющий рассуждения на основании знаний, заложенных в БЗ. Машина вывода является неизменной частью ЭС,

*Редактор базы знаний* - модуль, предназначенный для разработчиков ЭС, позволяющий добавлять новые знания в БЗ и редактировать существующие,

*Интерфейс пользователя* - модуль, взаимодействующий с пользователем, через который система запрашивает необходимые для ее работы данные и выводит результат.

**Рисунок 1.1 - Основные компоненты экспертной системы**

В процессе функционирования ЭС загружает информацию из своей базу знаний и пытается провести логический вывод решения поставленной перед ней задачей с помощью машины вывода. База знаний заполнена знаниями описанными с помощью языка представления знаний.

### Представление знаний и логический вывод

Под *представлением знаний* понимается методика и форма структурированного описания и хранения в БЗ знаний человека-эксперта. Существует большое количество способов представления знаний, и при разработке новой экспертной системы может быть выбран один из них, или использоваться комбинация нескольких способов.

В данной работе применяется комбинация двух способов представления знаний, такие как продукционные правила и объектно-ориентированный подход. Объектно-ориентированный подход используется для записи фактов, описывающих состояние предметной области, составляющие её объекты и свойства. Продукционные правила описывают способы решения задачи.

Продукционные правила являются выражениями вида:

“**ЕСЛИ** *<условие>* **ТО** *<действие>*”

Экспертные системы основанные на продукционных правилах, называются продукционными ЭС. Они наиболее подходящие в те случаях, когда знания в предметной области возникают на основе эмпирических ассоциации, накопленных за многие годы работы по решению задач в данной области. В качестве условия и действия может быть, например, предположение о наличии того или иного свойства, принимающее значение истина или ложь. Когда как термин действие следует трактовать широко: это может быть как  директива к выполнению какой-либо операции, рекомендация, или модификация базы знаний.

Примером продукции может быть следующее выражение, взятое из базы знаний, разработанной в данной работе:

**ЕСЛИ** оптическая система с вынесенным назад зрачком и величиной заднего фокального отрезка, больше половины и меньше одного фокусного расстояния

**ТО** добавить в структурную схему базовый элемент, первая поверхность которого типа P или лежит в 1-й зоне, а вторая - типа A и лежит во 2-й зоне

Факты поступают в систему через интерфейс пользователя или выводятся в процессе поиска решения задачи. И когда в процессе интерпретации правил МВ какой-либо факт  согласуется с частью правила ЕСЛИ, то выполняется действие, определенное частью ТО этого правила. Последовательное сопоставление частей правил ЕСЛИ с фактами порождает *цепочку вывода* и называется *логическим выводом*.

Существует два основных способа выполнения правил в системе: прямая цепочка рассуждений или прямой вывод и обратная цепочка рассуждений или обратный вывод. В данной работе используется только прямой вывод на знаниях, он применим в задачах, где на основании имеющихся фактов необходимо определить тип объекта или явления, выдать рекомендацию, определить диагноз и т.д.

Продукционные правила обеспечивают естественный способ описания процессов, управляемый сложной и изменяющейся средой. Через правила можно описать ход решения задачи, не имея заранее алгоритма этого решения. Более того, можно корректировать способ решения, добавляя новые правила, не изменяя существующих, что обеспечивает высокую модульность базы знаний.

Однако, несмотря на то, что с помощью продукционных правил можно представить решение любой задачи, при большом количестве правил становится сложно отслеживать непротиворечивость базы знаний.

### Состав команды разработчиков ЭС

В разработке ЭС участвуют представители следующих специальностей:

* эксперт в проблемной области, задачи которой будет решать ЭС,
* инженер по знаниям — специалист по разработке ЭС (используемые им технологии (методы) называют технологией (методами) инженерии знаний),
* программист по разработке инструментальных средств (ИС), предназначенных для ускорения разработки ЭС,
* специалист по тестированию с квалификацией инженера по знаниями дополнительными навыками тестирования ПО.

Необходимо отметить, что отсутствие среди участников разработки инженеров по знаниям (т.е. их замена программистами) либо приводит к неудаче процесс создания ЭС, либо значительно удлиняет его.

Эксперт определяет знания (данные и правила), характеризующие проблемную область, обеспечивает полноту и правильность введенных в ЭС знаний.

Инженер по знаниям помогает эксперту выявить и структурировать знания, необходимые для работы ЭС; осуществляет выбор того ИС, которое наиболее подходит для данной проблемной области, и определяет способ представления знаний в этом ИС; выделяет и программирует (традиционными средствами) стандартные функции (типичные для данной проблемной области), которые будут использоваться в правилах, вводимых экспертом [[1]](https://docs.google.com/document/d/1K8JLzBVCg_ty5BBj8g5v_8m960awBC4h4GGWNQLUmSc/edit" \l "bookmark=id.klhzy8isybnw).

Программист разрабатывает ИС (если ИС разрабатывается заново), содержащее в пределе все основные компоненты ЭС, и осуществляет его сопряжение с той средой, в которой оно будет использовано.

### Этапы разработки ЭС

В ходе работ по созданию ЭС сложилась определенная технология их разработки, включающая шесть следующих этапов [[1]](https://docs.google.com/document/d/1K8JLzBVCg_ty5BBj8g5v_8m960awBC4h4GGWNQLUmSc/edit" \l "bookmark=id.klhzy8isybnw):

* идентификацию,
* концептуализацию,
* формализацию,
* выполнение,
* тестирование,
* опытную эксплуатацию.

