**Министерство образования науки**

**Российской федерации**

**Рязанский государственный радиотехнический университет**

**Кафедра ВПМ**

**Учебная дисциплина:** Проектирование информационных систем

(наименование дисциплины)

**Контрольная работа**

**Вариант 4**

**Бюро по трудоустройству**

**Выполнил:**

**Студент группы:** 4041

Бирюков Николай Юрьевич

(Ф.И.О.)

**Принял:**

**Преподаватель:** Белов

Владимир Викторович

(Ф.И.О.)

**Оценка:**

**Рязань 2018**

Оглавление

[**Задание** 3](#_Toc503736382)

[**Определение рабочей области** 3](#_Toc503736383)

[**Описание работы системы** 3](#_Toc503736384)

[**Визуальное моделирование и UML** 4](#_Toc503736385)

[**Создание проекта** 6](#_Toc503736386)

[**Диаграмма вариантов** 6](#_Toc503736387)

[**Отношения между прецедентами и актерами** 10](#_Toc503736388)

[**Документирование элементов модели** 15](#_Toc503736389)

[**Диаграммы деятельности** 20](#_Toc503736390)

[**Основные элементы нотации диаграмм деятельности** 20](#_Toc503736391)

[**Создание диаграммы деятельности** 21](#_Toc503736392)

[**Диаграммы классов** 24](#_Toc503736393)

[**Основные элементы диаграмм классов** 24](#_Toc503736394)

[**Построение диаграммы классов** 26](#_Toc503736395)

[**Диаграммы последовательности** 26](#_Toc503736396)

[**Основные элементы нотации диаграмм последовательности** 27](#_Toc503736397)

[**Добавление диаграммы последовательности в модель** 29](#_Toc503736398)

[**Кооперативные диаграммы** 29](#_Toc503736399)

[**Диаграммы состояний** 30](#_Toc503736400)

[**Основные элементы диаграмм состояний** 30](#_Toc503736401)

[**Создание диаграммы состояний** 33](#_Toc503736402)

# **Задание**

Спроектируйте систему предоставления вакансий бюро по трудоустройству.

# **Определение рабочей области**

Бюро по трудоустройству занимается предоставлением вакансий по трудоустройству. Соискатель регистрируется/входит в систему публикует резюме, просматривает список вакансий и если какая либо из позиций его заинтересовала, то он может откликнуться.

Предполагаем, что потенциальный соискатель заходит на сайт по трудоустройству, где он регистрируется, либо входит в систему, затем он может просмотреть список вакансий и может откликнуться на заинтересовавшую позицию.

Для того чтобы корректно создать систему, отвечающую всем требованиям заказчика, мы должны абсолютно четко представить себе ее основные бизнес-функции и выяснить предъявляемые к системе требования. Для этого необходимо провести обследование компании и построить ее полную бизнес-модель. Поскольку наш пример является придуманным, мы не можем провести такое обследование и не имеем возможности общаться с заказчиком, то мы будем опираться на придуманное нами словесное описание системы.

# **Описание работы системы**

Каждая вакансия в каталоге описывается артикулом, размером заработной платой и кратким описанием.

Соискатель регистрируется/входит в систему, публикует резюме, затем он может открыть каталог вакансий. Каталог не содержит разделы, имеет блочную структуру, состоит из списка вакансий, размером заработной платой ценой и кратким описанием. Соискатель может откликнуться на заинтересовавшую позицию.

Данные по откликнувшемся соискателю поступают сотруднику HR, назовем его сотрудником отдела кадров, он созванивается и назначает дальнейшую встречу.

Для того чтобы ограничить масштаб задачи, мы не рассматриваем систему снабжения бюро по трудоустройству новыми вакансиями. Этим занимается другая система, назовем ее **Система HR**. Информация по откликнувшемся соискателям поступает также в **Систему для HR**.

Поскольку на протяжении всего процесса, от того как, соискатель откликнется на вакансию и до того, как HR свяжется с ним, кандидат проходит разные стадии, то будет разумно ввести понятие статуса. Сотрудники HR могут статус изменять, соискатель может проследить текущий статус. В таком случае наша система предоставляет еще одну функцию: текущий статус.

# **Визуальное моделирование и UML**

Визуальным моделированием (visual modeling) называется способ представления идей и проблем реального мира с помощью моделей.

**Модель** – это абстракция, описывающая суть сложной проблемы или структуры без акцента на несущественных деталях, тем самым делая ее более понятной.

Разработка программного обеспечения - не исключение. При построении сложной системы строятся ее абстрактные визуальные модели.

В настоящее время в области проектирования информационных систем с успехом применяется визуальное моделирование с помощью унифицированного языка моделирования UML.

**Унифицированный язык моделирования (Unified Modeling Language, UML)** является графическим языком для визуализации, специфицирования, конструирования и документирования систем, в которых большая роль принадлежит программному обеспечению.

С помощью UML можно детально описать систему, начиная разработку с концептуальной модели с ее бизнес-функциями и процессами, а также описать особенности реализации системы, такие как классы программного обеспечения системы, схему базы данных. Используя UML, мы также можем разрабатывать сложные системы быстро и качественно.

Как язык графического визуального моделирования UML имеет свою нотацию – принятые обозначения. Нотация обеспечивает семантику языка, является способом унификации обозначений визуального моделирования, обеспечивает всестороннее представление системы, которое сравнительно легко и свободно воспринимается человеком. Последняя версия нотации UML 2.4.1 опубликована в августе 2011 года.

Моделирование с помощью UML осуществляется поэтапным построением ряда диаграмм, каждая из которых отражает какую-то часть или сторону системы либо ее замысла.

