Моделирование прецедентов (вариантов использования)

Моделирование прецедентов (вариантов использования) традиционно разделяют на бизнес-моделирование (диаграммы бизнес-прецедентов) и системное моделирование (диаграммы прецедентов). В бизнес-моделировании делается акцент на саму организацию, тогда как в системном моделировании основное внимание нацелено на разрабатываемую систему. Элементы, используемые в диаграммах бизнес-прецедентов и в диаграммах прецедентов, а также их интерпретация приведены в таблице.

Понятие	Бизнес-моделирование	Системное моделирование
Вариант использования (прецедент, Use case)	Описывает особенности бизнеса	Описывает действия системы в контексте бизнеса
Действующее лицо (субъект, роль, Actor)	Внешнее по отношению к организации	Внешнее по отношению к системе (но может принадлежать организации)
Сотрудник (Business worker)	Принадлежит организации	Не используется

Рассмотрим концепцию системного моделирования на примере информационной системы резервирования авиабилетов. Клиенты должны получить возможность личного или интерактивного заказа билета, его замены или отказа от ранее зарезервированного билета.

Диаграмма прецедентов служит определенной цели: документирование действующих лиц (всего, что находится вне системы), вариантов использования (всего, что находится в пределах системы) и отношений (взаимосвязей) этих компонентов.

Действующие лица

Действующие лица — это все, кто взаимодействует с разрабатываемой системой. В UML действующее лицо обозначается фигуркой человека (см. рис.1).

Существуют три основных типа действующих лиц: пользователи системы, другие системы, взаимодействующие с данной, и время.

Первый тип — это физическое лицо, или пользователь. Это общепринятое действующее лицо, имеющееся практически в любой информационной системе. В системе резервирования авиабилетов действующими лицами станут люди, *непосредственно* использующие систему. Поскольку некоторые функции системы станут доступными через Интернет, клиенты смогут непосредственно обращаться к системе. Известно также, что клиент может обратиться к представителю службы по работе с клиентами для заказа билета. Представитель этой службы тоже непосредственно общается с системой, следовательно, от также является действующим лицом.

Присваивая имена действующим лицам, помните, что нужно отразить роль, а не должность. Каждый человек может играть несколько ролей. Джон Доу может быть представителем службы по работе с клиентами, но может и сам заказать для себя билет в интерактивном режиме (т.е. играть роль клиента). Использование названия роли вместо названия должности позволит получить более стабильную картину действующих лиц. Должности со временем меняются, но роли и ответственности сохраняются, хотя и переходят к другому человеку. Используя роли, вам не придется обновлять модель при любом изменении в должностном расписании.

Вторым типом действующего лица является другая система. Например, системе резервирования авиабилетов необходимо взаимодействовать с внешним приложением для проверки кредитной карточки клиента. В этом случае действующим лицом становится внешнее приложение. Эта система не будет изменяться, поскольку не попадает в рамки текущего проекта, хотя и взаимодействует с проектируемой системой. Любые подобные системы, находящиеся вне разрабатываемого проекта, станут действующими лицами.

Третий тип действующего лица — это время, поскольку от времени зависит запуск каких-либо событий в системе. Например, в рамках рекламной кампании можно выиграть бесплатный авиабилет. Ежедневно в 15:00 система автоматически и случайным образом выбирает одного из клиентов, который получает билет бесплатно. Время находится вне нашего контроля, поэтому становится действующим лицом.

Прецеденты

Прецеденты — описание "на высоком уровне" функций, предоставляемых системой, т.е. они иллюстрируют, как можно использовать систему.

Продолжим пример с системой бронирования авиабилетов. Кроме действующих лиц необходимо определить прецеденты. Безразлично, что будет специфицировано первым — более того, можно одновременно выявлять и действующих лиц, и прецеденты. Для описания прецедентов необходимо

ответить на вопрос: "В чем особенности работы системы, имеющие важное значение для внешнего мира?" Из приведенного выше краткого описания системы ясно, что она обязана обеспечивать продажу билетов, изменение бронирования и отмену заказа на билет. Эти условия становятся хорошими кандидатами для прецедентов, поскольку каждое такое действие имеет значение для конечных пользователей системы. В анализ не включаются некоторые прецеденты из унаследованной системы, например "Получить информацию о рейсе". Это выходящая из зоны анализа логика, которая не заботит конечного пользователя. Поэтому нельзя рассматривать такие информационные элементы, как прецеденты. С другой стороны, действия "Продажа билета", "Изменение заказа" или "Отмена заказа" имеют значение для конечного пользователя с точки зрения описания системы "на высоком уровне", поэтому такие действия должны быть отражены в модели через прецеденты. В UML прецедент изображается в виде овала (см. рис.1).