На этапе *идентификации* определяются задачи, которые подлежат решению, выявляются цели разработки, определяются эксперты и типы пользователей.

На этапе *концептуализации* проводится содержательный анализ проблемной области, выявляются используемые понятия и их взаимосвязи, определяются методы решения задач.

На этапе *формализации* выбираются ИС и определяются способы представления всех видов знаний, формализуются основные понятия, определяются способы интерпретации знаний, моделируется работа системы, оценивается адекватность целям системы зафиксированных понятий, методов решений, средств представления и манипулирования знаниями.

На этапе *выполнения* осуществляется наполнение экспертом базы знаний. В связи с тем, что основой ЭС являются знания, данный этап является наиболее важным и наиболее трудоемким этапом разработки ЭС. Процесс приобретения знаний разделяют на извлечение знаний из эксперта, организацию знаний, обеспечивающую эффективную работу системы, и представление знаний в виде, понятном ЭС. Процесс приобретения знаний осуществляется инженером по знаниям на основе анализа деятельности эксперта по решению реальных задач [2]. В режиме приобретения знаний общение с ЭС осуществляет (через посредничество инженера по знаниям) эксперт. В этом режиме эксперт, используя компонент приобретения знаний, наполняет систему знаниями, которые позволяют ЭС в режиме решения самостоятельно (без эксперта) решать задачи из проблемной области. Эксперт описывает проблемную область в виде совокупности данных и правил. Данные определяют объекты, их характеристики и значения, существующие в области экспертизы. Правила определяют способы манипулирования с данными, характерные для рассматриваемой области [3]. Отметим, что режиму приобретения знаний в традиционном подходе к разработке программ соответствуют этапы алгоритмизации, программирования и отладки, выполняемые программистом. Таким образом, в отличие от традиционного подхода в случае ЭС разработку программ осуществляет не программист, а эксперт (с помощью ЭС), не владеющий программированием.

В режиме *консультации* общение с ЭС осуществляет конечный пользователь, которого интересует результат и (или) способ его получения.

Необходимо отметить, что в зависимости от назначения ЭС пользователь может не быть специалистом в данной проблемной области (в этом случае он обращается к ЭС за результатом, не умея получить его сам), или быть специалистом (в этом случае пользователь может сам получить результат, но он обращается к ЭС с целью либо ускорить процесс получения результата, либо возложить на ЭС рутинную работу). В режиме консультации данные о задаче пользователя после обработки их диалоговым компонентом поступают в рабочую память. Интерпретатор на основе входных данных из рабочей памяти, общих данных о проблемной области и правил из БЗ формирует решение задачи. ЭС при решении задачи не только исполняет предписанную последовательность операции, но и предварительно формирует ее. Если реакция системы не понятна пользователю, то он может потребовать объяснения.

## Обзор платформы Drools

За время существования и использования технологии экспертных систем было создано множество инструментов позволяющих упростить, ускорить разработку ЭС и сделать её более эффективной. Эти инструменты как правило различаются тем, какие из четырех основных компонентов, описанных выше, они предоставляют.

В данной работе используется одна из наиболее популярных и развитых - платформа Drools[[1]](#footnote-1) от компании Red Hat. Она предоставляет большое количество инструментов, в том числе все основные компоненты ЭС, кроме пользовательского интерфейса, для разработки.

### Машина вывода

Платформа включает развитую машину вывода (Drools Expert), которая поддерживает как прямой, так и обратный методы логического вывода. Она тесно интегрирована с платформой Java, что позволяет управлять МВ через удобный программный интерфейс.

В качестве основного языка представления знаний используется собственный язык под названием Drools Rule Language (DRL), который активно использует конструкции и возможности языка Java для написания правил.

Пример правила написаного на DRL:

**rule "B1P1A"**

**when**

**Classification(d==1, s==2)**

**then**

**insert( ElementFactory.newElement( "B1P1A" ));**

**end**

**Как видно из примера, правила написанные на языке DRL имеют трабиционную для продукционных правил структуру, где часть условий описана после ключивого слова when, а часть действий - после then.**

### Инструменты для разработки ЭС

Для эффективной разработки баз знаний платформа Drools включает компонент под названием Guvnor, которые комбинирует в себе множество интрументов, таких как редакторы правил и таблиц решений, инструмент для тестирования, хранилище баз знаний и прочее. На Рисунок 1.2 представлен интерфейс Guvnor.

