SWD SOFTWARE

Разработка систем реального времени с использованием UML и каркасов приложений

Дмитрий Рыжов

Менеджер по продукту

d.ryzhov@swd.ru

Telelogic

Исходные предпосылки

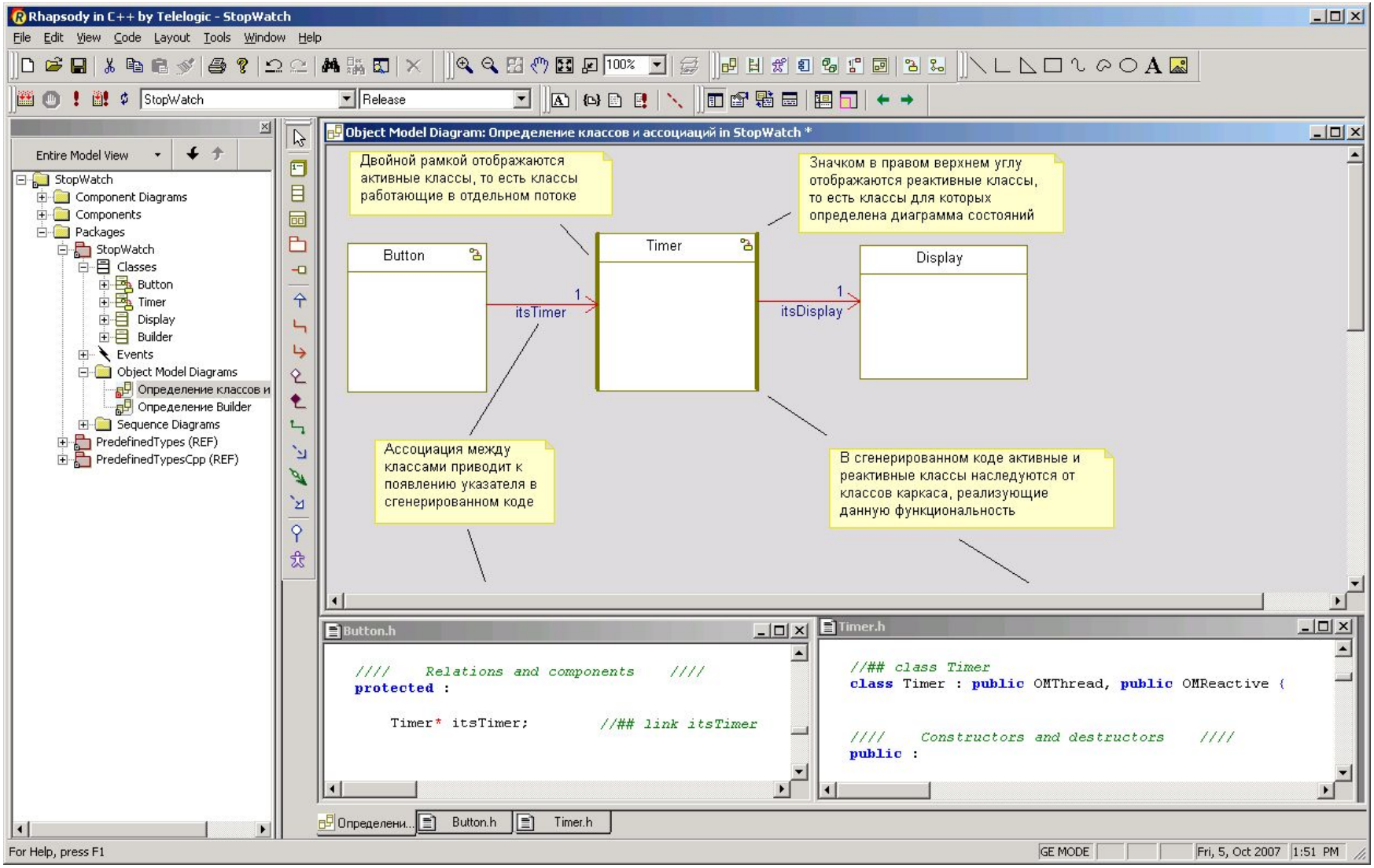
- Распространение языка моделирования UML
- Развитие инструментов разработки на основе визуального моделирования
- Применение инструментов на всех стадиях процесса разработки
- Специализация инструментов
- Инструменты для разработки встраиваемых систем и приложений реального времени
- Автоматическая генерация кода, тестирование, временной анализ и верификация

