# Лабораторная работа № 5

**Тема**: *разработка диаграмм вариантов использования*

**Цель**: *изучить принципы построения диаграмм вариантов использования UML, освоить построение диаграмм вариантов использования в CASE-средстве Enterprise Architect 8.0, освоить методику оценки трудоемкости разработки программного обеспечения на основе вариантов использования.*

## Краткая теория

Диаграмма вариантов использования, как правило, является исходной моделью, с которой начинается процесс моделирования на языке UML. Она описывает функциональное назначение системы в самом общем виде с точки зрения всех ее пользователей и заинтересованных лиц. Диаграмма вариантов использования пре6дставляет собой диаграмму, на которой изображаются варианты использования проектируемой системы, как правило, заключенные в границу субъекта и внешние актеры, а также определенные взаимоотношения между актерами и вариантами использования. Данная диаграмма предназначена для достижения следующих целей:

- определения общих границ функциональности проектируемой системы в контексте моделируемой предметной области;
- предъявления требований к функциональному поведению проектируемой системы в форме вариантов использования;
- разработке исходной концептуальной модели системы для ее последующей детализации в форме логических и физических моделей;
- подготовке исходной документации для взаимодействия разработчиков системы с ее заказчиками.

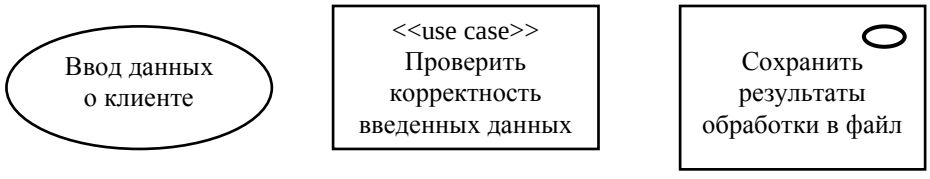
Таким образом, основным назначением диаграммы вариантов использования является спецификация функциональных требований к проектируемой системе. Так как требования выступают в качестве исходных данных для разработки системы, то визуализация их в форме диаграммы повышает наглядность представления и позволяет целенаправленно управлять процессом разработки других моделей на языке UML. Кроме того, данная диаграмма может использоваться как практическое средство для итеративного уточнения требований к проектируемой системе.

Если проектируемая система является достаточно сложной или имеет большие масштабы, то для упрощения ее модельного представления она может быть декомпозирована или разбита на отдельные подсистемы, которые рассматриваются как изолированные или обособленные части исходной системы. В этом случае для каждой из подсистем можно разрабатывать отдельную диаграмму вариантов использования.

Основными элементами (предметами языка UML), отображаемыми на диаграмме вариантов использования, являются: вариант использования, актер, системная граница, примечание.

Вариант использования представляет собой общую спецификацию совокупности выполняемых системой действий с целью предоставления некоторого наблюдаемого результата, имеющего значение для одного или нескольких актеров. Проще говоря, вариант использования представляет собой законченный фрагмент поведения системы с точки зрения тех или иных заинтересованных лиц без указания технических или физических особенностей его реализации. Описание этого фрагмента поведения называется сценарием. Сценарии могут задаваться в нескольких формах: в виде обычного неструктурированного текста, в виде упорядоченного списка действий или в виде текста не некотором формализованном языке.

Графически вариант использования в нотации языка UML обозначается эллипсом, внутри которого (или ниже) обязательно записывается имя в форме строки текста. Имя рекомендуется формировать в виде законченного предложения, начинающегося с глагола или отглагольного существительного. Альтернативным способом обозначения варианта использования является обозначение в виде обычного классификатора (прямоугольник), у которого либо указывается стереотип <<use case>> в верхней части, либо пиктограмма в виде эллипса в правом верхнем углу. Примеры обозначения варианта использования приведены на рисунке 1.

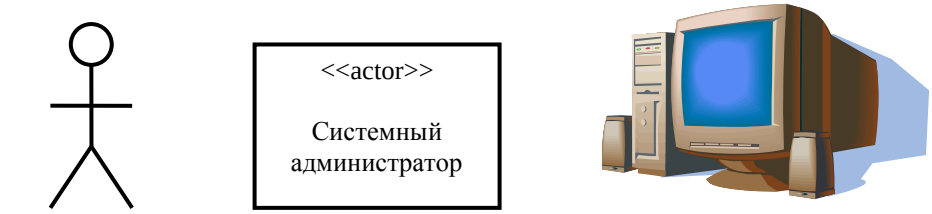



а) б) в)

Рисунок 1 – Графическое обозначение варианта использования
а) – стандартное обозначение; б) – со стереотипом <<use case>>; в) – с пиктограммой эллипса

Актер представляет собой любую внешнюю по отношению к проектируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач. Формально в контексте языка UML каждый актер специфицирует некоторую роль, которую играет пользователь или любая другая система, взаимодействующая с проектируемой системой.

Графически актер в нотации языка UML обозначается в форме «проволочного человечка», под которым записывается имя актера. Именем актера должно быть имя существительное (возможно с поясняющим прилагательным). Также актер может отображаться в виде классификатора со стереотипом <<actor>> или в форме предложенной разработчиком пиктограммы (рисунка). Примеры изображения актеров приведены на рисунке 2.