**Диаграмма** – это графическое представление множества элементов. Обычно диаграмма изображается в виде графа с вершинами (сущностями) и ребрами (отношениями). Диаграммы подчиняются нотации UML и изображаются в соответствии с ней.

Основные диаграммы UML:

* вариантов использования (use case diagram);
* классов (class diagram);
* кооперации (collaboration diagram);
* последовательности (sequence diagram);
* состояний (statechart diagram);
* деятельности (activity diagram);
* компонентов (component diagram);
* развертывания (deployment diagram).

Построения этих диаграмм достаточно для полного моделирования системы.

В данном пособии рассматриваются основные элементы нотации диаграмм и принципы их построения.

# **Создание проекта**

Создадим новый проект в PowerDesigner. Наша модель будет иметь четыре представления: Use Case, Logical, Component и Deployment. Данный подход по структуре представлений на наш взгляд наиболее соответствует методологии Rational Unified Procces (RUP), которая поддерживает итеративный процесс разработки информационных систем. Настоящая контрольная работа не претендует на полное соответствие этапам RUP: целью моделирования нашей системы является знакомство с нотацией UML и изучение приемов работы в CASE-средстве проектирования и моделирования UML в PowerDesigner. Мы выбираем подход Rational Approach для удобства дальнейшей работы.

# **Диаграмма вариантов**

Поведение системы (т.е. функциональность, которую она обеспечивает) описывают с помощью функциональной модели, которая отображает системные прецеденты (use cases, случаи использования), системное окружение (действующих лиц, актеров, actors) и связи между ними (use cases diagrams).

**Диаграмма вариантов использования (диаграмма прецедентов, use case diagram)** – это диаграмма, на которой изображаются отношения между актерами и вариантами использования.

С помощью этой диаграммы можно:

* Определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
* Сформулировать общие требования к функциональному поведению проектируемой системы;
* Разработать исходную **концептуальную модель** системы для ее последующей детализации в форме логических и физических моделей;
* Подготовить **исходную документацию** для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Диаграмма вариантов использования (прецедентов) представляет собой граф, в вершинах которого расположены актеры или прецеденты, связи между вершинами – это разного вида отношения.

**Актером (действующее лицо, actor)** называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне.

Это может быть человек, техническое устройство, программа или любая другая система, которая служит источником воздействия на моделируемую систему так, как определит разработчик. На диаграммах вариантов использования актер изображается в виде человечка, под которым записывается его имя.

**Вариант использования (прецедент, use case)** – описание множества последовательности действий (включая варианты), выполняемых системой для того, чтобы актер мог получить определенный результат.

Каждый вариант использования должен быть независимым в том смысле, что если он всегда выполняется совместно с другим вариантом использования, то, скорее всего, это один прецедент, а не два, либо они связаны отношением включения или расширения. Как следует из определения, прецедент (или вариант использования) должен обладать результирующей ценностью для актера, актер должен получить некоторый результат исполнения прецедента. Скорее всего, после исполнения прецедента в системе произойдут некоторые изменения: появятся новые данные, изменится поведение. Каждый вариант использования должен исполняться от начала до конца.

Прецедент описывает**, что делает** система, но никак **не уточняет, как** она это делает. Заметим, что диаграмма прецедентов не отображает последовательность, в которой будут выполняться варианты использования. На диаграмме прецедент изображается в виде эллипса. Имя прецедента может состоять из нескольких слов и знаков препинания (за исключением двоеточия), как правило, имя выбирают в виде словосочетания или глагольного выражения.

Одним из наиболее важных (и дорогостоящих) этапов проектирования информационных систем является этап определения требований к системе. Если требования заказчика информационной системы разработчиками будут определены не корректно, то в итоге заказчик может получить совсем не ту систему, которую он ожидал. Моделирование прецедентов и актеров помогает нам лучше понять требования, предъявляемые к системе, и согласовать их с заказчиком с помощью демонстрации и обсуждения диаграммы прецедентов. Прецеденты и актеры – это отражение требований к системе, они показывают, кто и для чего будет использовать будущую систему.

Итак, определим актеров и прецеденты системы бюро трудоустройства.

Напомним, что соискатель регистрируется/входит в систему, публикует резюме, затем он откликается на вакансию. Статус отклика, может быть следующий: ожидает действий от HR, приглашен на собеседовании, отказ, принят. Статус отклика изменяется автоматически либо сотрудником HR. Соискатель может узнать статус отклика в личном кабинете.

Система не занимается приглашением кандидата на собеседование. Этим занимается другая система, назовем ее **Система HR**.

Таким образом, с нашей системой взаимодействуют **соискатель**, **сотрудник HR** и внешняя система **Система HR**. С нашей системой взаимодействуют сотрудник HR, который проверяет откликнувшихся кандидатов и приглашает их на собеседование. Введем обобщающее сотрудников действующее лицо – **Сотрудник**.

**Соискатель, Сотрудник, Система HR**.

Соискатель должен иметь возможность зарегистрироваться для него определим прецедент **Регистрация**

Соискатель должен иметь возможность войти в систему, для него определим прецедент **Авторизация**

Соискатель должен иметь возможность добавлять, изменять резюме, для него определим прецедент **Управление резюме**.

Соискатель использует нашу систему для того, чтобы найти интересующую вакансию и откликнуться на нее. В конечном итоге результат использования системы соискателем будет получен, если он выполнил все эти действия от начала до конца. Поэтому не будем разделять отклик соискателя на несколько прецедентов, а выделим только один: **Отклик на вакансию**.

Соискатель, откликнувшийся на вакансию может в дальнейшем узнать статус своего заказа, это тоже случай использования системы, назовем его **Информация о статусе отклика**.

