Практическая работа № 2

Построение диаграммы вариантов использования UML при проектировании программного обеспечения.

Цели работы:

1.1. Закрепить теоретические знания по принципам создания диаграммы вариантов использования.

1.2. Получить практические навыки по построению диаграммы вариантов использования (диаграммы прецедентов).

Рекомендуемое программное обеспечение

1. «Draw.io»

Достоинства:

- Бесплатный доступ.
- Можно выбрать самому, как пользоваться сервисом: зайти на https://app.diagrams.net/ через браузер, добавить плагин для Chrome или установить приложение на смартфон или компьютер.
- Для сохранения проектов доступны несколько форматов: векторная (.svg) и растровая (.png) графика, веб-страницы (.html, .xml), собственный формат сервиса (.drawio).
- Вы можете указать, куда Draw.io следует сохранять созданный контент. Можно выбрать GitHub, GitLab, Dropbox, Google-диск, OneDrive или память вашего девайса.

2. «Figma.com» перейти по ссылке https://www.figma.com/

Достоинства:

- Позволяет совместно работать над документами — создавать и редактировать командой в режиме реального времени.
- Хранит документы в облаке — макеты не занимают место на диске, их не нужно заливать, чтобы показать коллегам или заказчику.
- Кроссплатформенность — возможность работать с редактором на Windows, Mac, Linux.

Краткие теоретические сведения.

Проектирование – один из важных шагов при разработке программы. Обычно при проектировании разработчики изображают систему графически, поскольку человеку легко разобраться в таком представлении. Именно поэтому вместо написания громоздких текстов про каждую возможность будущей программы разработчики строят различные диаграммы для описания своих систем. Это помогает им не забывать, что нужно реализовать в программе, и быстро вводить в курс дела своих коллег.

Когда разработчик создаёт своё приложение, он в первую очередь задумывается над двумя вопросами:

- Что будет делать приложение?
- Кто будет пользоваться этим приложением?

Некоторыми программами может пользоваться множество людей, поэтому часто необходимо выделять различные группы пользователей системы. У каждой такой группы могут быть свои права и возможности в системе.

Диаграмма прецедентов или диаграмма вариантов использования (англ. *use case diagram)* в UML — диаграмма, отражающая отношения между акторами и прецедентами и являющаяся составной частью ***модели прецедентов***, позволяющей описать систему на концептуальном уровне. Другими словами, **диаграмма вариантов использования**– диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей.

Прецедент — возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних требований к системе.

Пример построения диаграммы вариантов использования

Разберём элементы этой диаграммы, которые чаще всего применяются при построении, на множестве небольших примеров диаграмм и на примере одной большой диаграммы. Эта большая диаграмма будет использоваться при проектировании какой-нибудь программной системы. В качестве такой системы давайте выберем информационную систему для учебного заведения (можно рассматривать ее как сайт или как отдельное приложение). Пример, разумеется, демонстрационный и не претендует на законченность.

В этой системе можно выделить следующие группы пользователей:

- Обучающиеся
- Преподаватели
- Классные руководители
- Заместители директора

Обучающиеся могут:

- Смотреть расписание
- Просматривать свои оценки

Преподаватели могут:

- Размещать материалы для уроков
- Выставлять оценки в электронный журнал

Классные руководители могут делать все то же самое, что и преподаватели плюс:

- Составлять расписание родительских собраний

Заместители директора могут:

- Составлять расписание
- Публиковать посты с важной информацией

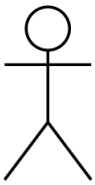
Кроме того, у системы есть функционал, который доступен всем группам пользователей. В разрабатываемой нами системе актуально будет добавить мессенджер, в котором можно будет быстро связываться с интересующим человеком. Получается, эта функциональность должна быть доступна каждому пользователю. Все пользователи могут:

- Отправлять сообщения

Получилось много пунктов, которые может быть сложно уложить в голове. Для того чтобы быстро ориентироваться в этих пунктах, мы и хотим научиться строить диаграммы вариантов использования.

Построение диаграммы

Каждая группа пользователей на *диаграмме вариантов использования* обозначается «человечком» - актёром, под которым записывается имя группы людей, которую он обозначает. Давайте изобразим группу пользователей "Преподаватели":



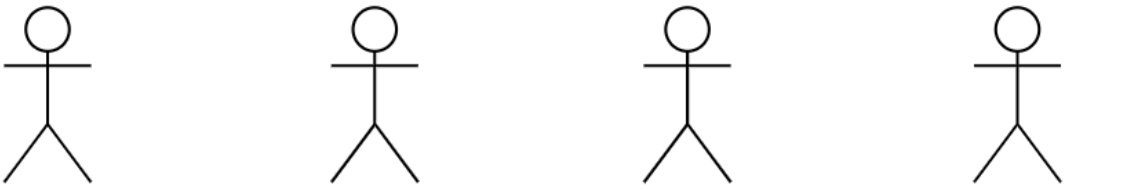
Преподаватель

Этот актор обозначает всех преподавателей, которые будут пользоваться системой

Обратите внимание, что имя группы записывается в единственном числе. Символ человечка уже обозначает *группу* пользователей, поэтому не нужно дополнительно отражать это в имени.