Пример разработки секундомера

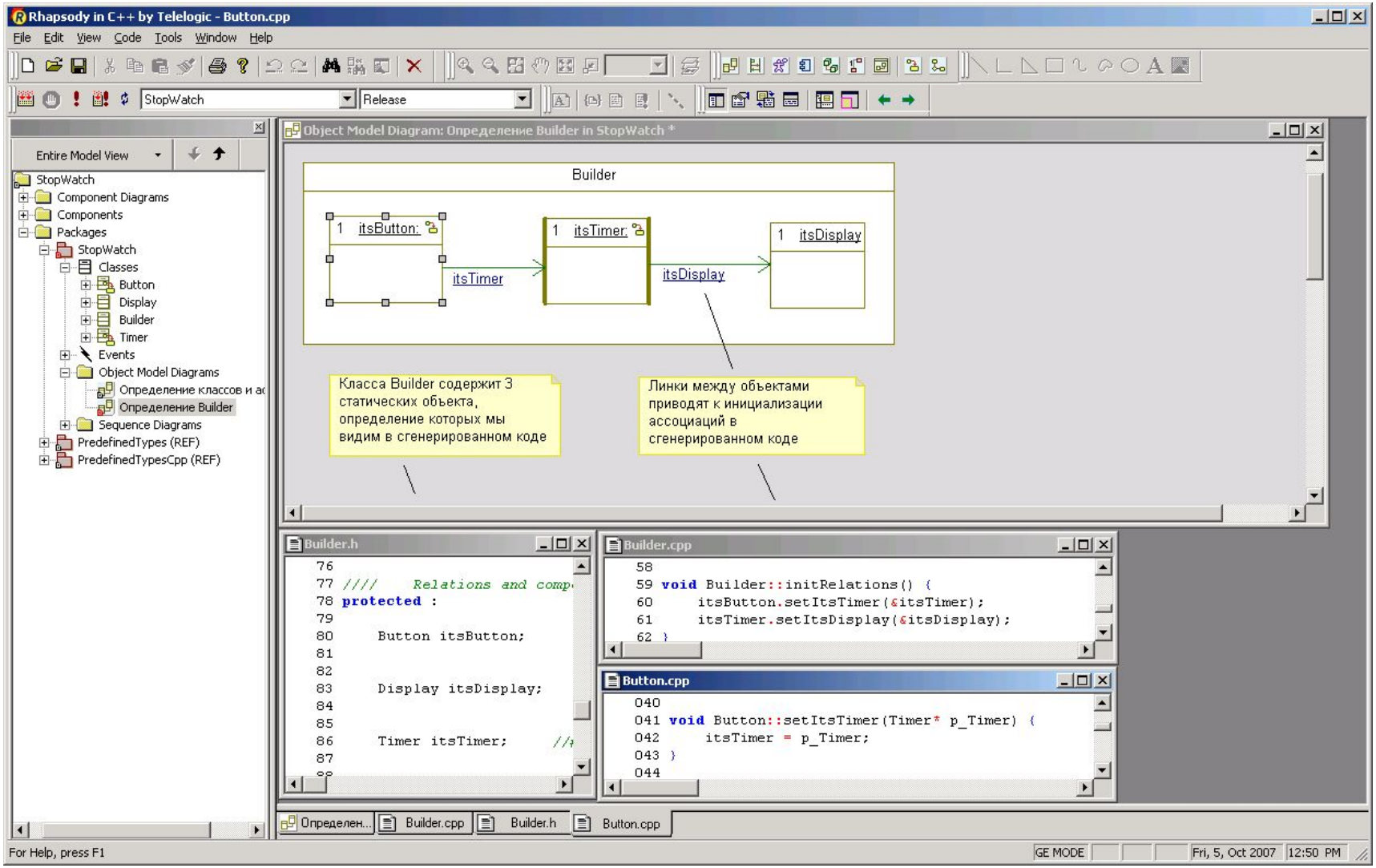
Требования к секундомеру

- Секундомер имеет одну кнопку для запуска и остановки и дисплей для отображения. Дисплей отображает минуты и секунды
- При нажатии и отпускании кнопки в течении 2 секунд секундомер запускается либо останавливается
- Если кнопка удерживается нажатой более чем 2 секунды, то секундомер сбрасывается в 0 и останавливается