Пользователь Удаленный пользователь

а) б) в)

Рисунок 2 – Графическое обозначение актеров
а) – в форме «проволочного человечка»;
б) – в форме классификатора со стереотипом <<actor>>; в) – в форме пиктограммы

Также на диаграмме вариантов использования может присутствовать элемент – системная граница, которая предназначена для обозначения субъекта моделирования (некоторого элемента модели, который обладает функциональным поведением). Графически системная граница в нотации языка UML обозначается с помощью прямоугольника, внутри которого располагаются отдельные варианты использования, относящиеся к этому субъекту моделирования. Актеры внутри системной границы присутствовать не могут, так как они по определению являются внешними по отношению к моделируемой системе сущностями. Вверху прямоугольника может указываться имя данного субъекта. Отображение субъектов моделирования с помощью системной границы необязательно, и, если ее изображение отсутствует, то предполагается, что все варианты использования относятся к системе в целом. Пример отображения субъекта моделирования с помощью системной границы приведен на рисунке 3.

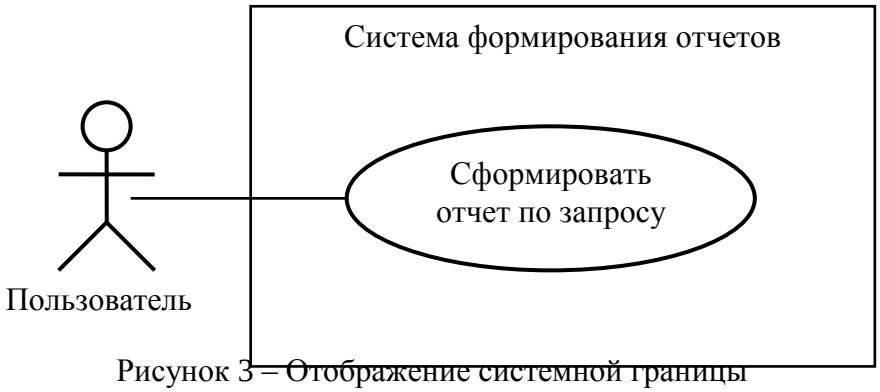



Рисунок 3 – Отображение системной границы

На диаграмме вариантов использования (как и на других диаграммах языка UML) могут использоваться примечания, которые предназначены для включения в модель произвольной текстовой информации, в качестве которой могут быть указаны, например, пояснения разработчика относительно элементов модели, рекомендации руководителя проекта по разработке модели, некоторая справочная информация (например, об авторе) и т.д.

Графически примечание отображается в виде прямоугольника с «загнутым» правым верхним уголком, внутри которого содержится текст комментария. Примечание может относиться к любому элементу диаграммы, который при этом называется аннотируемым элементом. Символ примечания должен быть соединен с аннотируемым элементом пунктирной линией. Примечание может относиться к нескольким элементам, тогда пунктирные линии от примечания проводятся ко всем элементам. Если примечание не соединено ни с одним из элементов диаграммы, то это означает: либо примечание относится к диаграмме в целом, либо из контекста примечания должно быть ясно к какому элементу оно относится. Примеры примечаний приведены на рисунке 4.

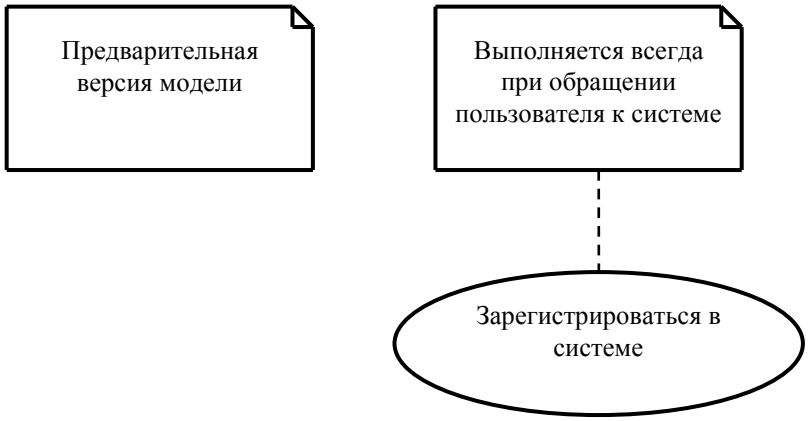



Рисунок 4 – Изображение примечания на диаграммах языка UML
а) – в виде отдельного примечания; б) – в виде примечания, присоединенного к аннотируемому элементу

Между элементами диаграммы вариантов использования могут существовать различные отношения. Под отношением в языке UML понимается произвольная семантическая взаимосвязь между отдельными элементами модели. В общем случае отношение является абстрактным понятием и ссылается на один или более связанных с ним элементов модели. Различные подклассы отношения имеют собственную семантику и графическую нотацию, которая в большинстве случаев является вариацией линии.

На диаграмме вариантов использования между актерами и вариантами использования могут применяться следующие виды отношений:

- ассоциация;
- обобщение;
- включение;
- расширение.

Отношение ассоциации на диаграмме вариантов использования предназначено только для обозначения взаимодействия актера и варианта использования. Отношение ассоциации на диаграмме вариантов использования, как и на других диаграммах языка UML, обозначается сплошной линией, соединяющей актера и вариант использования. Если направление взаимодействия актера и варианта использования для разработчика не имеет принципиального значения, то такое отношение может быть представлено в виде ненаправленной ассоциации (рисунок 5).

