6.3. Системная аналитика: прецеденты, сценарии, модели

Остап окончил свой труд, вынул из «дела Корейко»
чистый лист бумаги и вывел на нем заголовок: «ШЕЯ»
Многометражный фильм. Сценарий О. Бендера.
И.Ильф, Е.Петров. Двенадцать стульев

Функционалом системы занимается системная аналитика. После того, как в процессе бизнес-анализа построена модель предметной области, в нее встраивается программная система, для описания функционала которой необходимо:

- создать адекватное отображение структуры предметной области в системе (диаграмма классов анализа);
- описать интерфейс взаимодействия предметной области с системой (прецеденты);
- описать бизнес-процессы в системе, как продолжение бизнес-процессов предметной области (сценарии);
- база знаний фактов и свойств системы, касающихся как отдельных сценариев, так и системы в целом (функциональные требования).

Описание программной системы, включенной в бизнес процесс, на функциональном уровне называется **моделью анализа.** Между моделью предметной области и моделью анализа возможны различные варианты взаимоотношения:

- в простейших случаях можно не делать различия между предметной областью и системой. Предметная область может создаваться системой. Например, сетевая игра, социальная сеть. Материальная сторона бизнес-процессов может отсутствовать вовсе или сводиться к взаимодействию пользователей вне системы, например, передача ведомости доставки от курьера к менеджеру;
- включение системы в бизнес-процесс может поменять его структуру. Обычно это оптимизация и новые возможности, предоставляемые информационными технологиями. Это должно быть в общих чертах отражено в бизнес-архитектуре. Естественно, что модель анализа должны отражать соответствующие изменения функционала.

Диаграмма прецедентов

Прецедент (вариант использования, use case) - ограниченное по времени и функционалу атомарное взаимодействие пользователя с системой. Самым общим представлением функционала является диаграмма прецедентов - перечень атомарных сценариев взаимодействия (прецедентов) и связанных с ними пользователей (актеров) (рис.6-12). Между овалами прецедентов и фигурками актеров устанавливаются связи – ассоциации. Между двумя прецедентами может быть установлена зависимость одного из видов:

- включение (include) – целевой прецедент является частью прецедента-источника;
- расширение (extend) – целевой прецедент выполняется при определенных условиях исполнения прецедента-источника;

Между двумя актерами или двумя прецедентами может быть определено отношение обобщения (наследования) – целевой прецедент является более общим, а источник – его расширением.

На самом деле вместо термина **пользователь** применяется термин **актер (**actor – **актор, эктор).** Он имеет более широкий смысл: в качестве актеров могут выступать самые различные стимуляторы описываемого функционала:

- обычные пользователи. Реальный пользователь может соответствовать нескольким актерам, с каждым из которых связана определенная **роль** в системе. Например, при авторизации пользователь получает определенные привилегии, каждая из которых соответствует отдельной роли – актеру;
- внешние подсистемы по отношению к системе, реализующей этот функционал;
- внешние датчики/приемники сигналов, ассоциируемые с соответствующими физическими объектами управляемой системы;
- время, как актер для периодически реализуемого функционала.

Например, фоновые периодические процессы, такие как обновление или синхронизация данных могут быть описаны как внутренний актер, лишь бы у него была внятная роль**.** Такими актерами могут быть, например, автоматическое распределение заказов в службе управления вызовами такси, служба управления локальным журналом в системе управления выдачей ключей аудиторий и т.п..