Преимуществом описания системы через прецеденты является способность отделения реализации системы от побудительных причин для ее разработки. Можно сконцентрироваться на действительно важных аспектах — удовлетворении требований и ожиданий клиента, которые превалируют над деталями реализации. Анализ прецедентов позволит клиенту увидеть то, что он получит от системы, и согласовать границы действия системы в самом начале разработки проекта.

Применение прецедентов немного отличается от традиционных способов разработки. Разделение проекта по прецедентам ориентировано на процессы в системе, но не на их реализацию. В этом проявляется отличие данного метода от часто используемого декомпозиционного подхода. Хотя функциональная декомпозиция также предназначена для последовательного деления проблемы на части, с которыми будет работать система, прецеденты акцентируют то, что ожидает от системы клиент.

В начале работы над проектом возникает вопрос: "Как выявить прецеденты?" Прецеденты не зависят от реализации и предоставляют высокоуровневое описание системы. Обсудим каждую часть этого определения.

Во-первых, прецеденты не зависят от реализации. Прецеденты заостряют внимание на том, *что* должна делать система, а не на том, *как* она будет это делать.

Во-вторых, прецеденты дают высокоуровневую картину системы. Набор прецедентов должен ясно и просто показать пользователю всю систему. Прецедентов не должно быть много, чтобы пользователю не приходилось знакомиться с большим объемом документации только для того, чтобы узнать, что делает эта система. В то же время набор прецедентов обязан предоставить полное описание системы. В типичных системах выявляется от 20 до 70 прецедентов (если же их будет 3000, то теряется преимущество простоты описания). Допустимо применять разные типы отношений, чтобы при необходимости подразделять сложные прецеденты. Можно и группировать прецеденты в пакеты, чтобы упорядочить структуру описания.

Наконец, прецеденты должны акцентировать то, что получит пользователь от системы. Каждый вариант использования должен представлять собой законченную транзакцию между пользователем и системой, причем результат транзакции должен иметь важное значение для пользователя. Прецеденты именуются по терминологии пользователей, а не техническими названиями, которые не всегда понятны работающим с системой людям. Не должно быть прецедента "Взаимодействие с кредитной системой банка для проверки номера кредитной карточки". Клиент пытается купить билет, поэтому разумно ввести прецедент "Покупка билета" или "Купить билет". Обычно название прецедента приводится в формате "<отглагольное существительное><дополнение> " или "<глагол><существительное>" и определяет то, что в результате получает клиент. Пользователю безразлично количество систем, с которыми взаимодействует его система, какие специфические операции выполняются внутри системы и сколько строк кода будет написано для проверки кредитной карточки Visa. Пользователю важен только покупаемый им билет, поэтому акцентируются результаты действия системы, но не операции, приводящие к такому результату.

Получив окончательный список прецедентов, следует задуматься о его полноте и попытаться ответить на вопрос: существует ли для любого функционального требования хотя бы один прецедент? Если для требования нет прецедента, то такое требование не будет реализовано в системе.

Отношения

До сих пор прецеденты и действующие лица рассматривались отдельно. Теперь для получения более полной картины системы нужно отразить отношения между ними.

Ассоциативные отношения служат для показа взаимосвязей между прецедентом и действующим лицом.

Между прецедентами возможны отношения трех типов: включающие, расширяющие и обобщенные. Обобщенные отношения демонстрируют некоторые свойства прецедентов.

Между действующими лицами возможны только обобщенные отношения.

Ассоциативные отношения

Отношения между действующими лицами и прецедентами *ассоциативны*. В UML ассоциативные отношения показывают стрелкой:

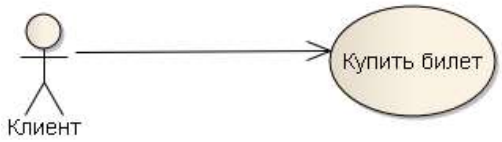

Рис.1

На рис.2 прецедент инициирует взаимодействие с действующим лицом "Кредитная система". Во время исполнения прецедента "Купить билет" система бронирования инициирует обращение к кредитной системе для проверки кредитной карточки и завершения транзакции. Хотя информация передается в обоих направлениях (от кредитной системы и к ней), стрелка показывает только тот объект, который инициирует коммуникацию.

