Анализ и проектирование на UML

Направление подготовки
“Информационные системы и технологии”

Максим Валерьевич Хлопотов

Представления

– Все аспекты моделируемой системы не удается описать с единой точки зрения.
– Моделировать сложную систему следует с нескольких различных точек зрения, каждый раз принимая во внимание один аспект моделируемой системы и абстрагируясь от остальных.
– Этот тезис является одним из основополагающих принципов UML.

Иерархия диаграмм UML

Представления

Выделяем три представления:
– представление использования (что делает система полезного?);
– представление структуры (из чего состоит система?);
– *представление поведения (как работает система?).*

Представление поведения

Определяющим признаком для отнесения элементов модели к представлению поведения является явное использования понятия времени, в частности, в форме описания последовательности событий/действий, то есть в форме алгоритма.

Описывается диаграммами *состояний* и *деятельности*, а также диаграммами взаимодействия в форме диаграмм *кооперации* и/или *последовательности*.

Представление поведения

– Модель поведения должна быть достаточно детальной для того, чтобы послужить основой для составления компьютерной программы.

– Модель поведения должна быть компактной и обозримой, чтобы служить средством общения между людьми в процессе разработки системы.

Представление поведения

– Модель поведения не должна зависеть от особенностей реализации конкретных компьютеров, средств программирования и технологий.

– Средства моделирования поведения в UML должны быть знакомы и привычны большинству пользователей языка и не должны противоречить требованиям наиболее ходовых парадигм программирования.

Представление поведения

– В UML предусмотрено несколько различных средств для описания поведения.
– Выбор того или иного средства диктуется типом поведения, которое нужно описать.

Например, для описания жизненного цикла конкретного объекта используется конечный автомат в форме **диаграммы состояний**.
При этом состояния конечного автомата соответствуют состояниям объекта (различным наборам значений атрибутов), а переходы соответствуют выполнению операций.

Представление поведения

– Диаграммы состояний можно составить не только для программных объектов — экземпляров отдельных классов, но и для более крупных конструкций, в частности, для всей модели приложения в целом или для более мелких — отдельных операций.

– Для описания последовательности выполняемых элементарных шагов при выполнении отдельной операции или реализации сложного варианта использования, удобно использовать диаграммы деятельности.

Представление поведения

– Взаимодействие нескольких программных объектов между собой описывается диаграммами взаимодействия в одной из двух эквивалентных форм (диаграммы кооперации и диаграммы последовательности).

– Для объектно-ориентированной программы поведение прежде всего определяется взаимодействием объектов, поэтому диаграммы данного типа имеют столь важное значение при моделировании поведения в UML.

Диаграмма состояний

- Диаграмма состояний — это основной способ детального описания поведения в UML. В сущности, диаграммы состояний представляют собой граф состояний и переходов конечного автомата, нагруженный множеством дополнительных деталей и подробностей.
- Диаграммы состояний в UML являются реализацией основной идеи использования конечных автоматов как средства описания алгоритмов.

Диаграмма состояний

- ***Конечный автомат (state machine)*** - модель для спецификации поведения объекта в форме последовательности его состояний, которые описывают реакцию объекта на внешние события, выполнение объектом действий, а также изменение его отдельных свойств.
- Вершинами графа *конечного автомата* являются *состояния*. Дуги графа служат для обозначения *переходов* из *состояния* в *состояние*.

Состояние

- На диаграммах состояний применяется всего один тип сущностей — состояния, и всего один тип отношений — переходы. Совокупность состояний и переходов между ними образует машину состояний.
- Машина состояний — термин, принятый в англоязычной литературе, но обозначающий тоже самое, что конечный автомат.

Состояние

- условие или ситуация в ходе жизненного цикла объекта, в течение которого он удовлетворяет логическому условию, выполняет определенную деятельность или ожидает события.

Диаграмма состояний

Состояния бывают: простые, составные, специальные

- Каждый тип состояний имеет дополнительные подтипы и различные составляющие элементы.

Переходы бывают простые и составные, и каждый переход может содержать :

- исходное состояние,
- событие перехода,
- сторожевое условие,
- действие на переходе,
- целевое состояние.

Состояние

Простое состояние имеет следующую структуру:

- имя;
- действие при входе;
- действие при выходе;
- внутренняя активность.