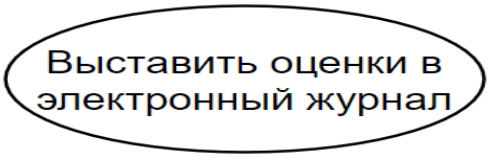
В терминологии UML, этот человечек называется актёром (англ. "actor"). В общем случае, актёр обозначает любые сущности, использующие систему. Этими сущностями могут быть люди, технические устройства или даже другие системы.

Так же изобразим актёров для оставшихся групп пользователей:



Здесь изображены все группы пользователей, которые могут пользоваться нашей системой

Каждая группа пользователей использует определённые функции системы. На *диаграмме вариантов использования* функция системы изображается эллипсом, внутри которого записывается имя функции в форме глагола с пояснительными словами.



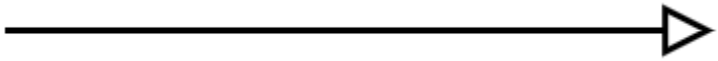
Этот эллипс представляет действие "Выставить оценки в электронный журнал"

В терминологии UML, этот эллипс называется *вариантом использования* (англ. "use-case"). В общем случае, вариант использования – набор действий, который может быть использован актёром для взаимодействия с системой.

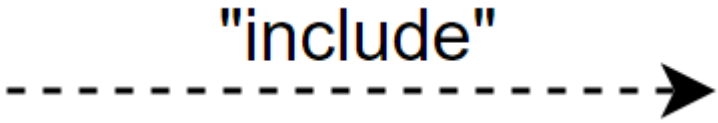
Связи между элементами

На диаграммах UML для связывания элементов используются различные соединительные линии, которые называются *отношениями*. Каждое такое отношение имеет собственное название и используется для достижения определённой цели. В качестве справочной информации перечислю все виды отношений, которые мы будем использовать в этой статье.

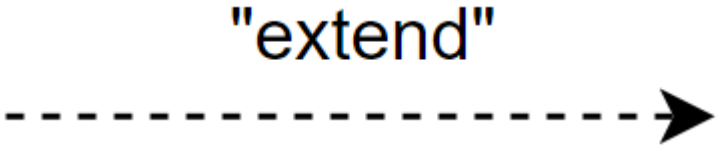
Отношение ассоциации (англ. "association relationship")



Отношение обобщения (англ. "generalization relationship")



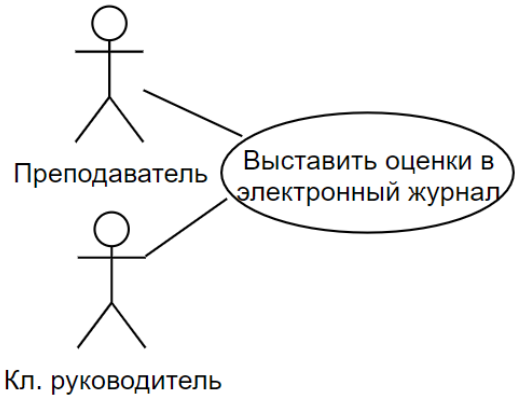
Отношение включения (англ. "include relationship")



Отношение расширения (англ. "extend relationship")

Отношение ассоциации

Мы хотим отображать на диаграмме информацию о том, какие варианты использования могут быть использованы каждым актёром. Сейчас, например, мы хотим показать, что выставлять оценки могут только преподаватели.

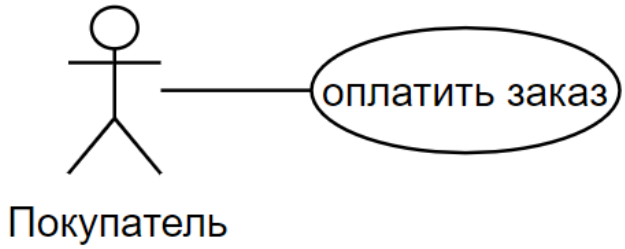


Изображаем на диаграмме информацию о том, что преподаватели могут выставлять оценки

Мы соединили актеров с вариантом использования с помощью сплошной линии без стрелки. Такая линия называется отношением ассоциации.

