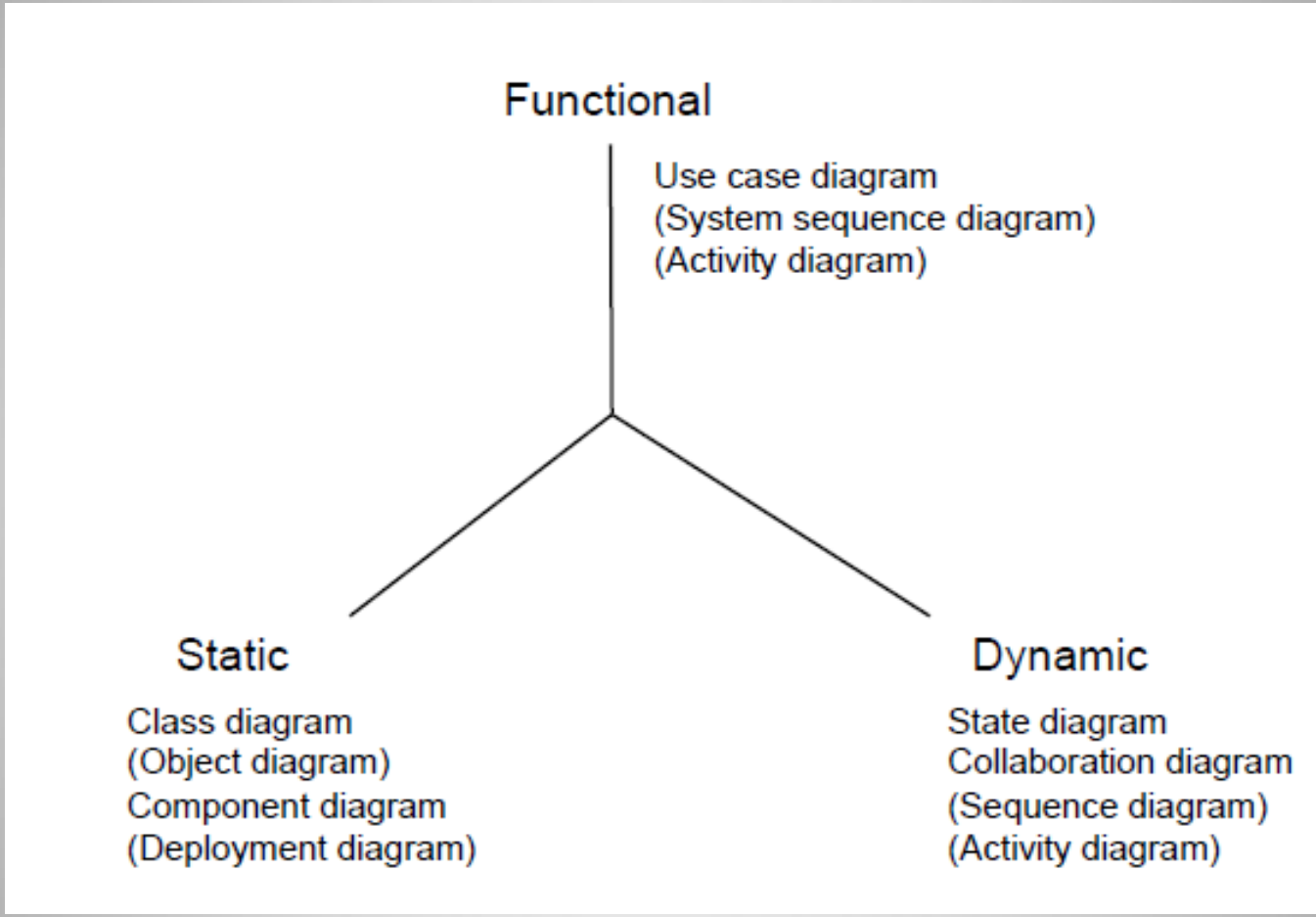
Анализ и проектирование на UML

Максим Валерьевич Хлопотов

Оси моделирования



Представление поведения

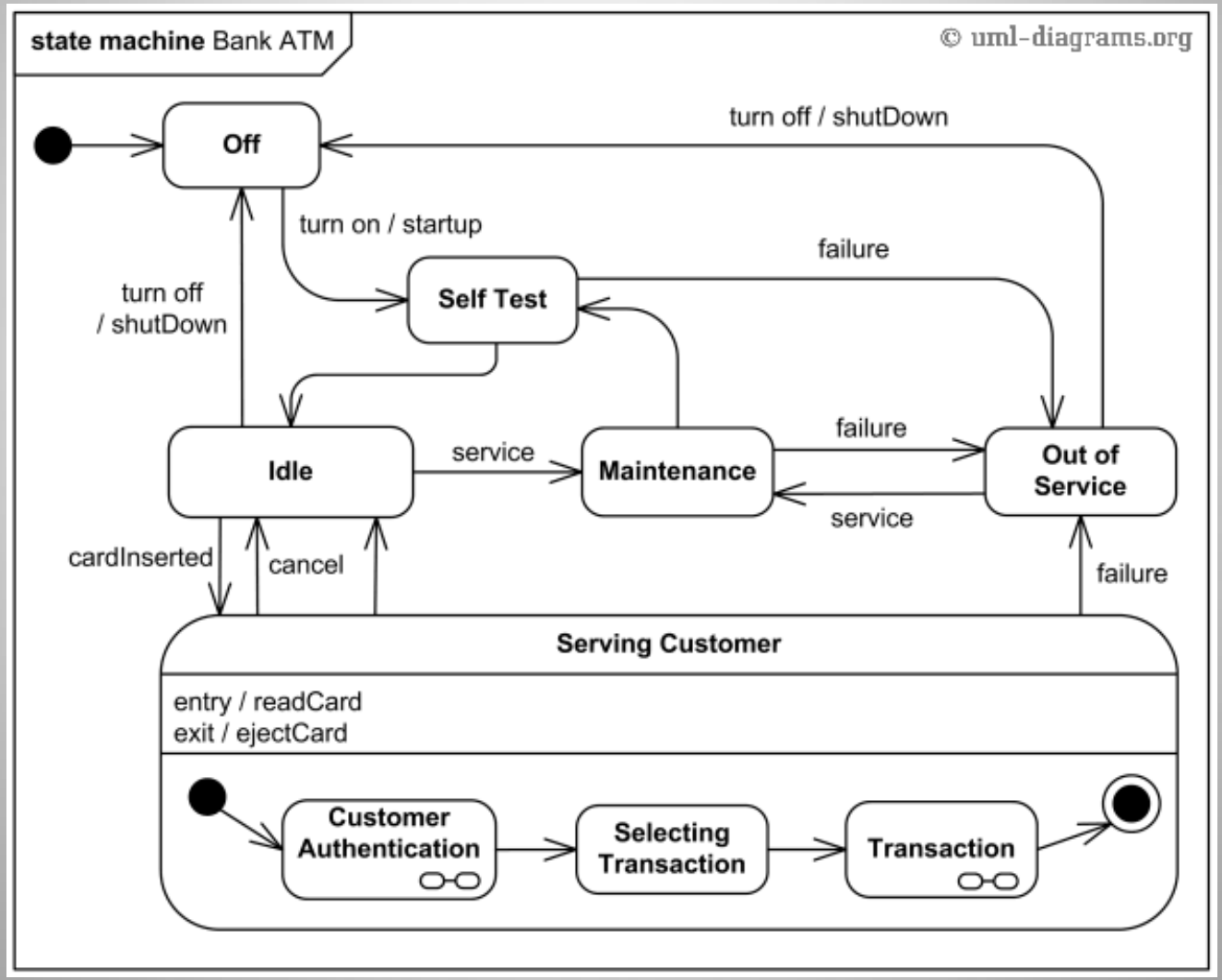
- Диаграммы состояний можно составить не только для программных объектов — экземпляров отдельных классов, но и для более крупных конструкций, в частности, для всей модели приложения в целом или для более мелких — отдельных операций.
- Для описания последовательности выполняемых элементарных шагов при выполнении отдельной операции или реализации сложного варианта использования, удобно использовать диаграммы деятельности.

Поведение приложения

- Для пользователя поведение приложения проявляется прежде всего в интерфейсе.
- В принципе возможны два архитектурных решения интерфейса с пользователем:

- инициатива принадлежит программе, т. е. программа, будучи запущенной, выполняет свою функцию, следуя заложенному в нее алгоритму решения задачи, запрашивая по мере необходимости информацию у пользователя (так называемый активный интерфейс);
- инициатива принадлежит пользователю, т. е. программа, будучи запущенной, находится в состоянии ожидания команд пользователя, которые пользователь подает, следуя имеющемуся у него в голове алгоритму решения задачи, а программа выполняет подаваемые команды (так называемый пассивный интерфейс).

Поведение приложения



Поведение приложения

- В настоящее время более распространенным, особенно в наиболее массовых приложениях для бизнеса, является второе решение (т. е. пассивный интерфейс). Это обусловлено целым рядом причин, в том числе:

- увеличением компактности многофункционального приложения при использовании пассивного интерфейса, т. к. программа должна обеспечить только выполнимость сравнительно небольшого числа сравнительно простых команд, а подбор последовательности выполнения команд, нужных для решения задачи, возлагается на пользователя;
