Моделирование на UML

Лекция 3

Моделирование использования

Сквозной пример

информационная система отдела кадров

Информационная система отдела кадров — это типичное офисное приложение из самого распространенного класса систем автоматизации делопроизводства.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Информационная система «Отдел кадров» (сокращенно ИС ОК) предназначена для ввода, хранения и обработки информации о сотрудниках и движении кадров.

Система должна обеспечивать выполнение следующих основных функций:

1. Прием, перевод и увольнение сотрудников.
2. Создание и ликвидация подразделений.
3. Создание вакансий и сокращение должностей.

На этом примере видны многие характерные "особенности" подобных документов.

С одной стороны, что-то написано, а с другой стороны не очень понятно, что делать дальше. Безо всяких объяснений заказчик использует термины свой предметной области – разработчик должен их знать и понимать. Требований к реализации нет вовсе. Функции не упорядочены по приоритетам. **Что делать?**

Можно заявить, что это техническое задание никуда не годится и отказаться работать с заказчиком, который его представил. Это простое и надежное решение, но...

Можно потребовать уточнить техническое задание, включив в него ответы на все вопросы разработчика. Это во всех отношениях идеальный вариант, только, к сожалению, так не бывает.

Можно, наконец, попытаться что-то сделать самим с имеющимся материалом.

Выберем третий путь и, **применяя UML, постепенно превратим расплывчатое описание приложения во вполне четкую модель, пригодную для реализации.**

Подходы к моделированию

1) Метод структурного проектирования:

- Программирование сверху вниз - исходная задача разбивается на подзадачи до тех пор, пока каждая отдельная подзадача не станет настолько простой, что ее реализация становится очевидной;
- **Программирование снизу вверх**, при котором уровень языка программирования повышается (например, с помощью определения модулей) до тех пор, пока он не станет настолько высоким и близким к исходной задаче, что ее реализация станет очевидной.
- **программирование вширь**, когда, начиная с самого первого шага, создается и на всех последующих шагах поддерживается работоспособная версия программы.

Подходы к моделированию

2) Моделирование баз данных.

первый шаг хорошо известен: после получения требований нужно составить схему базы данных, то есть определить состав таблиц базы и полей в таблицах, назначить первичные ключи в таблицах и установить связи между таблицами с помощью внешних ключей.

3) Объектно-ориентированный подход.

Апологеты этого подхода первый шаг проектирования описывают примерно так: нужно выделить словарь предметной области (то есть набор основных понятий), сопоставить этим понятиям классы проектируемой системы, определить их атрибуты и операции и дальше все пойдет как по маслу.

4) Моделирование использования.

Преимущества моделирования использования

Простые утверждения. Моделирование использования фактически позволяет переписать исходное техническое задание (или просто записать, если никакой исходной формулировки требований не было) в строгой и формальной, но, в тоже время, очень простой и наглядной графической форме, как совокупность простых утверждений относительно того, что делает система для пользователей. Простое утверждение имеет следующую грамматическую форму: подлежащее – сказуемое – прямое дополнение. Или, в логических терминах, субъект – предикат – объект. Например: начальник увольняет сотрудника, директор создает отдел. Конечно, использование такой формы не гарантирует от ошибок (вряд ли гарантия от ошибок вообще возможна), но благодаря простоте и наглядности формы их легче заметить.

Абстрагирование от реализации. Моделирование использования предполагает формулирование требований к системе абсолютно независимо от ее реализации. Другими словами, представление использования описывает только, **что** делает система (но не **как** это делается и не **зачем** это нужно делать). Заметим, что другие подходы, используя на первых шагах термины и понятия реализации (структура программы, структура данных, структура взаимодействующих объектов) накладывают невольные ограничения на реализацию, которые не вытекают из существа задачи, а значит, могут служить источником неэффективности и ошибок.

Декларативное описание. Каждый вариант использования описывает (а вернее сказать, именует) некоторое множество последовательностей действий, доставляющих значимый для пользователя результат. Однако никакого императивного описания представление использования не содержит, в модели нет указаний на то, какой вариант использования должен выполняться раньше, а какой позже, то есть, нет описания алгоритма, а значит, нет алгоритмических ошибок.

Выявление границ. Представление использования определяет границы системы и постулирует существование во внешнем мире использующих ее агентов (действующих лиц). Описание системы в виде черного ящика с определенными интерфейсами кажется очень похожим на представление использования, но здесь есть важное различие, которое часто упускается из вида. Если ограничиться только описанием интерфейсов, то очень легко допустить ошибки следующего типа: предусмотреть интерфейс, который не нужен, потому что им никто не пользуется. Или, аналогично, забыть интерфейс, который необходим определенной категории пользователей. На диаграмме использования одинокие и покинутые действующие лица и варианты использования обнаруживаются с первого взгляда.

Вывод: моделирование использования безопаснее и надежнее альтернативных методов, то есть при прочих равных условиях позволяет совершить меньше грубых проектных ошибок на ранних стадиях проектирования. В этом заключается основное преимущество данного метода.

Действующие лица

С синтаксической точки зрения *действующее лицо* (actor) – это стереотип классификатора, который обозначается специальным значком.

Для действующего лица указывается только имя, идентифицирующее его в системе.

Семантически *действующее лицо* — это множество логически взаимосвязанных ролей.

***Роль* (role) в UML — это *интерфейс* (interface), поддерживаемый данным *классификатором* (classifier) в данной *ассоциации* (assocation).**

С прагматической точки зрения главным является то, что **действующие лица находятся вне проектируемой системы** (или рассматриваемой части

Имеет смысл отнести пользователей к разным категориям, если наблюдаются следующие признаки:

- пользователи участвуют в разных (независимых) бизнес-процессах;
- пользователи имеют различные права на выполнение действий и доступ к информации;
- пользователи взаимодействуют с системой в разных режимах: от случая к случаю, регулярно, постоянно.

Применительно к нашему примеру мы в первом приближении склонны выделить две категории пользователей:

- менеджер персонала, который работает с конкретными людьми;
- менеджер штатного расписания, который работает с абстрактными должностями и подразделениями.

Бизнес-процесс пользователя первой категории включает в себя не только работу с приложением, но и беседы с конкретными людьми, интервью и тому подобное, чем явно отличается от других бизнес-процессов предприятия.