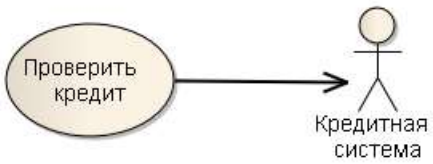

Рис.2

За исключением включающих и расширяющих отношений, любой прецедент обязан инициироваться действующим лицом.

Включающие отношения

Включающие отношения позволяют одному прецеденту предоставлять функции другим прецедентам. Такие отношения необходимы в двух случаях.

Во-первых, если несколько прецедентов имеют во многом идентичную функциональность, ее можно разделить на отдельные прецеденты, которые включаются (входят в состав) общего прецедента.

Во-вторых, включающие отношения помогут в ситуации, когда один из прецедентов обладает необычно широкой функциональностью. В этом случае в модели появятся два меньших прецедента.

Включающие отношения отмечаются пунктирной стрелкой со словом «include» (см. рис. 3).

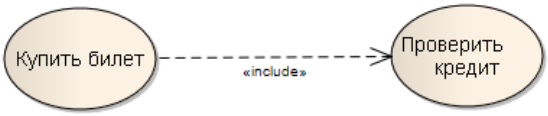

Рис. 3

В этом примере прецедент "Проверить кредит" отвечает за проверку правильности номера кредитной карточки и наличия на ней достаточной суммы для завершения транзакции. Эта функциональность применяется и в процессе "Купить билет" и в процессе "Изменить заказ", поэтому два прецедента связаны включающим отношением.

Включающие отношения предполагают, что один прецедент ***всегда пользуется функциональностью другого прецедента.*** Вне зависимости от последовательности исполнения прецедента "Купить билет" всегда выполняется прецедент "Проверить кредит".

Расширяющие отношения

В отличие от включающих расширяющие отношения позволяют одному варианту использования дополнить свою функциональность за счет другого. Эти отношения похожи на включающие, поскольку оба типа отношений выделяют некоторые функции в отдельном варианте использования.

В UML расширяющие отношения отмечаются пунктирной стрелкой со словом «extend» (см. рис. 4).

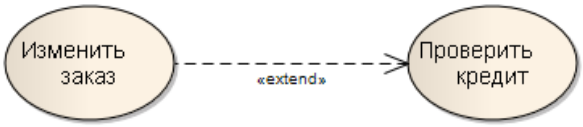

Рис. 4

В этом примере прецедент "Проверить кредит" расширяет прецедент "Изменить заказ". Во время

исполнения прецедента "Изменить заказ" прецедент "Проверить кредит" выполняется тогда и только тогда, когда происходит изменение количества заказанных мест, иначе прецедент "Проверить кредит" не запускается.

Учитывая необязательность выполнения прецедента "Проверить кредит", между двумя прецедентами возникают расширяющие отношения. Стрелка направлена от обязательного прецедента "Изменить заказ" к необязательному "Проверить кредит".

Обобщенные отношения

Обобщенные отношения служат для показа общих черт прецедентов или действующих лиц. Например, могут существовать клиенты двух категорий: корпоративные и индивидуальные. Для моделирования такого отношения служит нотация, показанная на рис. 5.

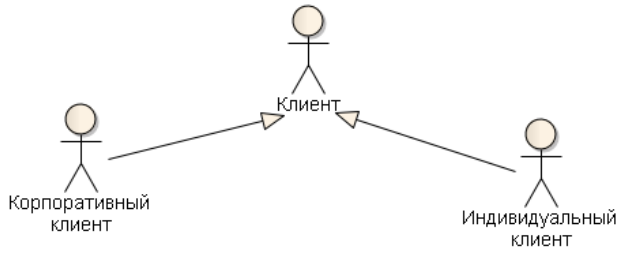

Рис. 5

На диаграмме показаны клиенты двух типов: корпоративные и индивидуальные. Поскольку соответствующие действующие лица могут непосредственно формироваться как экземпляры, они называются конкретными. Если же действующее лицо не может создаваться в виде экземпляра, то оно является абстрактным. Абстрактное действующее лицо служит только для показа существования конкретных действующих лиц.