- наличием в современных системах программирования и операционных системах для данной архитектуры развитого механизма поддержки, называемого событийным управлением, который описан ниже;
- устоявшейся за последние несколько лет привычкой массового пользователя к пассивному интерфейсу.

Поведение приложения

- При всех своих достоинствах пассивный интерфейс обладает и определенными недостатками, а именно: предполагается, что у пользователя в голове имеется алгоритм решения задачи и что пользователь в состоянии транслировать этот алгоритм в последовательность команд приложения. Другими словами, в приложениях с пассивным интерфейсом предполагается, что пользователь знает, что нужно делать, чтобы решить задачу.

Поведение приложения

- *Событийное управление* — это способ структуризации программного кода, основанный на следующей идее:
- Имеется некоторое предопределенное множество поименованных *событий.* События могут быть явным образом связаны с объектами, а могут быть связаны неявным образом или быть связаны с неявными объектами, в таком случае события обычно называют *системными.*

Поведение приложения

- События могут *возникать.* Возникновение события подразумевает, что состояние системы изменилось определенным образом.
- С событием может быть связана процедура, которая называется *реакцией на событие.* При возникновении события автоматически вызывается процедура реакции.

Поведение приложения

- В современных системах программирования, поддерживающих событийное управление, предусматривается большое число самых разнообразных событий, реакции на которые могут быть определены в программе, например: нажатие клавиши на клавиатуре, попадание указателя мыши в определенную область экрана, достижение внутренним таймером заданного значения, открытие заданного файла и т. д.