Пользователи второй категории, очевидно, должны иметь специальные права доступа, поскольку вряд ли допустимо, чтобы кто угодно мог создавать и уничтожать подразделения на предприятии.

Рис. 1 - **Действующие лица ИС ОК**

Семантически *вариант использования* (use case) – это описание множества возможных последовательностей действий (событий), приводящих к значимому для действующего лица результату.

Каждая конкретная последовательность действий называется *сценарием* (scenario).

В нашем примере простой анализ текста технического задания выявляет семь вариантов использования:

1. прием сотрудника;
2. перевод сотрудника;
3. увольнение сотрудника;
4. создание подразделения;
5. ликвидация подразделения;
6. создание вакансии;
7. сокращение должности;

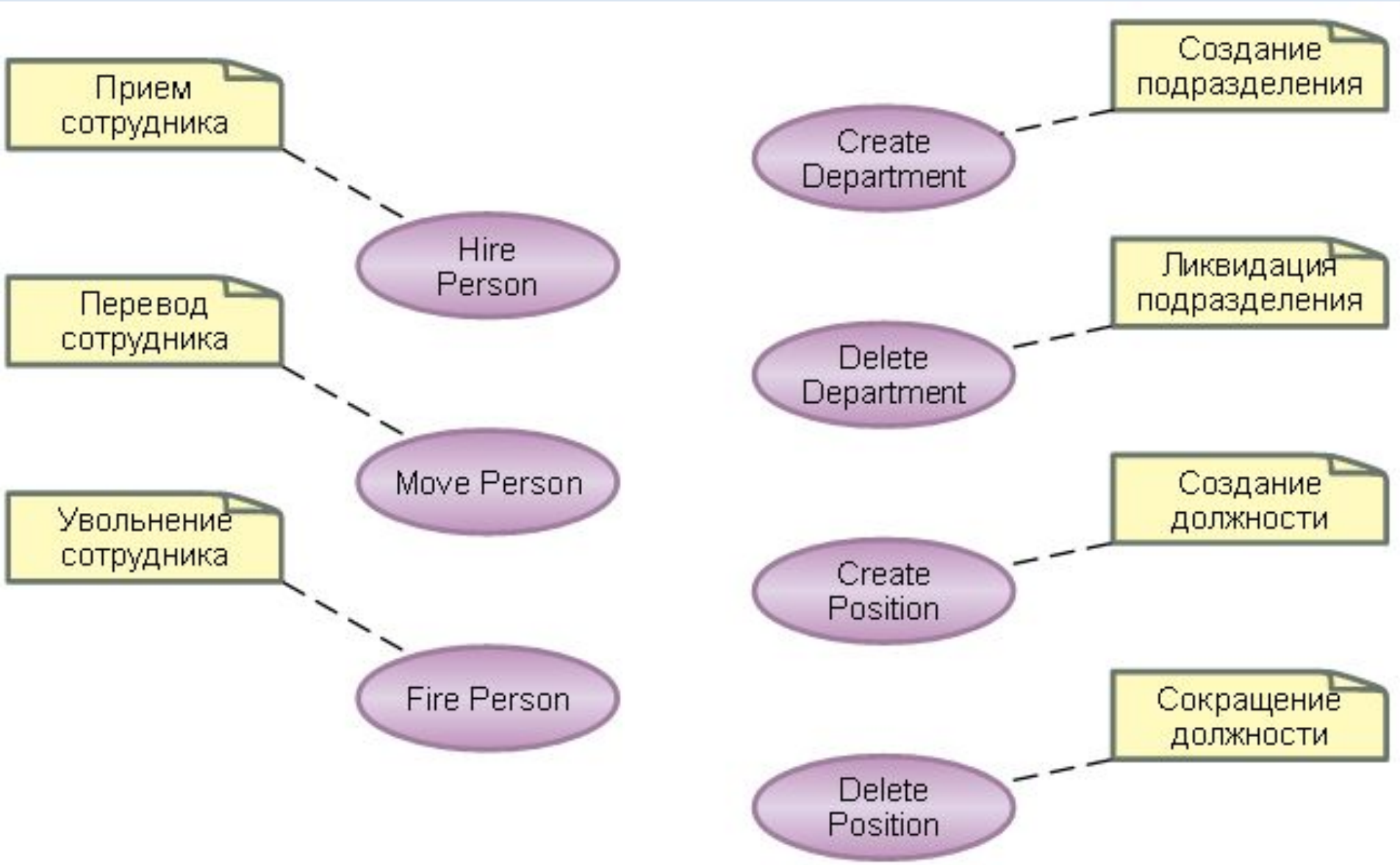


Рис. 2- **Варианты использования ИС ОК**

Комментарии

Третьим типом сущности, применяемым на диаграмме использования, является **комментарий** (comment). Заметим, что комментарии являются очень важным средством UML, значение которого часто недооценивается начинающими пользователями.

В отличие от большинства языков программирования комментарии в UML синтаксически оформлены с помощью специальной нотации и выступают на тех же правах, что и остальные сущности.

В UML последовательно проводится следующий важный принцип: **вся информация, которую пользователь вносит в модель, должна быть сохранена инструментом во внутреннем представлении модели** и предъявлена по первому требованию, даже если инструмент не умеет обрабатывать эту информацию. Комментарии являются

Отношения на диаграммах использования

На диаграммах использования применяются следующие основные типы отношений:

- ассоциация между действующим лицом и вариантом использования;
- обобщение между действующими лицами;
- обобщение между вариантами использования;
- зависимости между вариантами использования;

Ассоциация между действующим лицом и вариантом использования показывает, что действующее лицо тем или иным способом взаимодействует (предоставляет исходные данные, получает результат) с вариантом использования.