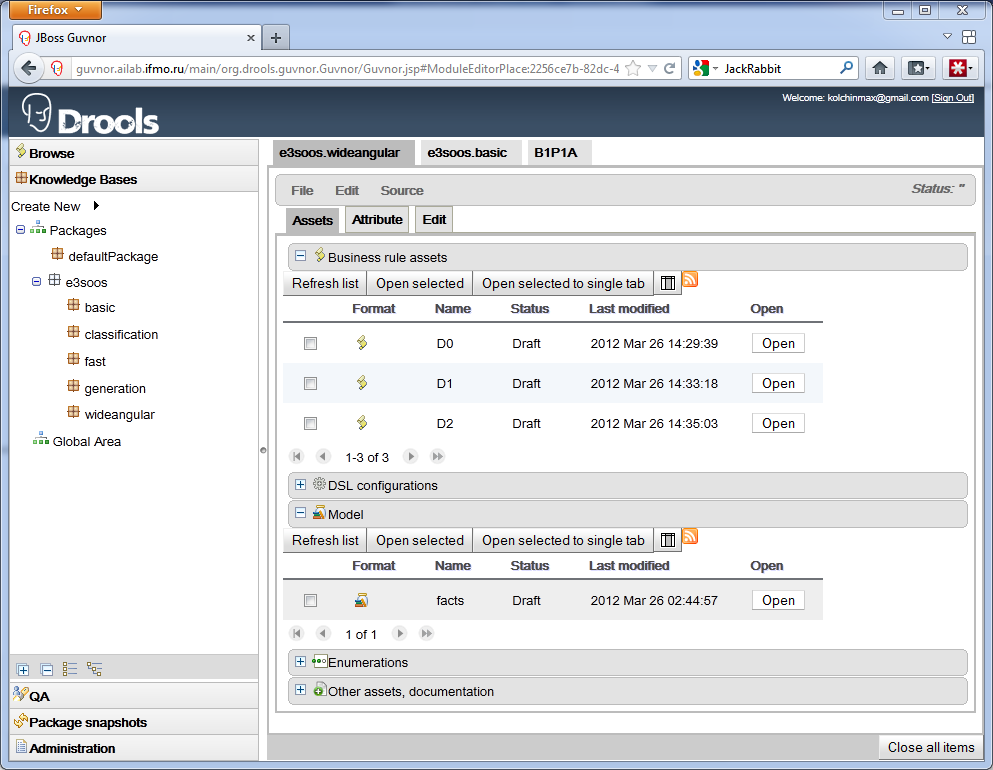


Рисунок 1.2 - Интерфейс Guvnor

Так же Guvnor предоставляет возможность для разработки предметно-ориентированного языка (DSL), которые может быть использован при написании правил. Рассмотри пример правила из предыдущего раздела, но записанного спомощью конструкция DSL языка, пример на Рисунок 1.3

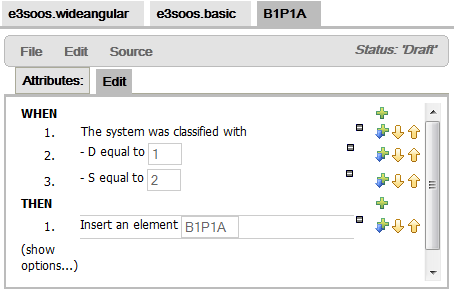


Рисунок 1.3 - Пример правила описанного с помощью DSL языка

Как было сказано выше, платформа Drools тесно интегрирована с языком программирования Java, поэтому позволяет использовать описание классов на этом языке, в качестве моделей фактов при написании правил. Модели фактов описаны во второй главе данной работы.

Для хранения правил и моделей фактов используется репозиторий данных Apache JackRabbit[[2]](#footnote-2), поддерживающий версионность хранимых данных, полнотекстовый поиск, транзакции. Таким образом обеспечивая надежное хранение баз знаний.

А для доступа к разработанным БЗ, Guvnor содержит веб-сервис, предоставляющий RESTful[[3]](#footnote-3)-интерфейс, позволяющий клиенту скачать необходимую версию базы знаний.

## Проектирование оптических систем

Расчет оптических систем сводится к умению разработчиков оптимальным образом расположить элементы оптической схемы, чтобы конечный результат - качество изображения оптической системы можно было бы довести до заданного предела в соответствующих геометрических размерах.

Разработка любой оптической системы начинается с анализа технического задания - требований на ее проектирование, позволяющих на конечном этапе выработать критерии пригодности оптической системы. В соответствии с этими требованиями проводится синтез принципиальной оптической схемы.

Проектирование ОС можно представить как поэтапный процесс принятия решений, в соответствии с рисунком 1.1.1 включающий в себя следующие этапы:

1. анализ технического задания,
2. структурный синтез,
3. параметрический синтез,
4. оптимизация параметров системы,
5. анализ качества изображения системы.

Причем на этапах оптимизации и анализ ОС может быть принято решение вернуться на предыдущие этапы проектирования. Если оптимизировать оптическую систему не удалось, то необходимо вернуться на этап структурного синтеза, где выбрать другую структурную схему. А если оптимизация прошла успешно, но система не удовлетворила требованиям качества изображения, то может быть решено вернуться на этап оптимизации, что бы рассмотреть систему с другими параметрами.

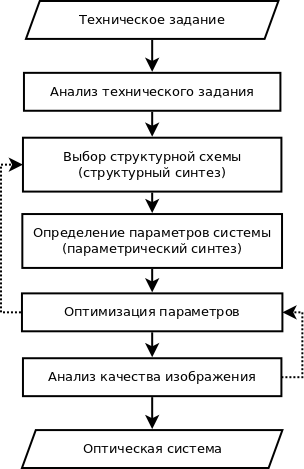


Рисунок 1.4 - Основные этапы проектирования оптической системы

Из рисунка 1.1.1 видно, что определены два этапа синтеза ОС:

* структурный синтез - процедура выбора типов, количества и взаимного расположения оптических элементов, из которых состоит ОС.
* параметрический синтез - процедура определения конструктивных параметров оптических элементов, входящих в ОС, таких как радиус, толщина, положение воздушных промежутков, типов оптических сред и т.д.

Определяющим является этап выбора структурной схемы, поскольку при ее удачном выборе общий результат достигается гораздо быстрее за счет более быстрой сходимости процесса оптимизации параметров. В то время, как приблизится к оптимальному решению при неудачном выборе схемы практически не представляется возможным, так как подобная ситуация сводится к оценке множества комбинаций различных значений свободных параметров системы. На практике это приводит к обычному методу проб, требующему при большом количестве конструктивных параметров схемы, неоправданно больших затрат машинного времени, обычно не приводящих к успеху.