Поведение приложения

- В программе, целиком управляемой событиями, нет основного потока управления, он находится вне программы (то есть там, где реализован механизм возникновения событий).
- Управление в программу попадает только в форме вызова процедуры реакции. Такая организация программы обеспечивает высокую модульность, прозрачность, сбалансированность структуры и другие полезные свойства. Понятно, что если связать события с командами приложения (как обычно и делается), то событийное управление как нельзя лучше подходит для реализации пассивного интерфейса.

Поведение приложения

- Однако приложения для бизнеса с пассивным пользовательским интерфейсом являются хотя и распространенным, но не единственным типом приложений.
- Нетрудно привести примеры важных типов приложений, в которых реакция на внешние события отсутствует или играет второстепенную роль: компиляторы, программы инженерно-технических и научных расчетов и др. В таких программах для описания поведения традиционно используется понятие потока управления.

Поведение приложения

- *Поток управления* — это последовательность выполнения операторов (команд) в программе.
- Если программа представляет собой просто последовательность операторов, то операторы в программе выполняются по очереди в естественном порядке (от начала к концу). В этом случае поток управления просто совпадает с последовательностью операторов в программе.
- Однако обычно это не так.

Поведение приложения

- Во-первых, на поток управления оказывают влияние различные управляющие конструкции: операторы перехода, условные операторы, операторы цикла и т. д.
- Во-вторых, в большинстве практических систем программирования используется понятие подпрограммы: при выполнении оператора вызова подпрограммы выполнение операторов программы приостанавливается, управление передается в подпрограмму, т. е. в поток управления попадают операторы подпрограммы, а при выходе из подпрограммы возобновляется выполнение операторов программы. При этом считается, что вызванная подпрограмма *активизирована и остается таковой,* пока не возобновится выполнение вызвавшей программы или подпрограммы.
- Кроме того, на поток управления влияют факторы, находящиеся вне данной программы, например, поток управления меняется при обработке прерываний и при возникновении событий.

Поведение приложения

- Различаются однопоточные (т. е. с одним потоком управления) и многопоточные программы.
- Характерным признаком однопоточной программы является то, что в каждый момент времени можно указать единственный оператор программы, который выполняется в данный момент — говорят, что этот оператор имеет *фокус управления.*
- В многопоточной программе имеется несколько потоков управления и сосуществуют несколько фокусов управления, соответственно. При этом совершенно необязательно, чтобы различные потоки управления выполнялись физически одновременно на разных процессорах: достаточно иметь в операционной системе механизм, обеспечивающий переключение процессора на выполнение разных потоков управления.

Диаграммы взаимодействия

- Диаграммы взаимодействия предназначены для моделирования поведения путем описания взаимодействия объектов для выполнения некоторой задачи или достижения определенной цели.
- Взаимодействие происходит путем обмена **сообщениями**. Диаграммы взаимодействия применяются на разных уровнях моделирования: как для описания поведения отдельных операций, так и целых вариантов использования.

Диаграммы взаимодействия

- Данный тип диаграмм позволяет описывать не только взаимодействие программных объектов (экземпляров классов), но и взаимодействие экземпляров иных классификаторов: действующих лиц, вариантов использования, подсистем, компонентов, узлов.
- Диаграммы взаимодействия графически изображаются в двух формах: диаграммы последовательности и диаграммы кооперации.
- Оба типа диаграмм моделируют поведение "по индукции", от частного к общему, путем описания конкретного протокола передачи сообщений (т. е. сценария).

Диаграммы взаимодействия

- Сильная сторона состоит в том, что в объектно-ориентированной парадигме обмен сообщениями — это и есть само выполнение программы, поэтому протокол передачи сообщений является наиболее точной моделью поведения. Диаграммы взаимодействия находятся "ближе" к реальному выполнению программы, чем другие средства описания поведения.
- Слабость диаграмм взаимодействия состоит в том, что это диаграммы описывают поведение на уровне объектов, а не классов, на уровне протоколов выполнения алгоритма, а не самого алгоритма. Диаграммы взаимодействия менее "алгоритмичны", чем машины состояний и диаграммы деятельности.

Диаграммы взаимодействия

- На диаграммах обоих типов основными сущностями являются объекты: экземпляры классификаторов — классов и действующих лиц. Отношениями же являются связи, т. е. экземпляры ассоциаций, по которым передаются сообщения.
- Фактически, это такое же описание последовательности обмена сообщениями взаимодействующих объектов, только выраженное другими графическими средствами.

Диаграммы взаимодействия

- Диаграммы кооперации и диаграммы последовательности семантически эквиваленты, хотя графически выглядят совсем по-разному.
- Семантически эти диаграммы эквиваленты потому, что описывают одно и то же: последовательность передачи сообщений между объектами в процессе взаимодействия объектов.
- Выглядят по-разному они потому, что в диаграмме последовательности графически подчеркивается упорядоченность во времени передаваемых сообщений, в то время как в диаграмме кооперации на передний графический план выдвигается структура связей между объектами, по которым передаются сообщения.