Состояние

- Имя состояния является обязательным. Все остальные составляющие простого состояния не являются обязательными.
- Действие при входе (обозначается при помощи ключевого слова entry) — это указание атомарного действия , которое должно выполняться при переходе автомата в данное состояние.

Состояние

- Действие при выходе (обозначается при помощи ключевого слова exit) — это указание атомарного действия, которое должно выполняться при переходе автомата из данного состояние. Действие при выходе выполняется до всех других действий, предписанных переходом, выводящим автомат из данного состояния.

Состояние

- Внутренняя активность (обозначается при помощи ключевого слова do) — это указание деятельности, которая начинает выполняться при переходе в данное состояние после выполнения всех действий, предписанных переходом, включая действие на входе.
- Внутренняя активность либо заканчивается по завершении, либо прерывается в случае выполнения перехода (в том числе и внутреннего перехода).
- В классической модели конечный автомат, находясь в некотором состоянии, ничего не делает: он находится в состоянии ожидания перехода.

Переход

- Простой переход всегда ведет из одного состояния в другое состояние.
- Существует несколько ограничений для специальных состояний: для начального состояния не может быть входящих переходов, а для заключительного — исходящих.
- В остальном переходы между состояниями могут быть определены произвольным образом.

Переход

- Прочие составляющие — событие перехода, сторожевое условие и действия на переходе не являются обязательными.
- Если они присутствуют, то изображаются в виде текста в определенном синтаксисе рядом со стрелкой, изображающей переход. Синтаксис описания перехода следующий:

Событие [Сторожевое условие] / Действие

Переход

- Событие перехода — это тот входной символ (стимул), который вместе с текущим состоянием автомата определяет следующее состояние.
- UML допускает наличие переходов без событий — такой переход называется переходом по завершении.

Переход

- Сторожевое условие — это логическое выражение, которое должно оказаться истинным для того, чтобы возбужденный переход сработал.
- Для каждого возбужденного перехода сторожевое условие проверяется ровно один раз, сразу после того, как переход возбужден и до того, как в системе произойдут какие-либо другие события.
- Если сторожевое условие ложно, то переход не срабатывает. Даже если впоследствии сторожевое условие станет истинным, переход сможет сработать только если повторно возникнет событие перехода.

Переход

- В UML предусмотрены синтаксические средства, до некоторой степени облегчающие семантически правильное построение сторожевых условий за счет более наглядного их изображения.
- Таковыми являются:
- сегментированные переходы;
- символы ветвления;
- переходные состояния;
- предикат else.

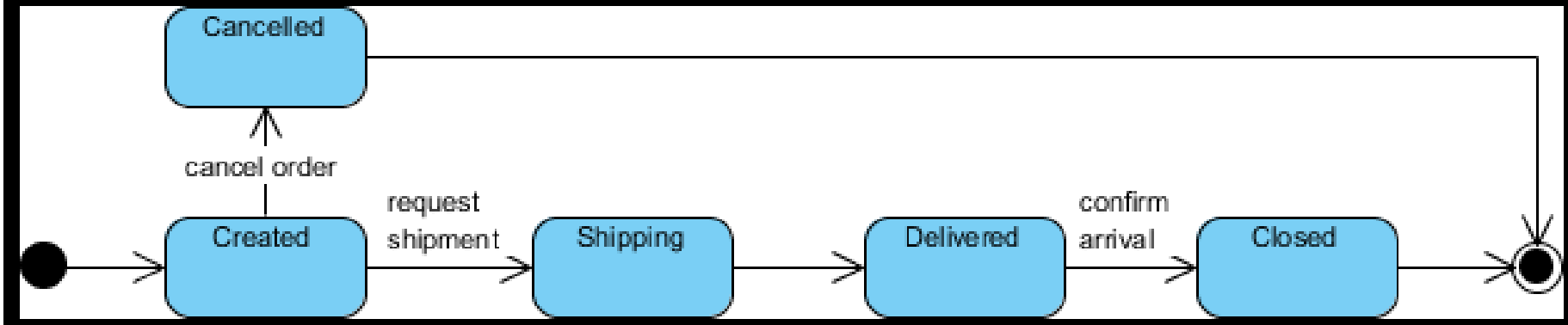
Переход

- Линия перехода может быть разбита на части, называемые сегментами.
- Сегменты перехода — части, на которые может быть разбита линия перехода.