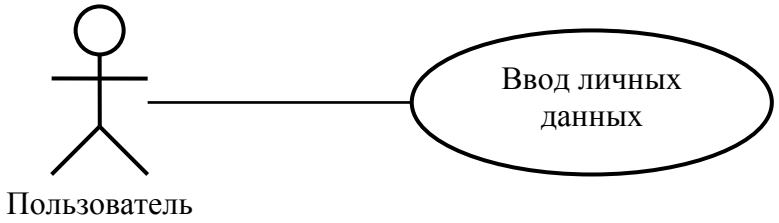



Рисунок 5 – Пример ненаправленной ассоциации

Если направление взаимодействия между актером и вариантом использования, по мнению разработчика, имеет значение, то такое отношение может быть представлено в форме направленной ассоциации, которая обозначается сплошной стрелки с V-образным наконечником. Направленная ассоциация от актера к варианту использования указывает на тот факт, что данный актер является инициатором выполнения соответствующего варианта использования. На рисунке 6 приведен пример того, что актер «Пользователь» инициирует выполнения варианта использования «Регистрация в системе».

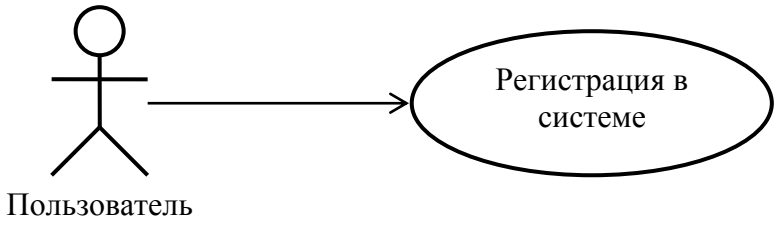



Рисунок 6 – Пример ассоциации, направленной от актера к варианту использования

Также ассоциация может быть направлена от варианта использования к актеру. Это означает, что в процессе своего выполнения система предоставляет актеру некоторую информацию.

Один актер может взаимодействовать с несколькими вариантами использования, что означает возможность обращения этого актера к разным функциям проектируемой системы. С одним вариантом использования может быть связано несколько актеров. В таком случае разработчику требуется дополнительно специфицировать условия выполнения данного варианта использования: параллельное или последовательное участие в нем актеров.

В общем случае отношение ассоциации может иметь собственное имя, а концевые точки ассоциации – имя и кратность (мощность). Но, как правило, на диаграммах вариантов использования эти характеристики не специфицируются.

Отношения включения и расширения на диаграмме вариантов использования являются частными случаями отношения зависимости, которое определяется как форма взаимосвязи между двумя элементами модели, специфицирующее то обстоятельство, что изменение одного (независимого) элемента приводит к измерению другого (зависимого) элемента. Отношение

зависимости в языка UML обозначается пунктирной стрелкой с V-образным наконечником, направленной от зависимого элемента к независимому элементу.

Отношения включения на диаграмме вариантов использования специфицирует тот факт, что некоторый вариант использования содержит поведение, определенное в другом варианте использования. То есть поведение или выполнение одного варианта использования включается в качестве составного фрагмента в поведение или выполнение другого варианта использования. Графически данное отношение обозначается в виде отношения зависимости со стереотипом <<include>>, направленным от включающего к включаемому варианту использования. На рисунке 7 приведен пример: вариант использования «Внесение изменений в базу данных» включает в себя вариант использования «Выполнение авторизации».

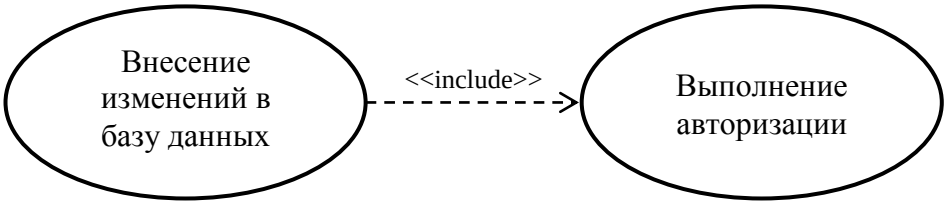



Рисунок 7 – Пример отношения включения

На практике отношение включения используется для моделирования ситуаций, когда существуют общие части поведения двух или более вариантов использования. Тогда эта общая часть может быть выделена в отдельный вариант использования, чтобы затем включить его во все базисные варианты использования, имеющие эту часть в своем поведении. В общем случае вариант использования может быть связан отношением включения с несколькими включаемыми вариантами использования, а также самому быть включаемым в другие варианты использования.

Отношение расширения определяет взаимосвязь одного варианта использования с некоторым другим вариантом использования, функциональность или поведение которого задействуется не всегда, а только при выполнении некоторых дополнительных условий. Графически данное отношение обозначается в виде отношения зависимости со стереотипом <<extend>>, направленного от расширяющего к расширяемому варианту использования. На рисунке 8 приведен пример: вариант использования «Печать отчета» расширяет поведение варианта использования «Формирование отчета за месяц».