Отношение ассоциации
Association relationship

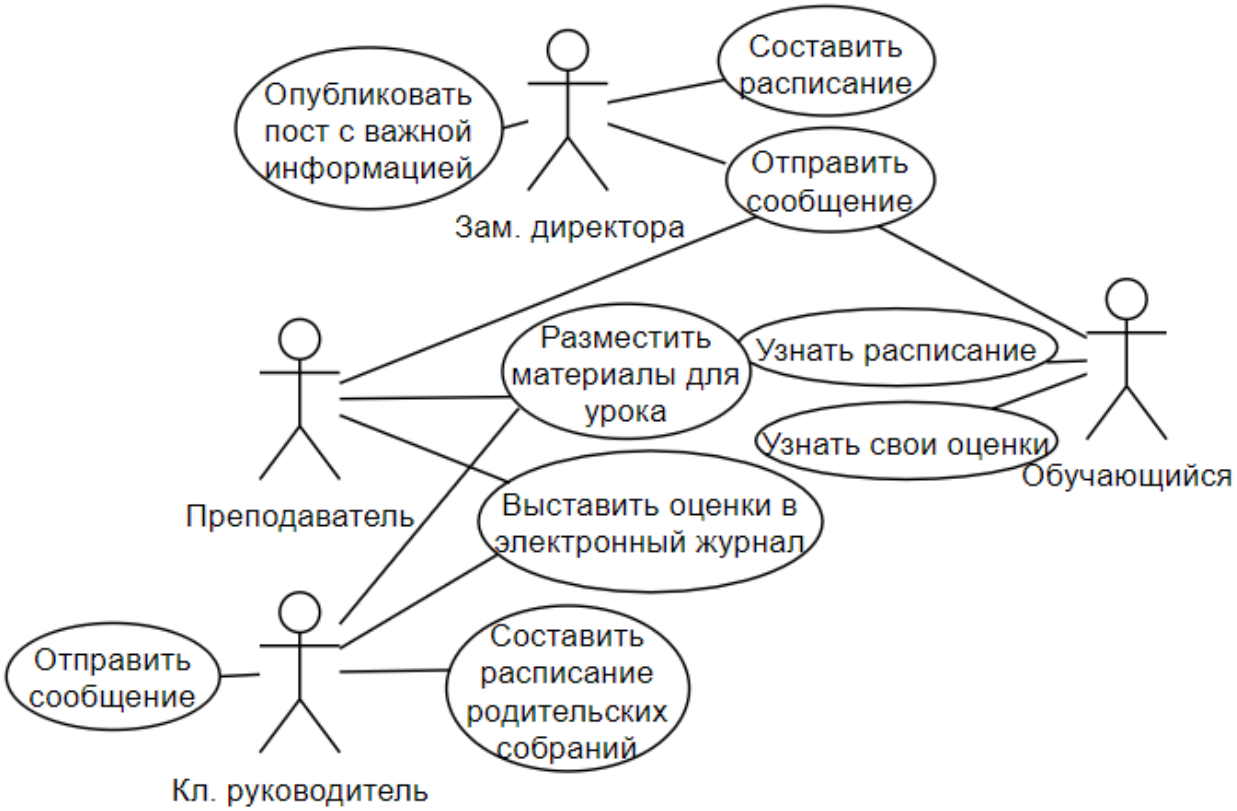
Отношение ассоциации предназначено **только** для соединения актёров и вариантов использования. Нет никакого смысла соединять отношением ассоциации двух актёров или два варианта использования.



Изображаем на диаграмме возможность покупателей оплачивать заказы

Если на диаграмме вариантов использования актёр соединен с вариантом использования с помощью *отношения ассоциации*, это означает, что данный актёр может выполнять действия, описанные вариантом использования.

Добавим еще вариантов использования и соединим их с соответствующими актёрами:



Первая версия диаграммы.

Отношение обобщения

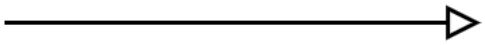
Заметим, что в нашей системе группы пользователей «Преподаватель» и «Классный руководитель» обладают схожими возможностями. Чтобы изобразить это на диаграмме, мы можем пойти одним из трёх путей:

1. Дублировать варианты использования, чтобы связать их с каждым схожим актёром (очевидно, неудачный вариант).

2. Соединить каждого актёра со всеми нужными вариантами использования. Это может породить множество пересечений линий, что не самым лучшим образом скажется на читаемости диаграммы.
3. Показать с помощью одного из видов отношений, что актёры связаны между собой. Это будет означать, что один из них может пользоваться всеми вариантами использования, с которыми соединён другой актёр.

Последний вариант похож на принцип повторного использования кода при написании программ или на наследование классов в ООП (Объектно-ориентированное программирование). Преимущество этого варианта в том, чтобы уменьшить количество связей на диаграмме.

Разумеется, мы воспользуемся третьим путём. В этом нам поможет, так называемое, *отношение обобщения*. Отношение обобщения обозначается сплошной линией с полой треугольной стрелкой.

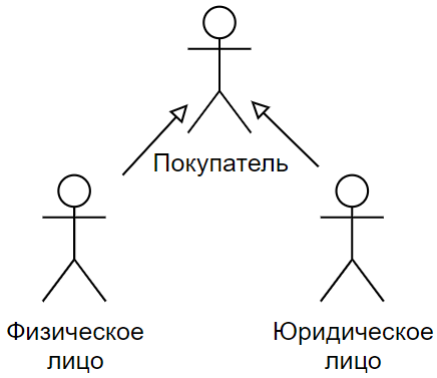


Отношение обобщения
Generalization relationship

Отношение обобщения означает, что некоторый актёр (вариант использования) может быть *обобщён* до другого актёра (варианта использования). Стрелка направлена от частного случая(специализации) к общему случаю.

Ниже представлены несколько примеров использования отношения обобщения.