Диаграмма последовательности

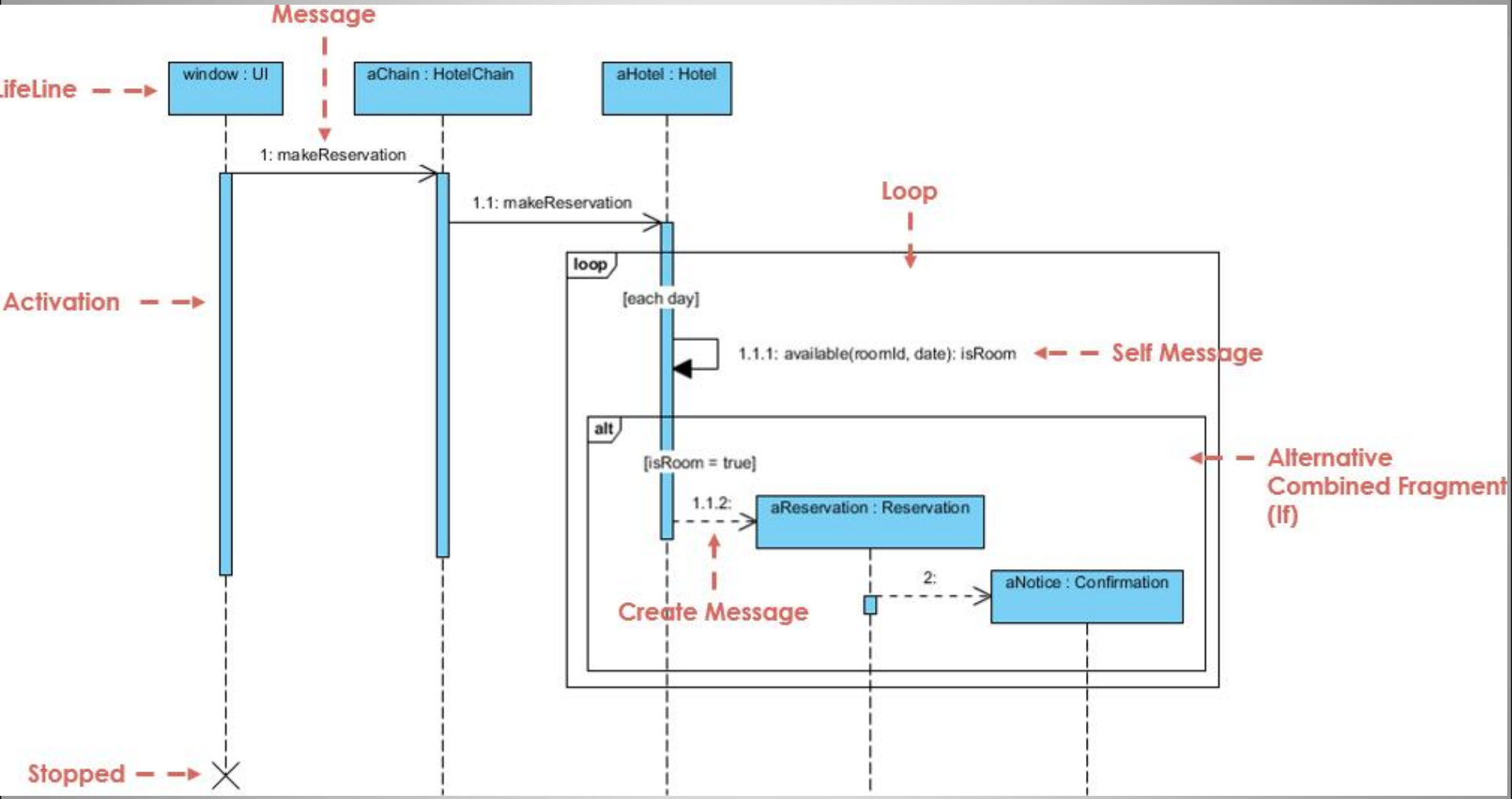


Диаграмма последовательности

Таблица 4.1. Общепринятые операторы для фреймов взаимодействия

Оператор	Значение
`alt`	Несколько альтернативных фрагментов (alternative); выполняется только тот фрагмент, условие которого истинно (рис. 4.4)
`opt`	Необязательный (optional) фрагмент; выполняется, только если условие истинно. Эквивалентно `alt` с одной веткой (рис. 4.4)
`par`	Параллельный (parallel); все фрагменты выполняются параллельно
`loop`	Цикл (loop); фрагмент может выполняться несколько раз, а защита обозначает тело итерации (рис. 4.4)
`region`	Критическая область (critical region); фрагмент может иметь только один поток, выполняющийся за один прием
`neg`	Отрицательный (negative) фрагмент; обозначает неверное взаимодействие
`ref`	Ссылка (reference); ссылается на взаимодействие, определенное на другой диаграмме. Фрейм рисуется, чтобы охватить линии жизни, вовлеченные во взаимодействие. Можно определять параметры и возвращать значение
`sd`	Диаграмма последовательности (sequence diagram); используется для очерчивания всей диаграммы последовательности, если это необходимо