Не всегда необходимо формировать обобщенные отношения. В общем случае они необходимы только при различиях в поведении разных действующих лиц с точки зрения анализа системы. Если корпоративный клиент будет инициировать некоторые прецеденты, которые не способен инициировать индивидуальный клиент, то обычно не требуется обобщения действующего лица. Если же оба типа клиентов выполняют одинаковые прецеденты, возможно, потребуется обобщение. Если оба типа выполняют одинаковые прецеденты, но немного по-разному, то обобщения опять не потребуется. Эти небольшие отличия документируются в спецификации нужного прецедента. Если обобщенные отношения будут полезны команде разработчиков, то следует их ввести, иначе не рекомендуется перегружать диаграмму ненужными подробностями. Это же справедливо для прецедентов. Если есть некоторый набор функций, от которого отталкиваются несколько прецедентов, можно создать обобщенный прецедент и наследовать его свойства в других прецедентах через обобщенные отношения.

Спецификация прецедентов

Прецеденты начинают описание того, что будет делать система. Однако для реального проектирования системы потребуются более подробные данные, которые отражены в спецификациях прецедентов. Спецификация прецедента должна подробно определять то, что будут делать с системой пользователи, и то, что делает сама система.

Несмотря на детализацию, спецификация прецедента остается независимой от реализации. Целью такой спецификации является описание действий системы, а не того, *как* выполняются эти действия. Обычно спецификация содержит:

- ***Краткое описание***
- ***Действующие лица***
- ***Предусловия (Preconditions)***
- ***Основной поток событий***
- ***Альтернативный поток событий***
- ***Постусловия (Postconditions)***

Познакомимся с этими компонентами.

Описание

Любой прецедент должен иметь краткое описание, объясняющее действия в этом прецеденте. Описанием прецедента "Купить билет" может быть: "Позволяет клиенту получить информацию о рейсах, узнать о наличии свободных мест и произвести покупку билета с оплатой по кредитной карточке".

Описание должно быть кратким, но в него необходимо включить сведения о разных типах пользователей, выполняющих данный прецедент, и ожидаемый результат. Во время работы над проектом

(особенно если проект сложный) эти описания будут напоминать членам команды, почему тот или иной прецедент был включен в проект и что он должен делать. Четко документируя таким образом цели каждого прецедента, можно уменьшить неразбериху, возникающую среди разработчиков.

Предусловия

В *предусловиях* прецедента перечислены все условия, который должны быть выполнены ***прежде, чем данный прецедент начнет работать***. Например, предусловием может стать исполнение другого прецедента, внесение другими прецедентами в базу данных какой-либо информации, наличие у пользователя определенных прав доступа, требуемых для запуска данного прецедента. Не у всех прецедентов есть предусловия.

Диаграммы прецедентов не предназначены для показа последовательности исполнения прецедентов, но предусловия позволят документировать подобную информацию, например, необходимость последовательного выполнения двух прецедентов.

Основной и альтернативный потоки событий

Конкретные детали прецедентов отражены в основном и альтернативном потоках событий. Поток событий — это поэтапное описание того, что происходит во время выполнения прецедента. Поток событий уделяет внимание тому, *что* делает система, причем описывает это с точки зрения пользователя. Основной и альтернативный потоки событий содержат:

- Процедуру начала прецедента
- Различные этапы исполнения прецедента
- Нормальный (основной) поток событий в прецеденте
- Все отклонения от основного потока событий, которые называются альтернативными потоками событий
- Все потоки ошибок
- Процедуру завершения прецедента

Помимо описания потоков событий в текстовом виде часто используются диаграммы Деятельности. В этом разделе будут обсуждаться только текстовые описания, а диаграммы будут рассмотрены позже.

Существуют потоки трех типов: основной (primary), альтернативный (alternate) и поток ошибок (error flows). Основной поток описывает наилучший сценарий либо наиболее используемый путь исполнения прецедента. Для покупки билета таким потоком станет процесс успешной покупки. Альтернативный поток специфицирует отклонения от основного потока, которые не рассматриваются как ошибочные. Например, кредитная карточка клиента может оказаться недействительной либо он закажет билет на несуществующий рейс. Оба эти сценария допустимы и должны обрабатываться системой, поскольку нет никаких причин для того, чтобы считать саму систему неправильной. Наконец, потоком ошибок считается девиация от основного или альтернативного потока, которая рассматривается как условие формирования ошибки. Например, система не сможет проверить кредитную карточку или наличие свободных мест на определенный рейс. Поток ошибок описывает проблемы в самой системе.