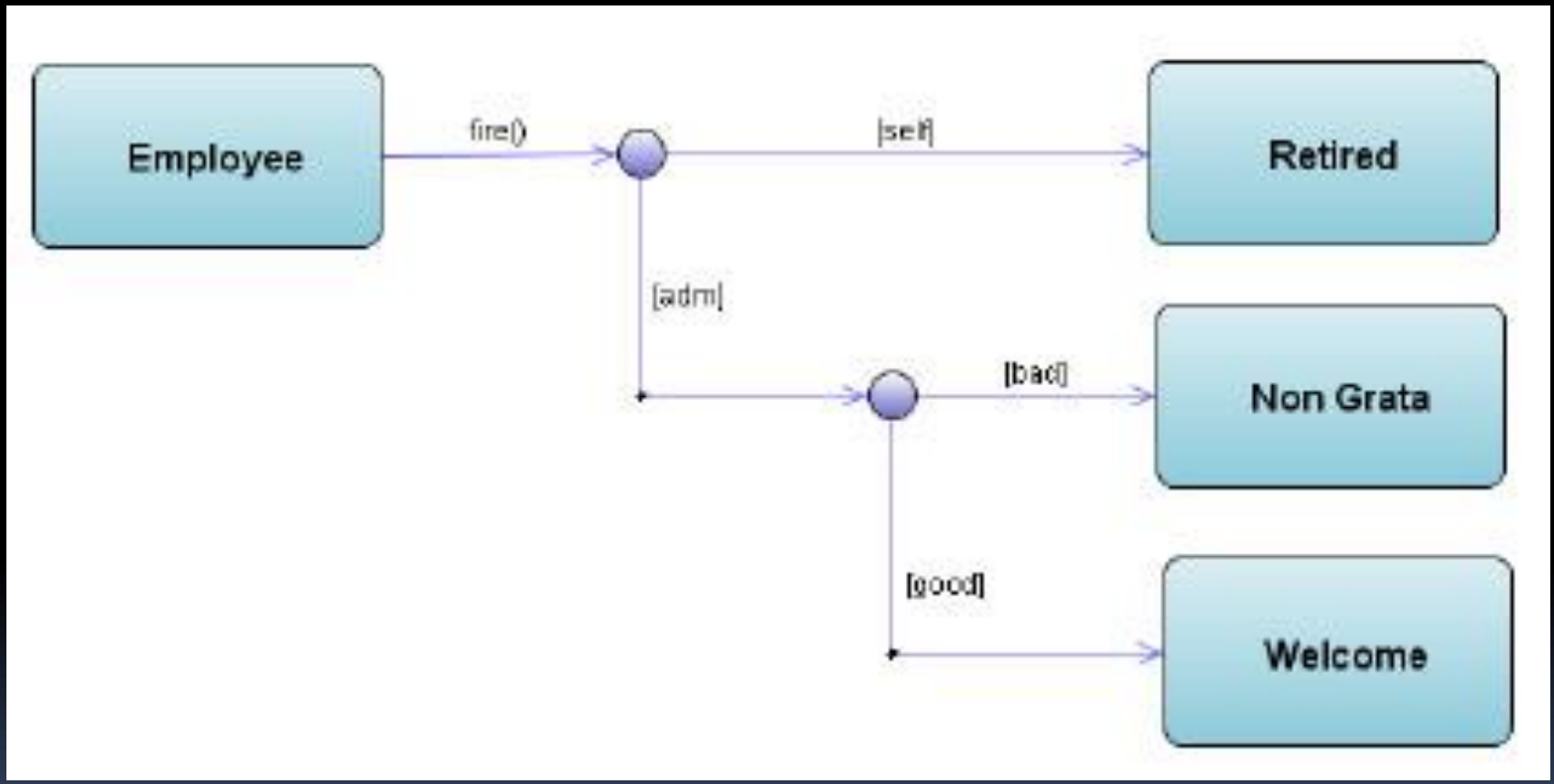
Разбивающими элементами являются следующие фигуры:

- переходное состояние (изображается в виде небольшого кружка);
- ветвление (изображается в виде ромба).