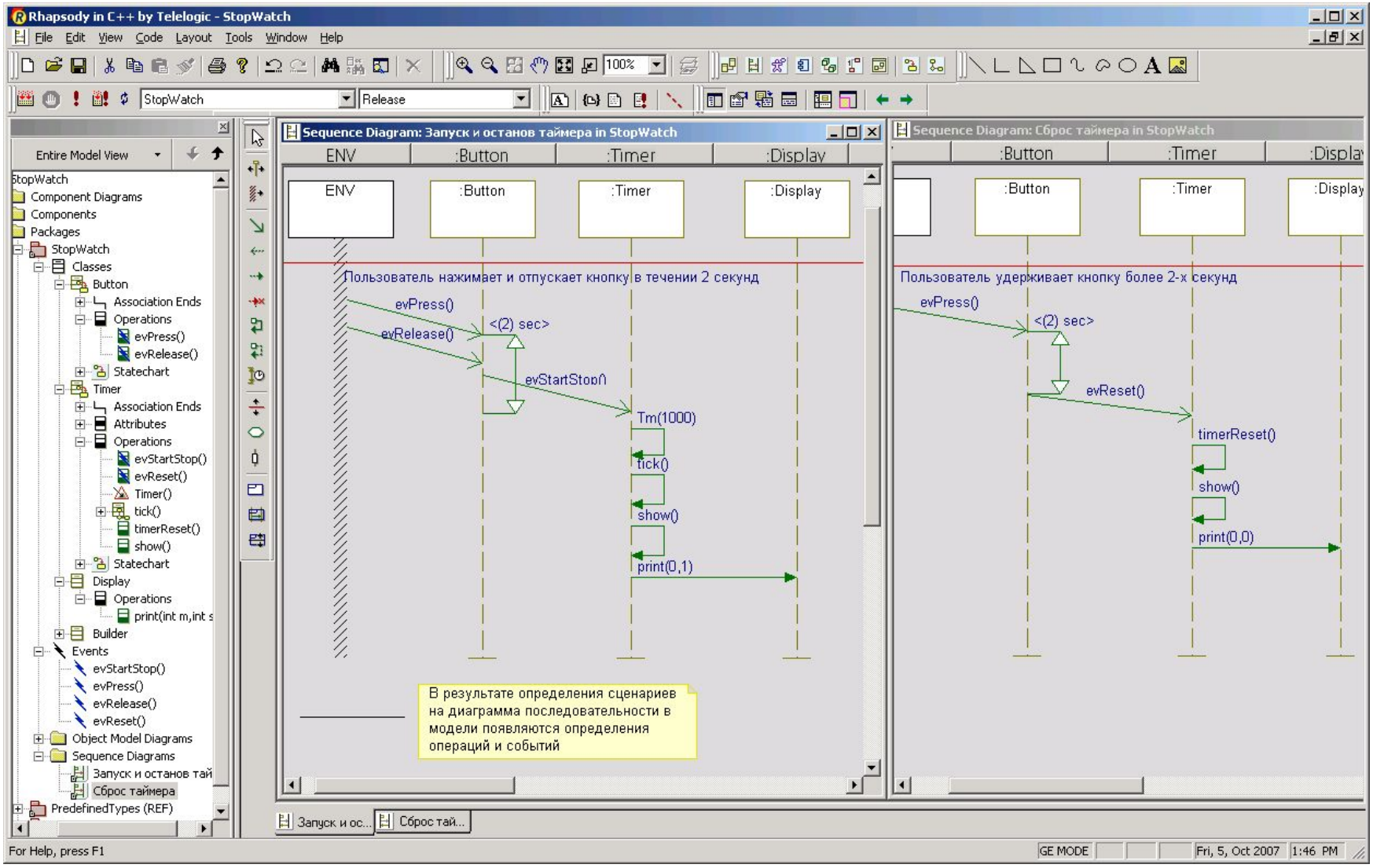
Определение классов и связей между ними



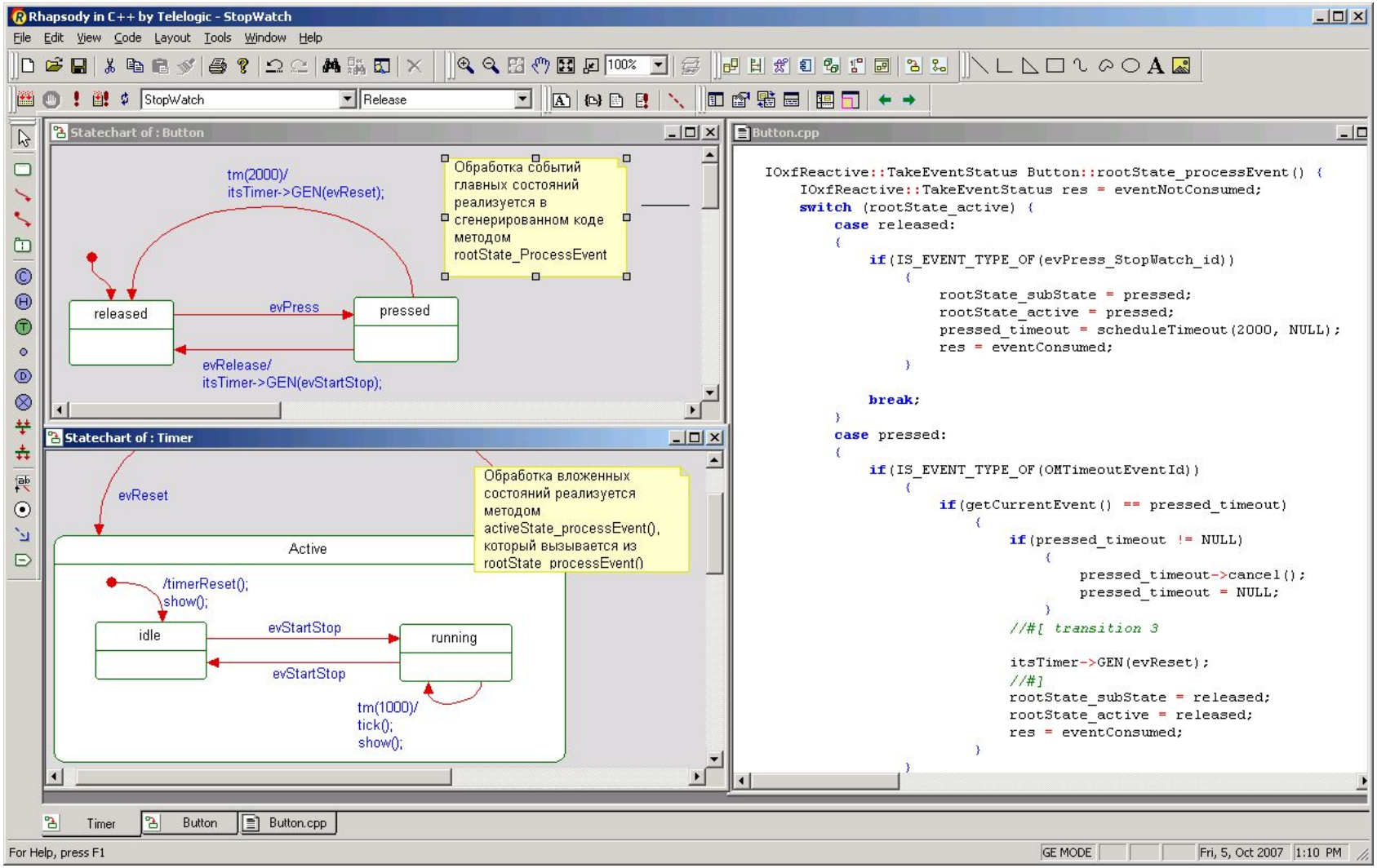
Создание объектов и инициализация связей



Определение сценариев для запуска и сброса секундомера



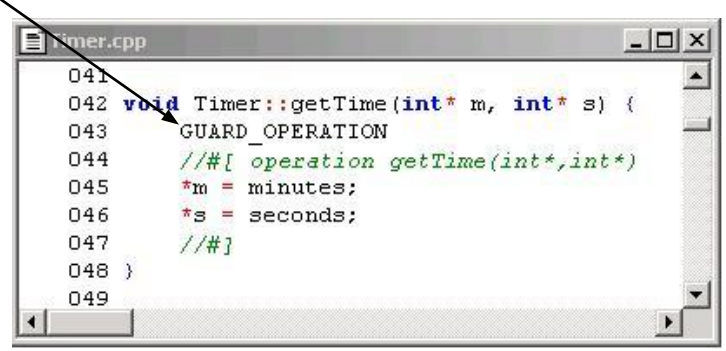
Определение реакции на события на диаграммах состояний



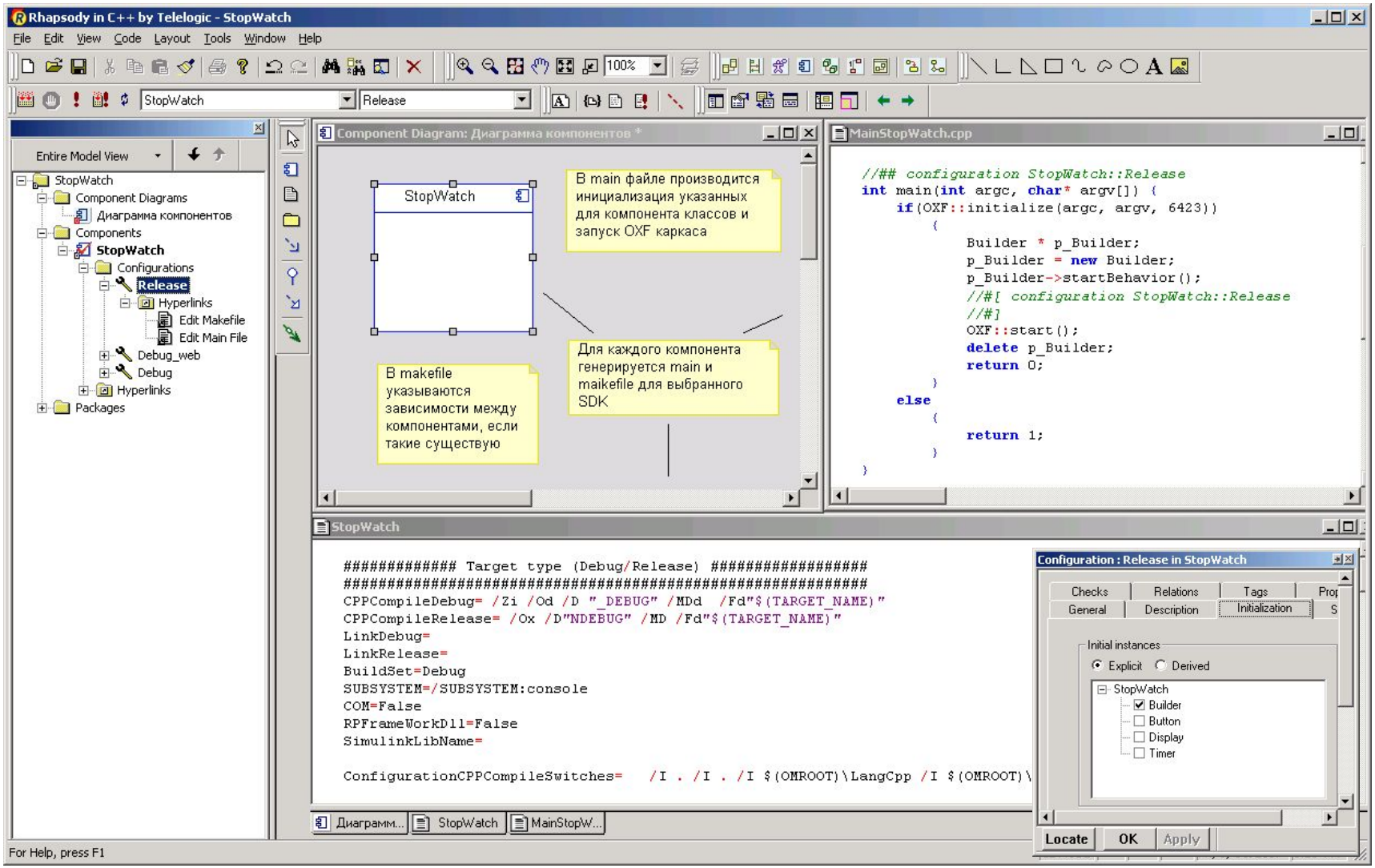
Зашита синхронных операций

- Установка для таймера свойства Concurrency=guarded
- В коде появляется макрос GUARD_OPERATION
- Обработка асинхронных событий становится защищенной (eventGuard != 0)