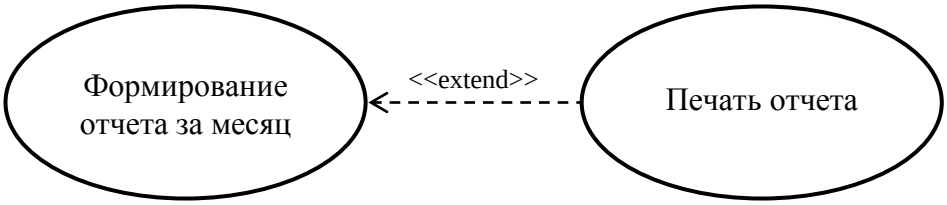



Рисунок 8 – Пример отношения расширения

В общем случае отношение расширения позволяет моделировать тот факт, что базовый (расширяемый) вариант использования может присоединять к своему поведению некоторое дополнительное поведение, определенное в форме расширяющего варианта использования. Наличие этого отношения всегда предполагает проверку некоторого условия и ссылку на точку расширения в базовом варианте использования. Таким образом, выполнение последовательности действий базового варианта использования осуществляется до первой точки расширения, у которой проверяется условие, и, если оно истинно, то выполнение базового варианта использования приостанавливается и начинается выполнение последовательности действий расширяющего варианта использования. После завершения выполнения расширяющего варианта использования возобновляется выполнение базового варианта использования до следующей точки расширения или до конца. Если же условие точки расширения не выполняется, то базовый вариант использования продолжает свое выполнение без прерывания до следующей точки расширения или до конца.

Точки расширения в варианте использования могут быть описаны в сценарии, указаны с помощью произвольного текста (как это показано на рисунке 9) или с помощью структурированного текста (как это показано на рисунке 10).

В общем случае базовый вариант использования может иметь несколько точек расширения, с каждой из которых должен быть связан расширяющий вариант использования. Один расширяющий вариант использования может быть связан отношением расширения с несколькими базовыми вариантами использования, а также иметь в качестве собственных расширений другие варианты использования.

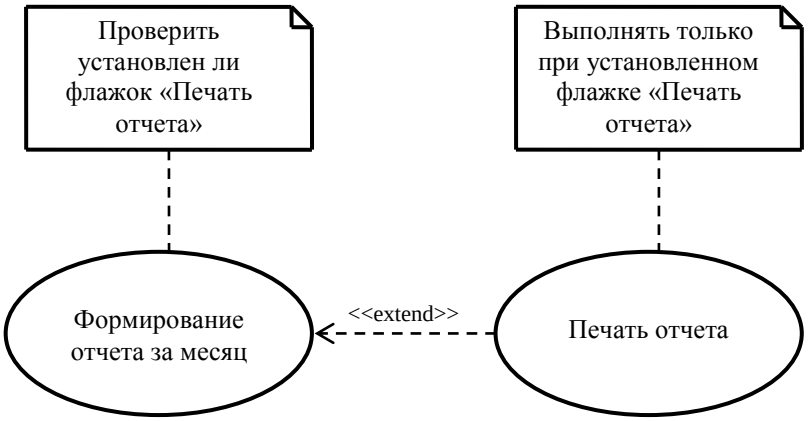



Рисунок 9 – Пример отношения расширения с указанием условия выполнения в форме произвольного текста

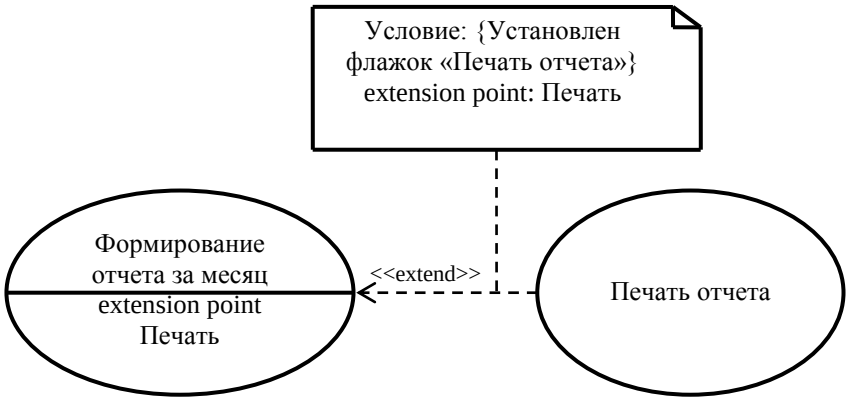



Рисунок 10 – Пример отношения расширения с указанием условия выполнения в форме структурированного текста

Отношение обобщения предназначено для спецификации того факта, что один элемент модели является специализацией другого элемента модели. Главной особенностью отношения обобщения является то, что оно может связывать между собой только элементы одного типа. Графически это отношение обозначается сплошной линей со стрелкой в форме не закрашенного треугольника, направленной на общий элемент модели.

На рисунке 11 приведен пример отношения обобщения между вариантами использования «Отправка сообщения пользователю» и «Отправка сообщения пользователю по e-mail». В данном примере вариант использования «Отправка сообщения пользователю» является общим, а вариант использования «Отправка сообщения пользователю по e-mail» его специализацией (уточняется метод отправки сообщения).