Для установления соответствия между требованиям к оптической системе, содержащихся в техническом задании на её проектирование, и теми средствами, при помощи которых эти требования реализуются, была определена классификация оптических систем. Тем самым она позволяет установить связь между классификацией ОС и их структурными схемами.

## Классификация оптических систем

### Общая классификация ОС

Существует несколько видов классификации оптических систем, одна из них это общая классификация ОС по расположению объекта и изображения (таблица 1.1).

**Таблица 1.1 - Общая классификация ОС по расположению объекта и изображения**

|  |  |
| --- | --- |
| **Условное обозначение класса** | **Наименование класса ОС** |
| 00 | Телескопическая система |
| 01 | Фотообъектив |
| 10 | Микробъектив |
| 11 | Оборачивающая система |

Как видно из таблицы, в соответствии с этой обобщенной классификацией ОС, существует 4 класса оптических систем, различающихся по положению объекта и его изображения. При более подробном рассмотрении в каждом из этих классов обнаружится множество подклассов, которые определяются уточненными классификациями.

### Классификация объективов по техническим характеристикам

Существует множество классификаций объективов, так как классификация играет важную роль в понимании процесса проектирования. В работе рассматриваются те из них, которые будут полезны для инженера по знаниям при формализации процесса компоновки объективов.

Различаются два вида характеристик: технические и обобщенные.

Рассмотрим основные технические характеристики объективов, которые соответствуют конкретным значениям физических величин:

* F - фокусное расстояние,
* W - угловое поле,
* J - относительное отверстие,
* L - спектральный диапазон,
* Q - показатель качества изображения,
* S - задний фокальный отрезок,
* D - конструктивные особенности (положение апертурной диафрагмы).

В таблице 1.2 представлены единицы измерения для технических характеристик объективов.

**Таблица 1.2 - Единицы измерения технических характеристик объективов**

|  |  |  |
| --- | --- | --- |
| Условное обозначение | Наименование | Единицы измерения |
| J | Светосила | безразмерная величина |
| W | Угловое поле | угловые единицы |
| F | Фокусное расстояние | миллиметры |
| L | Спектральный диапазон | нанометры |
| Q | Качество изображения | части длины волны |
| S | Задний фокальный отрезок | миллиметры |
| D | Положение апертурной диафрагмы | миллиметры |

### Классификация объективов по обобщенным характеристикам

Кроме классификации по техническим характеристикам существует классификация объективов по обобщенным характеристикам, в соответствии с которой каждой технической характеристике присваивается не конкретное ее значение, а некоторое условное число, характеризующее диапазон, в котором находится данная техническая характеристика, рисунок 1.2.2.

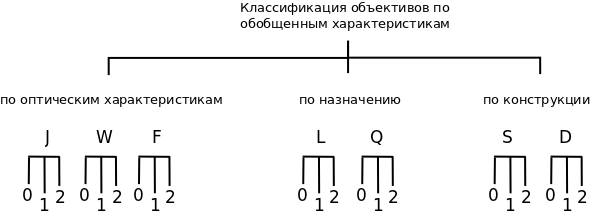


Рисунок 1.5 - Классификация объективов по обобщенным характеристикам

Числа “0”,”1”,”2” на рисунке 1.2.2 являются условными обозначениями обобщенных характеристик объективов, они неявно связанны с выбором структурной схемы ОС.

«0» —соответствует ОС с таким значением технической характеристики, для реализации которой достаточно простейшей оптической схемы;

«2» — соответствует ОС со значением технической характеристики объектива, схема которого максимально сложна и имеет наибольшее количество элементов для достижения требуемых высоких значений технических характеристик;

«1» — характеризует ОС, которая занимает промежуточное положение по сложности реализации между «0» и «2».

Использование этой классификации объективов позволяет описать 37 = 2187 классов оптических систем.

Связь между обобщенной и технической классификациями объективов представлена в таблице 1.2.3.

**Таблица 1.3 - Связь обобщенной и технической характеристик**

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | Число “0” | Число “1” | Число “2” |
| J | свыше 2.8 | от 1.5 до 2.8 | менее 1.5 |
| W | менее 15º | от 15º до 60º | свыше 60º |
| F | менее 50 мм | от 50 до 100 мм | более 100 мм |
| L | менее 5 нм | от 5 до 200 нм | более 200 нм |
| Q | Геометрически - ограниченное,  d > 5dэ | Промежуточное,  2dэ <= d <= 5dэ | Дифракционное,  d < 2dэ |
| S | менее F/2 | от F/2 до F | более F |
| D | внутри системы | вынесен назад | вынесен вперед |

## Функциональные типы оптических элементов

Основной принцип, который положен в основу процедуры синтеза принципиальной схемы оптической системы, заключается в том, что каждый оптический элемент функциональным устанавливается строго в соответствии с его функциональным назначением. Такой принцип позволяет исключить возможность попадания в оптическую систему заведомо "лишних" элементов.

В своей теории М.Русинов и И.Лившиц выделяют четыре типа элементов в соответствии с их функциональным назначением: базовые, широкоугольные, светосильные и коррекционные.

Предлагается обозначать указанные четыре основных типа оптических элементов, используемых при проектировании объективов, следующим образом:

* B - базовые элементы,
* Y - широкоугольные элементы,
* T - светосильные элементы,
* C - коррекционные элементы.

Элементы располагаются в строго определенном порядке, называемый “формулой синтеза” (рисунок 1.4).