Сотрудник должен изменять статус отклика, для него определим прецедент **Управление статусом отклика**.

**Система HR** должна получать информацию о сделанных откликах, для нее также должен быть доступен прецедент Получение информации об отклике.

Итак, прецеденты системы бюро по трудоустройству **Регистрация**, **Авторизация, Управление резюме, Отклик на вакансию, Информация о статусе отклика**, **Управление статусом отклика**.

# **Отношения между прецедентами и актерами**

Связи и взаимоотношения, существующие между элементами модели, в UML описываются с помощью отношений, изображаемых на диаграммах.

**Отношение (relationship)** – это семантическая связь между отдельными элементами модели.

Между актерами и прецедентами диаграммы вариантов использования могут существовать разного рода отношения, показывающие, что экземпляры действующих лиц и вариантов использования взаимодействуют друг с другом. Действующие лица могут взаимодействовать с несколькими прецедентами и между собой, равно как и прецеденты могут быть связаны между собой особого типа отношениями.

В основном на диаграммах прецедентов используются следующие типы отношений:

* Ассоциации (association relationship);
* Включения (include relationship);
* Расширения (extend relationship);
* Обобщения (generalization relationship).

**Ассоциация** – это структурное отношение, показывающее, что объект неким образом связан с другим объектом.

Отношение этого типа используется не только на диаграммах прецедентов, но и на других диаграммах. Если между элементами модели показано отношение ассоциации, то это означает, что между ними существует семантическая связь. Ассоциативное отношение может быть направленным. В этом случае направление связи показывает, кто является инициатором. Если отношение направленно от актера к прецеденту, то это означает, что актер инициирует выполнение прецедента.

Между прецедентами также возможны взаимоотношения, которые описываются отношениями двух типов: включения и расширения (дополнения).

**Включение (include)** говорит о том, что исходный прецедент явным образом включает в себя поведение целевого.

Другими словами, включение создается, когда один прецедент использует другой. При этом исполнение базового прецедента невозможно без исполнения используемого. Изображается включение в виде пунктирной стрелки с надписью <<include>>, которая направлена от базового элемента к используемому.

**Расширение (extend)** показывает, что целевой прецедент расширяет поведение исходного.

Используемый прецедент выполняется не всегда вместе с базовым, а только при выполнении дополнительных условий, таким образом, расширяя функциональность базового элемента. Изображается расширение пунктирной стрелкой с надписью <<extend>>, направленной от используемого варианта использования к базовому.

Обозначения отношений <<include>> и <<extend>> есть не что иное, как обозначения стереотипов, которые широко используются в UML для создания новых элементов модели путем расширения функциональности базовых элементов.

**Стереотип (Stereotype)** – это механизм, позволяющий категоризировать элементы модели.

С помощью стереотипов мы можем создавать своего рода подтипы типов. Это позволяет UML иметь минимальный набор элементов, которые могут быть дополнены при необходимости для создания связующих базовых элементов в системе. В UML стереотип обозначается именем, которое записывается в двойных угловых скобках: <<имя стереотипа>>.

В UML мы можем создавать собственные стереотипы на основе уже имеющихся типов, но также существуют и стандартные, заранее определенные стереотипы нотации UML. Так, отношение зависимости (о котором мы еще будем говорить) расширяется для прецедентов и актеров с помощью двух стереотипов <<include>> и <<extend>>.

**Ассоциация** – это коммуникативное отношение, которое соответствует стереотипу <<communicate>>, который, впрочем, всегда опускается.

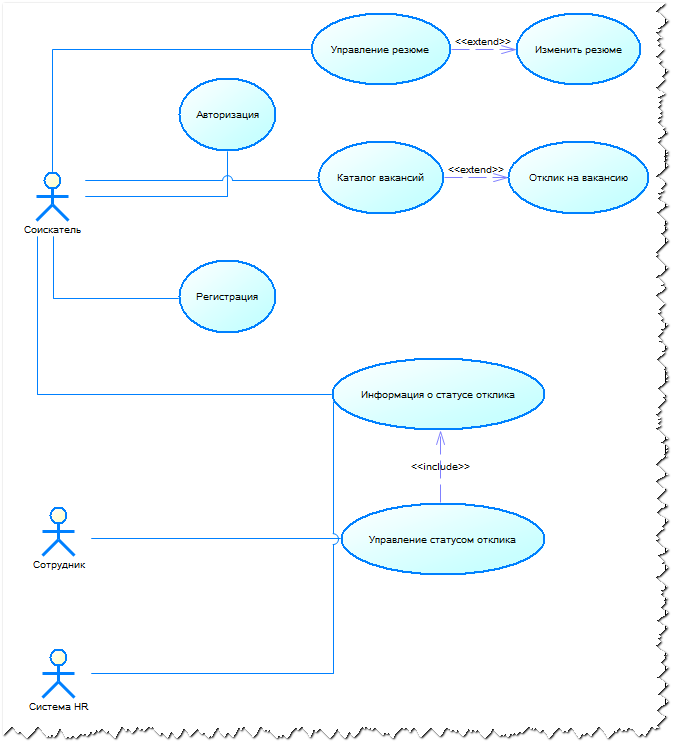
Два и более актера могут иметь общие свойства, т.е. взаимодействовать с одним и тем же множеством вариантов использования одинаковым образом. Такая общность свойств и поведения представляется в виде отношения обобщения с другим, возможно, абстрактным актером, который моделирует соответствующую общность ролей.

**Обобщение (Generalization)** – это отношение между общей сущностью и ее конкретным воплощением.

На диаграммах обобщение обозначается стрелкой с не закрашенным треугольником на конце, направленной от частного элемента к общему.