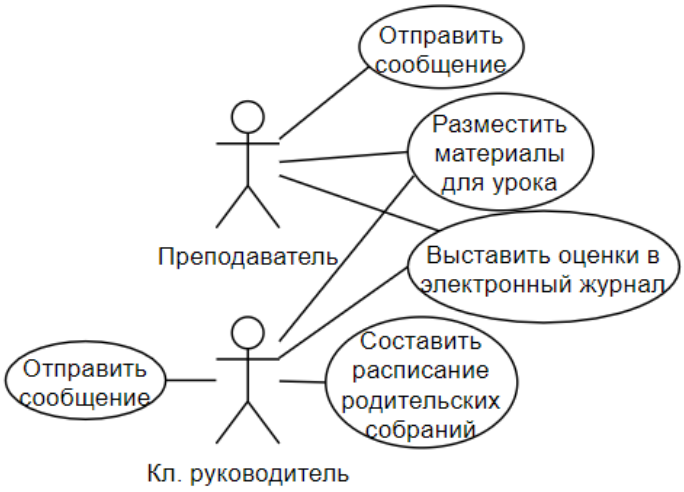
Покупка горного и скоростного велосипеда - ЧАСТНЫЙ случай покупки велосипеда



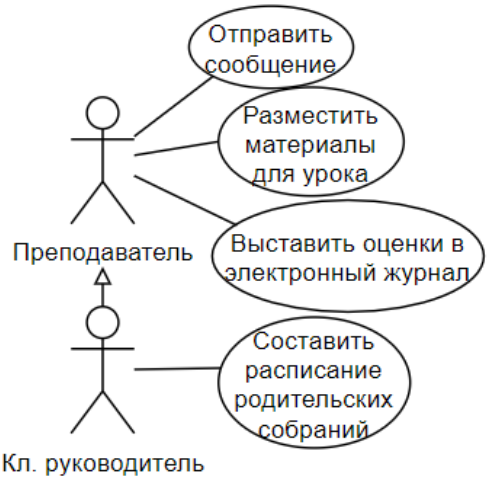
Физическое лицо и юридическое лицо можно ОБОБЩИТЬ до обычного покупателя

Как можно заметить, отношение обобщения используется, чтобы показать, что одно действие является *частным случаем* другого действия или что одну группу людей можно *обобщить* до другой группы.

Вернёмся к нашему основному примеру. Изобразим отношение обобщения от актёра "Кл. руководитель" к актёру "Преподаватель".



До использования отношения обобщения

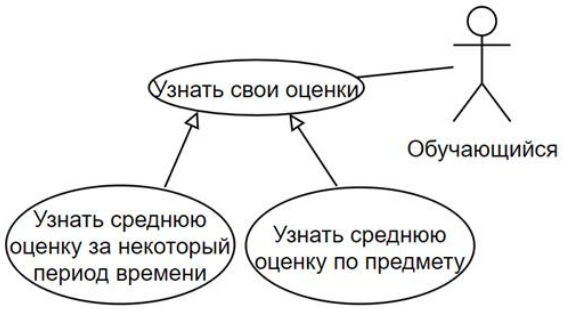


После добавления отношения обобщения

На рисунке сверху сразу видно, насколько понятнее становится диаграмма при использовании отношения обобщения: исчезли все повторы вариантов использования и пересечения линий. Разумеется, это огромный плюс для тех, кто будет читать эту диаграмму в дальнейшем.

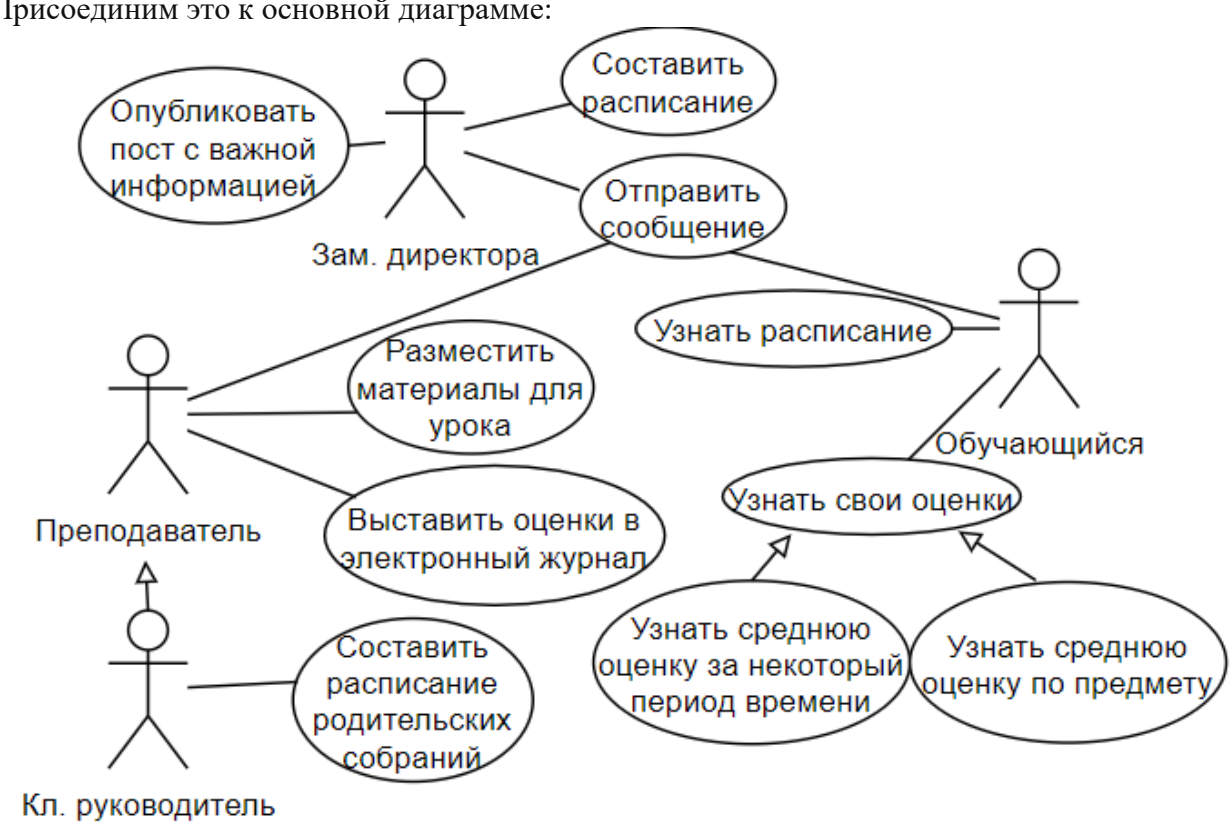
Давайте обратим внимание на действие «Узнать свои оценки». Логично предположить, что обучающиеся захотят не только знать список своих оценок, но и знать свою среднюю оценку за некоторый период времени или среднюю оценку по определённому предмету.

