UML. Диаграмма состояний

Диаграмма состояний

Для моделирования динамических аспектов системы используются **диаграммы взаимодействий** и **автоматы**.

Диаграммы взаимодействий моделируют поведение **сообщества** совместно работающих **объектов**.
Автоматы моделируют поведение **отдельного объекта**.

Автомат может показывать:

- *передачу потока управления от одного состояния объекта к другому (**диаграмма состояний**)*
- *передачу потока управления от одной деятельности к другой (**диаграмма деятельности**)*

Диаграмма состояний (state machine diagram) — это диаграмма, которая показывает автомат, фокусируя внимание на его потенциальных состояниях и переходах между ними

Назначение

Диаграммы состояний служат для моделирования поведения реактивных объектов.

Диаграммы состояний могут существовать автономно, и могут использоваться для моделирования поведения классов, прецедентов и системы в целом.

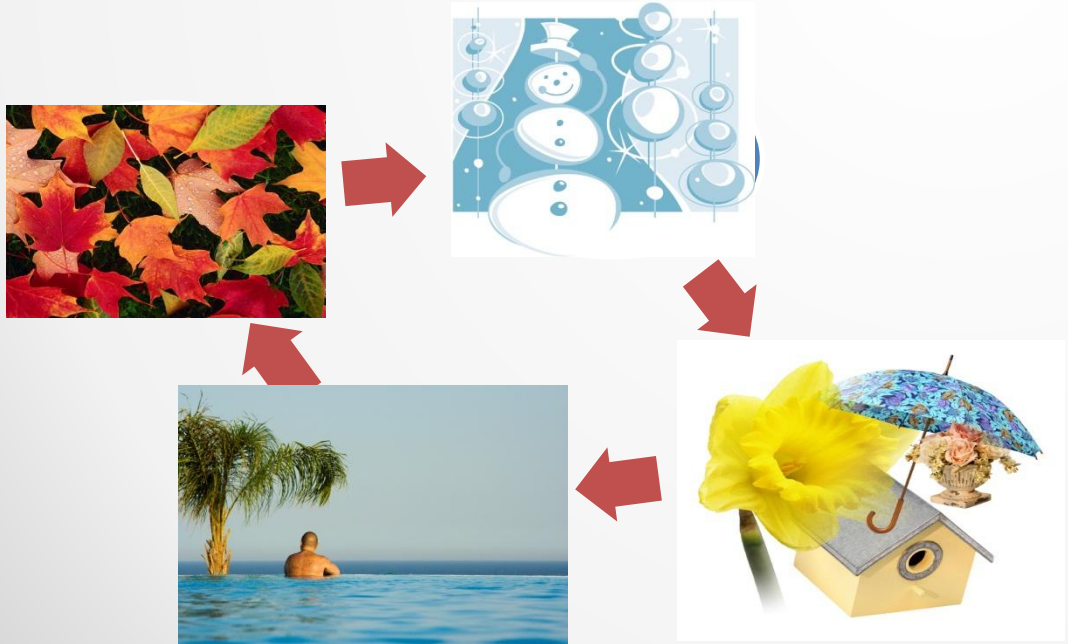

Диаграмма состояний может содержать:

- Состояния
- *Переходы*
- Примечания

Элементы диаграммы состояний

Состояние

Состояние объекта (state) — это ситуация в его жизни, на протяжении которой он удовлетворяет некоторому условию, осуществляет определенную деятельность или ожидает какого-то события.

Состояние определяют следующие элементы:

- ***имя*** *- текстовая строка, которая отличает одно состояние от всех остальных. Имя состояния должно быть уникальным внутри объемлющего пакета*
- **внутренние действия** - действия, выполняемые при входе в состояние и выходе из него, а также выполняемые без выхода из состояния

Список внутренних действий в данном состоянии

- ***подсостояния*** *- внутри состояния могут существовать подсостояния, как непересекающиеся (активизируемые последовательно), так и параллельные (активные одновременно)*
- **отложенные события** - список событий, которые не обработаны в этом состоянии, а отложены и поставлены в очередь для обработки объектом в некотором другом состоянии

Внутреннее действие

Внутреннее действие — текстовая строка формата

<метка действия / выражение действия>

Перечень меток действия имеет фиксированные значения, которые не могут быть использованы в качестве имен событий:

entry *— указывает на действие, которое выполняется в момент входа в данное состояние*

exit — указывает на действие, которое выполняется в момент выхода из данного состояния

include *— используется для обращения к подавтомату, при этом следующее за ней выражение действия содержит имя этого подавтомата*

do — специфицирует выполняющуюся деятельность, которая выполняется в течение всего времени, пока объект находится в данном состоянии, или до тех пор, пока не закончится вычисление, специфицированное следующим за ней выражением действия

Переход

Переход (transition) — это отношение между двумя состояниями, показывающее, что объект, находящийся в **исходном состоянии**, должен выполнить определенное **действие** и перейти в **целевое состояние**, как только произойдет указанное **событие** и будут выполнены определенные **сторожевые условия**

Каждый переход может помечен строкой текста, которая имеет следующий общий формат:

<сигнатура события>[<сторожевое условие>] <выражение действия>

Сигнатура события описывает некоторое событие с необходимыми аргументами:

<имя события>(<список параметров, разделенных запятыми>)

Событие-триггер, сторожевое условие и действие

Событие-триггер (trigger event) — событие, при получении которого объектом, находящимся в исходном состоянии, может сработать переход

Сторожевое условие (guard condition) *— логическое выражение, которое вычисляется при возникновении события-триггера. Если значение истинно, то переходу разрешено сработать, если ложно - переход не срабатывает*

Действие (action) — это атомарное вычисление, которое выполняется в том случае, когда переход срабатывает, и приводит к изменению состояния объекта или возврату значения

Псевдосостояния

Псевдосостояние объекта (pseudo-state) — абстракция для различных типов вершин, используемая в диаграмме состояний

Псевдосостояния не содержат никаких внутренних действий

- **Начальное состояние (initial)**
- ***Конечное состояние (final)***
- **Точка входа (entry point)**
- ***Точка выхода (exit point)***
- **Выбор (choice)**
- ***Историческое состояние (history)***
- **Соединение (junction)**
- ***Прекращение (terminate)***
- **Ветвление (fork)**

Составные состояния и подсостояния

Простое состояние — состояние, не имеющее внутренней структуры

Составное состояние (composite state) *— сложное состояние, которое состоит из других вложенных в него состояний*

Подсостояние (substate) — состояние, являющееся частью другого состояния

Последовательные подсостояния

Последовательные подсостояния (sequental substates) используются для моделирования такого поведения объекта, во время которого в каждый момент времени объект может находиться в одном и только одном подсостояний

Начальное и конечное состояния объекта должны быть единственными в каждом составном состоянии

Параллельные подсостояния

Параллельные подсостояния (concurrent substates) позволяют специфицировать два или более автомата, которые выполняются параллельно в контексте объемлющего объекта

- Объемлющий объект может одновременно находиться в каждом из параллельных подсостояний
- Если какой-либо из подавтоматов пришел в свое конечное состояние раньше других, то он должен ожидать, пока другие подавтоматы не придут в свои конечные состояния

Советы

При моделировании состояний соблюдайте следующие правила:

- *Дайте диаграмме имя, соответствующее назначению*
- Сначала смоделируйте устойчивый состояния объекта, затем переходите к допустимым переходам состояний
- *Ветвления и параллельность моделируйте на отдельной диаграмме*
- Располагайте элементы так, чтобы число

Пример

Домашнее задание

1. Разработать диаграммы состояний для ваших БП