Актеры, прецеденты и отношения – это основные элементы нотации диаграмм вариантов использования. Диаграмма вариантов использования помогает отобразить основные требования к моделируемой системе и обеспечить взаимопонимание функциональности системы между разработчиком и заказчиком. Можно построить одну, главную диаграмму прецедентов, на которой будут отражены границы системы (актеры) и ее основная функциональность (прецеденты). Для более подробного представления системы допускается построение вспомогательных диаграмм прецедентов.

Итак, для системы бюро по трудоустройству мы определили актеров **Соискатель, Сотрудник, Система HR** и **прецеденты Регистрация, Авторизация, Управление резюме, Отклик на вакансию, Информация о статусе отклика, Управление статусом отклика.** Построим основную диаграмму прецедентов



Для актера Покупатель и прецедента **Регистрация, Авторизация, Управление резюме, Отклик на вакансию** установили отношение направленной ассоциации. Все эти прецеденты **Покупателем**. **Сотрудник** имеет возможность управлять статусом отклика, при этом он непременно участвует в прецеденте **Информация о статусе отклика**. Направленную ассоциацию от **Информация о статусе отклика** к актеру **Система HR** можно понимать как автоматическую передачу данных из моделируемой системы в систему снабжения вакансиями **Система HR**

В модель нужно включить краткое описание каждого актера или прецедента, делается это для того, чтобы между разработчиком и заказчиком системы не оставалось «белых пятен» и расхождений в понимании функциональности системы и ролей взаимодействующих с ней актеров. Для каждого актера описывается роль, которую он играет в системе, а для каждого прецедента – его назначение и функциональность. Также можно уточнить, каким актером запускается прецедент.

# **Документирование элементов модели**

Для актеров и прецедентов системы бюро по трудоустройству сделаем краткое описание в программе PowerDesign.

**Соискатель** – это человек, который может откликнуться на вакансию, с помощью проектируемой системы.

**Сотрудник** – это все сотрудники системы бюро по трудоустройству, которые могут получать информацию об откликах на размещению вакансию через **Систему HR** и изменять статус отклика в системе в зависимости от того шага, на котором находится обработка данного запроса.

**Систему HR** – это внешняя система, которая получает информацию о сделанных откликах, а также через эту систему происходит управление вакансиями.

**Регистрация** – этот прецедент запускается соискателем для того, чтобы зарегистрироваться в бюро по трудоустройству.

**Авторизация** – этот прецедент запускается соискателем для того, чтобы авторизоваться в бюро по трудоустройству.

**Управление резюме** – этот прецедент используется соискателем для просмотра и изменения резюме.

**Отклик на вакансию** – этот прецедент запускается соискателем для того, чтобы откликнуться на вакансию в системе бюро по трудоустройству

**Управление статусом отклика** – этот прецедент используется сотрудниками HR для изменения статуса заказа в процессе его обработки.

**Информации о статусе отклика** – прецедент используется всеми актерами для просмотра информации о статусе отклика.

**Поток событий**

Одним из требований языка UML является самодостаточность диаграмм для представления информации о моделях проектируемых систем. Однако, как уже отмечалось выше, диаграммы вариантов использования описывают то, что делает система, без уточнения того, как она это делает.

Для реального описания системы потребуются более специфические данные, которые отражены в потоке событий. Потоки событий уточняют или детализируют последовательность действий, совершаемых системой при выполнении ее вариантов использования, а также описывают логику переходов через варианты использования.

**Поток событий** – это определенная последовательность действий, которая описывает действия актеров и поведение моделируемой системы в форме обычного текста.

Потоки событий – это текстовые описания пошагового выполнения прецедентов, они понятны не только разработчику, но и стороннему читателю. Их задача – еще больше детализировать описание функциональности системы до того, как разработчики приступят к написанию программного кода, и устранить возможное недопонимание требуемой функциональности, как можно больше сблизить представления разработчика о системе и заказчика.

Потоки событий бывают трех типов: основной, альтернативный и поток ошибок.

Основной (главный) поток описывает наилучший сценарий либо наиболее используемый путь исполнения прецедента.

Альтернативный поток специфицирует отклонения от основного потока, которые не рассматриваются как ошибочные.

Поток ошибок рассматривается как отклонение от альтернативного или основного, которое порождает условия формирования ошибки.

---------------------------------------------------------

Опишем потоки событий прецедента Регистрации.

**Основной поток событий.**

1. Соискатель открывает страницу регистрации.
2. Соискатель вводит номер телефона и пароль.
3. Система подтверждает введенный номер телефона и пароль.

Покупатель просматривает каталог и запускает поток «добавление товара в корзину»

4. Система открывает корзину.

5. Покупатель нажимает кнопку

**Основной поток событий.**

1. Прецедент начинается с выбора покупателем режима показа каталога.
2. Система открывает каталог.
3. Покупатель выбирает режим показа корзины.

Покупатель просматривает каталог и запускает поток «добавление товара в корзину»

4. Система открывает корзину.

5. Покупатель нажимает кнопку «оформить заказ».

А2. Покупатель просматривает корзину и запускает поток «изменение корзины».

А3. Покупатель решает вернуться в каталог.

6. Система переходит к первому шагу оформления заказа: запрашивает у покупателя личные данные и телефон.

7. Покупатель вводит личные данные и телефон.

8. Система переходит ко второму шагу оформления заказа: показывает содержимое заказа и просит подтвердить заказ.

9. Покупатель подтверждает заказ.

А4. Покупатель возвращается в корзину.

10. Система переходит к третьему шагу оформления заказа: запрашивает тип кредитной карты, ее номер, секретный код, имя владельца и дату завершения срока действия.