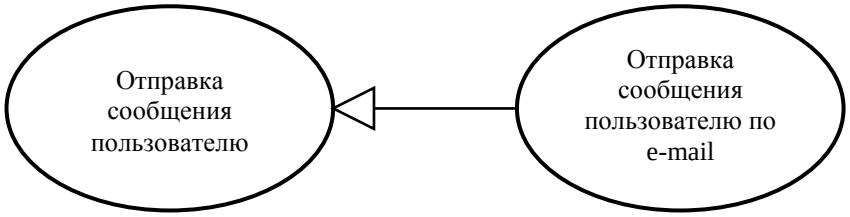



Рисунок 11 – Пример отношения обобщения между вариантами использования

Семантика отношения обобщения применительно к вариантам использования указывает на тот факт, что вариант использования Б является специализацией варианта использования А. При этом вариант использования А называется предком (или родителем) по отношению к варианту использования Б, а вариант использовании Б – потомком (или дочерним) по отношению к варианту использования А. Отношение обобщения между вариантами использования применяется в том случае, когда необходимо отметить, что дочерние варианты использования обладают всеми особенностями поведения родительских вариантов. При этом потомки участвуют во всех отношениях родительских вариантов использования. В свою очередь, потомки могут наделяться новыми свойствами поведения, которые отсутствуют у родительских вариантов использования, а также уточнять или модифицировать наследуемые от них свойства поведения. При этом, с одной стороны вариант использования может иметь нескольких родителей (множественное наследование), а с другой стороны – у одного варианта использования может быть несколько потомков.

Отношение обобщения также может существовать и между актерами. Данное отношение от актера Б к актеру А отмечает тот факт, что каждый экземпляр актера Б является одновременно экземпляром актера А и обладает всеми его свойствами. При этом актер Б обладает способностью играть такое же множество ролей, что и актер А, а также и свои специализированные роли. На рисунке 12 приведен пример отношения обобщения между двумя актерами, где актер «Посетитель магазина» является предком для актера «Покупатель». В данном случае актер «Покупатель», так же как и актер «Посетитель магазина» может, например, просматривать список товаров, а может и оформить покупку товара.

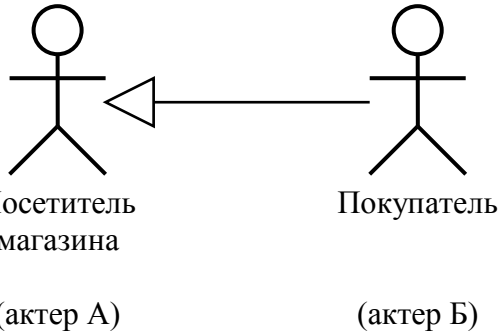



Рисунок 12 – Отношение обобщения между актерами

В заключение рассмотрим пример построения диаграммы вариантов использования (рисунок 13) для информационной системы справочная автовокзала. Целью данной системы является предоставление справочной информации о маршрутах городских, пригородных и междугородних автобусов. Обращаться к этой системе могут потенциальные пассажиры (пользователи) с целью получения справки, а также служащие автовокзала (администраторы) для создания и последующего внесения изменений в расписание движения автобусов. Доступ служащих автовокзала к системе осуществляется после процедуры аутентификации. Пользователю дополнительно должна быть предоставлена возможность просмотра информации по каждому маршруту отдельно.

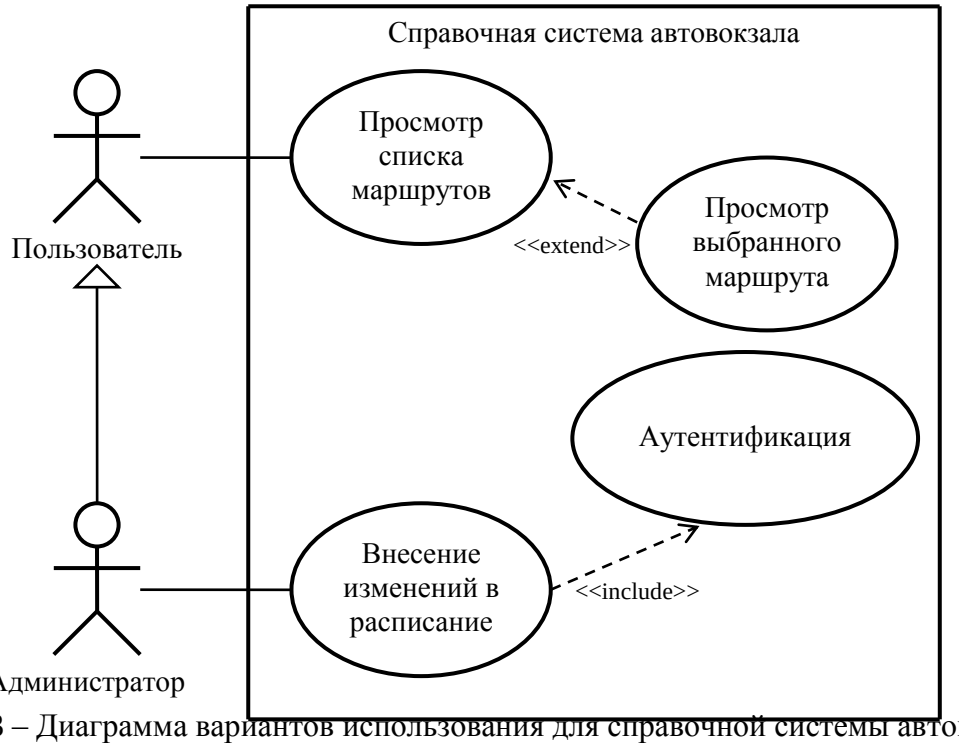



Рисунок 13 – Диаграмма вариантов использования для справочной системы автовокзала

На диаграмме показано, что пользователь системы может просматривать список маршрутов автобусов и, при желании, ознакомиться с каждым маршрутом более подробно (показано с помощью отношения расширения). Администратор системы тоже пользователем (показано с помощью отношения обобщения), т.е. он тоже может просматривать список маршрутов и их самих, а также может вносить изменения в расписание, предварительно выполнив процедуру аутентификации (показано с помощью отношения включения).