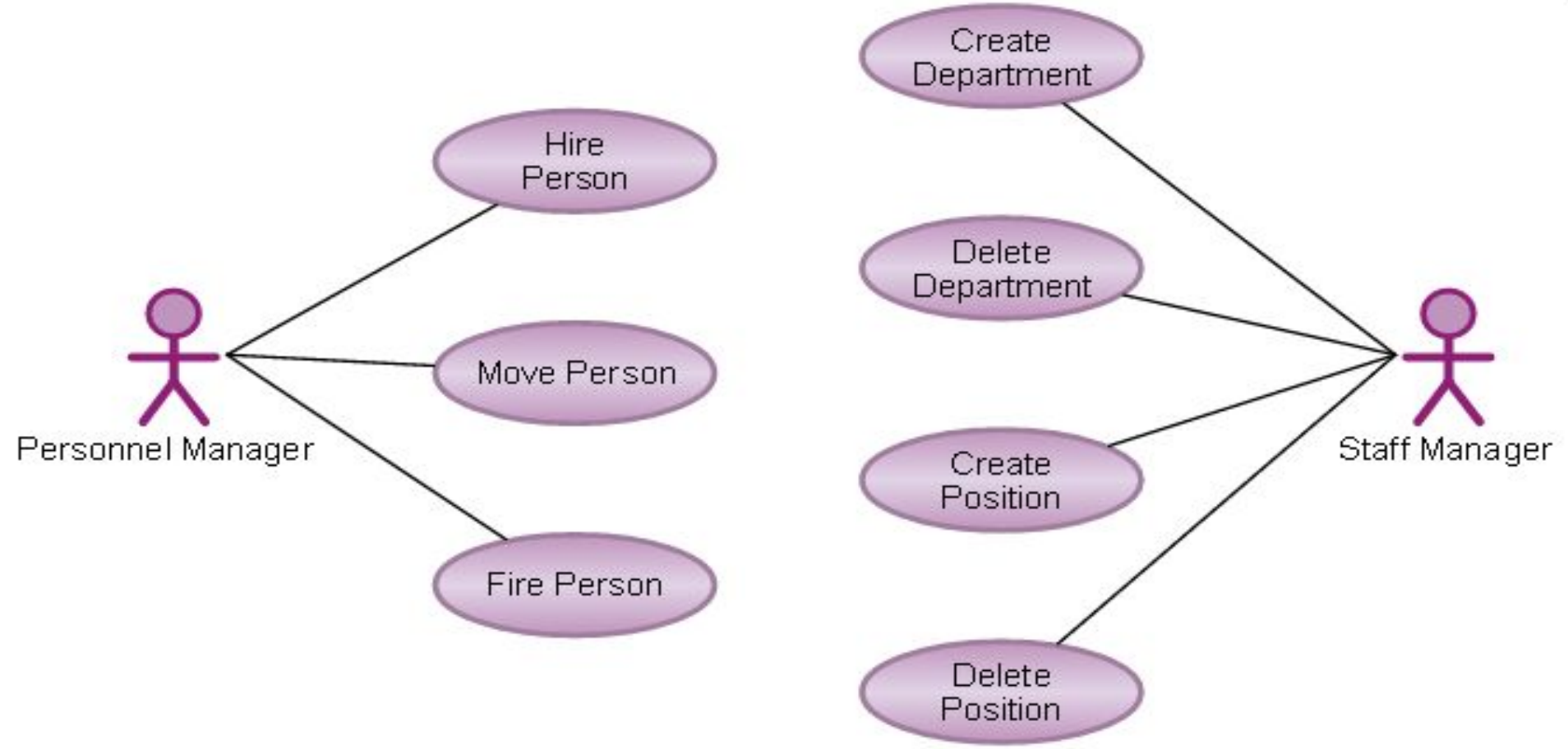
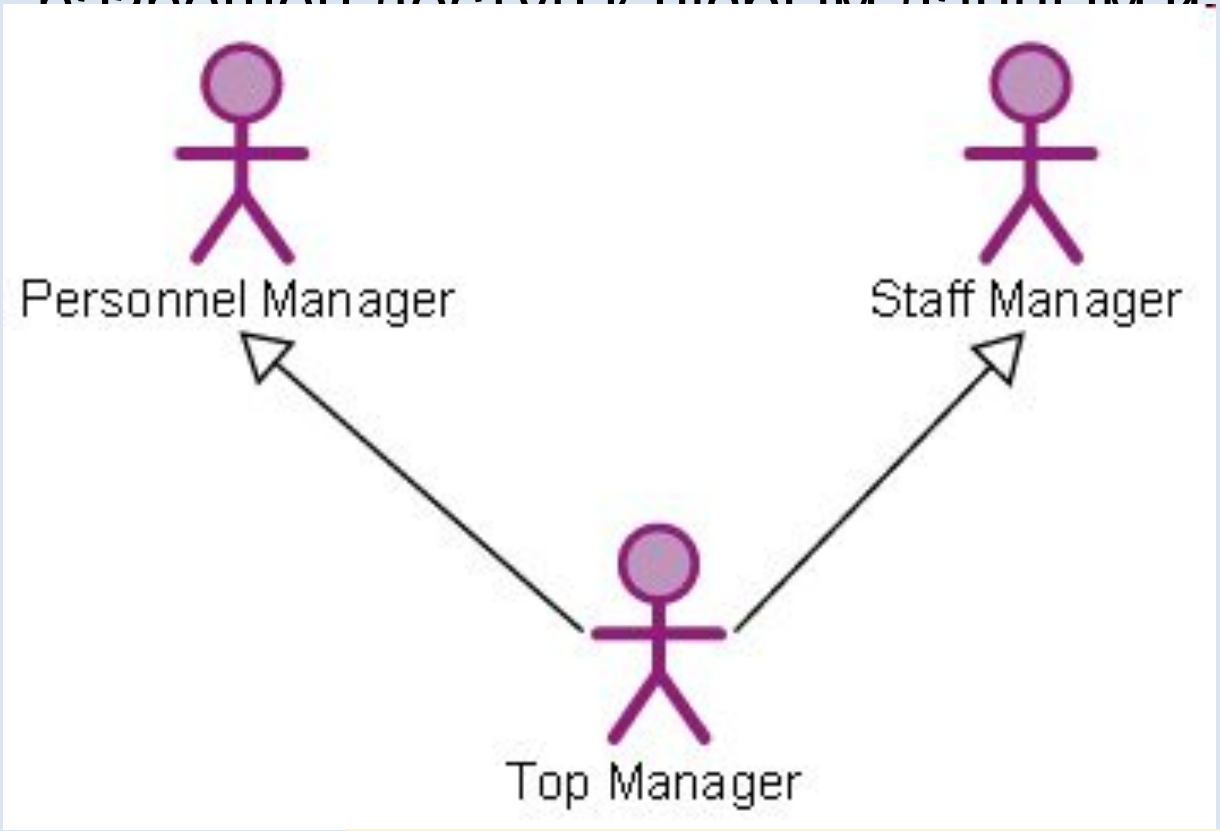


Рис. 4 - **Ассоциации между действующими лицами и вариантами использования**

ИЗМЕНЕНИЯ В ТЕХНИЧЕСКОМ ЗАДАНИИ

Среди всех пользователей информационной системы следует выделить особую категорию пользователей (высшее руководство), которой разрешен доступ к любым данным и операциям.