11. Покупатель вводит тип кредитной карты, ее номер, секретный код, имя владельца и дату завершения срока действия.

12. Система переходит к четвертому шагу оформления заказа: подтверждает оплату.

А5. Счет пользователя не найден.

А6. Недостаточно денег на счете.

Е1. Платежная система недоступна.

13. Система присваивает заказу номер и отправляет его вместе с подтверждением заказа на электронный адрес покупателя.

29

14. Вариант использования завершается.

Альтернативные потоки.

А1. добавление товара в корзину

1. Покупатель выбирает размер.

2. Покупатель выбирает количество.

3. Покупатель нажимает кнопку Добавить в корзину

4. Система помещает выбранный товар в корзину.

А7. Покупатель не выбрал размер.

А8. Покупатель не выбрал количество.

5. Система выводит сообщение о том, что товар добавлен в корзину.

6. Поток возвращается на второй этап основного потока.

А2. Изменение корзины

1. Покупатель нажимает кнопку «Удалить» напротив одного выбранного товара.

А9. Покупатель изменяет количество позиций одного выбранного товара.

2. Система удаляет товар из корзины.

3. Поток возвращается к этапу 4 основного потока.

А3. Покупатель решает вернуться в каталог

1. Поток возвращается к этапу 2 основного потока.

А4. Покупатель возвращается в корзину

1. Поток возвращается к этапу 4 основного потока.

А5. Счет пользователя не найден

1. Система выводит сообщение о том, что счет пользователя не обнаружен.

2. Поток возвращается к этапу 11 основного потока.

А6. Недостаточно денег на счете

1. Система выводит сообщение о том, что на счете пользователя недостаточно денег для совершения операции.

2. Поток возвращается к этапу 11 основного потока.

А7. Покупатель не выбрал размер

1. Система выводит сообщение о необходимости выбора размера.

2. Поток возвращается на 1 этап потока А1.

А8. Покупатель не выбрал количество

1. Система выводит сообщение о необходимости выбора количества товара.

30

2. Поток возвращается на 2 этап потока А1.

А9. Покупатель изменяет количество позиций одного выбранного товара

1. Покупатель увеличивает количество товара.

2. Система обновляет корзину.

А10. Недостаточно товара в наличии.

3. Поток возвращается к этапу 4 основного потока.

А10. Недостаточно товара в наличии

1. Система выводит сообщение о том, сколько позиций может заказать покупатель.

2. Поток возвращается к этапу 4 основного потока.

Потоки ошибок.

Е1. Платежная система недоступна

1. Система выводит сообщение о недоступности платежной системы.

2. Поток возвращается к этапу 11 основного потока.

5.1 Добавление потока событий к модели в StarUML

---------------------------------------------------------

# **Диаграммы деятельности**

Диаграммы деятельности обеспечивают еще один способ моделирования потока событий. С помощью текстового описания можно рассказать о потоке, но трудно будет понять логику событий в сложных и запутанных потоках с множеством альтернативных ветвей.

Диаграммы деятельности создаются также на разных этапах жизненного цикла системы для отражения последовательности выполнения операций.

# **Основные элементы нотации диаграмм деятельности**

Рассмотрим основные элементы нотации диаграмм деятельностей. На них иллюстрируются деятельности, переходы между ними, элементы выбора и синхронизации.

**Деятельностью** называется исполнение определенного поведения в потоке управления системы. В UML деятельность изображается в виде скругленного прямоугольника с текстовым описанием внутри.

Деятельности, помещенные между двумя жирными линиями на диаграмме деятельности, исполняются синхронно, одновременно.

Два состояния на диаграмме деятельности - начальное и конечное - определяют продолжительность потока. Начальное состояние обязательно должно быть отмечено на диаграмме, оно определяет начало потока. Конечных состояний может быть несколько или не одного. Оно определяет точку завершения потока. Конечных состояний может быть несколько, но начальное должно быть только одно. Начальное состояние изображается жирной точкой, а конечное – жирной точкой в окружности.

При моделировании управляющих потоков системы часто бывает необходимо показать места их разделения на основе условного выбора. Выбор на диаграмме показывается ромбом, помещенным на переходе. Ограничительные условия, от которых зависит выбор направления перехода, помещаются обычно над ромбом. В нотации UML условия записываются в квадратных скобках.

**Синхронизация** – это способ показать, что две или более ветвей потока выполняются параллельно.

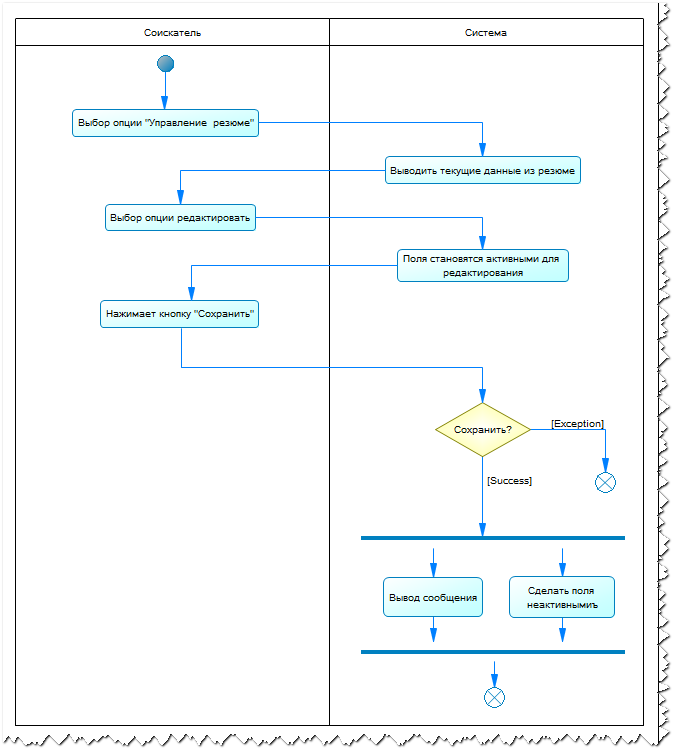
Деятельности, помещенные между двумя жирными линиями на диаграмме деятельности, исполняются синхронно, одновременно.