Диаграмма последовательности

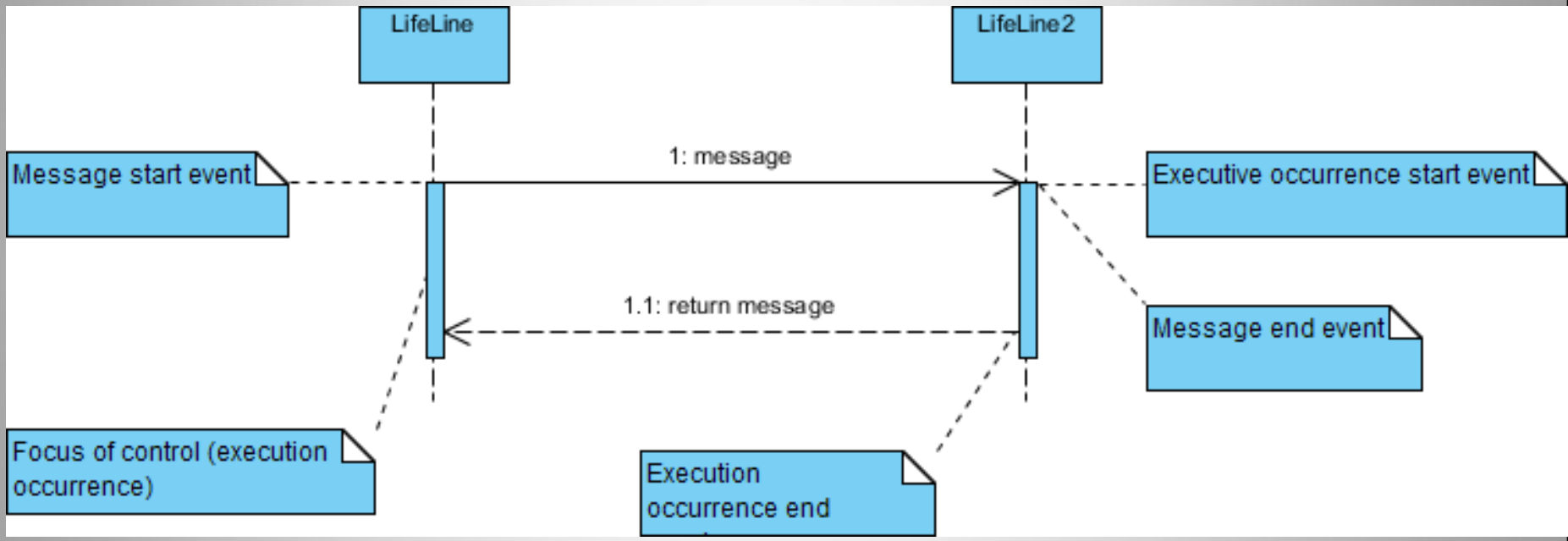


Диаграмма последовательности