Обобщение между действующими лицами показывает, что одно действующее лицо наследует все свойства (в частности, участие в ассоциациях) другого действующего лица.

Рис. 5 - **Иерархия категорий пользователей ИС ОК**

ИЗМЕНЕНИЯ В ТЕХНИЧЕСКОМ ЗАДАНИИ

Информационная система должна предоставлять возможность просматривать данные без внесения в них каких-либо изменений.

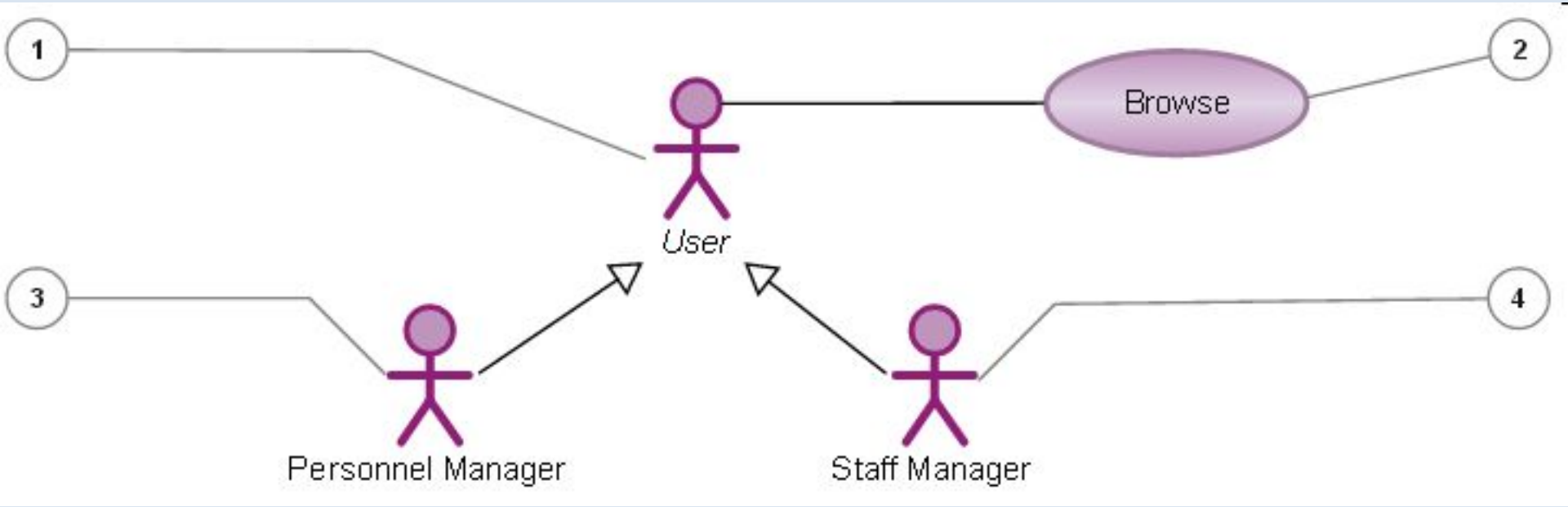


Рис. 6 - **Абстрактное действующее лицо**

Обобщение между вариантами использования показывает, что один вариант использования является частным случаем (подмножеством множества сценариев) другого варианта использования.

ИЗМЕНЕНИЯ В ТЕХНИЧЕСКОМ ЗАДАНИИ

Система должна поддерживать два способа увольнения сотрудника: по инициативе администрации и по собственному желанию.

Рис. 7 - **Обобщение вариантов использования**

Зависимость между вариантами использования **показывает, что один вариант использования зависит от другого варианта использования.**

ИЗМЕНЕНИЯ В ТЕХНИЧЕСКОМ ЗАДАНИИ

При увольнении сотрудника должна быть осуществлена выплата денежной компенсации за неиспользованный отпуск. В случае вынужденного сокращения возможна выплата выходного пособия.

Учетная запись сотрудника при увольнении должна быть заблокирована.