**Секции** делят диаграмму деятельности на несколько участков. Это нужно для того, чтобы показать, кто отвечает за выполнение деятельности и в каком порядке.

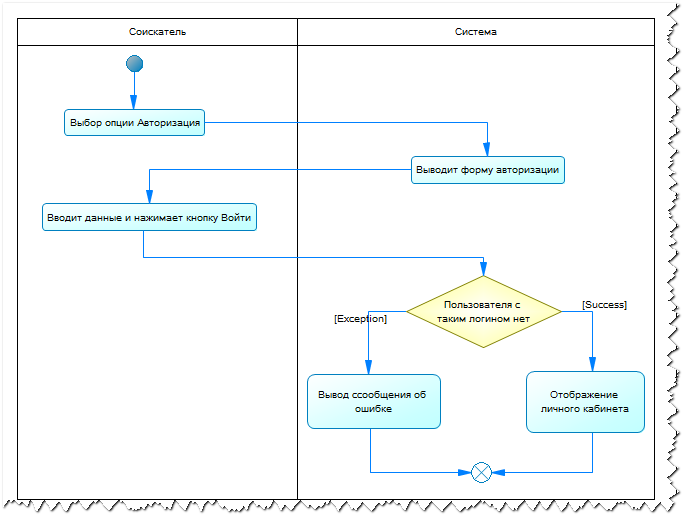
# **Создание диаграммы деятельности**

Построим диаграмму деятельности для прецедента **Регистрация** актера Соискатель.

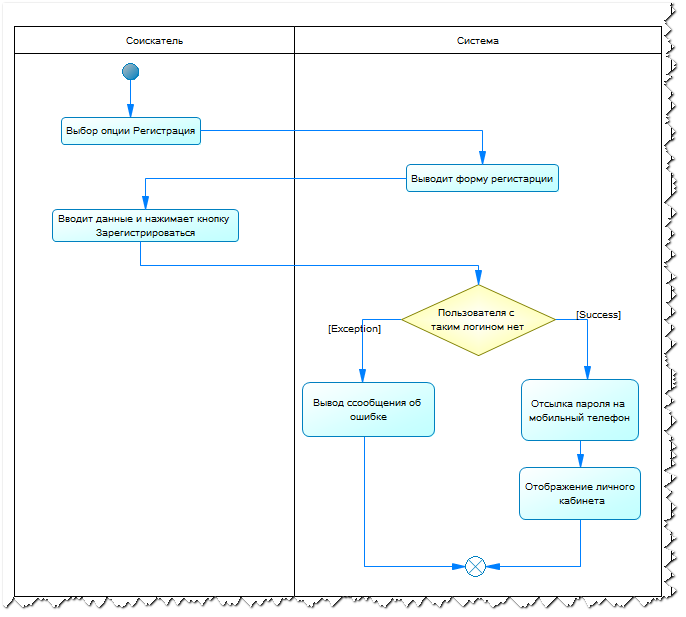
Построим диаграмму деятельности для прецедента **Управление резюме** актера Соискатель. Управление резюме включает просмотр редактирование своих личных данных (ФИО, дата рождения, предыдущие и текущие места работы). Управление резюме начинается из личного кабинета соискателя он выбирает опцию «**Управление резюме**»



***Авторизация***



***Регистрация***



# **Диаграммы классов**

Диаграмма классов является частью логической модели системы и представляет статическую картину системы.

Для каждой системы строится не одна, а несколько диаграмм классов: возможно, что для каждого прецедента или сценария своя. На одних показывают подмножества классов, объединенные в пакеты, и отношения между ними, на других – отображают те же подмножества, но с атрибутами и операциями классов. Для представления системы разрабатывается столько диаграмм классов, сколько потребуется.

# **Основные элементы диаграмм классов**

Дадим некоторые определения и опишем основные элементы нотации диаграмм классов.

**Объект** – это некоторая сущность реального мира или концептуальная (абстрактная) сущность.

Объект имеет четко определенные границы и значение для системы и характеризуется состоянием, поведением и индивидуальностью.

**Состояние объекта** – это одно из условий, в котором он может находиться. Состояние обычно изменяется со временем и характеризуется набором свойств, которые называются атрибутами.

**Поведение** определяет, как объект реагирует на запросы других объектов и что может делать сам объект. Поведение характеризуется операциями объекта.

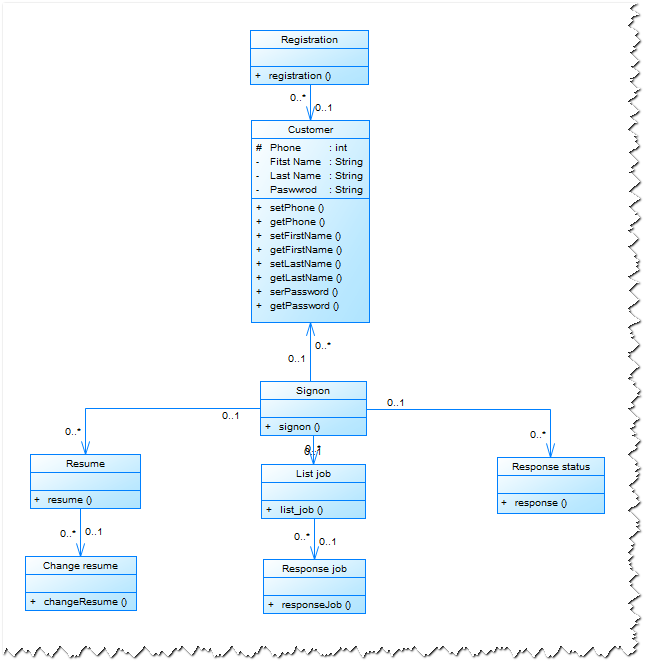
**Индивидуальность** означает, что каждый объект уникален, даже если его состояние идентично состоянию другого объекта.