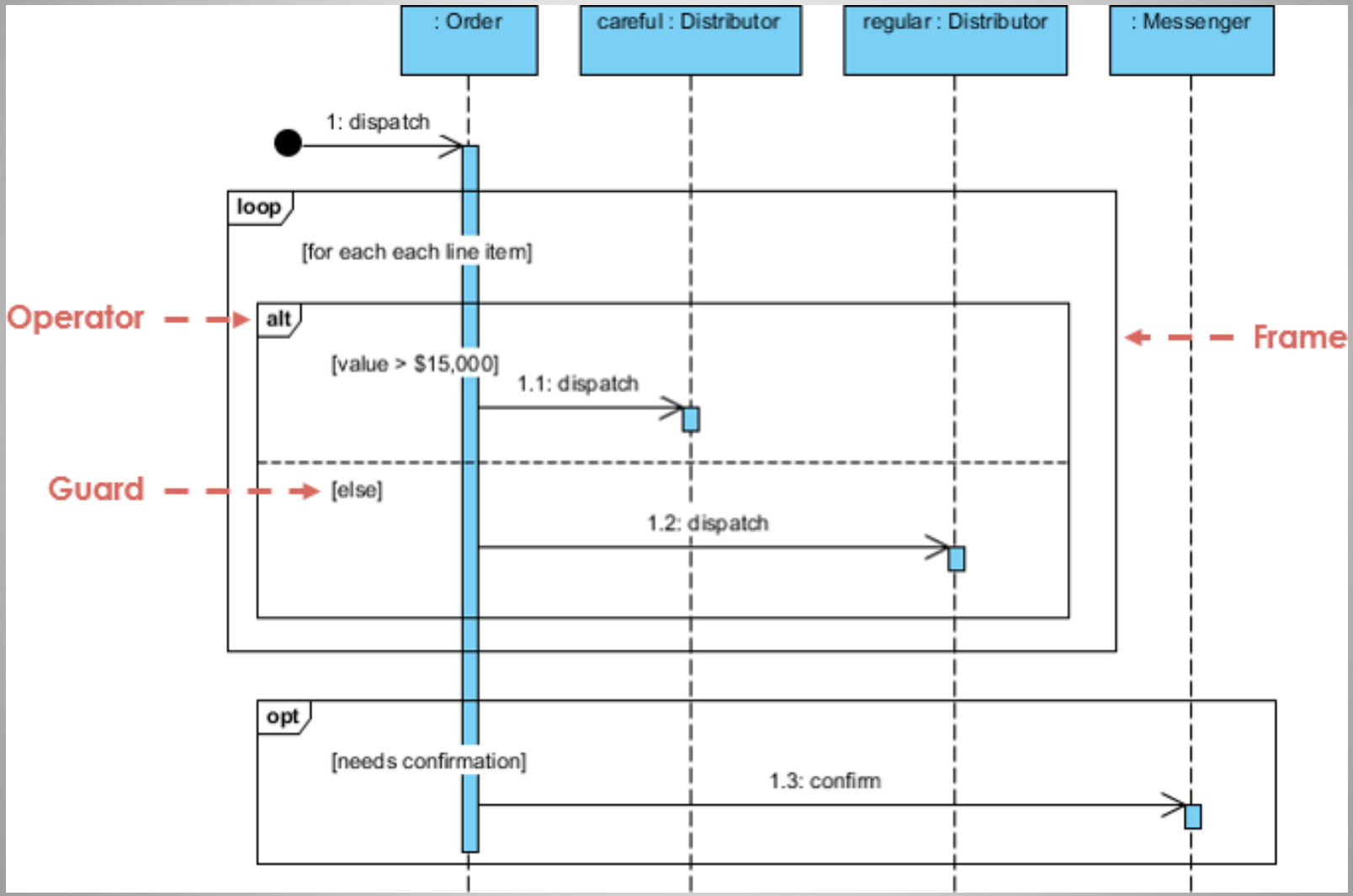
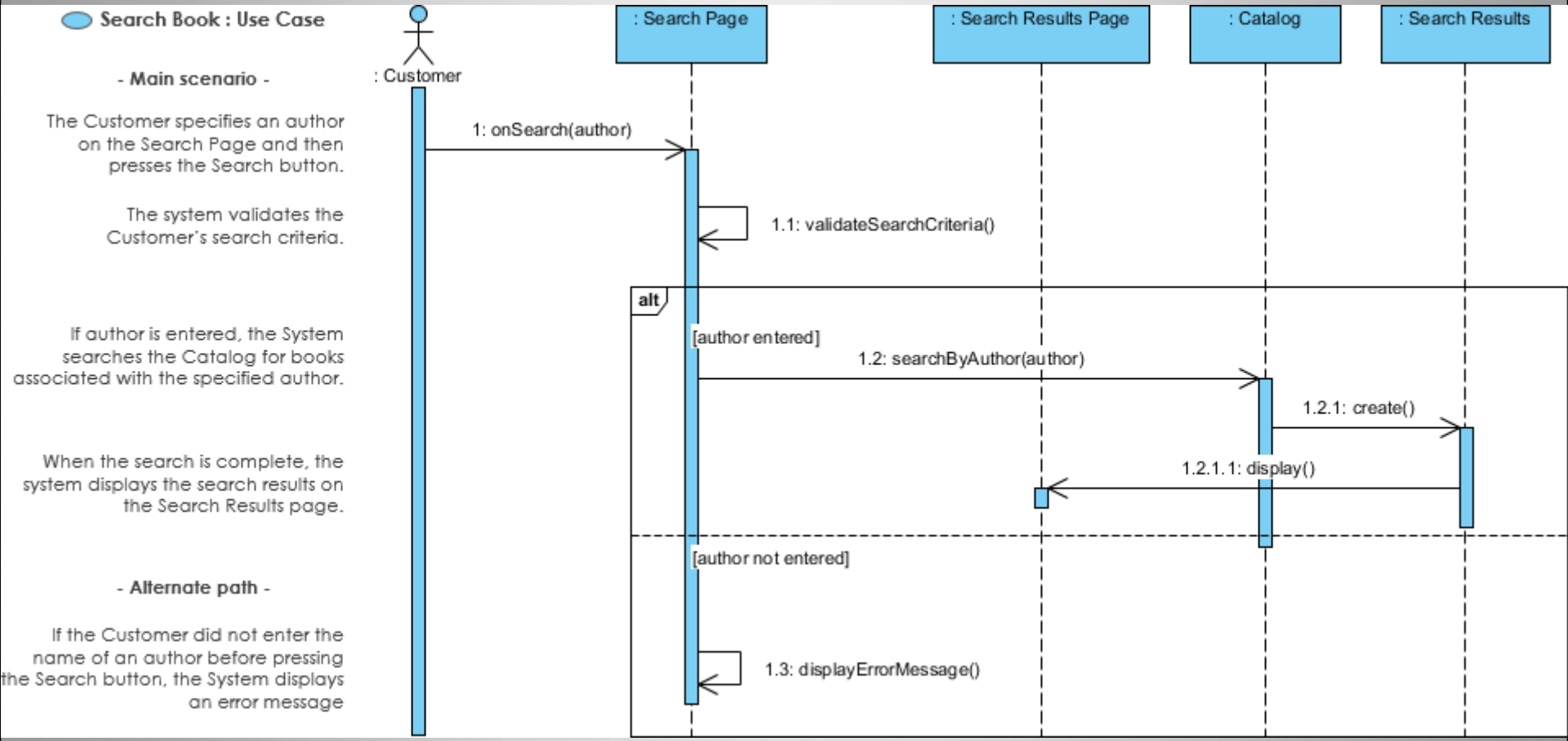