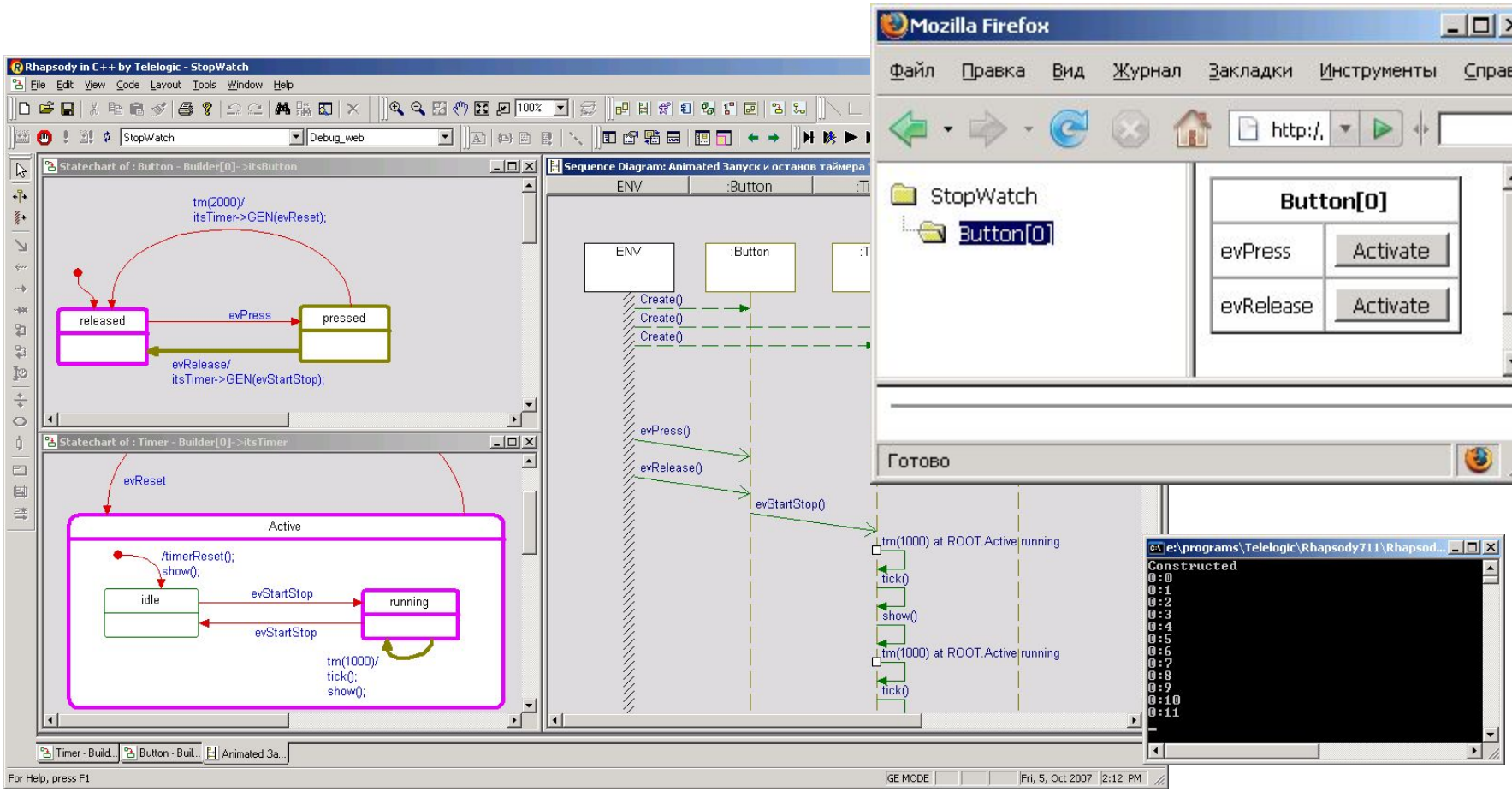

```
TakeEventStatus OMReactive::handleEvent(IOxfEvent* ev) {
  TakeEventStatus eventConsumeResult = eventNotConsumed;
  if (eventGuard != 0)
    eventGuard->lock(); // guard against synchronous ev
  eventConsumeResult = processEvent(ev);
  if (eventGuard != 0)
    eventGuard->unlock();
  return eventConsumeResult;
}
```



Определение компонента и сборка приложения



Отладка и тестирование приложения на уровне модели



Каркас приложения в Telelogic Rhapsody

Каркасы приложений

- Набор предопределённых взаимодействующих классов
- Предоставляют сервисы при разработке приложений определённого типа
- Разработка приложений путём наследования и переопределения

Преимущества использования каркасов

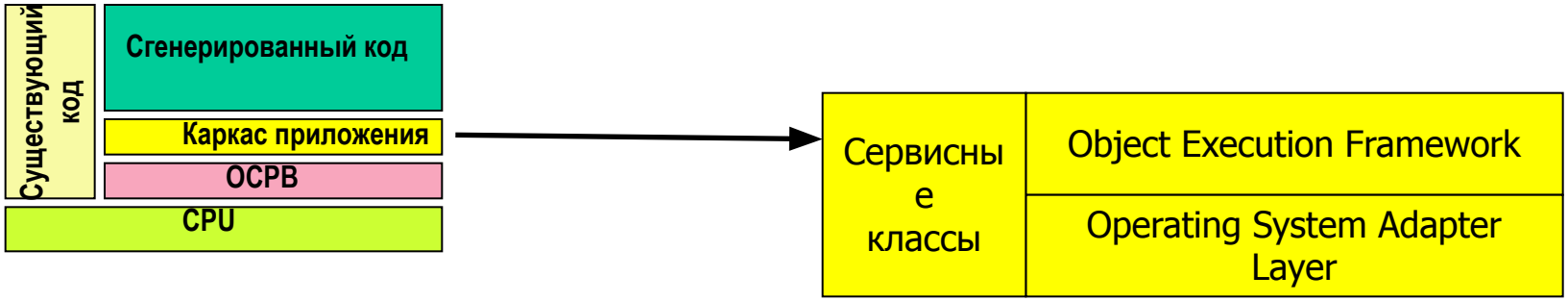
- Отсутствие необходимости создания приложений с нуля
- Определяют архитектуру целевых систем
- Представляют открытые конструкции, могут переопределяться в приложениях

Генерация кода приложений реального времени на основе каркаса

- В сгенерированном коде используется API каркаса
- Каркас реализует основные абстракции приложений реального времени
- Значительная часть функциональности содержится в классах каркаса
- Классы каркаса могут быть адаптированы под конкретные нужды
- Каркас – это библиотека, независимая от генератора кода
- Каркас не ограничивает приложения от использования других библиотек и сервисов ОС