**Класс** – это описание группы объектов с общими свойствами (атрибутами), поведением (операциями), отношениями с другими объектами и семантикой.

Каждый класс является шаблоном для создания объекта. А каждый объект – это экземпляр класса. Важно помнить, что каждый объект может быть экземпляром только одного класса!

В нотации UML классы и объекты изображаются в виде прямоугольников. Прямоугольник класса всегда делится на три секции (раздела), имя класса помещается в первую секцию, каждое слово в названии класса принято писать с большой буквы. Во второй и третьей секциях могут указываться атрибуты и операции класса соответственно, эти секции могут быть пустыми. Названия классов выбираются в соответствии с понятиями предметной области. Это должно быть существительное или словосочетание в единственном числе, наиболее точно характеризующее предмет. Класс должен описывать только одну сущность.

# **Построение диаграммы классов**



# **Диаграммы последовательности**

Как правило, поток событий описывает не одну последовательность действий, а несколько возможных, это отражается наличием главного потока событий и альтернативных потоков. Чаще всего невозможно описать прецедент с помощью только одной последовательности действий.

**Сценарий (Scenario)** – это некоторая последовательность действий, иллюстрирующая поведение системы.

**Сценарий** – это экземпляр потока событий. Он представляет собой одиночный проход по потоку событий для прецедента. Для графического отображения сценария используются диаграммы последовательностей.

Диаграмма последовательности действий отображает взаимодействие объектов, упорядоченное по времени.

# **Основные элементы нотации диаграмм последовательности**

На диаграммах последовательности изображаются объекты, классы и последовательность сообщений, которыми обмениваются объекты в ходе выполнения сценария

На диаграмме последовательностей могут также изображаться экземпляры действующих лиц. Для того чтобы поместить действующее лицо на диаграмму, нужно найти его в навигаторе модели справа и перетащить на поле диаграммы последовательностей

Действующие лица, присутствующие на диаграммах взаимодействия, выделяются из потока событий как сущности, запускающие процессы. На одной диаграмме их может быть несколько.

Для того чтобы поместить экземпляр уже созданного ранее на диаграмме прецедентов действующего лица на диаграмму взаимодействия, просто перетащите его с навигатора модели на рабочее поле диаграммы.

Каждый объект или действующее лицо на диаграмме последовательностей имеет свою линию жизни, которая обозначается пунктиром.

**Линия жизни объекта (object lifeline)** – вертикальная пунктирная линия на диаграмме последовательности, которая представляет существование объекта в течение определенного периода времени.

**Фокус управления (активность, focus of control)** – специальный символ на диаграмме последовательности, указывающий период времени, в течение которого объект выполняет некоторое действие, находясь в активном состоянии. Фокус управления изображается тонким прямоугольником, расположенным на линии жизни.

Объекты и действующие лица на диаграммах последовательности обмениваются сообщениями. Сообщения обозначаются стрелками, идущими от отправителя к получателю.

**Сообщение (message)** – спецификация передачи информации от одного элемента модели к другому с ожиданием выполнения определенных действий со стороны принимающего элемента.

Для сообщений на диаграммах последовательностей, как и для других элементов модели, доступен ряд спецификаций.

Во-первых, у каждого сообщения должно быть имя, соответствующее его цели.

Во-вторых, сообщения на диаграммах последовательностей можно соотнести с операциями, определенными для классов. Если от одного объекта к другому направлено сообщение, то это означает, что объект-источник вызывает операцию объекта-приемника. Объект не может вызвать произвольную операцию: она должна быть доступна этому объекту.

В особых случаях сообщение не становится операцией: например, ввод логина и пароля подразумевает их печать в соответствующих полях, и сообщение будет реализовано в виде поля ввода в окне программы.

Процедура создания операций из сообщений будет описана ниже.

В-третьих, мы можем для каждого сообщения установить тип синхронизации. Каждому типу соответствует его обозначение.

**Вызов операции (процедуры) (call)** вызывает операцию того объекта, к которому направлено. Объект может вызвать свою операцию. Тогда стрелка начинается и заканчивается на линии жизни одного и того же объекта, такое сообщение называется рефлексивным.

Синхронное сообщение обозначается стрелкой с закрашенным наконечником.

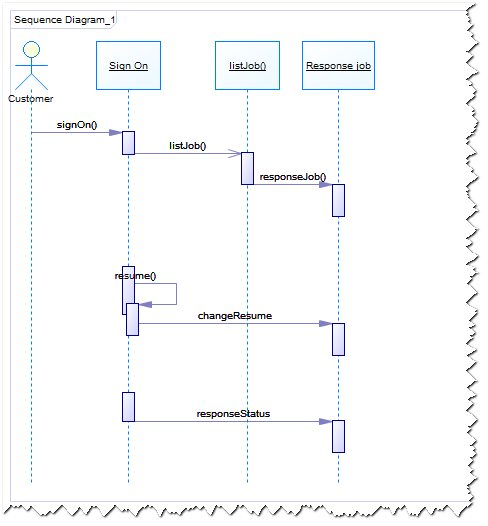
**Асинхронное сообщение (send)** посылает объекту сигнал. При этом источник не ждет отклика приемника или подтверждения получения, а продолжает свою работу. Обозначается нежирной стрелкой.