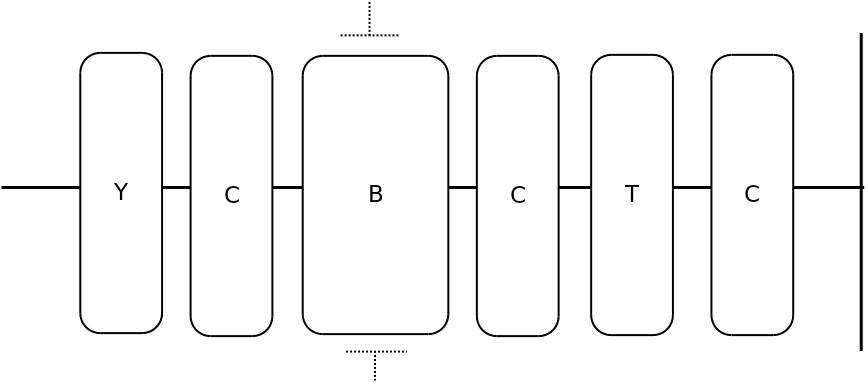


Рисунок 1.6- Функциональный порядок расположения оптических элементов

## Типы оптических поверхностей

Каждый оптический элемент имеет две поверхности. Экспертами выявлено шесть основных типов поверхностей оптических элементов, которые предложены к применению (таблица 1.1.4.1).

**Таблица 1.4 - Типы оптических поверхностей**

|  |  |
| --- | --- |
| Обозначение | Наименование поверхности |
| O | Плоская поверхность |
| P | Поверхность, концентричная центру входного зрачка |
| A | Апланатическая поверхность |
| F | Поверхность, концентричная точке фокуса |
| I | Поверхность, расположенная в близфокальной зоне |
| V | Произвольная поверхность |

Поверхности, входящие в состав оптической системы, могут располагаться различным образом. Предлагается рассматривать следующие варианты их расположения [[2]](https://docs.google.com/document/d/1K8JLzBVCg_ty5BBj8g5v_8m960awBC4h4GGWNQLUmSc/edit" \l "bookmark=id.jyinr9i8qnfo), условно названные зонами, обозначаемые в работе следующим образом в соответствии с Рисунок 1.7:

* “1” - первая зона, занимающая пространство от предмета до апертурной диафрагмы,
* “2” - вторая зона, занимающая пространство вблизи апертурной диафрагмы или непосредственно после неё,
* “3” - третья зона, занимающая близфокальное пространство.

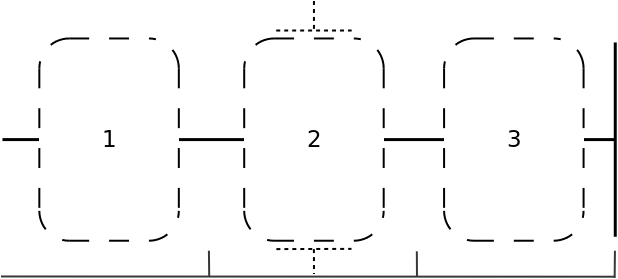


Рисунок 1.7 - Зоны расположения оптических поверхностей

## Обозначение оптических элементов

Таким образом, предложенными четырьмя функциональными типами элементов и шестью типами поверхностей, причем у каждого элемента по две поверхности, можно описать 4\*62 = 254 простых элемента. Простым элементов называется элемент состоящий только из одного элемента, когда как сложный элемент, представляет из себя комбинацию нескольких простых элементов.

Принимая во внимание принятые условия обозначения типов элементов, оптических поверхностей и зон их расположения предлагается обозначать следующим образом: в начале указывается тип элемента, затем указывается зона, в которой находится первая оптическая поверхность, далее тип этой поверхности, после указывается зона второй поверхности и тип второй поверхности:

**e, x, s, x, s**

где буквы в формуле это:

* e - функциональный тип оптического элемента,
* x - номер зоны, в которой находится поверхность,
* s - тип оптической поверхности,

Ниже приведены примеры записей оптических элементов:

B202P - базовый элемент, обе поверхности которого расположены справа от апертурной диафрагмы, первая поверхность - плоская, а вторая - концентрична центру входного зрачка.

T2А2P - светосильный элемент, обе поверхности которого расположены справа от апертурной диафрагмы, первая поверхность - апланатическая, а вторая концентрична центру входного зрачка.

C1P1P - коррекционный элемент, состоящий из двух концентричных центру входного зрачка поверхностей, расположенный в пространстве до апертурной диафрагмы.

Y1O2A - широкоугольный элемент, первая поверхность которого плоская и расположена до апертурной диафрагмы,     а вторая – апланатическая и расположена вблизи апертурной диафрагмы.