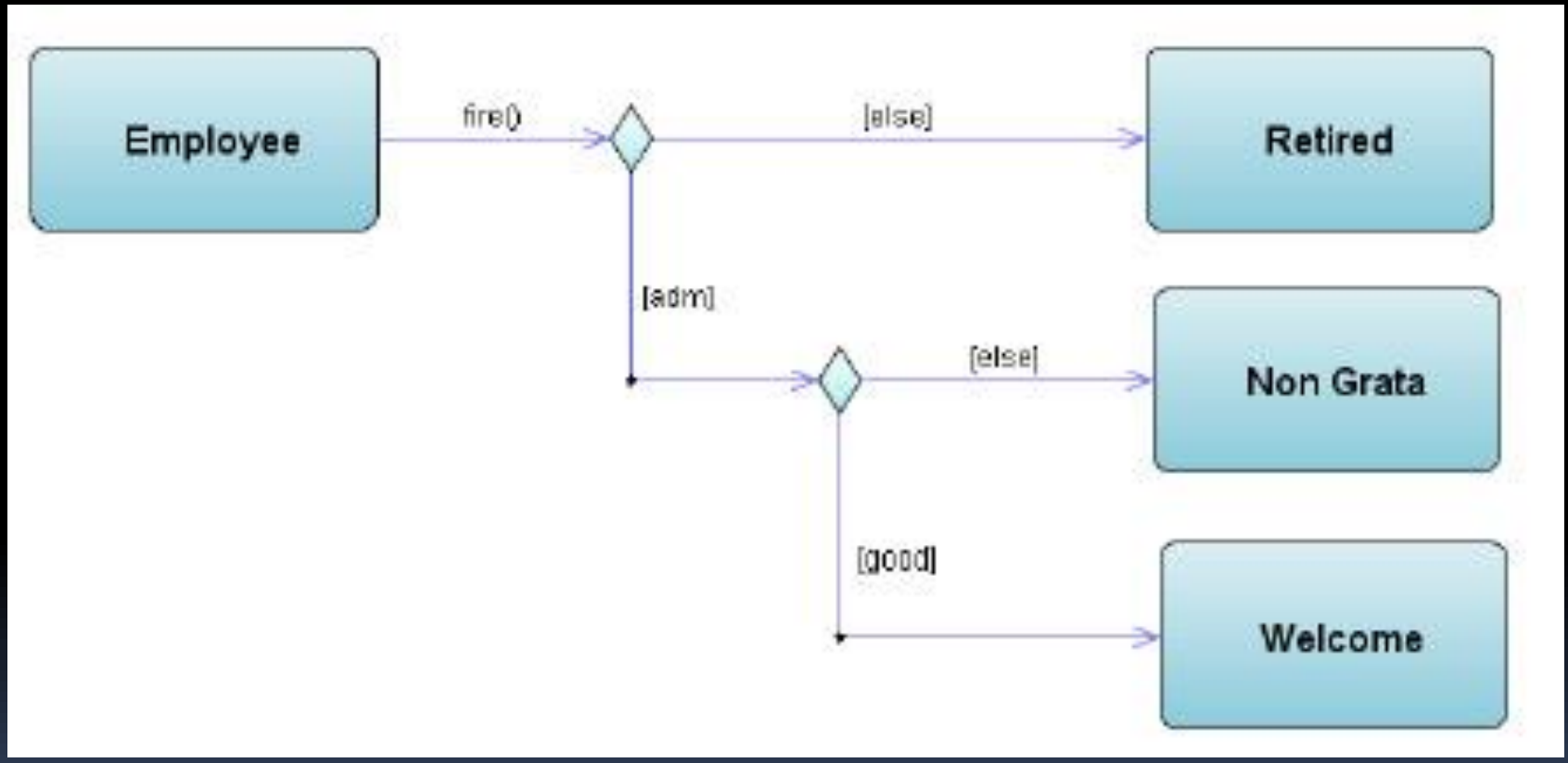
Примеры


Cancelled
cancel order
Created
request
shipment
Shipping
Delivered
confirm
arrival
Closed

Примеры



Примеры



Составное состояние

- *Составное состояние* — это состояние, в которое вложена машина состояний.
- Глубина вложенности в UML неограниченна, т. е. состояния вложенной машины состояний также могут быть составными.