## Оценка трудозатрат на разработку ПО на основе диаграммы UseCase

Шаг 1. Определение весовых показателей действующих лиц

Все действующие лица системы делятся на три типа: простые, средние и сложные.

Простое действующее лицо представляет внешнюю систему с четко определенным программным интерфейсом (API).

Среднее действующее лицо представляет либо внешнюю систему, взаимодействующую с данной системой посредством протокола наподобие TCP/IP, либо личность, пользующуюся текстовым интерфейсом (например, ASCII-терминалом).

Сложное действующее лицо представляет личность, пользующуюся графическим интерфейсом (GUI).

Подсчитанное количество действующих лиц каждого типа умножается на соответствующий весовой коэффициент, затем вычисляется общий весовой показатель A.

Таблица 1 – Весовые коэффициенты действующих лиц

| Тип действующего лица | Весовой коэффициент |
|---|---|
| Простое | 1 |
| Среднее | 2 |
| Сложное | 3 |

Шаг 2. Определение весовых показателей вариантов использования

Все варианты использования делятся на три типа: простые, средние и сложные, в зависимости от количества транзакций в потоках событий (основных и альтернативных). В

данном случае под транзакцией понимается атомарная последовательность действий, которая выполняется полностью или отменяется.

Подсчитанное количество вариантов использования каждого типа умножается на соответствующий весовой коэффициент, затем вычисляется общий весовой показатель UCP (табл. 2).

Другой способ определения сложности вариантов использования заключается в подсчете количества классов анализа, участвующих в их реализации (табл. 3).

Таблица 2 – Весовые коэффициенты вариантов использования

| Тип варианта использования | Описание | Весовой коэффициент |
|---|---|---|
| Простой | 3 или менее транзакций | 5 |
| Средний | От 4 до 7 транзакций | 10 |
| Сложный | Более 7 транзакций | 15 |

Таблица 3 – Весовые коэффициенты вариантов использования

| Тип варианта использования | Описание | Весовой коэффициент |
|---|---|---|
| Простой | Менее 5 классов | 5 |
| Средний | От 5 до 10 классов | 10 |
| Сложный | Более 10 классов | 15 |

В результате получаем показатель UUCP (unadjusted use case points):

$$UUCP = A + UC$$

Шаг 3. Определение технической сложности проекта

Техническая сложность проекта (TCF — technical complexity factor) вычисляется с учетом показателей технической сложности (табл. 4).

Таблица 4 – Показатели технической сложности

| Показатель | Описание | Вес |
|---|---|---|
| T1 | Распределенная система | 2 |
| T2 | Высокая производительность (пропускная способность) | 1 |
| T3 | Работа конечных пользователей в режиме on-line | 1 |
| T4 | Сложная обработка данных | 1 |
| T5 | Повторное использование кода | 1 |
| T6 | Простота установки | 0.5 |
| T7 | Простота использования | 0.5 |
| T8 | Переносимость | 2 |
| T9 | Простота внесения изменений | 1 |
| T10 | Параллелизм | 1 |
| T11 | Специальные требования к безопасности | 1 |
| T12 | Непосредственный доступ к системе со стороны внешних пользователей | 1 |
| T13 | Специальные требования к обучению пользователей | 1 |

Каждому показателю присваивается значение 7} в диапазоне от 0 до 5 (0 означает отсутствие значимости показателя для данного проекта, 5 — высокую значимость). Значение TCF вычисляется по следующей формуле:

$$TCF = 0.6 + (0.01 * (\sum T_i \cdot Bec_i)).$$

Шаг 4. Определение уровня квалификации разработчиков

Уровень квалификации разработчиков (EF — environmental factor) вычисляется с учетом следующих показателей (табл. 5).

Таблица 5 – Показатели уровня квалификации разработчиков

| Показатель | Описание | Вес |
|---|---|---|
| F1 | Знакомство с технологией | 1.5 |
| F2 | Опыт разработки приложений | 0.5 |
| F3 | Опыт использования объектно-ориентированного подхода | 1 |
| F4 | Наличие ведущего аналитика | 0.5 |
| F5 | Мотивация | 1 |
| F6 | Стабильность требований | 2 |
| F7 | Частичная занятость | -1 |
| F8 | Сложные языки программирования | -1 |

Каждому показателю присваивается значение в диапазоне от 0 до 5. Для показателей F1 — F4 0 означает отсутствие, 3 — средний уровень, 5 — высокий уровень. Для показателя F5 0 означает отсутствие мотивации, 3 — средний уровень, 5 — высокий уровень мотивации. Для F6 0 означает высокую нестабильность требований, 3 — среднюю, 5 — стабильные требования. Для F7 0 означает отсутствие специалистов с частичной занятостью, 3 — средний уровень, 5 — все специалисты с частичной занятостью. Для показателя F8 0 означает простой язык программирования, 3 — среднюю сложность, 5 — высокую сложность.

Значение EF вычисляется по следующей формуле:

$$EF = 1.4 + (-0.03 \cdot \left(\sum F_i \cdot Bec_i\right).$$

В результате получаем окончательное значение UCP (use case points):

UCP = UUCP * TCF * EF.