Компоненты каркаса Telelogic Rhapsody

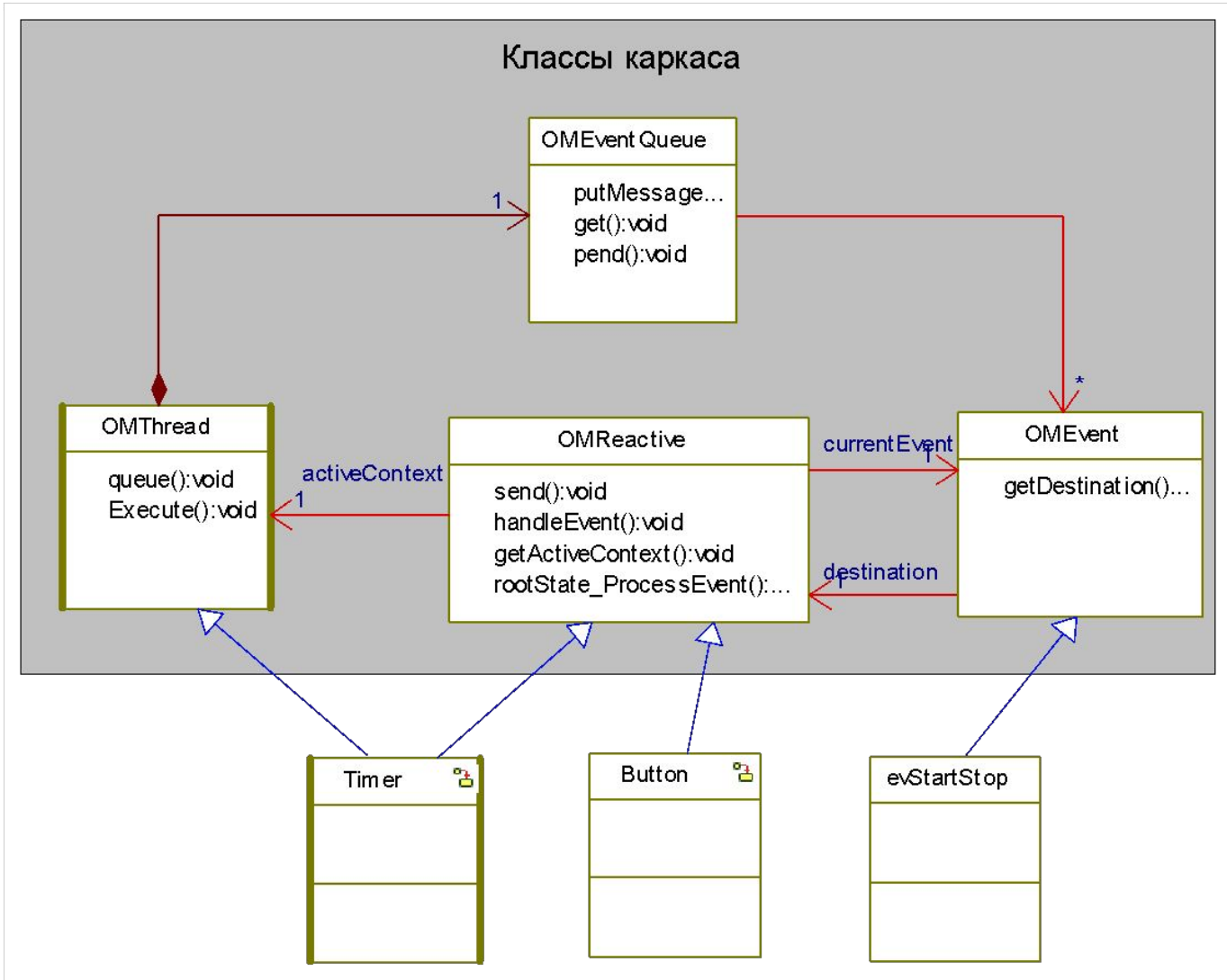
- Object Execution Framework (OXF)
- OS adapter level
- Сервисные классы
- Animation framework



Виды событий

- Асинхронные события
- События времени
- События вызова (синхронные)

