[СЛАЙД 2]

Моделирование прецедентов (вариантов использования) – это другой, дополнительный способ выявления и документирования требований.

Результат этой деятельности – модель прецедентов. В этой модели че­тыре компонента:  
• Граница системы – прямоугольник, очерчивающий прецеденты для обозначения края, или границы, моделируемой системы. В UML 2 эту границу называют контекстом системы (subject).   
• Актеры – роли, выполняемые людьми или сущностями, исполь­зующими систему.  
• Прецеденты – то, что актеры могут делать с системой.   
• Отношения – значимые отношения между актерами и прецедента­ми.

[СЛАЙД 3]

Первое, что необходимо сделать при построении системы, надо определить, что является частью сис­темы и что находится вне системы. Точное определение границ системы обычно играет важную роль в выявлении функциональных (а иногда и нефункциональ­ных) требований. В UML границу системы называют контекстом системы (subject). Контекст изображается в виде прямоугольника с именем системы. Актеры размещаются вне границ блока, а прецеденты – внутри. В нача­ле моделирования прецедентов имеется лишь предварительное пред­ставление о том, где находятся границы системы. По мере выявления актеров и прецедентов контекст системы обретает все более четкие очертания.

[СЛАЙД 4]

Актер определяет роль, которую выполняет некоторая внешняя сущ­ность при непосредственном взаимодействии с данной системой. Он может представлять роль пользователя или роль, исполняемую другой системой или частью аппаратных средств, которые касаются границ системы.

При моделировании актеров необходимо помнить следующие моменты.  
• Актеры всегда являются внешними по отношению к системе, находятся вне вашего контроля.   
• Актеры взаимодействуют непосредственно с системой – так они по­могают в определении контекста системы.   
• Актеры представляют роли, исполняемые людьми или сущностями по отношению к системе, а не конкретных людей или сущностей.   
• Один человек или сущность может играть по отношению к системе множество ролей одновременно или последовательно во времени.  
• У каждого актера должно быть короткое, осмысленное с прикладной точки зрения имя.   
• Каждого актера должно сопровождать краткое описание (одна или две строчки), объясняющее, что данный актер из себя представляет с прикладной точки зрения.

[СЛАЙД 5]

Прецедент – это что-то, что должна делать система по желанию актера. Это «вариант использования» системы конкретным актером:

• прецеденты ВСЕГДА инициируются актером;

• прецеденты ВСЕГДА описываются с точки зрения актеров

Чтобы найти прецедент, надо спросить:

«Как каждый из актеров использует систему?» и «Что система делает для каждого актера?»

[СЛАЙД 6]

Прецедент определен как «Описание последовательности действий, включая альтернативные и ошибочные последовательности, которые система, подсистема или класс могут осуществлять, взаимодействуя с внешними актерами». Прецедент описывает поведение, демонстрируемое системой с целью получения значимого результата для одного или более актеров. Прецедент – это что-­то, что должна делать система по желанию акте­ра.

Обычно прецеденты рассматриваются на уровне системы, но согласно определению они могут применяться также и для описания «варианта использования» подсистемы (части системы) или даже отдельного класса. Прецеденты также могут быть очень эффективными при моде­лировании бизнес-процессов, хотя данный вопрос не рассматривается в этой книге.

[СЛАЙД 7]

Отношение «include» выносит шаги, общие для нескольких прецедентов, в отдельный прецедент, который потом включается в остальные.

Включающий прецедент мы называем базовым, а тот прецедент, который включается, включаемым. Включаемый прецедент предоставляет поведение своему базовому прецеденту. В базовом прецеденте необходимо точно указать место, где должно быть включено поведение включаемого прецедента. Синтаксис и се­ мантика отношения «include» немного напоминают вызов функции.

Отношение «include» имеет простую семантику. Базовый пре­ цедент выполняется до момента включения. Затем выполнение пере­ ходит во включаемый прецедент. По завершении включаемого прецедента управление вновь возвращается в базовый прецедент.

[СЛАЙД 8]

Отношение «extend» предоставляет возможность ввести новое поведе­ние в существующий прецедент. Базовый прецедент предоставляет набор точек расширения, в которые может быть добавлено новое поведение. А расширяющий прецедент предоставляет ряд сегментов вставки, которые можно ввести в базовый прецедент в места, указанные точками входа. Как вскоре бу­дет показано, отношение «extend» может использоваться для того, чтобы точно указать, какие именно точки расширения базового прецедента подлежат расширению. В отношении «extend» любопытно то, что базовый прецедент ничего не знает о расширяющих прецедентах, он просто предоставляет для них точки входа. Базовый прецедент абсолютно полон и без расширений. Это существенно отличает «extend» от отношения «include», где базовые прецеденты остаются неполными без включаемых прецедентов.