Шаг 5. Оценка трудоемкости проекта

В качестве начального значения предлагается использовать 20 человеко-часов на одну UCP. Эта величина может уточняться с учетом опыта разработчиков. Приведем пример возможного уточнения.

Рассмотрим показатели F1—F8 и определим, сколько показателей F1—F6 имеют значение меньше 3 и сколько показателей F7-F8 имеют значение больше 3. Если общее количество меньше или равно 2, следует использовать 20 чел.-ч. на одну UCP, если 3 или 4—28. Если общее количество равно 5 или более, следует внести изменения в сам проект, в противном случае риск провала слишком высок.

***Опытные данные компании Rational*** Проект среднего размера (приблизительно 10 разработчиков, более чем 6—8 месяцев) может включать приблизительно 30 вариантов использования. Это соответствует тому, что средний вариант использования содержит 12 UCP, и каждая UCP требует 20-30 ч. Это означает общую трудоемкость 240-360 чел.-ч. на вариант использования. Таким образом, 30 вариантов использования потребуют приблизительно 9000 чел.-ч. (10 разработчиков в течение 6 месяцев). Однако прямой пропорции не существует: очень большой проект со 100 разработчиками и сроком 20 месяцев не начнется с 1000 вариантов использования из-за проблем размерности.

Использование описанной выше методики для простых и сложных систем хорошо согласуется с опытными данными компании Rational (приблизительно 150—350 ч. на один вариант использования). Самая простая система (весовой показатель UC = 5, A = 2, UUCP = 7) дает (при 20 чел.-ч. на UCP) приблизительно 140 чел.-ч. Сложная система (весовой показатель UC = 15, A = 3, UUCP = 18) дает приблизительно 360 чел.-ч.

## Ход работы

**1.** Создать проект в CASE-средстве Enterprise Architect 8.0. Для этого выбрать пункт главного меню File → New Project.

**2.** В окне Project Browser создать новый пакет (Ctrl+W). В появившемся окне (рис. 14) указать название пакета и его иконку (UseCase).

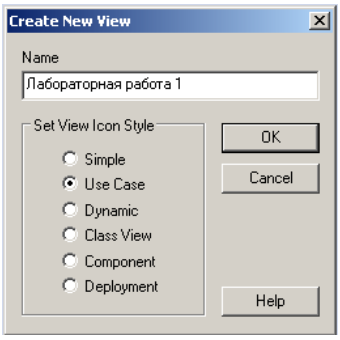



Рис. 14. Окно создания пакета

**3.** В окне Project Browser выделить созданный пакет и создать в нем диаграмму вариантов использования (рис. 15).

**4.** Разработать диаграмму вариантов использования для своего варианта задания.

**5.** Для каждого варианта использования в окне свойств указать сценарии их поведения (рис. 16).

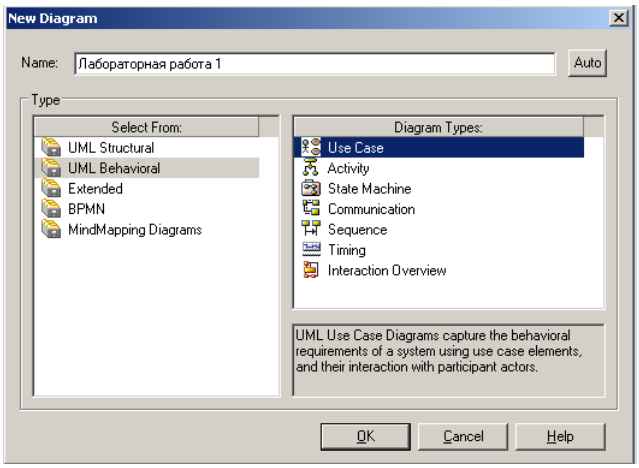



Рис. 15. Окно создания диаграммы

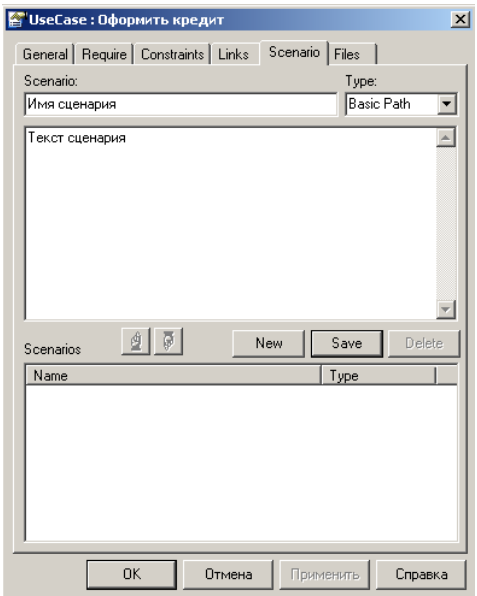



Рис. 16. Окно создания сценариев

**6.** В зависимости от сложности основного сценария для каждого варианта использования и актеров диаграммы указать их сложность (рис. 17).

**7.** Определить показатели технической сложности проекта и уровня квалификации разработчиков, а также норму одного часа рабоы в окне настроек факторов оценки (рис. 18). Вызов через главное меню программы Settings → Estimation Factors.

**8.** Рассчитать оценку трудозатрат на разработку программного обеспечения (рис. 19), вызов через главное меню Project → Use Case Metrics.