Наследование от классов каркаса



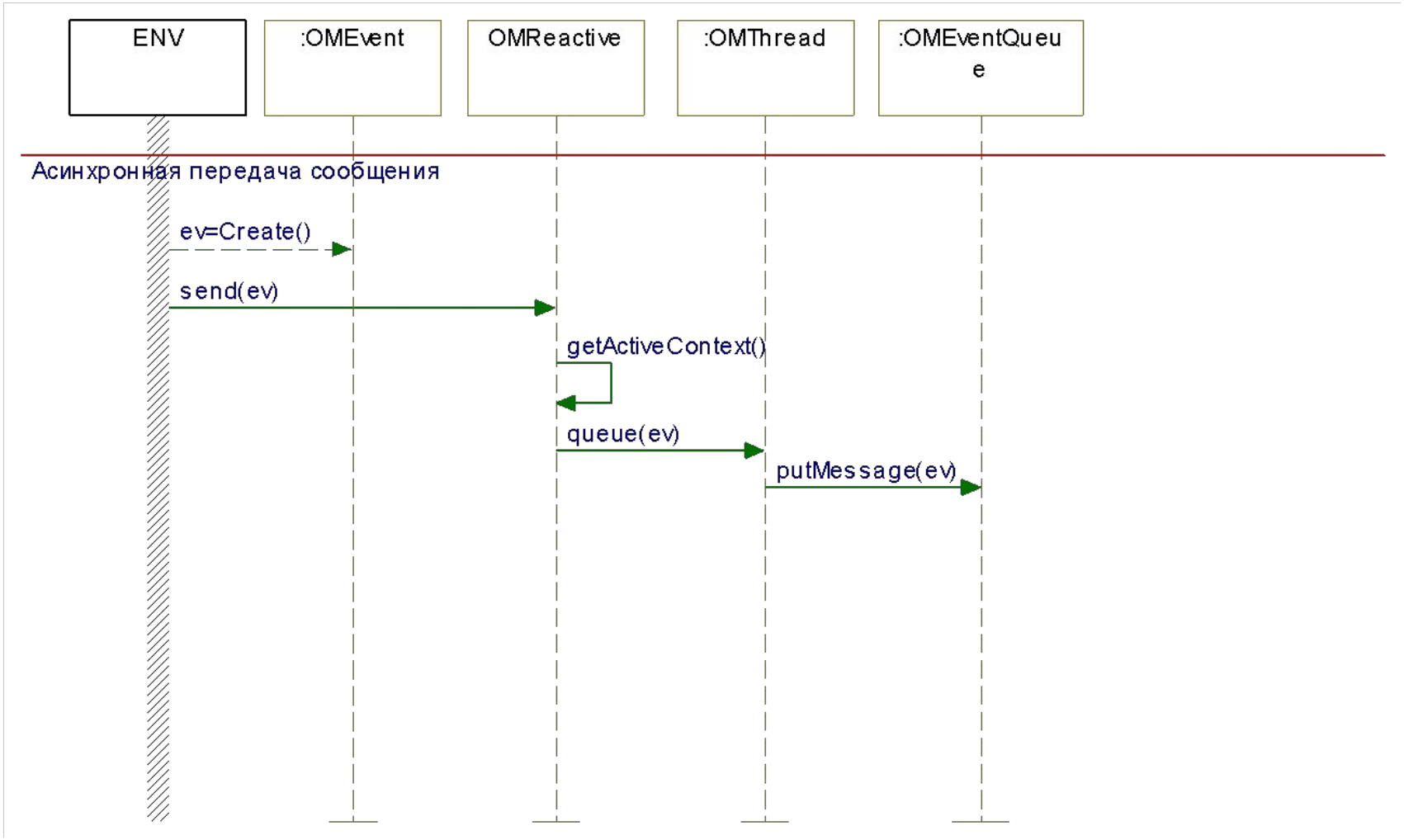
Активный класс

- Наследуется от класса OMThread каркаса
- Запускает в отдельном потоке функцию Execute
- Содержит очередь событий
- Предоставляет функцию queue для помещения событий в очередь
- В Execute разгребает очередь, передавая события адресатам на обработку в функцию handleEvent
- Позволяет перекрыть Execute для реализации другого поведения

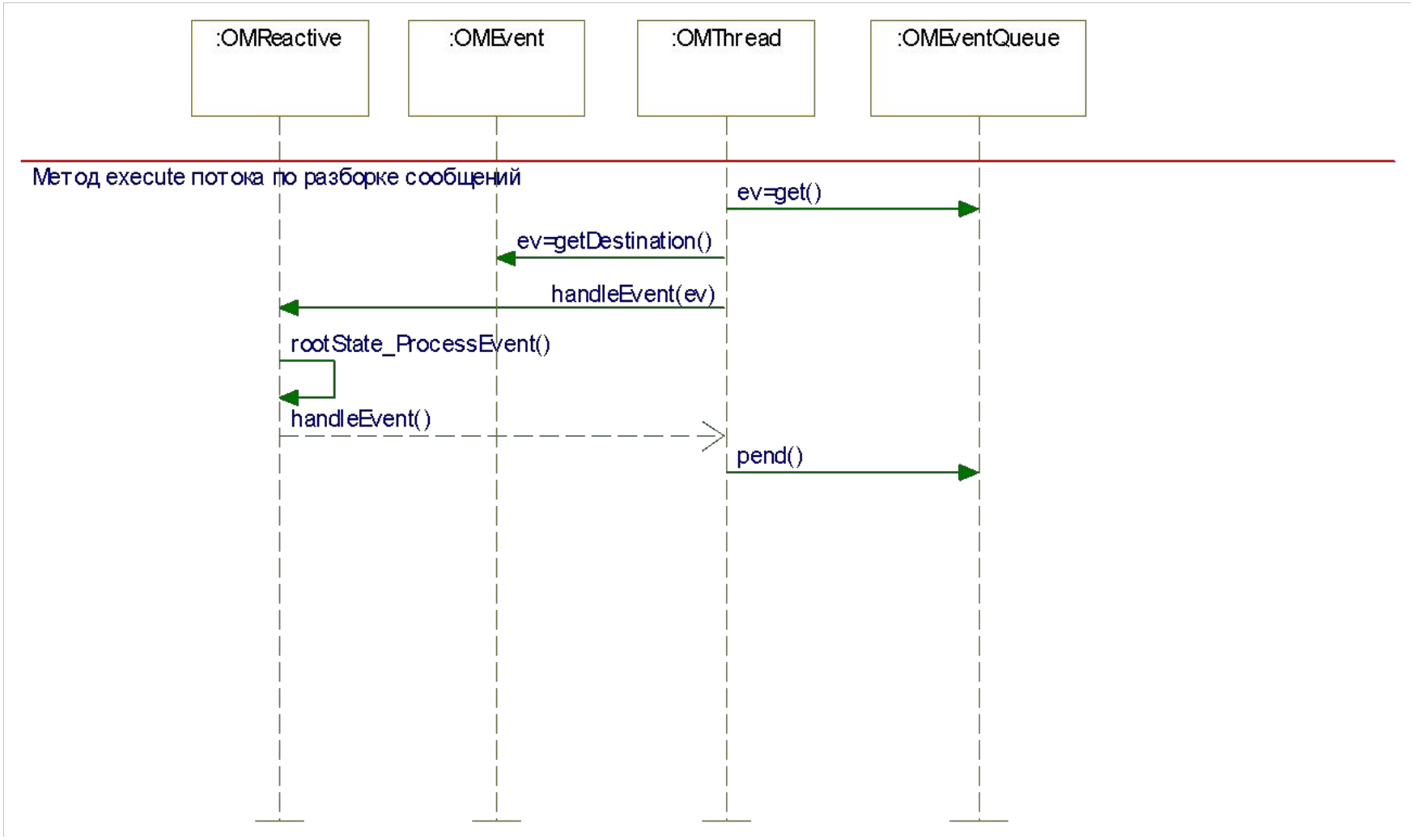
Реактивный класс

- Наследуется от класса OMReactive каркаса
- Предоставляет функцию send для передачи классу асинхронных событий
- Помещает полученные асинхронные события в связанный с ним активный класс для диспетчеризации
- Получает события от активного класса на обработку, вызывающего его функцию handleEvent
- Вызывает виртуальную функцию rootState_processEvent для обработки событий
- По умолчанию код для функции rootState_processEvent генерируется на основании диаграммы состояний