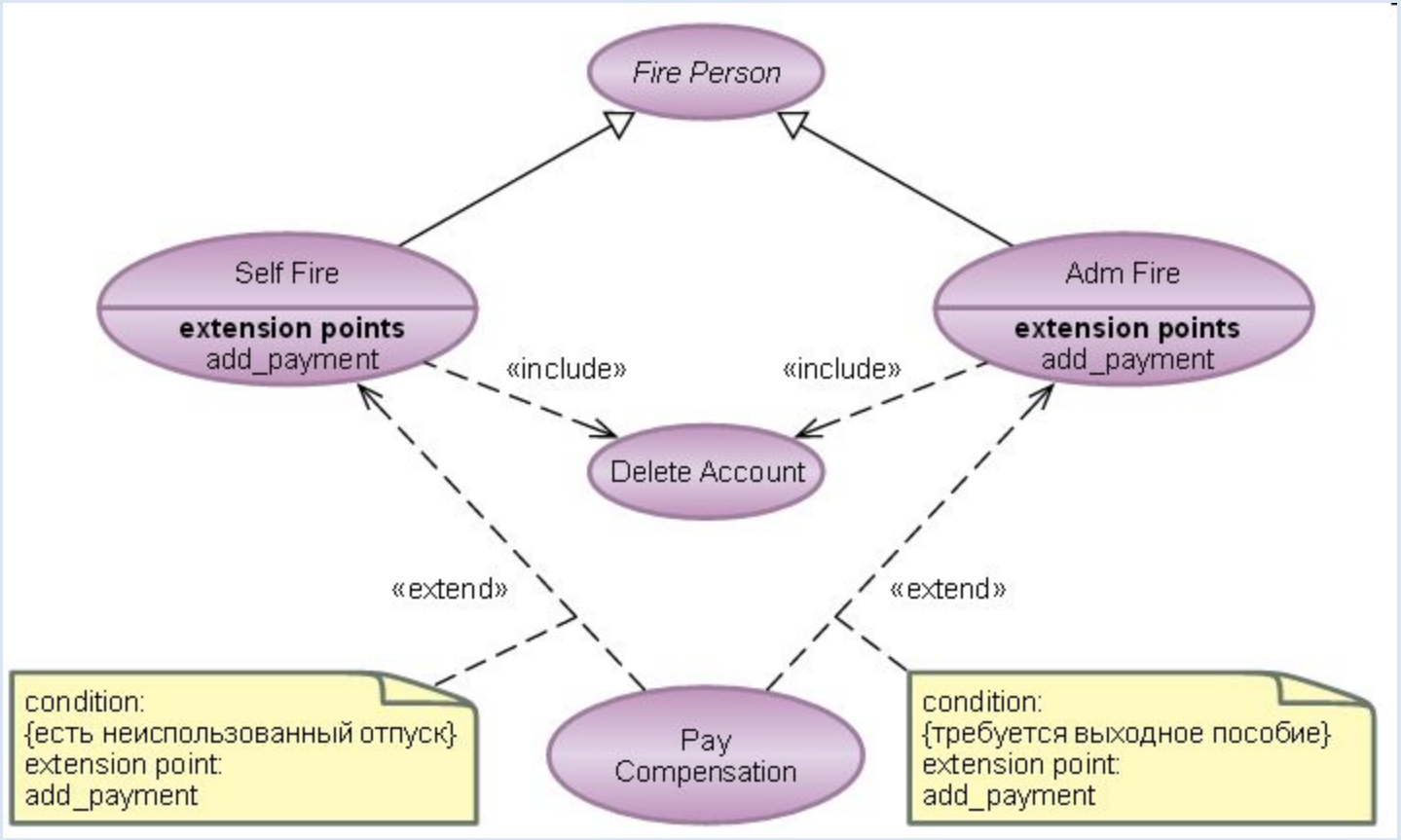


Рис. 8 - **Зависимости между вариантами использования**

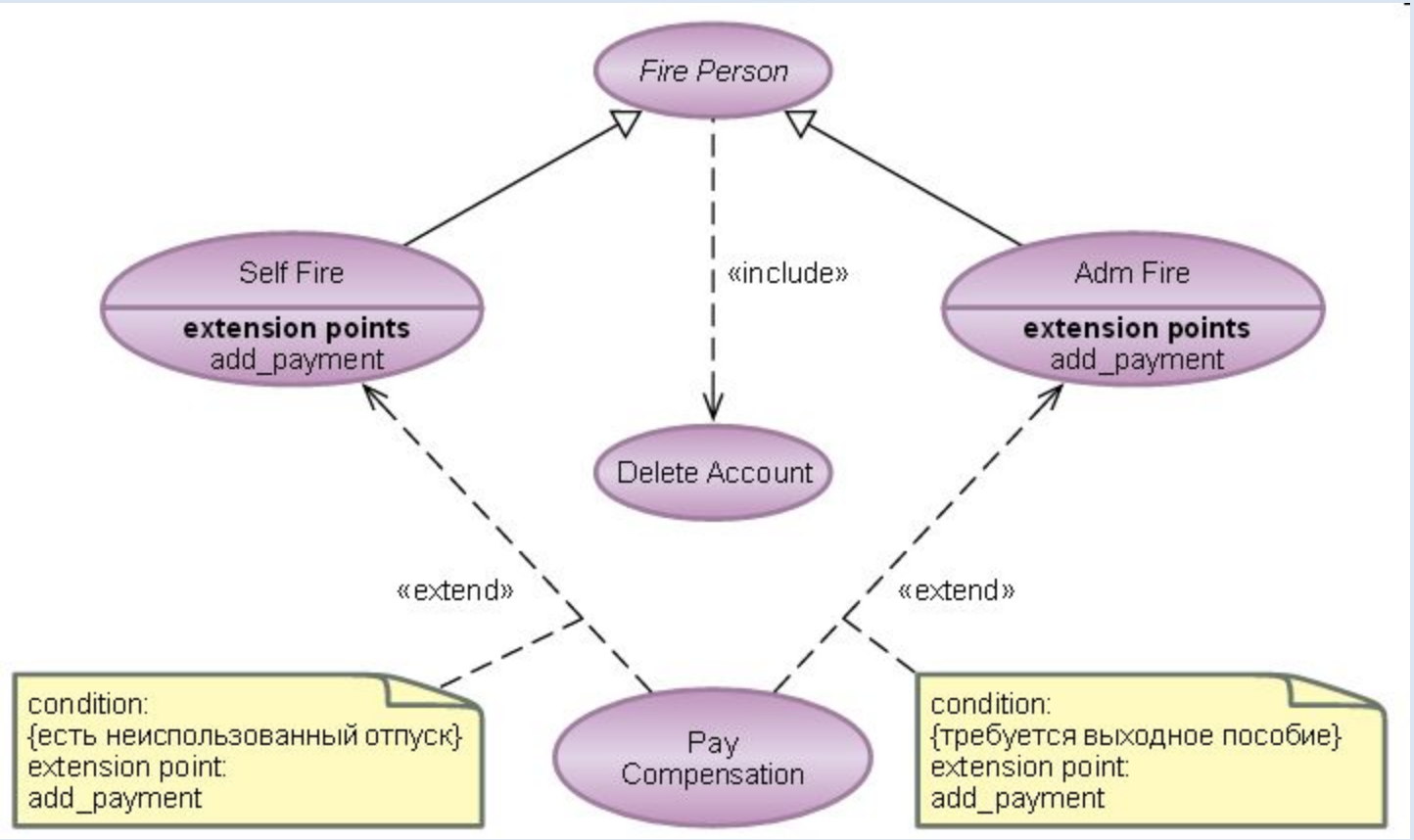


Рис. 9 - **Комбинация отношений обобщения и зависимости**

Способы применения моделей использования

Границы системы **(system boundary) — это графический комментарий в форме прямоугольной рамки, применяемый на диаграммах использования и отделяющий внутреннюю часть системы от ее внешнего окружения.**

Субъект **(subject) — это классификатор, который реализует поведение, декларируемое вариантами использования.**