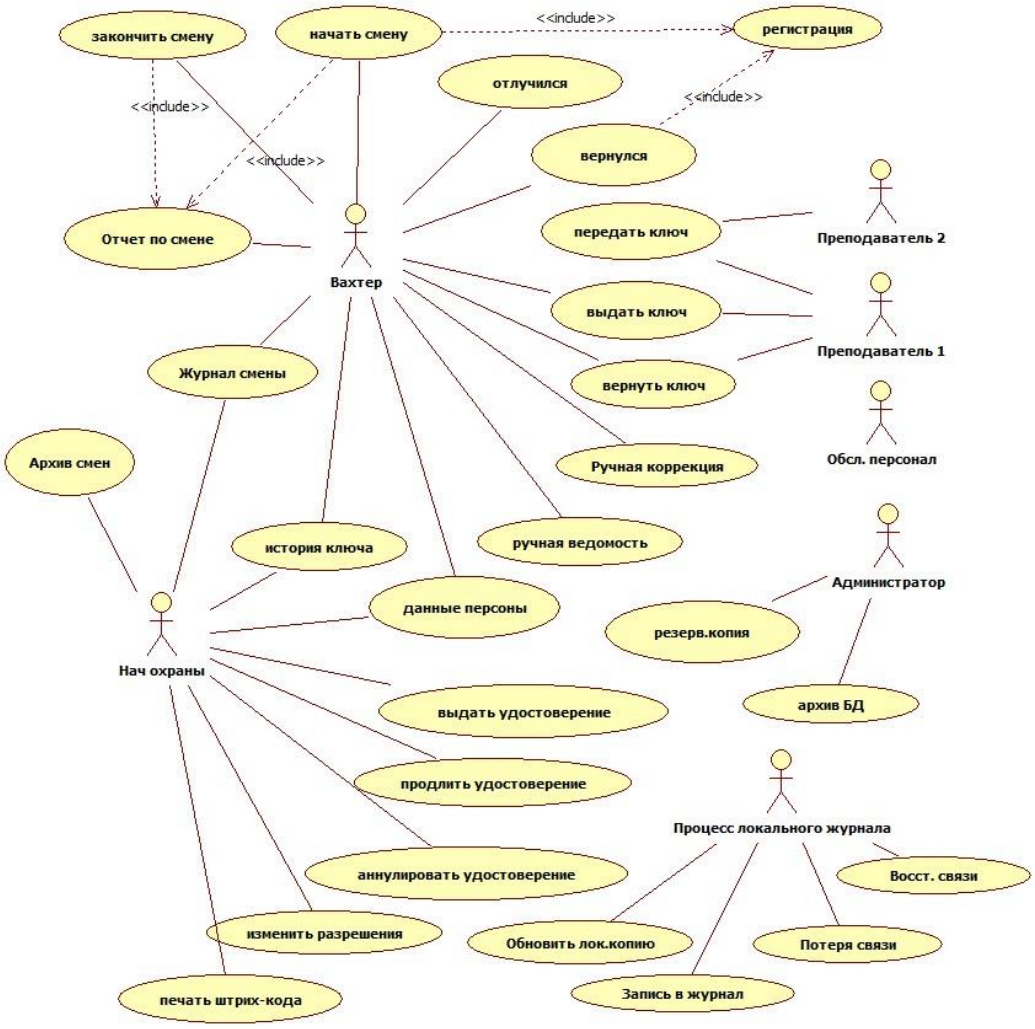

Рис. 6-12. Диаграмма прецедентов системы выдачи ключей аудиторий

Следует обращать внимание на полноту списка прецедентов, дабы не допускать пропуска видов работ, выполняемых пользователями (рис.6-13).