**9.** Составить отчет о проделанной работе

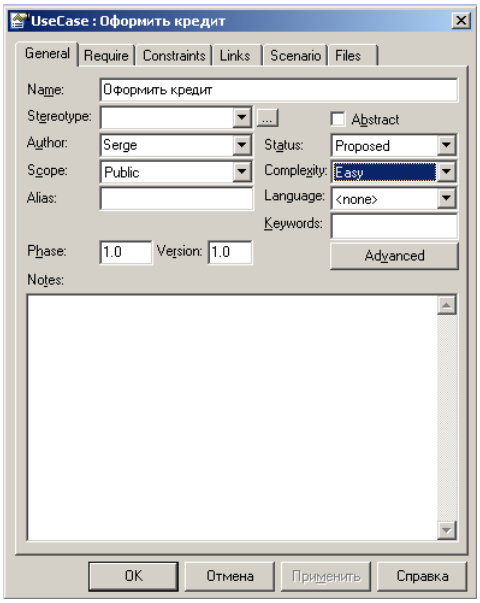



Рис. 17. Основное окно свойств варианта использования (или актера).

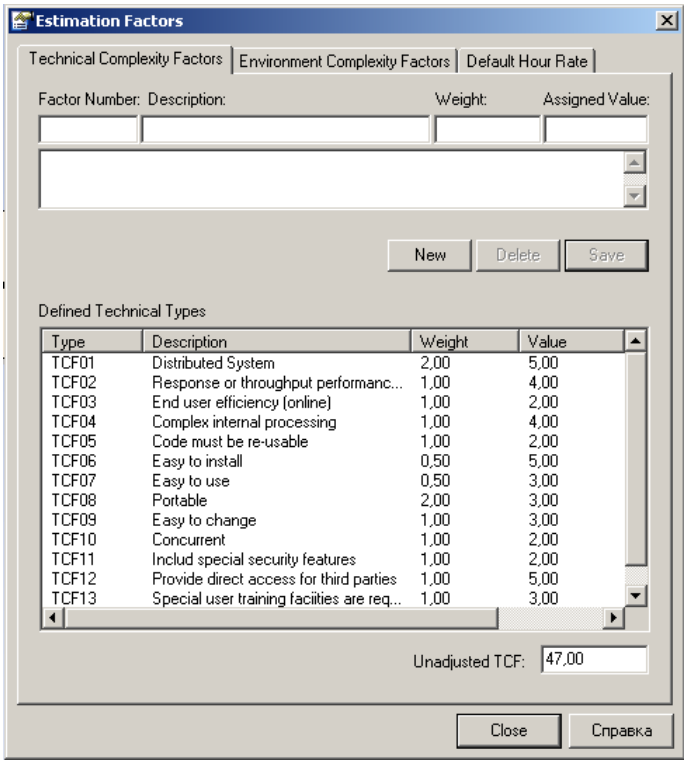



Рис. 18. Окно настроек факторов оценки

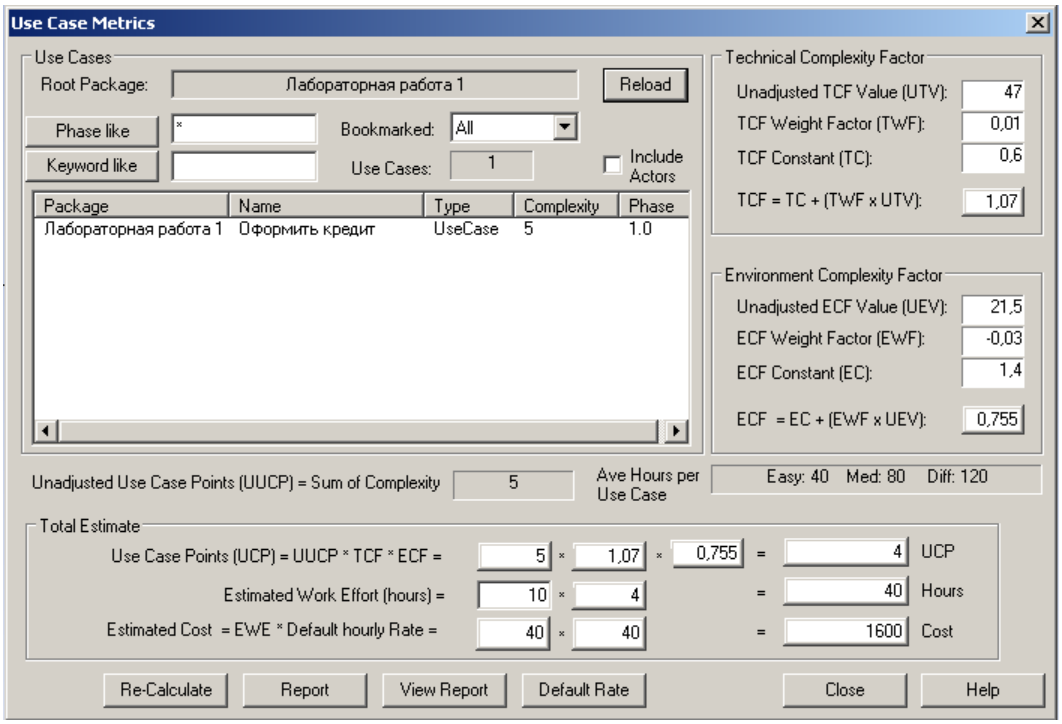



Рис. 19. Окно расчета оценки трудозатрат