Продолжим пример с прецедентом "Купить билет" и обсудим потоки по каждому из сценариев.

Основной поток

Этапы основного потока событий:

1. Прецедент начинается с выбора клиентом режима показа информации о рейсах.
2. Система показывает сведения об аэропортах отправления и назначения, а также датах вылета и возвращения (для билета в обратную сторону).
3. Клиент вводит название городов отправления и назначения, дату вылета и возвращения.
4. Система показывает список доступных рейсов, включая стоимость билетов.
   A1: Нет нужного рейса.
5. Пользователь выбирает рейс, билет на который он хотел бы зарезервировать.
6. Система показывает нее доступные варианты стоимости билетов на этот рейс.
7. Пользователь выбирает категорию билета для резервирования.
   A2: Пользователь выбрал бесплатный билет, предоставляемый членам клуба постоянных клиентов.
8. Система показывает цену, которую должен заплатить пользователь.
9. Пользователь подтверждает цену.
10. Система запрашивает тип кредитной карточки, ее номер, имя владельца и дату завершения срока действия.
11. Пользователь вводит тип кредитной карточки, ее номер, имя владельца и дату завершения срока действия.
12. Система подтверждает продажу по кредитной карточке.
    A6: Счет пользователя не найден
    A7: Недостаточно денег на счете
    E1: Кредитная система недоступна
13. Система резервирует место для пользователя на выбранный рейс

14. Система генерирует и показывает пользователю код подтверждения
15. Пользователь подтверждает получение кода.
16. Прецедент завершается.

Альтернативные потоки

A1: Нет нужного рейса

1. Система выводит сообщение, что для введенных города отправления и значения, а также для указанных дат нет мест на рейсах авиакомпании
2. Пользователь подтверждает просмотр сообщения.
3. Поток возвращается па этап 2 основного потока.

A2: Пользователь выбрал бесплатный билет, предоставляемый членам клуба постоянных клиентов

1. Система запрашивает идентификационный номер в клубе постоянных клиентов.
2. Пользователь вводит этот номер.
3. Система подтверждает правильность введенного номера.
   A3: Неправильный идентификационный номер
4. Система подтверждает, что расстояние, которое налетал пользователь на самолетах авиакомпании, позволяет предоставить бесплатный
   A4: Недостаточное расстояние для предоставления бесплатного билета
   A5: Нет бесплатных билетов
5. Цена билета устанавливается в $0.
6. Поток возвращается на этап 8 основного потока.

A3: Неправильный идентификационный номер

1. Система выводит сообщение о некорректном идентификационном номере.
2. Пользователь повторяет ввод номера ИЛИ отказывается от запроса на бесплатный билет.
3. Если пользователь повторяет ввод номера, то поток возвращается на этап 1 альтернативного потока A2.
4. Если пользователь отказывается от запроса на бесплатный билет, поток возвращается на этап 6 основного потока.

A4: Недостаточное расстояние для предоставления бесплатного билета

1. Система выводит сообщение о том, что расстояние недостаточно для предоставления бесплатного билета. В сообщении указано это расстояние и расстояние, необходимое для предоставления бесплатного билета
2. Поток возвращается на этап 6 основного потока.

A5: Нет бесплатных билетов

1. Система выводит сообщение о том, что бесплатные билеты на выбранный рейс не предоставляются.
2. Поток возвращается на этап 6 основного потока.

A6: Счет пользователя не обнаружен

1. Система выводит сообщение о том, что счет пользователя не обнаружен
2. Поток возвращается на этап 10 основного потока.

A7: Недостаточно денег на счете

1. Система выводит сообщение о том, что остаток на кредитной карточке не позволяет завершить транзакцию.
2. Поток возвращается на этап 10 основного потока.

Потоки ошибок

E1: Кредитная система недоступна

1. Система выводит сообщение о недоступности кредитной системы.
2. Поток возвращается на этап 10 основного потока.