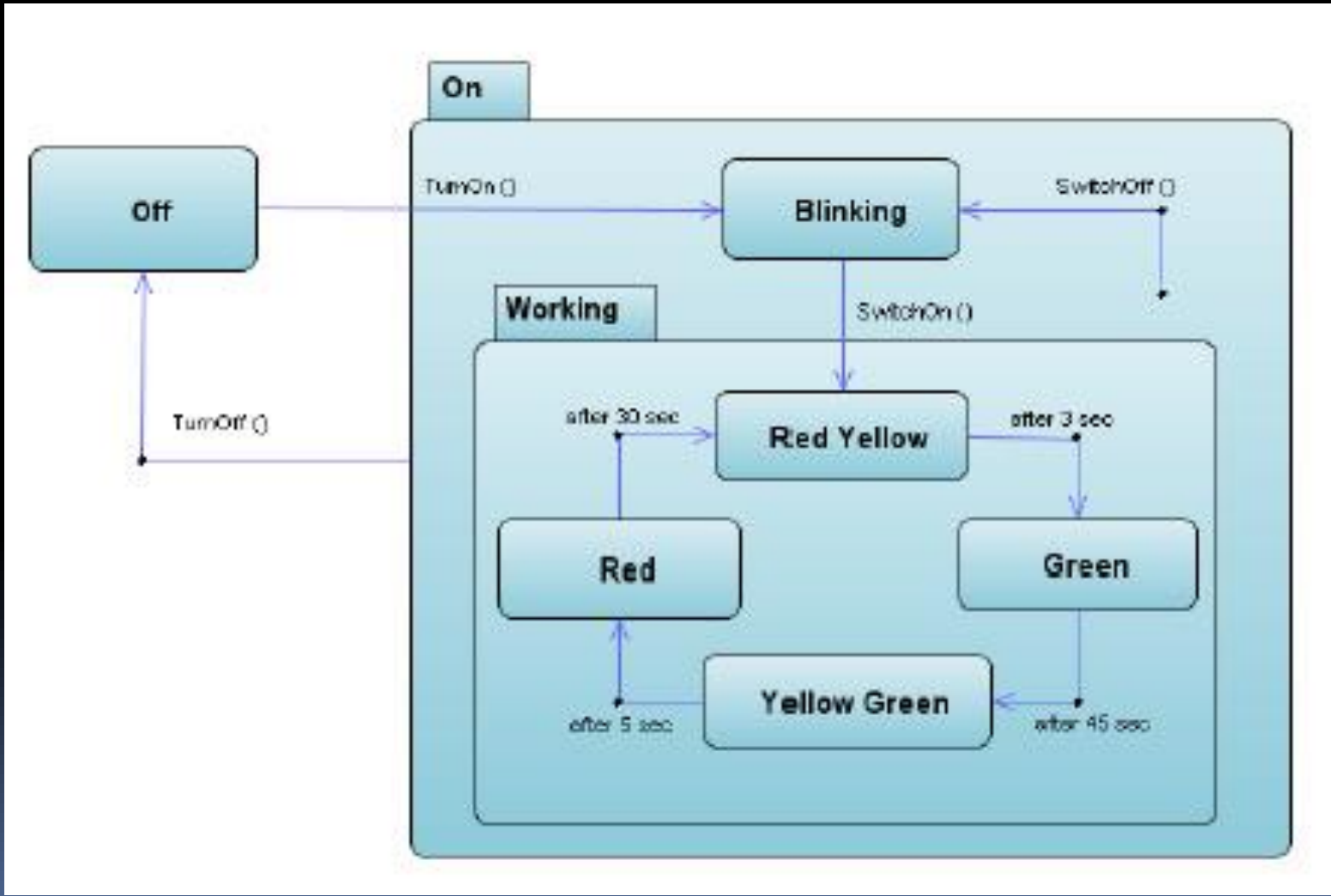
Составное состояние

- Рассмотрим все известный прибор: светофор.
- Он может находится в двух основных состояниях:
- Off — вообще не работает — выключен или сломался, как слишком часто бывает;
- On — работает. Но работать светофор может по-разному:
- Blinking — мигающий желтый, дорожное движение не регулируется;
- Working — работает по-настоящему и регулирует движение.