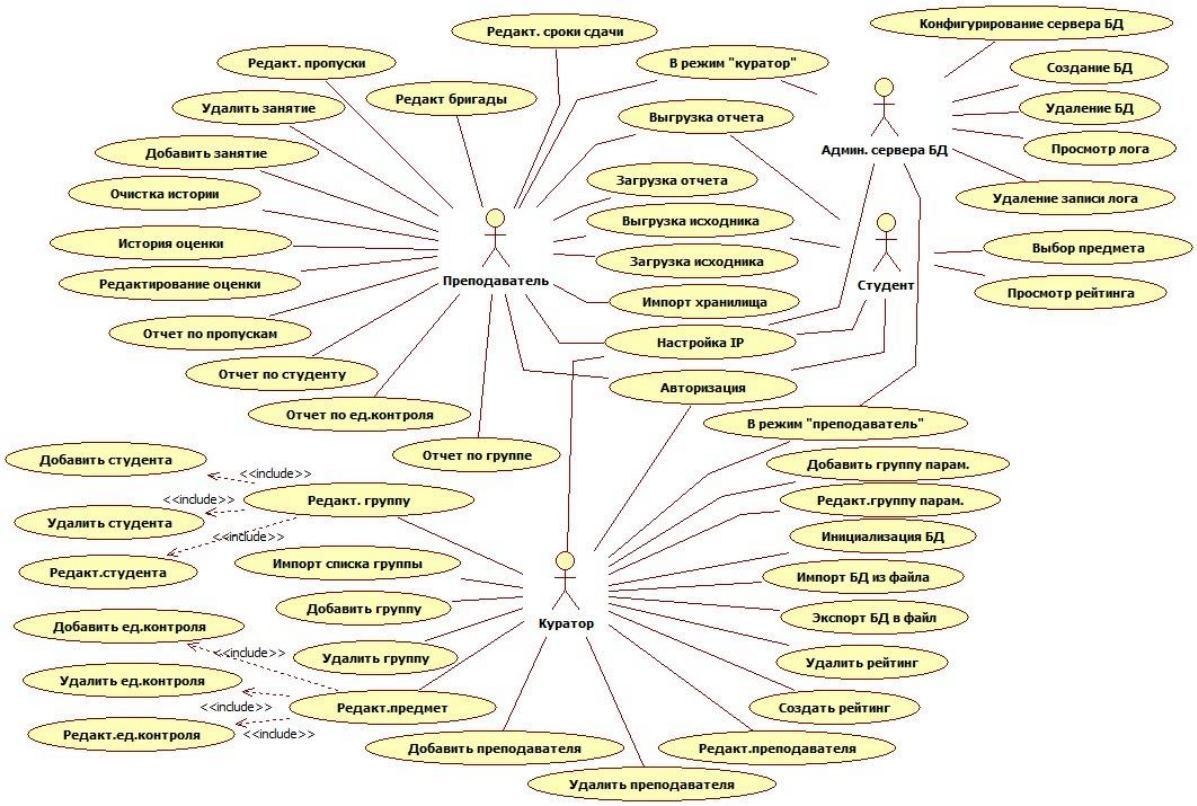

Рис. 6-13. Диаграмма прецедентов системы учета рейтинга успеваемости

Модель анализа. Диаграмма классов анализа

Рассмотрим в качестве примера, как будет эволюционировать модель анализа в системе выдачи ключей аудиторий. Исходная модель предметной области, приведенная на рис. 6-14, разработана на основе анализа бизнес-процессов (см.6.2) и выделения в нем бизнес-сущностей.