Таким образом, формализовав обозначение оптических элементов, составляющих структурную схему оптической системы, предлагается описывать структурную схему следующим образом, описанным ниже в форме Бэкуса - Наура:

**структурная схема ::= <Y>” + “<B>” + “<T>**

**<Y> ::= ”” | <Y\_расш> | <Y\_расш><C>**

**<Y\_расш> ::= ”Y”<список\_поверхностей>**

**<B> ::= <B\_расш> | <B\_расш><C>**

**<B\_расш> ::= ”B”<список\_поверхностей>**

**<T> ::= ”” | <T\_расш> | <T\_расш><C>**

**<T\_расш> ::= ”T”<список\_поверхностей>**

**<C> ::= ”” | <C\_расш>**

**<C\_расш> ::= ”C”<список\_поверхностей>**

**<список\_поверхностей> ::= <поверхность><поверхность>**

**<поверхность> ::= <зона><тип\_поверхности>**

**<зона> ::= ”1” | ”2” | ”3”**

**<тип\_поверхности> ::= ”O” | ”P” | ”A” | ”F” | ”I” | ”V”**

Пример структурной схемы объектива: **Y1O1P + C2P2P + B2A3P + T3F3O**

## Обзор аналогов

Существует несколько различных попыток решить проблему выбора структурной схемы оптической системы, такие как:

* *различные каталоги* - структурированные каталоги, содержащие примеры схем и схемы разработанных оптических систем: патентные, отраслевые, базы данных предприятий,
* *генетические алгоритмы* - программы, использующие генетические алгоритмы для генерации исходных схем,
* *экспертные системы* - программы, имитирующие решение задачи специалистом,
* *собственный опыт* - самостоятельно на основе имеющегося в проектировании ОС опыта, своего или чужого.

Несмотря на многочисленные публикации по данной теме, на данный момент не существует программного обеспечения решающего проблему выбора исходной оптической схемы.

## Постановка задачи

Задачей данной работы является разработка продукционной экспертной системы, автоматизирующей выбор структурной схемы оптической системы, класса фотообъективов.

### Нефункциональные требования

Система должна удовлетворять следующим нефункциональным требованиям:

1. реализации экспертной системы должна быть основана на использовании компонентов платформы Drools;
2. доступ к системе должен предоставлять через Интернет в виде веб-приложения;
3. разграничение прав доступа к функциям системы;
4. ограничение доступа к системе именем и паролем пользователя;

### Функциональные требования

Система должна иметь следующие функциональные возможности:

* автоматическая генерация структурных схем по введенным техническим требованиям:
  + ввод технических требований,
  + вывод кодового представления схемы.
* отрисовка графического изображения структурных схем, на основе их кодового представления,
* предоставление отладочной информации по выполненной генерации, только для пользователя с ролью “администратор”,
* ведение базы пользователей:
  + добавление и удаление пользователя,
  + редактирование данных пользователя,
  + добавление и удаление ролей пользователю.

# Проектирование

Данная глава посвящена проектированию системной и программной архитектур системы. В ней описывается архитектура системы, выбор инструментов разработки и архитектуры базы данных, а также приведены схемы и иллюстрации, как формализующие решаемую проблему, так и отображаемые архитектурные особенности разрабатываемой системы.

## Системная архитектура

Согласно техническим требованиям необходимо разработать веб-приложение предоставляющее доступ пользователя к функциям экспертной системы, автоматизирующей структурный синтез ОС. Данное требование предъопределяет архитектуру системы.

Веб-приложение – это клиент-серверное приложение, где клиентом выступает браузер, а сервером – веб-сервер.

Перед начало разработки был проведен обзор существующих инструментов разработки веб-приложений и выбраны наиболее подходящие.

### Выбор инструментов разработки клиента

Наиболее популярным и универсальным средством разработки клиентской части веб-приложения является использование комбинации языка разметки документов – HTML[[4]](#footnote-4), языка описания внешнего вида документа – CSS[[5]](#footnote-5) и интерпретируемого языка программирования JavaScript[[6]](#footnote-6), назовем этот подход «легким». А другие менее универсальные подходы, такие как использование технологий Flash[[7]](#footnote-7), Silverlight[[8]](#footnote-8) и Java-апплетов[[9]](#footnote-9), назовем «тяжелым» подходом.

Существенная разница между «легким» и «тяжелым» подходами в том что для работы «тяжелого» подхода необходима установка специальных плагинов для браузеров, являющихся основой для работы с этими технологиями. Когда как основа для «легкого» подхода, основа для его работы встроена как правило во все современные браузеры. Под основой следует понимать набор программных интерфейсов, интерпретатор или компилятор, если они необходимы и т.п.

Большим плюсом «тяжелого» подхода является то что использование специальных плагинов скрывает многие несовместимости браузеров и предоставляет больший контроль над интерфейсом, но зависимость от плагинов является и минусом данного подхода, так как их поддержка на множестве существующих операционных системах и платформах сложна и дорога, поэтому развитие «тяжелого» подхода значительно отстает от «легкого». Развитие «легкого» подхода значительно увеличилось с разработкой нового стандарта HTML5[[10]](#footnote-10), поддерживаемого организацией W3C[[11]](#footnote-11), который нацелен на строгую стандартизацию и внедрению новых технологий, позволяющих веб-приложениям конкурировать с традиционными приложениями.

В свете описанного выше, было принято решении об использовании «легкого» подхода (HTML + CSS + JavaScript) в разработке клиентской части веб-приложения, как наиболее гибкого и функционального решения.

### Выбор инструментов разработки сервера

Следуя предъявленным требованиям в использовании платформы Drools для создания экспертной системы, языком программирования для разработки серверной части веб-приложения был выбран язык Java. Так как программные интерфейсы к машине вывода (Drools Expert), с которыми необходимо работать, существуют только для языка Java.

Для разработки веб-сервера был выбран Play![[12]](#footnote-12) Фрэймворк по следующим причинам:

1. *Быстрота разработки.* Play! разрабатывается следуя парадигме проектирования программного обеспечения под названием «соглашение по конфигурации» (Convention over configuration[[13]](#footnote-13)), которая стремиться уменьшить количество решений, которые разработчик должен принять во время разработки, засчет упрошения конфигурирования, предоставления готовых модулей и т.п.;
2. *Использование проверенных Java компонентов.* Фрэймворк использует провереные компоненты Java платформы, такие как JPA (для доступа к БД) и прочее;
3. *Поддержка MVC паттерна.* Так же для упрощения разработка ведётся следуя принципам известному паттерну «Модель-Представление-Поведение».