Обратите внимание на структуру потоков событий: пользователь что-нибудь выполняет, система откликается, затем процесс повторяется. Следование этой структуре помогает полностью понять взаимодействие между пользователем и системой. Во время документирования потока событий рекомендуется применять нумерованный список (как в показанном примере) либо воспользоваться делением на абзацы, диаграммами или блок-схемами. Можно задокументировать поток событий в

структуре пользователь—система в виде таблицы:

Действие пользователя	**Ответ системы**
Выбор варианта просмотра информации о рейсах	Показ городов и дат отправления и прибытия
Ввод городов и дат отправления и прибытия	Показ номера рейса, время вылета и время прибытия для всех подходящих рейсов
……	…….

Уровень *детализации*

Классический вопрос при документировании потока событий — это вопрос об уровне детализации. Для ответа на него следует помнить о том, для кого предназначен разрабатываемый документ. Это три категории пользователей:

1. Заказчики просматривают этот документ, чтобы проверить, насколько точно он отражает их предположения о системе. Документ о потоке событий должен быть достаточно детализированным, чтобы разработчик и пользователи системы имели её одинаковое видение. Чем меньше будет подробностей, тем вероятнее расхождение в предположениях о возможностях системы. В то же время не следует показывать детали реализации, которые непонятны или неинтересны пользователям. Кратким ответом, с точки зрения пользователя, может быть такой: "Уровень детализации должен быть настолько высоким, чтобы без него стала невозможной реализация". Попробуйте избегать фраз, подобных "система учитывает постоянных клиентов". Что значит "учитывает"? Необходимо, чтобы разработчики и заказчики одинаково понимали все описания потоков событий.

2. Разработчики системы используют потоки событий для проектирования и реализации системы. Поэтому поток должен давать достаточное количество информации для понимания последовательности событий, происходящих в прецеденте. Хотя поток событий не привязан к реализации системы (попробуйте не использовать понятия "меню", "окно", "древовидная структура" и т.д.), он предоставляет достаточно информации для понимания того, как должна действовать система. Необходимо избегать несогласованности между видением системы со стороны пользователей и со стороны разработчиков.

3. Сотрудники контроля качества используют потоки событий для создания тестовых сценариев. Поскольку поток событий описывает пошаговые действия системы, его описание становится исходным материалом для сравнения того, что система делает, и того, что она *должна* делать. Поток событий, конечно, не является тестовым сценарием, но служит исходным материалом для его создания.

Рассмотрим кулинарный рецепт. В нем говорится: "Возьмите два яйца", но незачем приводить подробное описание: "Откройте холодильник, возьмите с полки первое яйцо и разбейте его о край чашки ..." Для описания потока событий достаточно указать: "Проверить идентификатор пользователя", но не нужно описывать, что процесс этой проверки предполагает обращение к определенной таблице в базе данных. Акцентируется информация, которой обмениваются пользователь и система, но не детали реализации системы.

Постусловия

Постусловия — это условия, которые всегда равны true после завершения исполнения прецедента. Как и предусловия, постусловия могут добавлять информацию о порядке исполнения прецедентов. Например, если один прецедент всегда исполняется после другого, то это можно отразить в постусловии. Постусловия требуются не для всех прецедентов.

РЕКОМЕНДАЦИИ

В заключение несколько рекомендаций по созданию диаграмм прецедентов:

• Не рекомендуется моделировать связи между действующими лицами (хотя допустимо в отдельных случаях для наглядности). По определению, действующие лица находятся вне сферы действия системы. Следовательно, коммуникации между ними тоже выходят за рамки разрабатываемой системы. Для описания связи между действующими лицами служат диаграммы деятельности.

• Не моделируйте прямые связи между двумя прецедентами (хотя допустимо моделирование включающих и расширяющих отношений). Диаграмма показывает доступные прецеденты, но не документирует порядок их выполнения, поэтому не требуется указывать ассоциации между прецедентами.

• Любой прецедент должен инициироваться действующим лицом, т.е. стрелка должна быть направлена от действующего лица к прецеденту. Исключением являются включающие и расширяющие отношения.

• Считайте базу данных уровнем, находящимся ниже всей диаграммы прецедентов. Можно ввести информацию в базу данных в одном прецеденте, а затем получить эту информацию в другом прецеденте. В этом случае информационный обмен между двумя прецедентами не нужно показывать ассоциацией, а следует отразить в Предусловии и Постусловии.