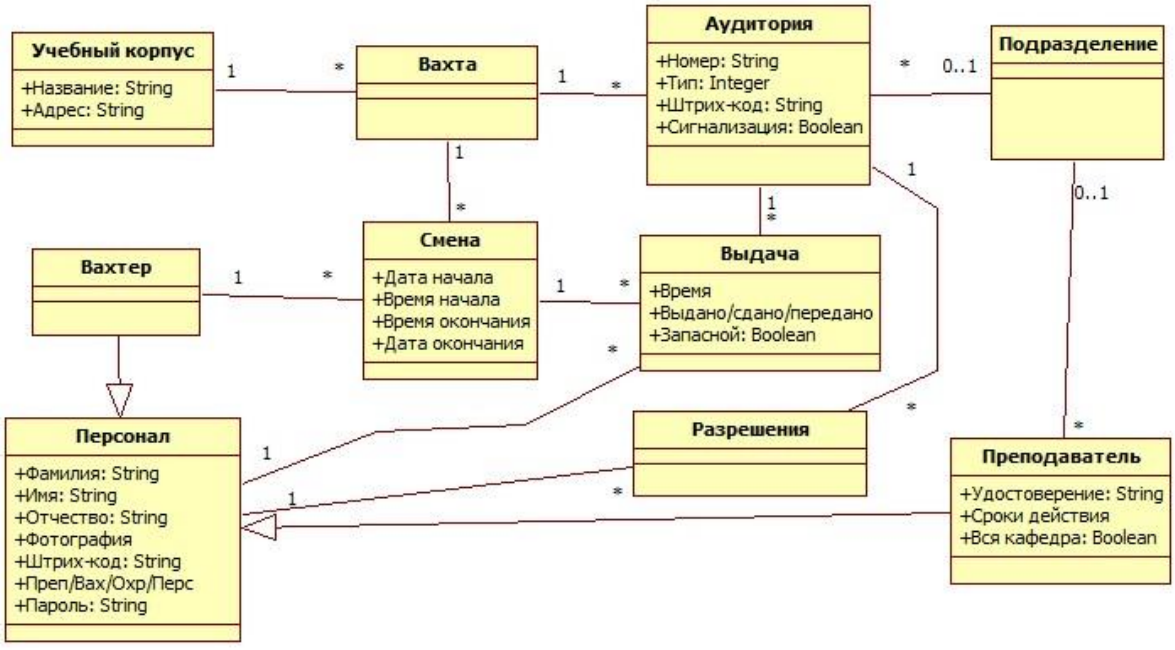

Рис. 6-14. Диаграмма классов предметной области системы выдачи ключей

В функционал системы вносятся следующие изменения, которые отражаются в структуре диаграммы классов модели анализа (рис.6-15):

- выдачу разрешений удобно производить не только по отдельным преподавателям, но и для подразделения в целом. В модели появляется базовый класс разрешений и два производных класса – разрешения **для персонала** и разрешения **для подразделений**;
- для установления состояния аудитории (открыта/закрыта) необходимо производить поиск связанного объекта с последней датой/временем классе **выдача**. К тому же возможны разные нестандартные ситуации, когда ключ сдан, а аудитория открыта, либо наоборот – ключ не сдан, но аудитория закрыта. Поэтому в класс **аудитория** необходимо продублировать данные о последней выдаче для основного и запасного ключей. Тогда класс *выдача* становится *чистым журналом событий***,** доступ к которому необязателен для ведения текущих операций.