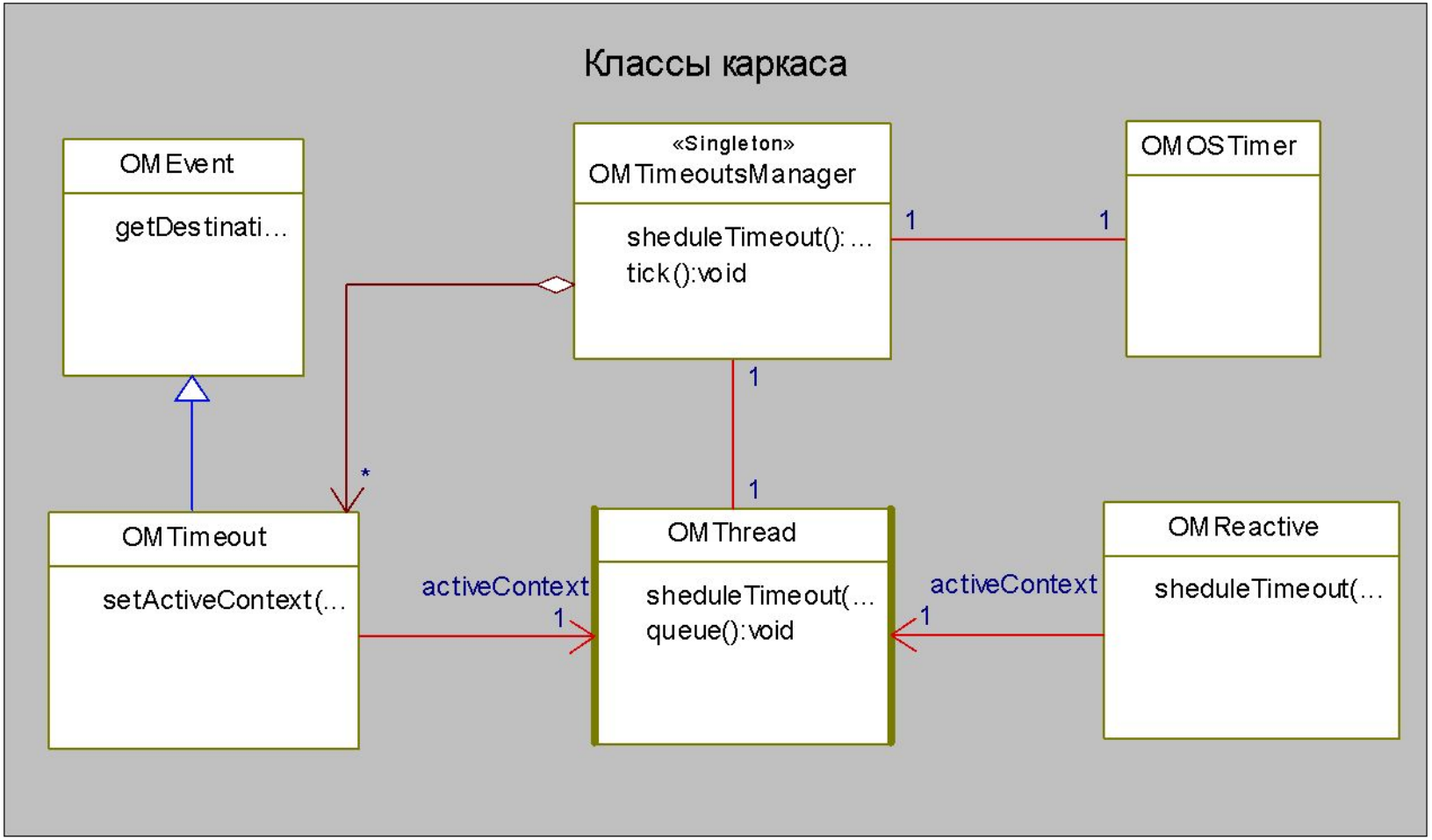
Посылка асинхронного события



Диспетчеризация асинхронных событий



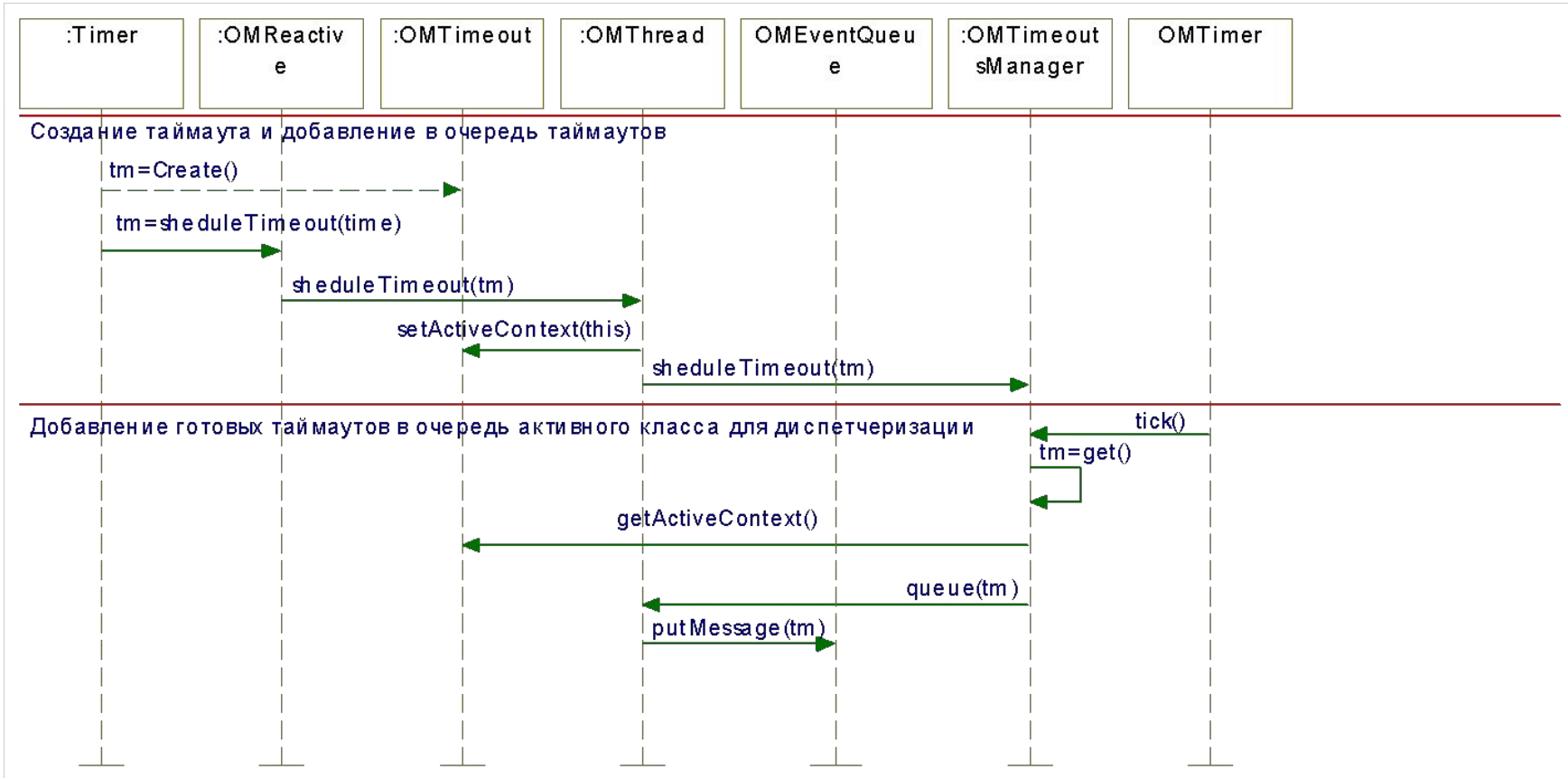
Классы каркаса для управления таймаутами