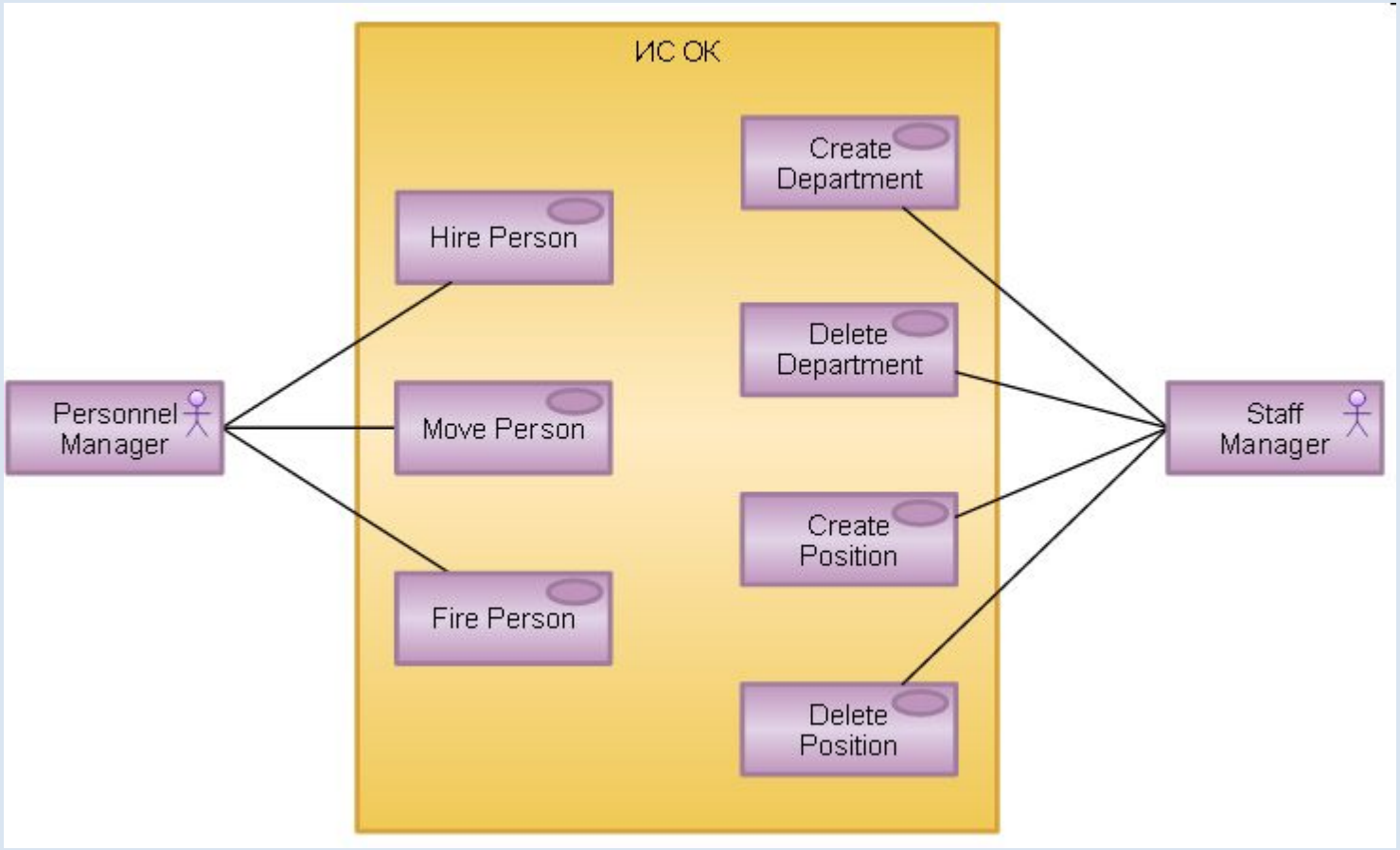


Рис. 10 - **Границы системы**

В качестве системы может выступать любая сущность, для которой можно определить функциональные или не функциональные требования.

Это может быть и подсистема главной системы, отдельный компонент и просто класс. Если мы рассматриваем модель использования некоторой подсистемы, то другие подсистемы (взаимодействующие с рассматриваемой) будут действующими лицами для рассматриваемой подсистемы. Если мы рассматриваем модель использования некоторого класса, то другие классы (взаимодействующие с рассматриваемым) будут действующими лицами для рассматриваемого класса.

ИЗМЕНЕНИЯ В ТЕХНИЧЕСКОМ ЗАДАНИИ

Система должна поддерживать интерфейс прикладного программирования (API) позволяющий внешним программным средствам получать доступ к информации, хранимой в системе.