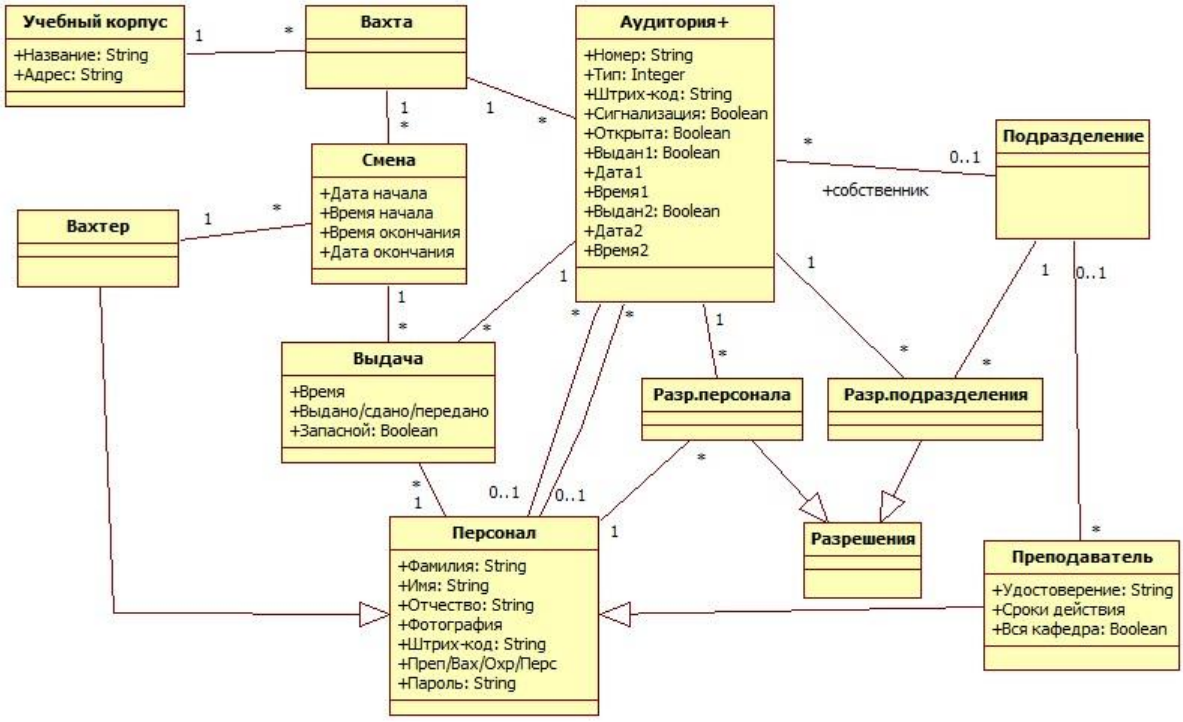

Рис.6-15. Модель классов анализа на основе модификации модели предметной области

Тестирование диаграммы классов модели анализа состоит в проверке возможности исполнения над ней прецедентов (сценариев): доступность связанных данных, правильность ассоциаций между классами.

Сценарии

Сценарий — это текстовое описание потока событий при выполнении прецедента, выражающее некий аспект поведения системы. Сценарии служат для описания функционального поведения системы.

Анализируются имена-существительные в тексте сценария. Некоторые из них будут действующими лицами, другие — объектами, а третьи — атрибутами объекта.

Сценарий содержит рад атрибутов (табл.6.1) и собственно описание потока событий в виде пар *стимул-реакция* (табл.6.1).

Таблица 6.1. Атрибуты сценария

Атрибут	**Значение**
Наименование прецедента	Выдать ключ
Актеры	Вахтер. Преподаватель
Предусловия	Вахтер зарегистрирован. Смена открыта. Открыт локальный журнал.
Активаторы	Преподаватель спрашивает ключ от аудитории
Цель	Получение ключа на руки
Краткое описание	Получение ключа с фиксацией события в БД и документированием факта выдачи

Таблица 6.2. Поток событий сценария

Действие пользователя (стимул)	**Реакция системы**
1.Вахтер: Сканирует штрих-код ключа на стенде	Если есть соединение с БД, система берет состояние ключа из БД, иначе из локального журнала.
	Если ключ уже выдан, выводится поле со справкой – когда и кому.
2. Вахтер: Сканирует штрих-код преподавателя	Если есть соединение с БД, данные берутся оттуда, иначе – из локального журнала. Если в локальном журнале нет данных, по **преподаватель заносится в локальный журнал** без разрешений на получение ключей по фамилии на штрих-коде.
	Если тип аудитории – лаборатория, то проверка разрешения. Если нет разрешения – отказ (держится 10 сек.) и завершение сценария, иначе выводится сообщение «Выдать ключ».
3. Вахтер: Сканирует штрих-код команды «выдать»	Если ключ отмечен, как выданный, выводится вопрос «Ключ на вахте?», иначе переход к п.6.
4. Вахтер: Сканирует штрих-код команды «да»	Оформляется фиктивный возврат ключа. Переход к п.6.
5. Вахтер: Сканирует штрих-код команды «нет»	Завершение сценария
6.	Если есть соединение с БД – редактируется объект *аудитория* и добавляется объект *выдача*. Редактируется локальный журнал. Печатается чек.
7. Преподаватель подписывает чек	

Архитектура или функционал?

При функциональном проектировании возникает один методологический вопрос. Если функциональное проектирование отвечает только на вопрос «что», но не отвечает на вопрос «как», то она принципиально не должна затрагивать вопросы программной архитектуры. Во-вторых, само понятие функционала также может быть размытым.