## Программная архитектура

Как бы описано выше система имеет клиент-серверную архитектуру, где клиентом выступает браузер пользователя, который загружает необходимые компоненты клиента с сервера по HTTP-протоколу и позволяет выполнять код клиента на стороне пользователя, тем самым уменьшает нагрузку на сервер.

Диаграмма развертывания системы представлена на Рисунок 2.1.

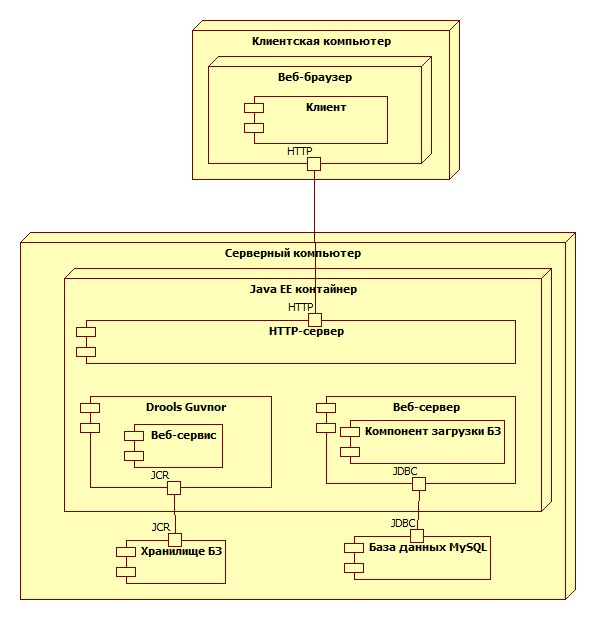


Рисунок 2.1 - Диаграмма развертывания системы

### Компоненты системы

Так как архитектура система клиент-серверная, то и описание компонентов разделим на клиентские и серверные.

#### Клиентские компоненты

Когда пользователь системы с помощью веб-браузера переходит по адресу, на котором расположена разработанная система, загружаются следующие клиентские компоненты, все из которых написаны на языке Javascript:

1. Компонент верификации данных – компонент, производящий верификацию данных вводимых пользователем,
2. Компонент коммуникации – осуществлюящий коммуникацию с сервером для отправки AJAX – запросов на сервер,
3. Компонент рисования – компонент, рисующий структурные схемы.

На Рисунок 2.2 представлена диаграмма клиентских компонентов их зависимости.

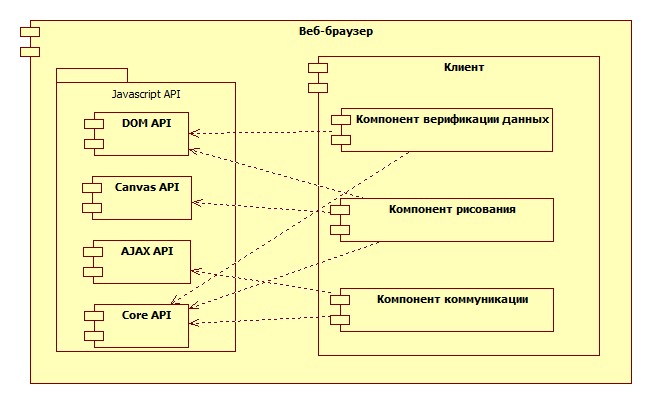


Рисунок 2.2 - Диаграмма компонентов клиента

Из диаграммы видно что для работы компоненты используют различные подсистемы браузера, получая к ним доступ через программный Javascript-интерфейс.

#### Серверные компоненты

В соответствии с требованиями сервер разворачивается на Java EE контайнере. И включает следующие компоненты:

1. Загрузчик баз знаний – компонент, отвечающий за загрузку необходимой верси базы знаний из хранилища,
2. Машина вывода,
3. Обработчик запросов – набор контролеров обрабатывающих HTTP-запросы пользователей,
4. Компонент управления МВ – компонент, отвечающий за работу с машиной вывода. Загружает и запускает обработку правил и фактов.

Диаграмма сервеных компонентов представлена на Рисунок 2.3.

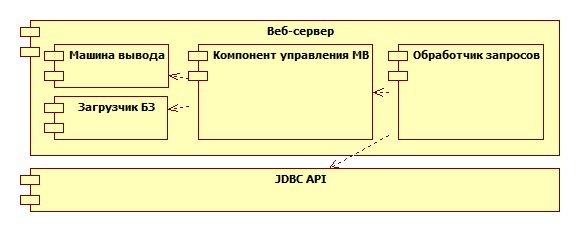


Рисунок 2.3 - Диаграмма компонентов сервера

## Архитектура данных

### Архитектура моделей фактов

Для представления фактов в базе знаний используются простые объекты языка Java, так называемые POJO – объекты. Классы для описания фактов могут быть описаны как на языке DRL, так и на языке Java, так как в конечном итоге все будет интерпритировано в язык Java.