Изобразим это на диаграмме. Для этого создадим два варианта использования "Узнать среднюю оценку за некоторый период времени" и "Узнать среднюю оценку по предмету" и соединим их с *вариантом использования "Узнать свои оценки"* отношением обобщения.



Уточняем на диаграмме, что у обучающихся есть возможность узнать среднюю оценку за некоторый период времени и средний балл по некоторому предмету

Присоединим это к основной диаграмме:



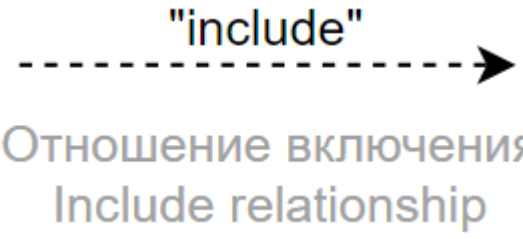
Вторая версия диаграммы

Отношение включения

Для заместителя директора мы отмечали, что ему нужно составлять расписания. *Условно* расписание можно поделить на три категории:

1. Расписание занятий
2. Расписание мероприятий
3. Расписание каникул

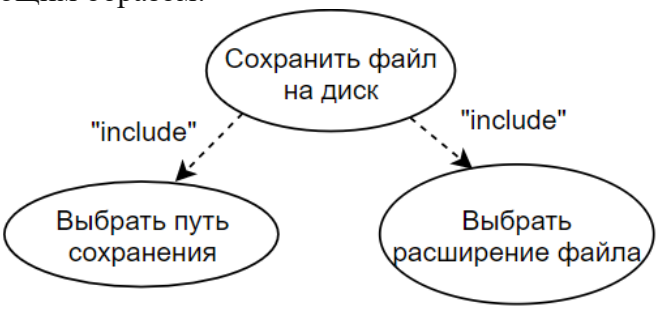
Всё это составляется заместителем директора, поэтому покажем это на диаграмме. Для этого будем использовать *отношение включения.* Отношение включения обозначается пунктирной линией с V-образной стрелкой на конце, над стрелкой добавляется надпись “include”.



В общем случае, отношение включения используется, чтобы показать, что некоторый вариант использования *включает* в себя другой вариант использования в качестве составной части.

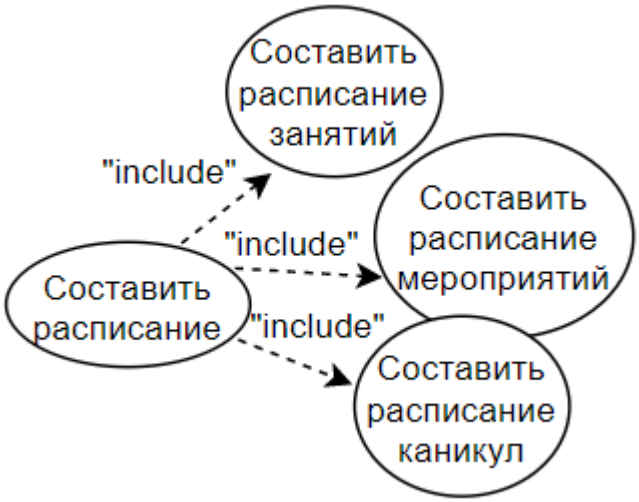
Когда мы используем отношение включения, мы подразумеваем, что составные варианты использования ОБЯЗАТЕЛЬНО входят в состав общего варианта использования.

Поясню смысл и этого отношения на небольшом примере. Когда пользователь сохраняет результаты своей работы в файл, он указывает место сохранения и расширение файла (например, если он редактировал фотографию в photoshop, он может сохранить ее в различных форматах). Этот процесс можно изобразить на диаграмме вариантов использования следующим образом:



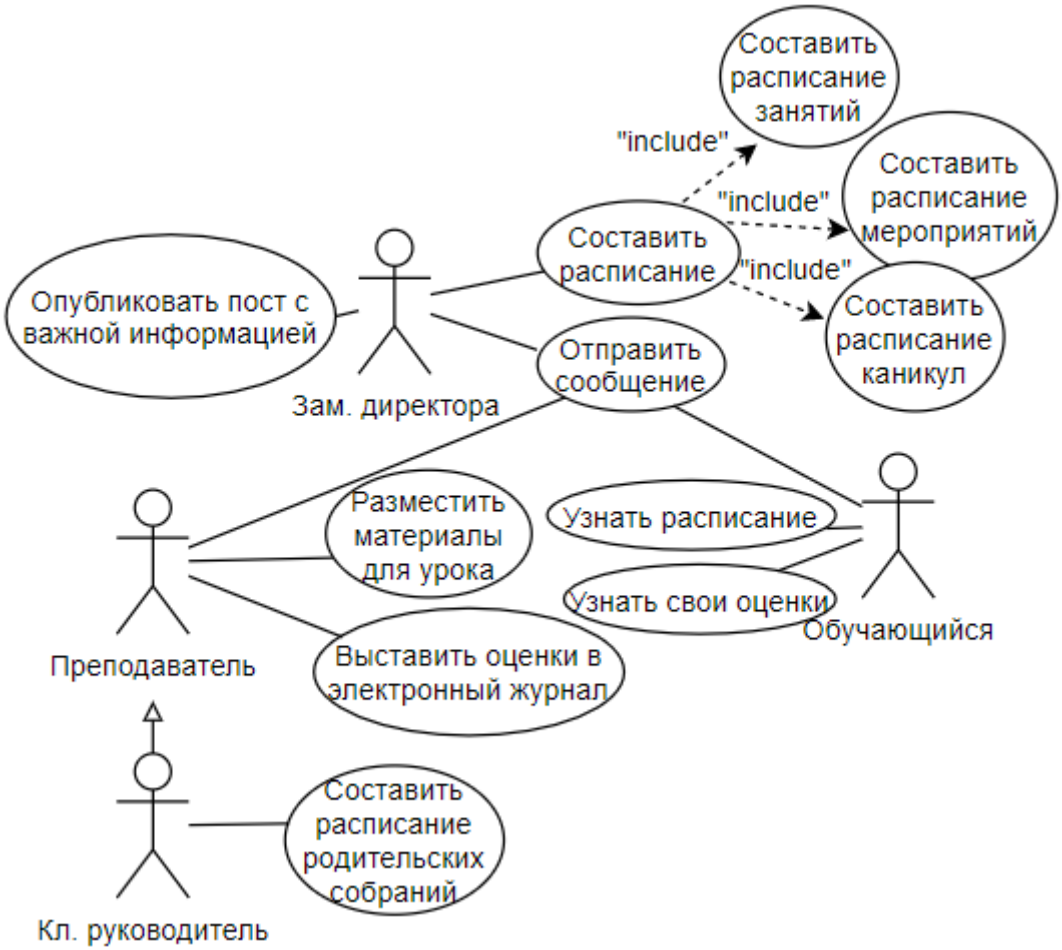
Отношение включения используется для изображения составного действия

Снова вернёмся к нашему основному примеру.



Составление расписания ВКЛЮЧАЕТ в себя составление расписания занятий, мероприятий, каникул(обязательно)

Как итог, наша диаграмма принимает следующий вид:

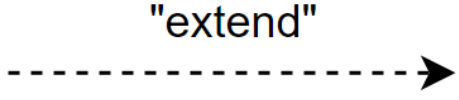


Третья версия диаграммы

Отношение расширения

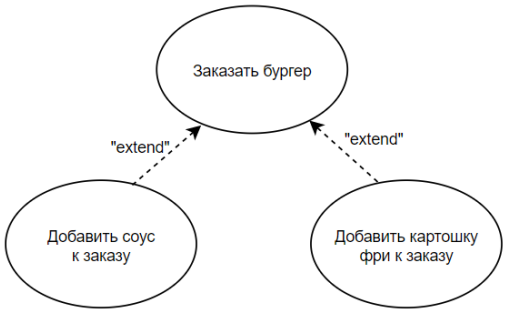
В диаграммах вариантов использования применяется ещё один вид связи – *отношение расширения*. На мой взгляд, применение отношение расширения несколько специфично, поскольку неправильное его использование может запутать читателя диаграммы. Тем не менее, для полноты картины мы всё равно рассмотрим применение этого отношения на практике.

Во время дистанционного обучения школьникам необходимо выполнять домашние задания и присылать их в виде архива или фотографий учителям. Получается, нужно добавить возможность прикреплять файл к сообщению в нашей системе. Чтобы отобразить это на диаграмме мы будем использовать *отношение расширения.* Отношение расширения обозначается пунктирной линией с V-образной стрелкой на конце (похоже на отношение включения), над стрелкой добавляется надпись **“extend ”.**



Отношение расширения
Extend relationship

Чтобы лучше понять этот тип отношений рассмотрим пример. Допустим, вы делаете заказ в сети быстрого питания. Вы хотите заказать бургер. Вам, скорее всего, вам предложат *расширить* ваш заказ картошкой фри или соусом. Давайте изобразим процесс заказа на диаграмме вариантов использования.