В качестве примера рассмотрим проблему повышения надежности системы выдачи ключей аудиторий. При пропадании соединения с сервером система должна продолжать работать с локальной копией журнала, синхронизируя внесенные изменения при восстановлении соединения – вести *локальный журнал*. Возможные варианты:

- считать локальный журнал элементом программной архитектуры, тогда он по определению является прозрачной (невидимой пользователю) компонентой и никак не проявляет себя на функциональном уровне;
- считать локальный журнал частью функционала. В этом случае логика его работы – предмет системной аналитики и он должен быть отражен в модели анализа и как элемент структуры, и как элемент поведения (см. приведенный выше сценарий). Что же касается конечного пользователя, то совсем не обязательно делать журнал видимым, он может быть частью скрытого функционала.

Еще одна проблема возникает при описании поведения системы на функциональном уровне. В сценариях описано, что происходит с бизнес-сущностями при его исполнении, но не указано, какие компоненты системы этим занимаются. Если такие компоненты **функциональной архитектуры** внести в описание сценариев, то получим полноценную систему функциональных классов трех видов - стереотипов:

- **boundary –** граничный класс, соответствующий актеру;
- **entity –** бизнес-сущность;
- **control –** управляющий класс, контроллер, элемент функциональной архитектуры.

Элементы функциональной архитектуры на уровне системной аналитики могут быть выделены только в самом общем виде, например, *подсистема авторизации, хранилище файлов, основная система контроля*. На этапе архитектурного проектирования они должны быть спроецированы в программную архитектуру.

С использованием этой триады можно разрабатывать UML - диаграммы *устойчивости (робастности, robust), последовательности или коммуникаций* для описания сценария реализации прецедента в системе (рис.6-16).