Таймауты

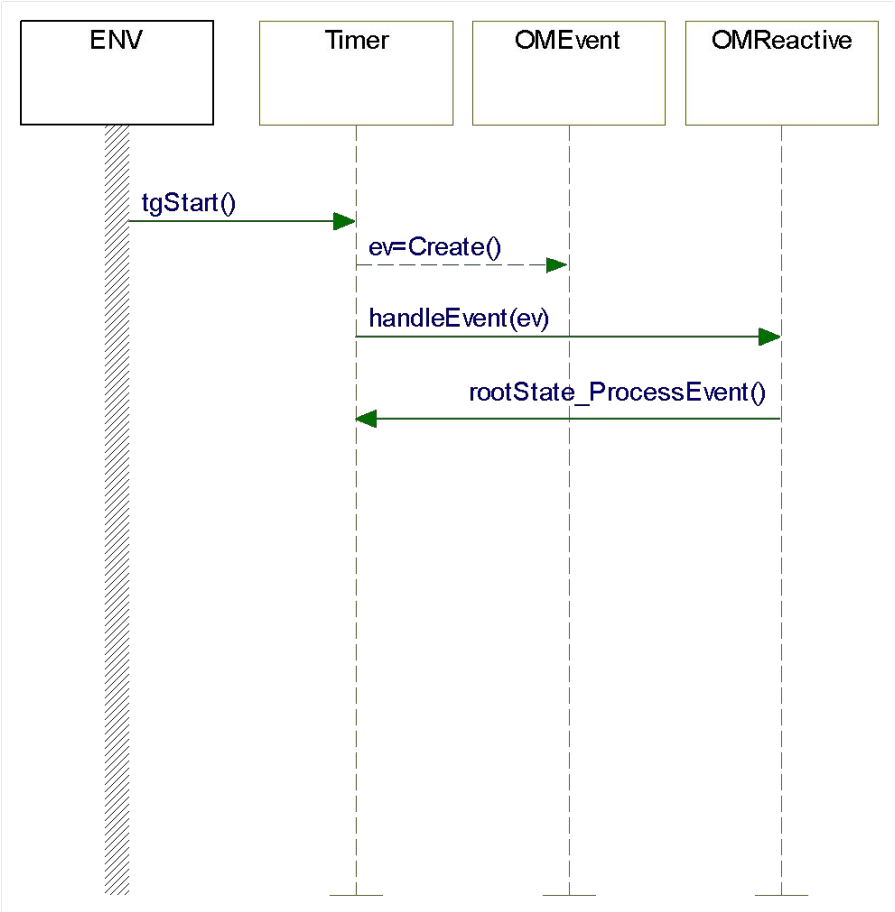
- Таймауты – это особый вид событий на которые можно определять реакции на диаграмме состояний
- Таймауты создаются в сгенерированном коде при входе в состояние и уничтожаются при выходе
- Всеми таймаутами управляет объект TimeoutsManager
- При истечении таймаута TimeoutsManager помещает его в очередь активного объекта для диспетчеризации
- Таймауты диспетчеризуются активными объектами наравне с другими событиями в очереди

Планирование таймаутов

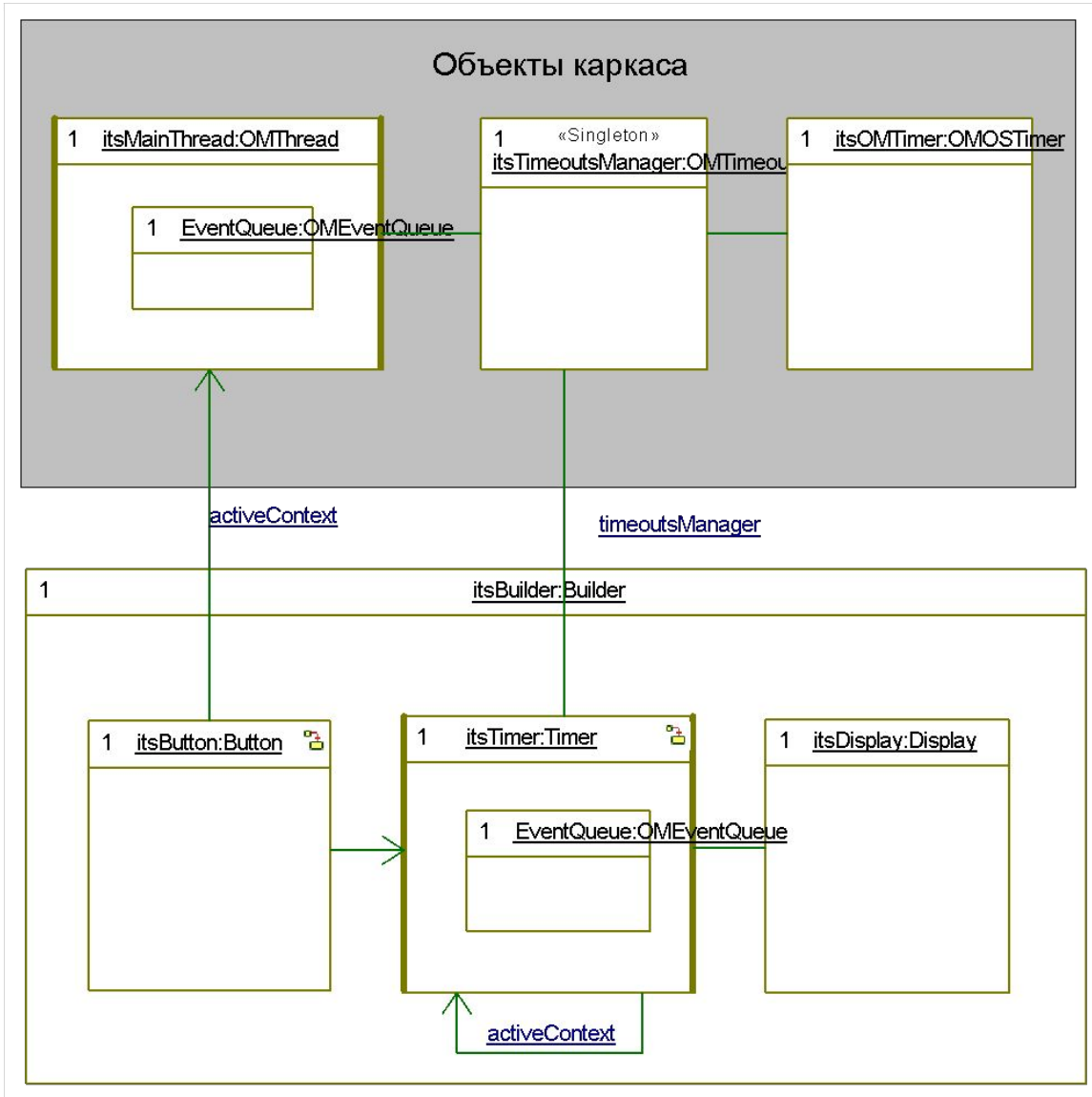


Синхронные события вызова

- События вызова генерируются при вызове тригерных операций класса
- В операции создаётся одноимённое событие и сразу же передаётся на обработку в handleEvent
- На диаграмме состояний можно определять переходы и реакции на такие события
- Реализация для тригерных операций генерируется автоматически



Объекты создаваемые каркасом



Связывание реактивных объектов с активными

- По умолчанию связывается с активным объектом MainThread
- Связывается с самим собой, если класс объявлен активным
- Связывается с содержащим его активным объектом
- Может быть связан с любым активным объектом путём вызова функции setActiveContext

Разновидности каркасов

- Object Execution Framework (использует ОС)
- Interrupt Driven Framework (не использует ОС)
- Synchronous Framework (не использует ОС)

Спасибо за внимание!

http://www.swd.ru/

196135, г. Санкт-Петербург,
пр. Юрия Гагарина 23
тел.: (812) 702-0833

115553, г. Москва,
пр. Андропова 22/30
тел.: (495) 780-8831