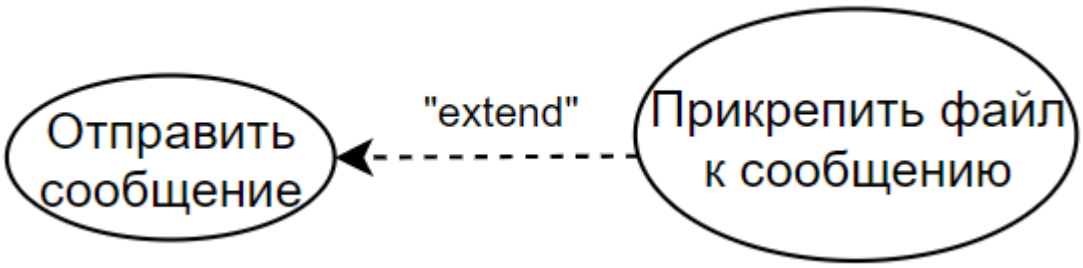


На диаграмме предполагается, что к заказу МОЖЕТ БЫТЬ добавлена картошка фри или соус (необязательно)

Два нижних варианта использования описывают возможные «расширения» для базового варианта использования. Исходя из этого примера, мы можем сделать важное замечание.
Можно сказать, что отношение расширения - это выборочное отношение включения. Если *отношение включения* обозначает, что элемент **обязательно** включается в состав другого элемента, то в случае *отношения расширения* это включение **необязательно.**

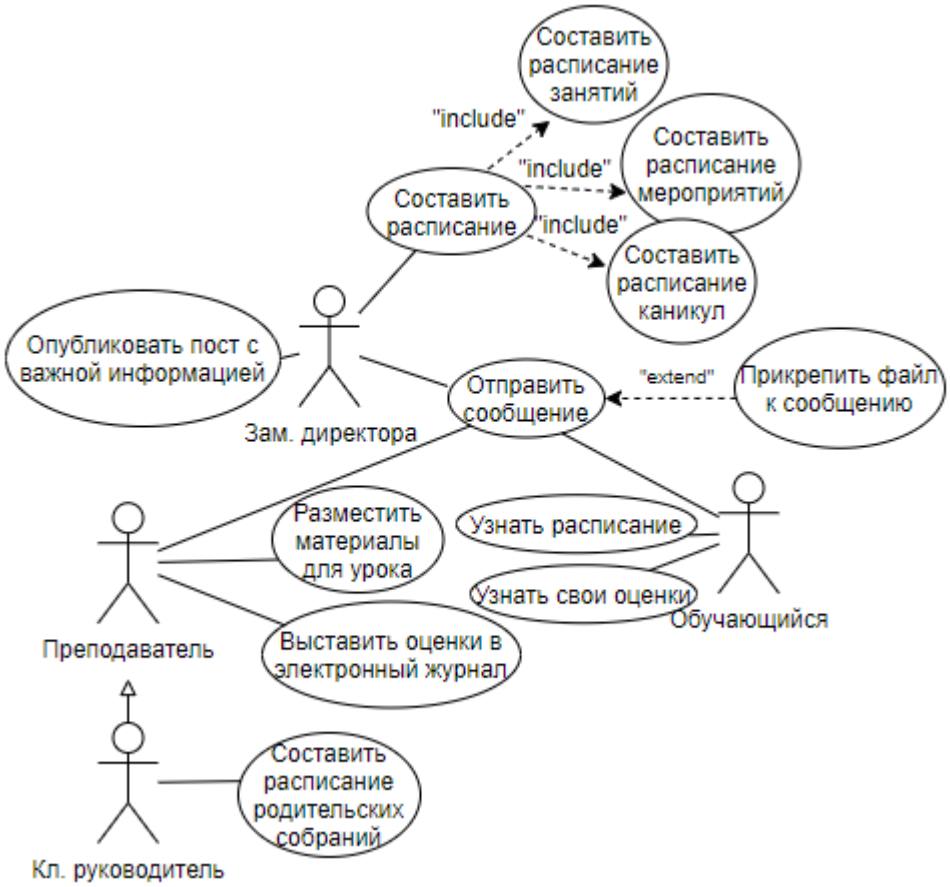
Понимание этого критически важно для грамотного использования этого вида отношений.

Вернёмся к нашему основному примеру. Мы хотим, чтобы действие «прикрепить файл к сообщению» *расширяло* действие «отправить сообщение». На диаграмме это изображается следующим образом:



Расширяем функционал отправки сообщений с помощью функции прикрепления файлов к сообщению (Необязательно прикреплять файл к каждому сообщению)

Как итог, получим такую диаграмму:



Четвёртая версия диаграммы

Общие рекомендации:

1. Диаграммы очень просто изменять. Не нужно пугаться того, что требования к программе могут измениться или что вы что-то забыли отобразить на диаграмме. Вы можете добавить элементы к диаграмме, когда вам угодно.
2. Не нужно засорять диаграмму слишком мелкими действиями. Объедините все общие действия в одну группу под общим названием, чтобы было просто читать диаграмму.
3. Старайтесь не допускать пересечений соединительных линий. Это может затруднить чтение диаграммы для вас и для ваших коллег.
4. Не дублируйте варианты использования на диаграмме. Если приходится дублировать варианты использования, то элементы диаграммы надо постараться расставить по-другому.
5. Пользуйтесь специальными компьютерными программами для построения диаграмм. Это существенно упростит весь процесс моделирования.

Задание на практическую работу.