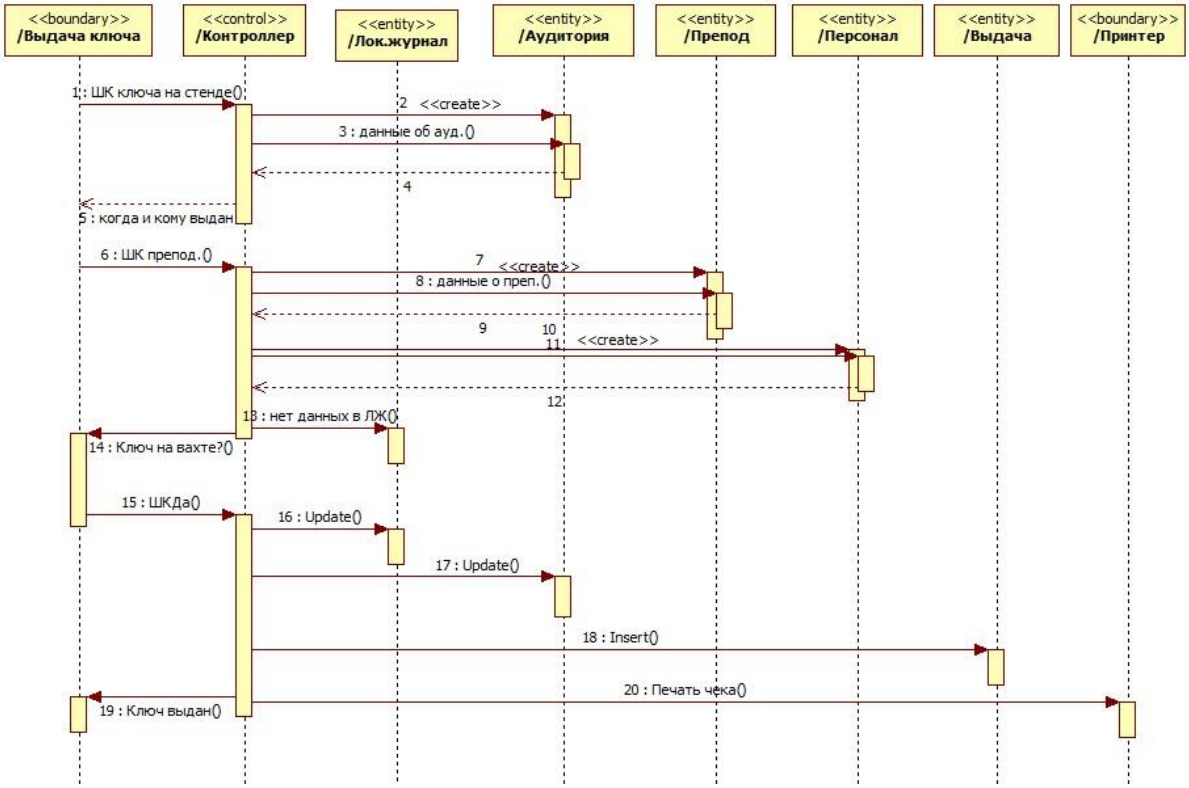

Рис.6-16. Диаграмма последовательности для прецедента

Данный пример является чисто номинальным, ибо функциональная архитектура представлена единственным объектом, т.е. фактически не проработана.

Замечание по теме. В небольших проектах, а также при использовании гибких методологий, ориентированных на коммуникации, в подробном расписыванием всех сценариев нет необходимости. Исполнитель, получив прецедент для его реализации вполне может сконструировать умозрительный сценарий. Для этого ему потребуется:

- диаграммы классов предметной области, анализа и проектирования (бизнес-объектов);
- спецификаций требований в виде иерархического справочника – базы знаний фактов, установленных для системы (см.6.4);
- прототип графического интерфейса.

Еще одним элементом функционального описания является определение состояний объектов бизнес-сущностей и актеров (граничных классов). Для класса определяется набор состояний объекта, переходы между ними производятся при исполнении прецедентов. Таким образом, получается автоматная модель, которая описывается UML-диаграммой состояний (рис.6-17).

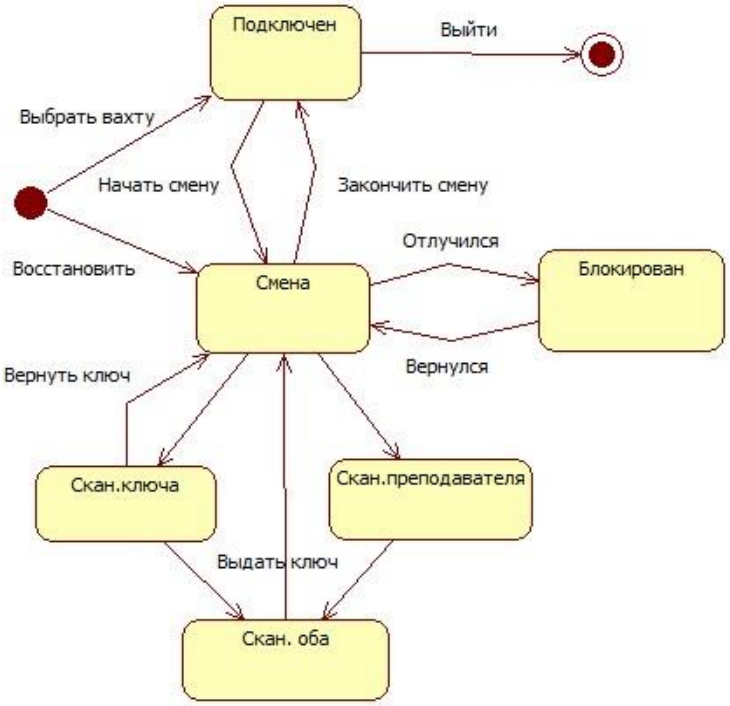

Рис.6-17. Диаграмма состояний граничного класса «вахтер»

Автоматная модель формирует интегрированное поведение сущности на принципах событийного моделирования. Состояние автомата проверяется и меняется в сценариях, реализующих прецеденты.