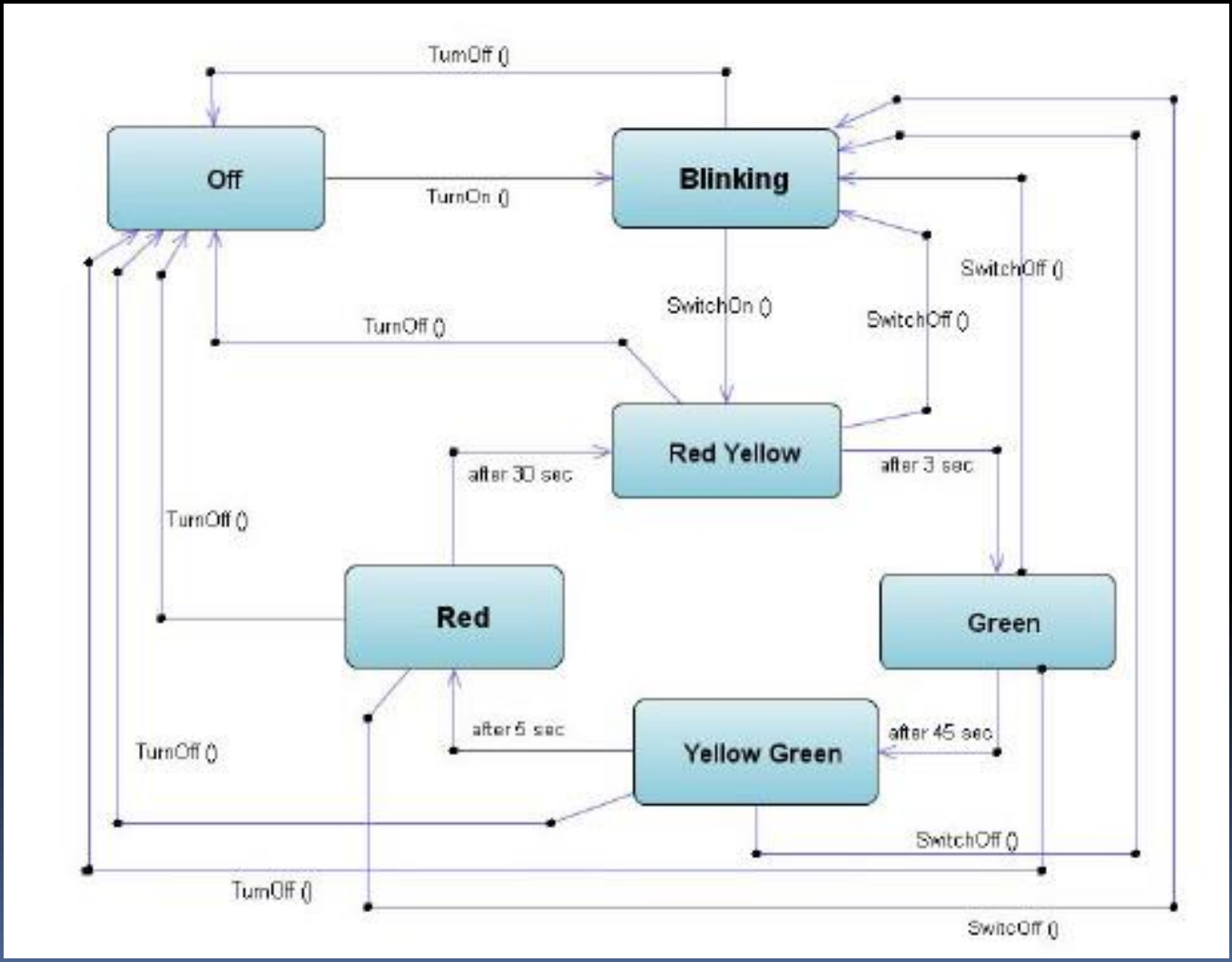
Составное состояние

- В последнем случае у светофора есть 4 видимых состояния, являющихся предписывающими сигналами для участников дорожного движения:
- Green — зеленый свет, движение разрешено;
- GreenYellow — состояние перехода из режима разрешения в режим запрещения движения (это настоящее состояние, светофор находится в нем заметное время);
- Red — красный свет, движение запрещено;
- RedYellow — состояние перехода из режима запрещения в режим разрешения движения (это состояние отличное от GreenYellow, светофор подает несколько иные световые сигналы и участники движения обязаны по другому на них реагировать).