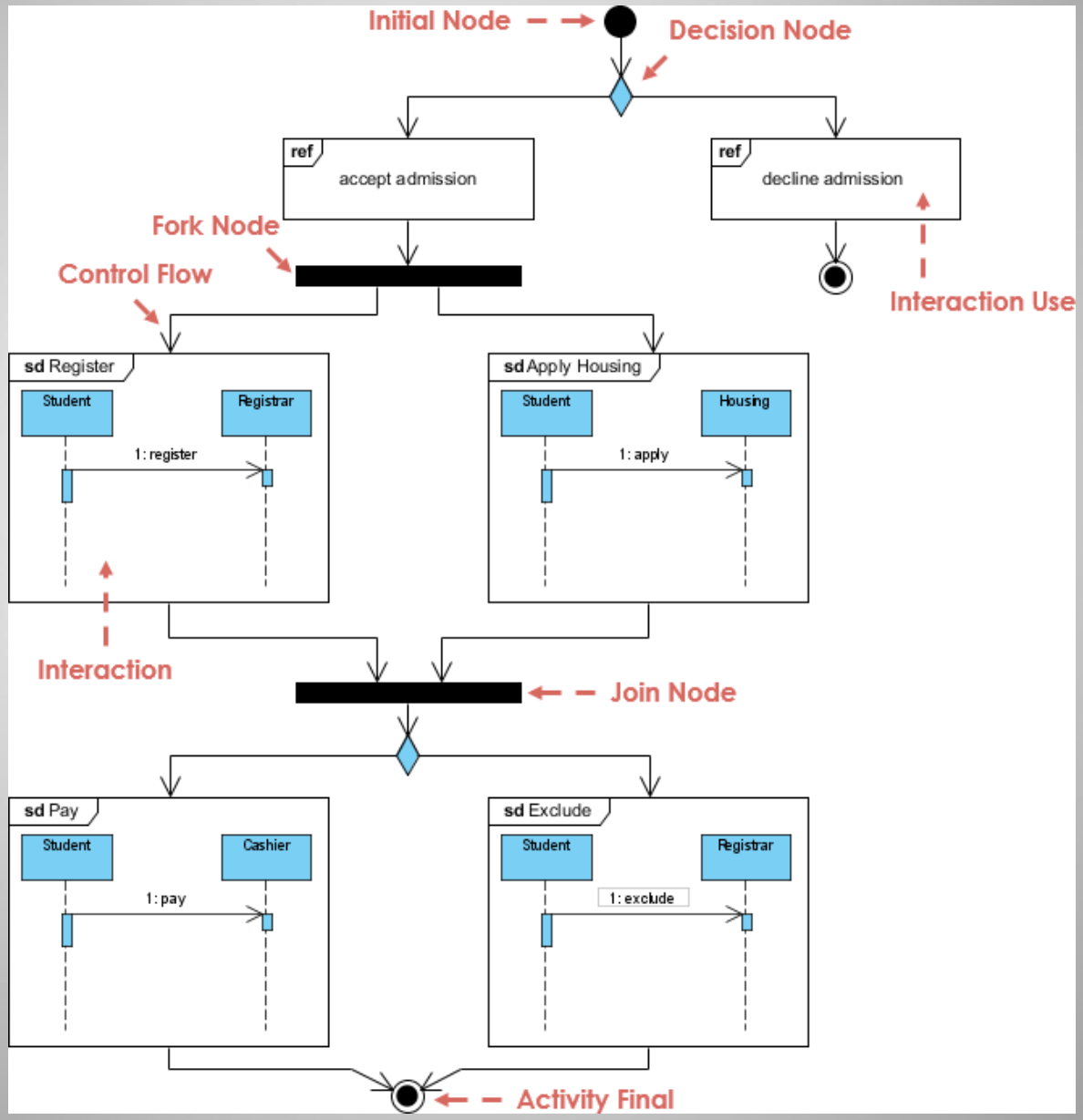
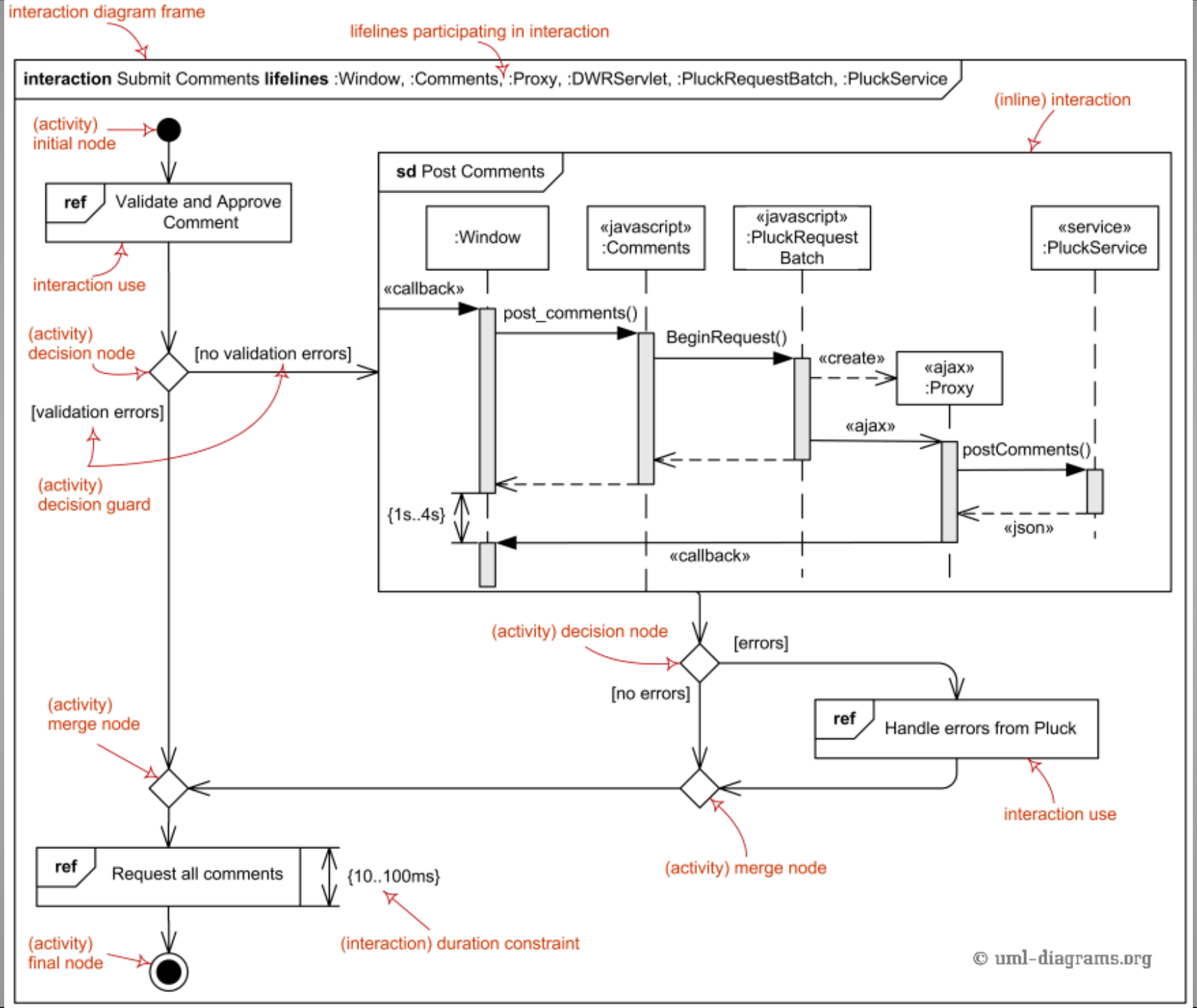