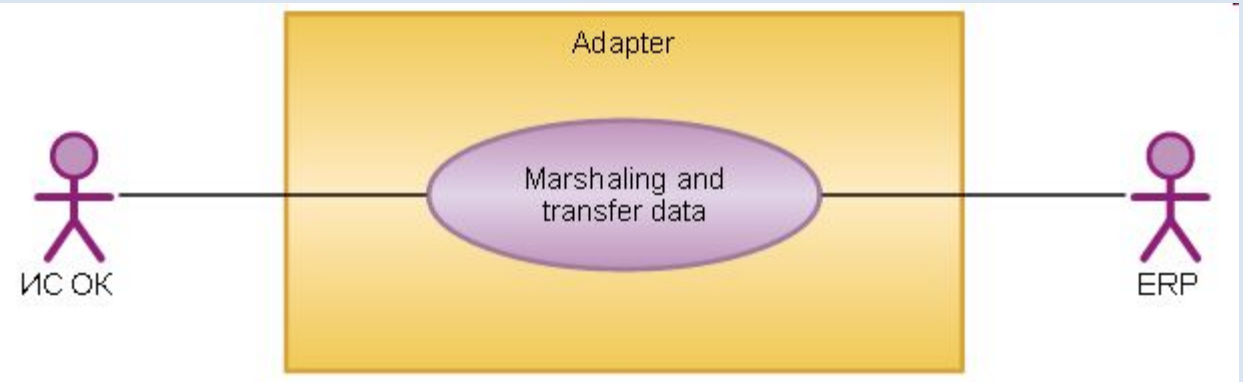


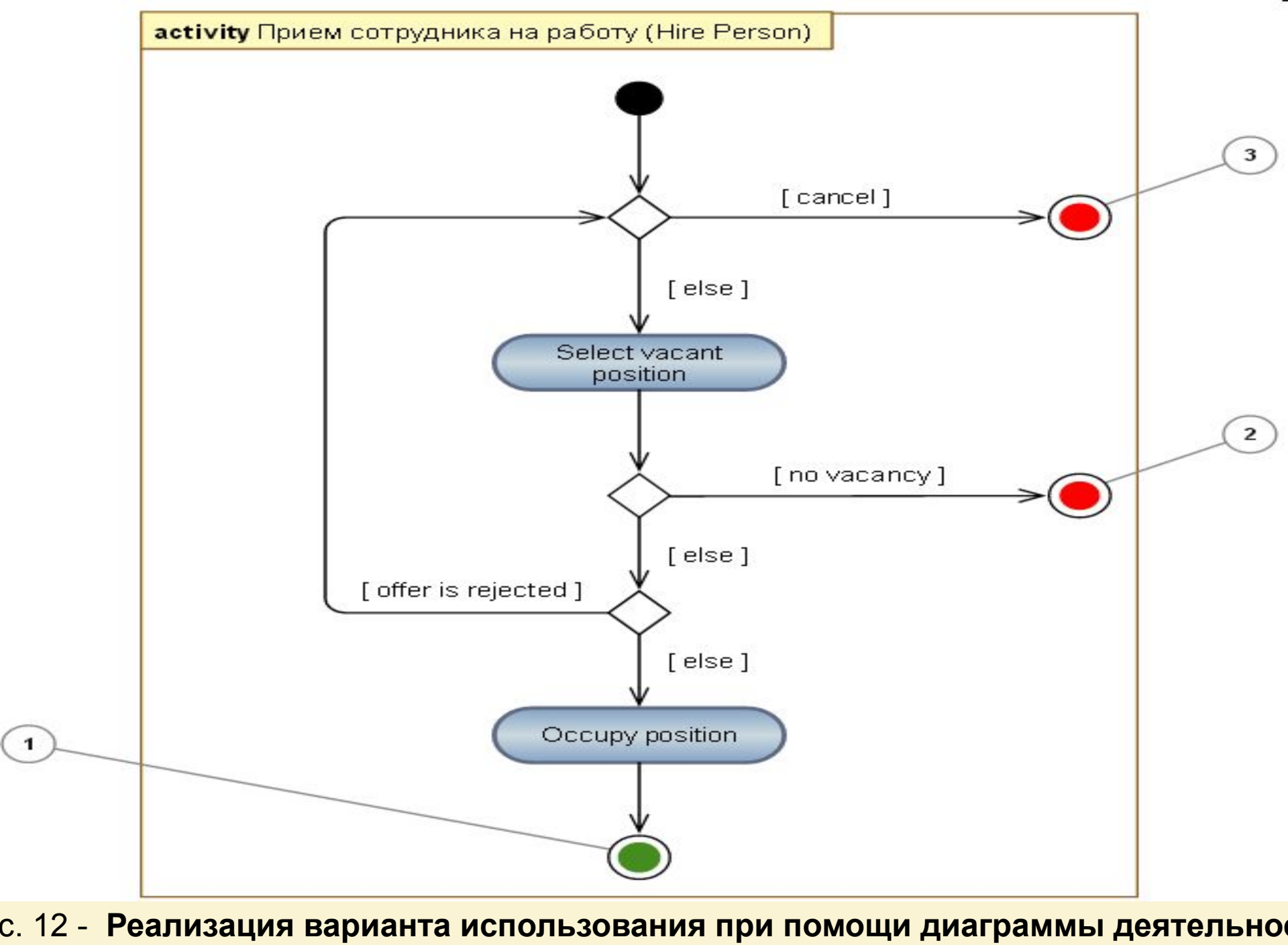
Рис. 11 - **Программные системы в качестве действующих лиц**

Реализация вариантов использования

Реализация варианта использования **(use case realization) — это описание всех или некоторых сценариев, составляющих вариант использования.**

Сценарий варианта использования **Увольнение по собственному желанию**

1. Сотрудник пишет заявление
2. Начальник подписывает заявление
3. **Если** есть неиспользованный отпуск, **то** бухгалтерия рассчитывает компенсацию
4. Бухгалтерия рассчитывает выходное пособие
5. Системный администратор удаляет учетную запись
6. Менеджер персонала обновляет базу данных