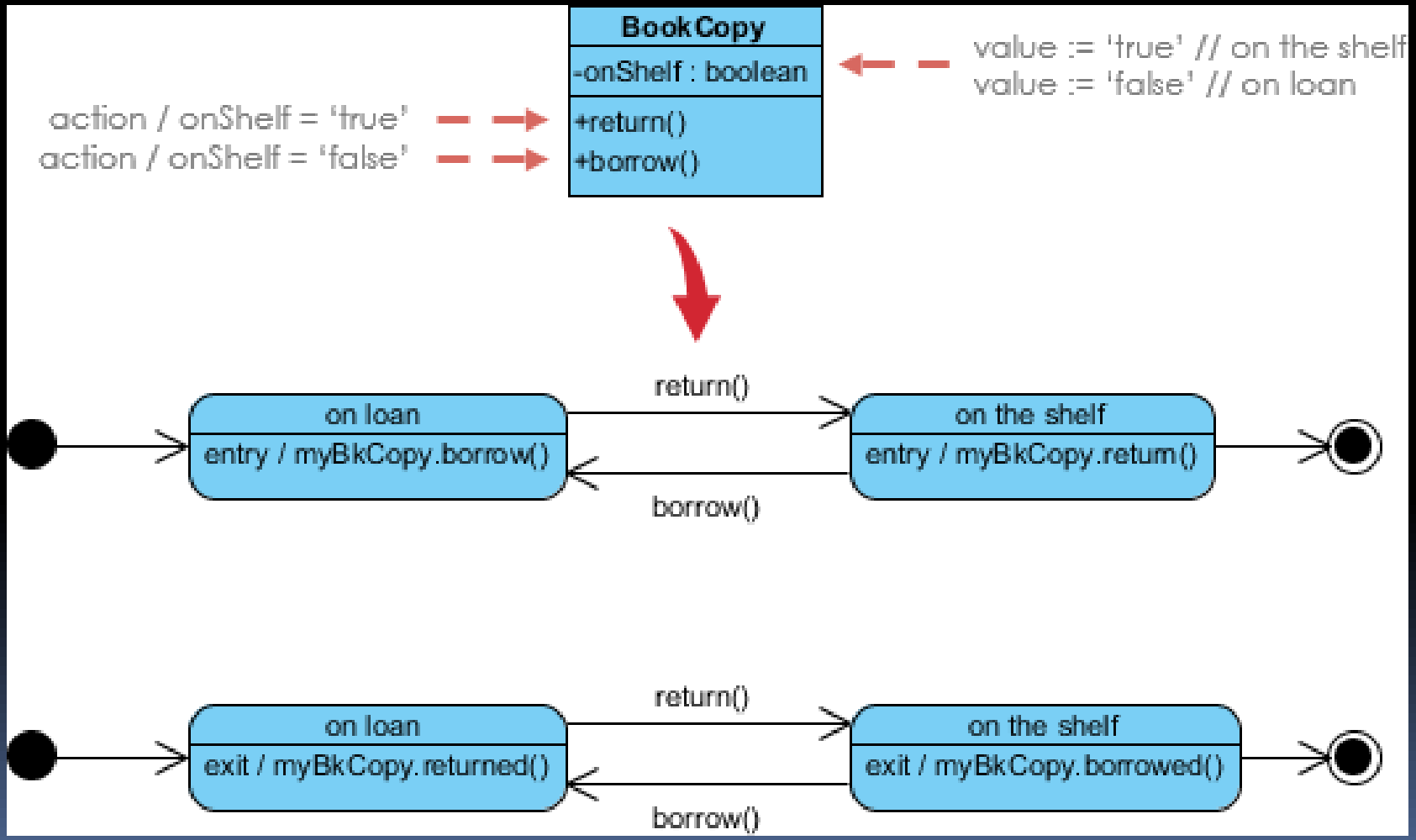
Составное состояние

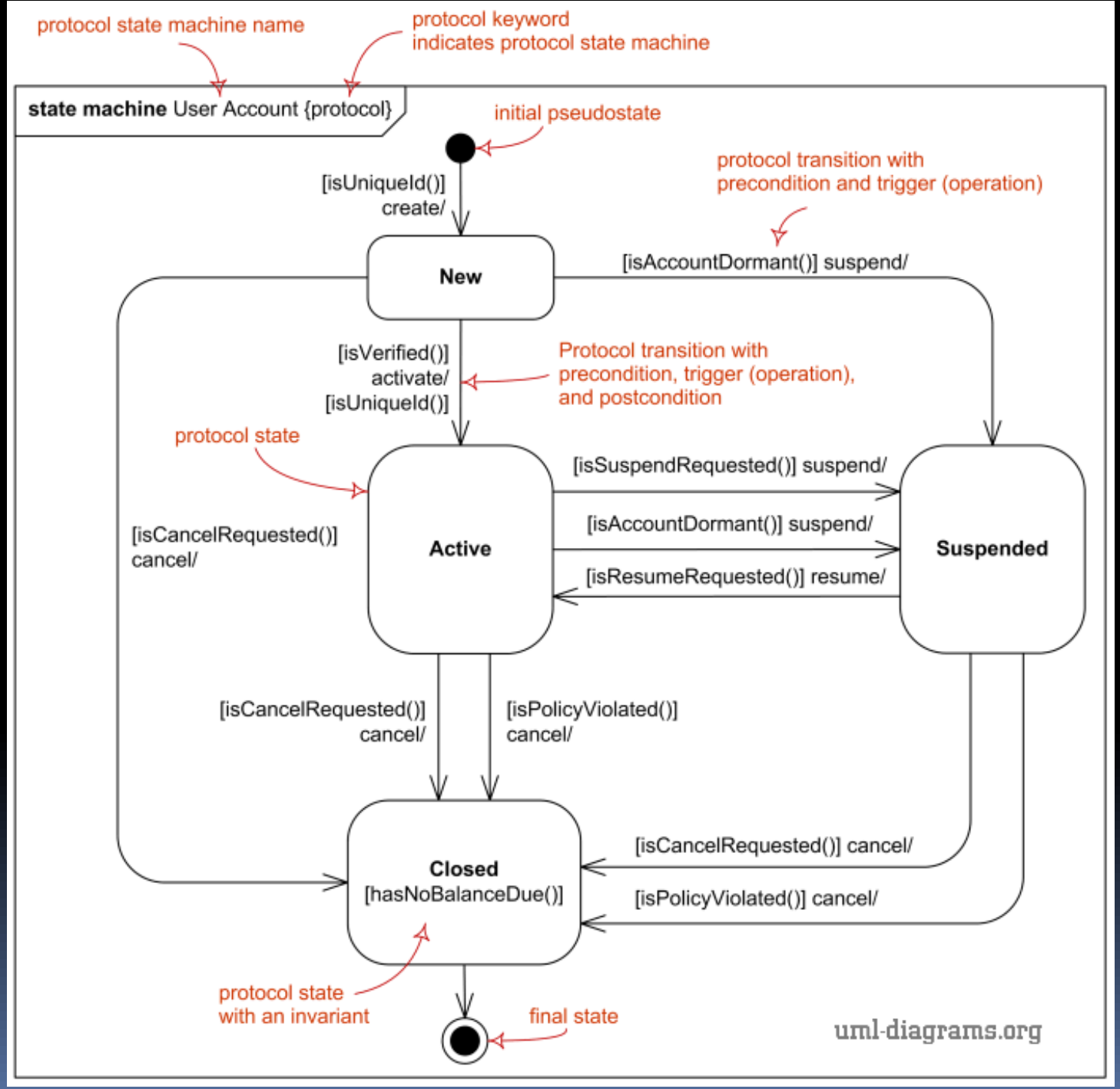


Составное состояние



Связь с диаграммой классов




protocol state machine name
protocol keyword
indicates protocol state machine
state machine User Account {protocol}
initial pseudostate
[isUniqueId()]
create/
protocol transition with
precondition and trigger (operation)
New
[isAccountDormant()] suspend/
[isVerified()]
activate/
[isUniqueId()]
Protocol transition with
precondition, trigger (operation),
and postcondition
protocol state
[isSuspendRequested()] suspend/
[isCancelRequested()]
cancel/
Active
[isAccountDormant()] suspend/
[isResumeRequested()] resume/
Suspended
[isCancelRequested()]
cancel/
[isPolicyViolated()]
cancel/
[isCancelRequested()] cancel/
Closed
[hasNoBalanceDue()]
[isPolicyViolated()] cancel/
protocol state
with an invariant
final state
uml-diagrams.org

Выводы

- Модель поведения — это описание алгоритма работы системы.
- В UML предусмотрено несколько различных средств для описания поведения.
- Выбор того или иного средства диктуется типом поведения, которое нужно описать.
- Для моделирования поведения используются диаграмма состояний, диаграмма деятельности, диаграммы взаимодействия.