Диаграмма последовательности

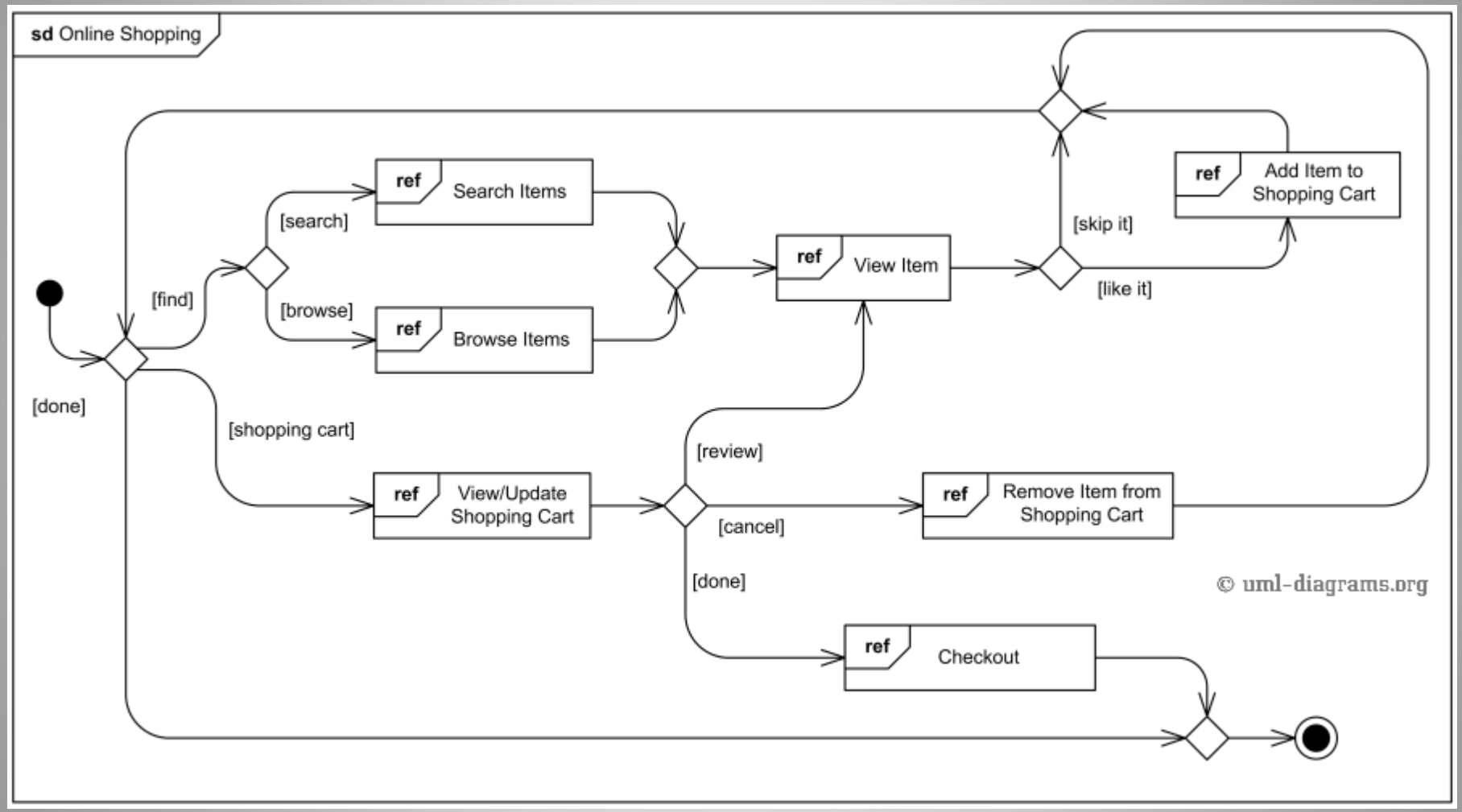


Обзорная диаграмма взаимодействия




interaction diagram frame
lifelines participating in interaction
interaction Submit Comments lifelines :Window, :Comments, :Proxy, :DWRServlet, :PluckRequestBatch, :PluckService
(inline) interaction
(activity) initial node
ref Validate and Approve Comment
interaction use
sd Post Comments
:Window
«javascript» :Comments
«javascript» :PluckRequest Batch
«service» :PluckService
«callback»
post_comments()
BeginRequest()
«create»
«ajax» :Proxy
«ajax»
postComments()
«json»
{1s..4s}
«callback»
(activity) decision node
[no validation errors]
[validation errors]
(activity) decision guard
(activity) decision node
[errors]
[no errors]
ref Handle errors from Pluck
interaction use
(activity) merge node
(activity) merge node
ref Request all comments
{10..100ms}
(interaction) duration constraint
(activity) final node
© uml-diagrams.org

Обзорная диаграмма взаимодействия



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.