с. 12 - **Реализация варианта использования при помощи диаграммы деятельно**

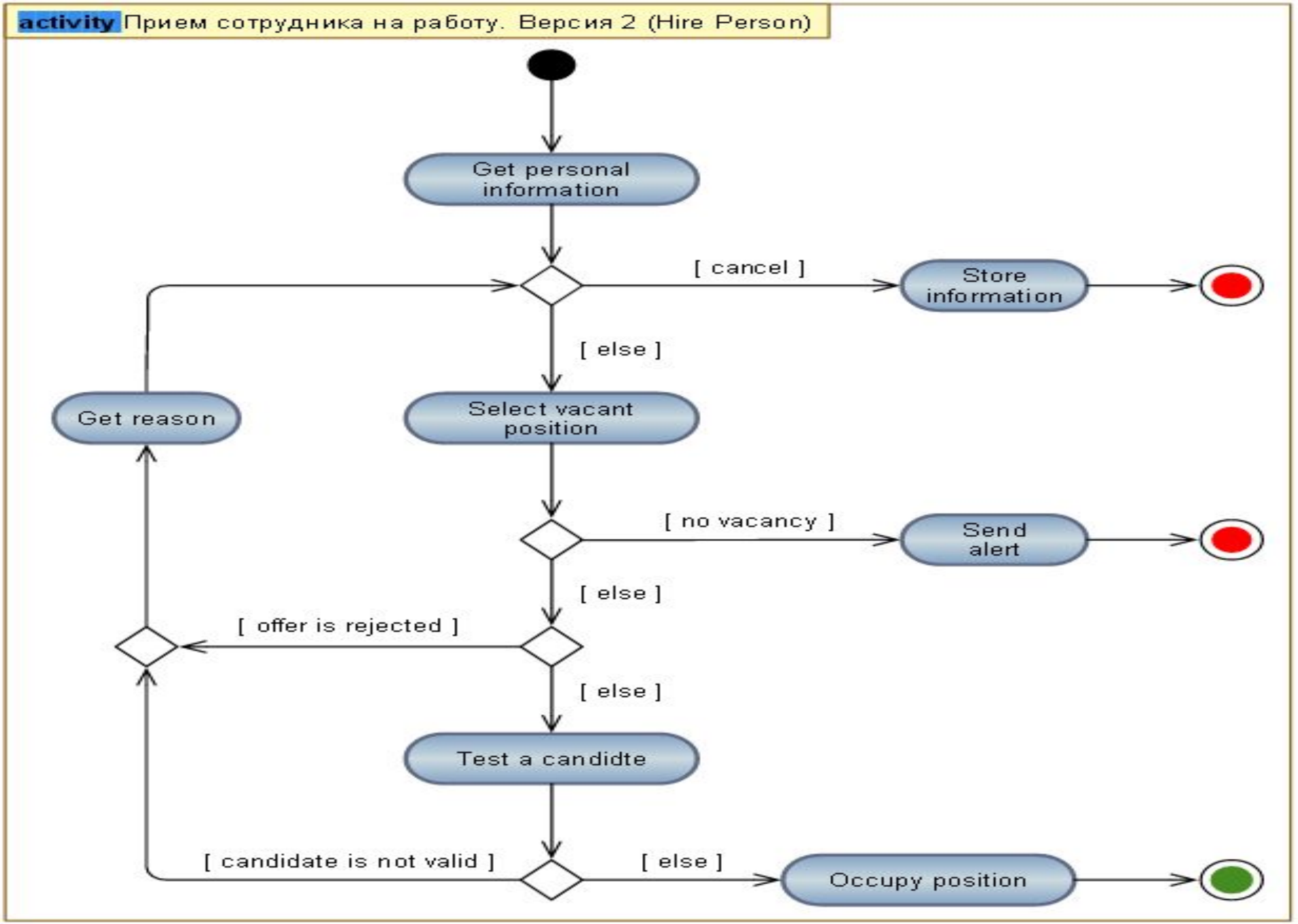


Рис. 14 - **Диаграмма последовательности для типового сценария**

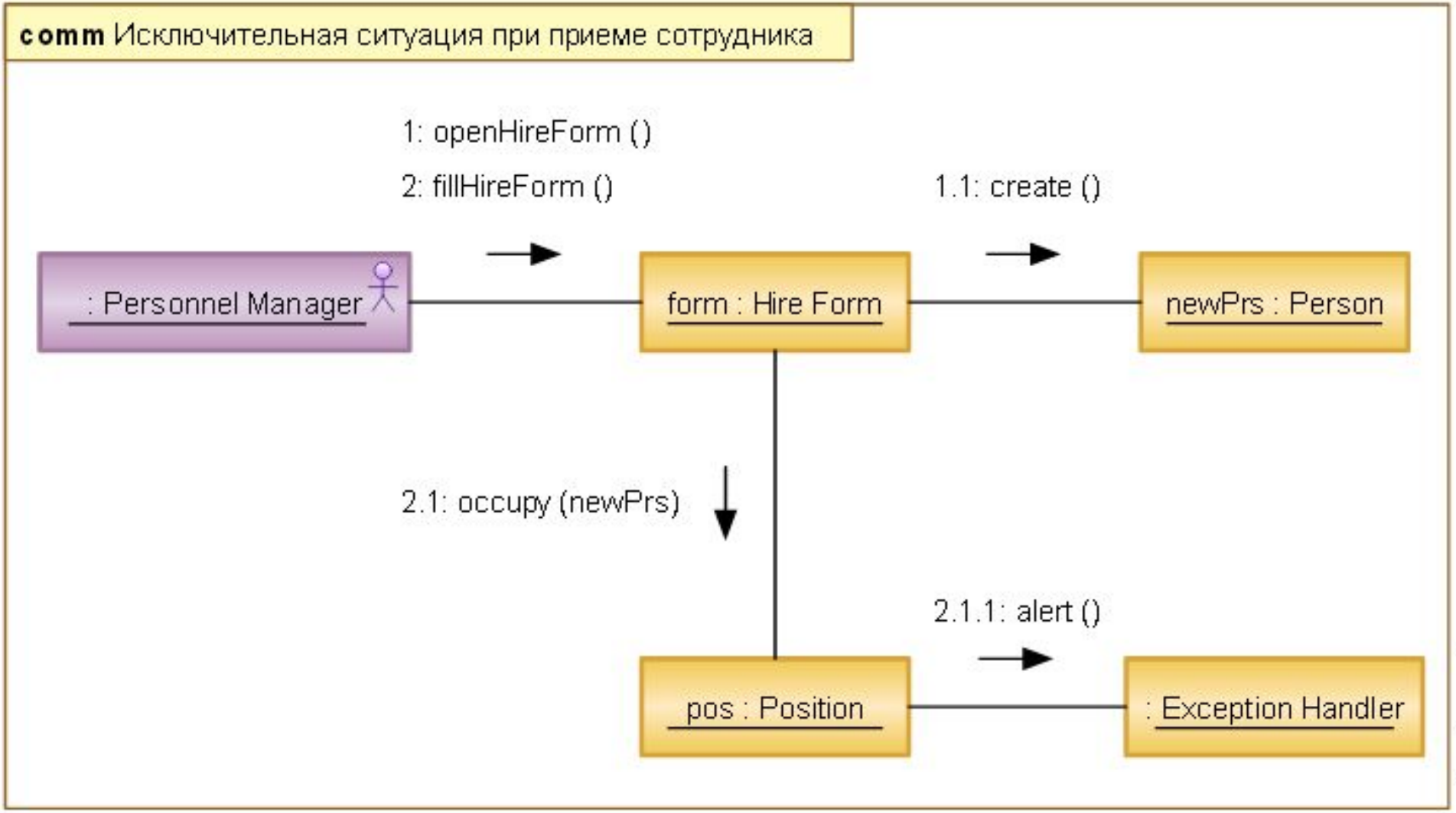


Рис. 15 - **Диаграмма коммуникации для исключительной ситуации**

Выводы

Реализация вариантов использования диаграммами взаимодействия является наиболее трудоемким и сложным методом, но этот метод лучше всего согласован с объектно-ориентированным подходом и в наибольшей мере приближает нас к конечной цели, а **моделирование использования – это универсальный способ представления функциональных требований**.