**Ответное сообщение (return)** возвращает значение из процедуры тому объекту, к которому направлено. Обозначается пунктирной стрелкой.

**Создать объект (create)** – создает новый объект. Обозначается стрелкой со стереотипом <<create>>.

**Уничтожить объект (destroy)** – удаляет объект. Объект может уничтожить сам себя. Обозначается стрелкой со стереотипом <<destroy>>. При уничтожении объекта на его линии жизни появляется символ разрушения, который обозначается крестом.

# **Добавление диаграммы последовательности в модель**



# **Кооперативные диаграммы**

**Диаграмма кооперации** – это альтернативный способ изображения сценария варианта использования. Этот тип диаграмм заостряет внимание на связях между объектами, отображая обмен данными в системе. А диаграммы последовательности отображают взаимодействие объектов во времени, поэтому ее следует читать сверху вниз и слева направо.

Диаграммы кооперации содержат все те же элементы, что и диаграммы последовательности: объекты, действующие лица, связи между ними и сообщения, которыми они обмениваются, но они уже не упорядочены во времени.

# **Диаграммы состояний**

Для представления взаимодействий объектов в системе используют диаграммы последовательности и кооперации. Прецеденты и сценарии описывают поведение системы, но если нам важна динамика класса, изменение состояний отдельного объекта, то для этой цели полезно построить диаграммы состояний.

Диаграмма состояний показывает положение одиночного объекта, события и сообщения, которые вызывают переход из одного состояния в другое, и действия, являющиеся результатом смены состояния. Диаграмма состояний показывает объект с момента его создания и до его уничтожения.

Диаграмму состояний строят не для каждого класса в системе, а только для классов с динамическим поведением, которые отсылают и принимают много сообщений, изменяют свое состояние. Программный код из диаграмм состояний не генерируются, но они важны для понимания динамики поведения класса, дают возможность понять логику изменений перед кодированием.

# **Основные элементы диаграмм состояний**

Диаграмму состояний часто рассматривают в контексте конечного автомата. Тогда можем сказать, что диаграмма состояний (Statechart diagram) показывает автомат, фокусируя внимание на потоке управления от состояния к состоянию.

**Автомат (State machine)** – это описание последовательности состояний, через которые проходит объект на протяжении всего жизненного цикла, реагируя на события, - в том числе описание реакций на эти события.

**Состояние (State)** – это ситуация в жизни объекта, на протяжении которой он удовлетворяет некоторому условию, осуществляет определенную деятельность или ожидает какого-то события.

Для поиска состояний класса можно просматривать атрибуты этого класса. Хорошим индикатором состояний является такой атрибут класса как «статус».

Диаграмма состояний изображается в виде графа с вершинами и ребрами. Состояние на диаграмме изображается прямоугольником со скругленными вершинами. Под именем состояния могут размещаться действия.

Кроме обычных состояний на диаграмме состояний могут размещаться псевдосостояния.

**Псевдосостояние (pseudo-state)** **–** вершина в конечном автомате, которая имеет форму состояния, но не обладает поведением.

Примерами псевдосостояний, которые определены в языке UML, являются начальное и конечное состояния.

**Начальное состояние (start state)** – разновидность псевдосостояния, обозначающее начало выполнения процесса изменения состояний конечного автомата или нахождения моделируемого объекта в составном состоянии.

В этом состоянии находится объект по умолчанию в начальный момент времени. Оно служит для указания на диаграмме состояний графической области, от которой начинается процесс изменения состояний. Графически начальное состояние в языке UML обозначается в виде закрашенного кружка, из которого может только выходить стрелка-переход.

**Конечное состояние (final state)** **–** разновидность псевдосостояния, обозначающее прекращение процесса изменения состояний конечного автомата.

Объект находится в конечном состоянии непосредственно перед уничтожением. Конечных состояний может быть несколько.

Находясь в каком-либо состоянии, объект может выполнять определенные действия, с состоянием можно связать такие действия как входное и выходное действия и внутренняя или просто деятельность.

**Действие Действие (Action)** – это одиночное вычисление, которое приводит к смене состояния или возврату значения.

В общем случае действие имеет следующий формат:

**<метка действия / выражение действия>**

**Входное действие (entry action)** – действие, которое выполняется в момент перехода в данное состояние. Обозначается с помощью ключевого слова - метки действия entry, которое указывает на то, что следующее за ней выражение действия должно быть выполнено в момент входа в данное состояние.

**Действие выхода (exit action)** – действие, производимое при выходе из данного состояния.

Обозначается с помощью ключевого слова - метки действия exit, которое указывает на то, что следующее за ней выражение действия должно быть выполнено в момент выхода из данного состояния.

**Внутренняя деятельность (do activity)** – выполнение объектом операций или процедур, которые требуют определенного времени.

Другими словами, деятельность – это поведение, которое реализуется объектом, когда он находится в данном состоянии. Деятельность – это прерываемое действие: оно может выполняться до конца или быть прервано переходом в другое состояние.

Обозначается с помощью ключевого слова - метки деятельности do, которое специфицирует так называемую "ду-деятельность", выполняемую в течение всего времени, пока объект находится в данном состоянии, или до тех пор, пока не будет прервано внешним событием. При нормальном завершении внутренней деятельности генерируется соответствующее событие.

# **Создание диаграммы состояний**

Отразим на диаграмме переход заказа между состояниями