1. Кратко описать выбранную предметную область (чем занимается предприятие, какие основные процессы в нем происходят. Описать организационную структуру).
   (вариант задания, соответствует номеру по журналу):
   1.1. Разработка информационной системы интернет-магазина "А". Реализация подсистем: отдел закупок, пользовательская поддержка, отдел продаж, отдел маркетинга, бухгалтерский отдел.
   1.2. Разработка информационной системы "Мотель". Реализация подсистем: отдел кадров, отдел аренды, отдел обслуживания, бухгалтерский отдел, отдел маркетинга.
   1.3. Разработка информационной системы "Стрелковый клуб Антеи". Реализация подсистем: арсенал, тир, отдел кадров, логистика, коммерческий отдел).
   1.4. Разработка информационной системы «Магазин бытовой техники». Реализация подсистем: бухгалтерия, отдел кадров, склад, доставка, отдел закупок.
   1.5. Разработка информационной системы "Издательство". Реализация подсистем: отдел кадров, редакция, доставка, производство, отдел закупки.
   1.6. «Разработка информационной системы «Турфирма». Реализация подсистем: бухгалтерия, отдел продаж, отдел кадров, отдел технической поддержки, отдел создания туров.
   1.7. Разработка информационной системы ресторана "Гранд". Реализация подсистем: отдел кадров, бухгалтерия, кухня, зал, отдел закупок.
   1.8. Разработка информационной системы "Поликлиника". Реализация подсистем: Регистратура, Лаборатория, Отдел кадров, Управление лекарственным обеспечением, Управление дневным стационаром.
   1.9. Разработка информационной системы "Магазин кухонных товаров". Реализация подсистем: бухгалтерия, склад, продажи клиентам, управление поставками, отдел управления клиентами и персоналом
   1.10. Разработка информационной системы "GameStart" - продажа цифровых товаров. Реализация подсистем: бухгалтерия, склад, отдел доставки, отдел кадров, дистрибьюция.
   1.11. Разработка информационной системы "Страховое агентство". Реализация подсистем: отдел кадров, юридический отдел, отдел страхование, бухгалтерия, клиенты.
   1.12. Разработка информационной системы «АвтоДилер». Реализация подсистем: отдел закупок, отдел сборки, отдел тестирования продукции, отдел продаж, отдел кадров.
   1.13. Разработка информационной системы "KinoYikes". Реализация подсистем: отдел закупок, отдел продаж, склад, отдел кадров, бухгалтерия.
   1.14. Разработка информационной системы "ТЦ Вектор". Реализация подсистем: отдел кадров, отдел менеджмента аренды, отдел маркетинга, бухгалтерия, отдел закупок и сбыта.
   1.15. Разработка информационной системы "Тренажерный зал". Реализация подсистем: отдел кадров, отдел продаж, отдел обслуживания, хозяйственный отдел, бухгалтерия.
   1.16. Разработка информационной системы «Магазин сотовой связи». Реализация подсистем: склад, отдел продаж, отдел закупок, бухгалтерия.
   1.17. Разработка информационной системы «Магазин автозапчастей». Реализация подсистем: отдел кадров, отдел продаж, отдел закупок, склад, отдел выявления брака.
   1.18. Разработка информационной системы «Магазин игровых аксессуаров». Реализация подсистем: склад, отдел продаж, отдел закупок, отдел кадров.

1.19. Разработка информационной системы «Государственная служба социальной поддержки безработных». Реализация подсистем: отдел кадров, консультационный отдел, распределительный отдел, каталог работ.
1.20. Разработка информационной системы «Vapeshop» Табакерка. Реализация подсистем: продукты, сотрудники, поставщики или контрагенты, бухгалтерия.
1.21. Разработка информационной системы «Атомная электростанция». Реализация подсистем: отдел кадров, отдел обслуживания, склад, технический отдел.
1.22. Разработка информационной системы «Оружейный магазин». Реализация подсистем: отдел закупок, отдел продаж, отдел контроля качества, склад
1.23. Разработка информационной системы «ГИБДД». Реализация подсистем: Водитель, Владелец, Транспортное средство, VIN, Протокол нарушений.
1.24. Разработка информационной системы «Детективное агентство». Реализация подсистем: отдел кадров, клиенты, делопроизводство, заказы.
1.25. Разработка информационной системы «Парикмахерская». Реализация подсистем: отдел кадров, отдел продажи, бухгалтерия, отдел поставок.
1.26. Разработка информационной системы «Пионерлагерь «Совёнок». Реализация подсистем: пляж, тир, отдел кадров, жилой фонд.
1.27. Разработка информационной системы «Компьютерный клуб». Реализация подсистем бухгалтерия, отдел кадров, склад, залы, посещения, компьютеры, посетители, услуги.
1.28. Разработка информационной системы «Платная медицинская клиника». Реализация подсистем: должность, доктор, услуги, отделение, пациент, регистрация, расписание, оплата услуг, лечение, история визитов.
1.29. Разработка информационной системы «Строительная компания «КИП». Реализация подсистем: технический отдел, оперативно производственный отдел, отдел снабжения, сметно-договорной отдел.
1.30. Разработка информационной системы «Дельфинарий». Реализация подсистем: посетители, вольеры животных, отдел клининга, отдел кадров.

2. Выделить потенциальных пользователей системы, описать функции, которые им будут доступны, указать цель, которую преследует каждый пользователь будущего приложения.
3. Построить четыре варианта диаграммы вариантов использования.
   3.1. Актёры, отношение ассоциации
   3.2. Включить отношение обобщения
   3.3. Включить отношение включение
   3.4. Включить отношение расширение

Требования к оформлению отчета

Отчет должен содержать:

- название и цели работы;

- выполненное задание на практическую работу (п.п.1-3);

- общие выводы, сделанные в процессе выполнения практической работы.