Разработано 5 классов и 4 классы перечисления:

* Classification –
* Requirements –
* Element –
* Schema –
* ElementFactory –
* ElementType –
* SurfaceType –
* ImageQuality –
* EntrancePupilPosition -

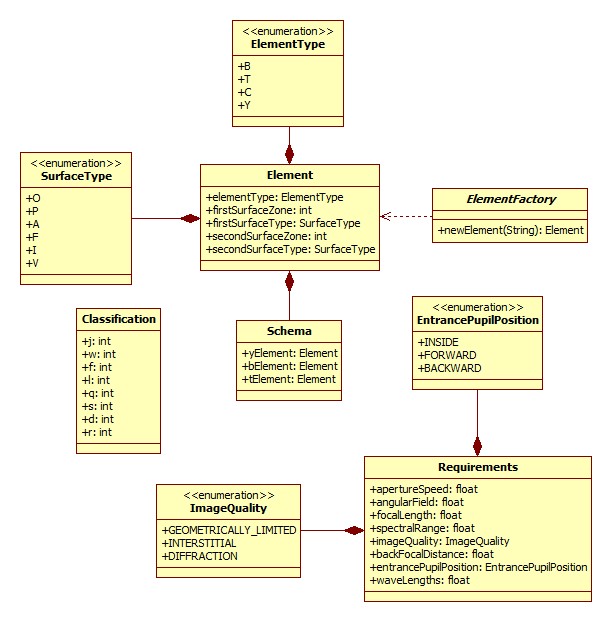


Рисунок 2.4 - Диаграмма классов фактов

# Реализация и тестирование

В данной главе представлены особенности реализации системы, составлен план тестирования и представлен графический интерфейс пользователя.

## Технологические особенности платформы

## Технологические особенности реализации

Технологической особенностью реализации разработанной системы является использование технологии экспертных систем в решении поставленной задачи. В частности использование платформы Drools и её инструментов.

### Алгоритм структурного синтеза ОС

Задача структурного синтеза, которая решается в данной работе состоит в том чтобы на основе входных технических требований предоставляемых пользователем, подобрать такие структурные схемы, которые бы наиболее полно подходят для разработки ОС по предъявленным требования.

Задача решается экспертной системой на основе продукционных правил, описанных в базе знаний. Выделяется несколько групп правил:

* Правила классификации - это правила, которые описывают классификацию разрабатываемой ОС, по входным техническим требованиям;
* Правила отбора оптических элементов - основываясь на классе ОС, устанавливают какие ОЭ элементы могут быть использованы в структурных схемах;
* Правила генерации структурных схем - правила прописывающие условия использования оптических элементов в схеме. Такие как взаимное расположение, порядок, количество элементов и т.д.

На рисунке 3.1 представлен алгоритм генерации структурных схем оптической системы.

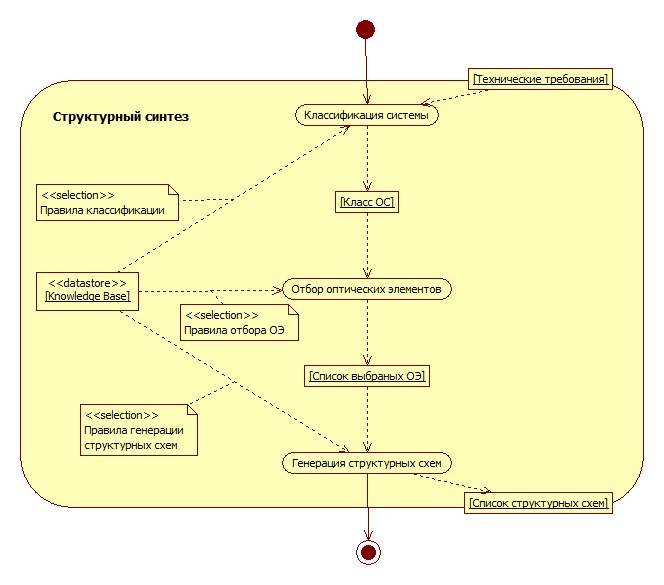


Рисунок 3. - Алгоритм структурного синтеза

### Правила классификации ОС

### Правила отбора ОЭ

### Правила генерации структурных схем

## Описание интерфейса пользователя

## Отладка алгоритмов

Описать режим отладки

## Тестирование

### Модульное тестирование

Рассказать про модульное тестирование

### Тестирование интерфейса

Придумать что-нить с Selenium

### Тестирование базы знаний

Описать тестирования БЗ

1. Drools - http://www.jboss.org/drools [↑](#footnote-ref-1)
2. Apache JackRabbit - http://jackrabbit.apache.org/ [↑](#footnote-ref-2)
3. REST - http://en.wikipedia.org/wiki/Representational\_state\_transfer [↑](#footnote-ref-3)
4. HTML - http://en.wikipedia.org/wiki/HTML [↑](#footnote-ref-4)
5. CSS - http://en.wikipedia.org/wiki/CSS [↑](#footnote-ref-5)
6. JavaScript - http://en.wikipedia.org/wiki/Javascript [↑](#footnote-ref-6)
7. Flash - http://en.wikipedia.org/wiki/Adobe\_Flash [↑](#footnote-ref-7)
8. Silverlight - http://en.wikipedia.org/wiki/Silverlight [↑](#footnote-ref-8)
9. Java-аплет - http://en.wikipedia.org/wiki/Java\_applet [↑](#footnote-ref-9)
10. HTML5 - http://en.wikipedia.org/wiki/HTML5 [↑](#footnote-ref-10)
11. W3C - http://en.wikipedia.org/wiki/W3c [↑](#footnote-ref-11)
12. Play! - http://www.playframework.org/ [↑](#footnote-ref-12)
13. Convention over configuration - http://en.wikipedia.org/wiki/Convention\_over\_configuration [↑](#footnote-ref-13)