

## Лабораторная работа № 4

### Создание Active Diagram

#### *Назначение диаграммы активности*

Active Diagram (диаграмма активности) позволяет моделировать последовательность бизнес-процессов или операций класса по принципу от активности к активности или от активности к состоянию.

Этот тип диаграмм может использоваться для моделирования различных типов действий. Например, финансовая компания может использовать данный тип диаграмм для моделирования потоков финансовых документов, прохождения оплаты счетов или заказов.

Компания, которая создает программные продукты, может использовать данный тип диаграмм для отслеживания процесса разработки и создания программного обеспечения.

Active Diagram — это специальная разновидность диаграммы состояний. В этом типе диаграмм большинство используемых знаков — это знаки активности, переходы между которыми вызваны завершением одних действий и началом других.

#### *Отличия между Activities и Statechart*

Главное отличие между Activities и Statechart Diagram в том, что в первом случае основное — действия, а во втором — статичное состояние. При этом Active Diagram больше подходит для моделирования последовательности действий, а Statechart для моделирования дискретных состояний объекта.

#### *Создание диаграммы активности*

Выберем пункт Browse=>State Machine Diagram и создадим новую диаграмму Active как показано на рис. 25.

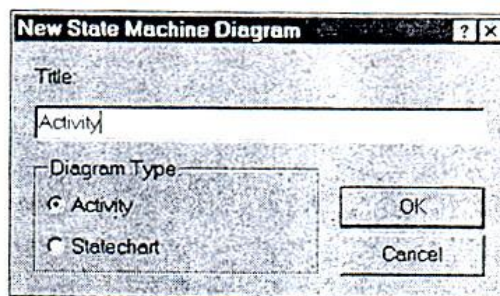


Рис.25. Создание диаграммы активности

#### *Строка инструментов*

После того как диаграмма будет активизирована, строка инструментов будет иметь следующий вид (рис. 26).



Рис.26. Строка инструментов диаграммы активности

#### *Activity (активность)*

Значок Activity обозначает выполняемые задачи или выполнение определенных действий в течение жизни объекта. Этот значок, в общем-то, похож на State, но если значок State обычно обозначает ожидание какого-либо события, то значок Activity обозначает непосредственное действие.

#### *State Transition (переход состояния)*

State Transition — переход из одного состояния в другое или по завершении выполнения определенного действия в начало другого. Этот значок также может характеризовать получение объектом некоторого сообщения с дальнейшей его

обработкой. State Transition может осуществляться как между Action-Action и State-State, так и между State-Action и Action-State.

Возможна также установка нескольких переходов между двумя состояниями или действиями. Каждый такой переход уникален и показывает реакцию объекта на определенное сообщение. Таким образом, нельзя создать несколько переходов между двумя состояниями с указанием одного и того же сообщения.

#### *Synchronizations (синхронизация)*

Значок Synchronizations позволяет определить независимо выполняемые действия. При этом действия разделяются на несколько выполняемых независимо и только по завершении всех действий объект продолжает работу. Этот значок представляет собой горизонтальную или вертикальную черту, обозначающую синхронизацию выполняемых работ.

#### *Decision (решение)*

Decision позволяет показать зависимость дальнейшей работы от внешних условий или решений. Этот значок аналогичен командам языка if или case и может иметь больше двух выходов, но обычно используют выбор из двух переходов, определенных Булевым выражением.

#### *Swimlanes (ограничение)*

Swimlanes (в прямом переводе «плавательные дорожки») позволяет моделировать последовательность действий различных объектов и связи между ними. При помощи этого элемента можно моделировать бизнес-процессы организации, отражая на диаграмме различные подразделения и объекты, играющие важные роли в модели бизнеса.

Swimlanes позволяет показать, кто выполняет те или иные роли в процессе. Для этого необходимо переместить соответствующие значки активности или состояний в зону определенного подразделения, отделенного от остальных Swimlane.

#### **Начало создания диаграммы**

Теперь посмотрим, как изменится диаграмма, если создавать не диаграмму состояния, а диаграмму активности для работы объекта EnvironmentController.

Создадим начало диаграммы, поместив на диаграмму значок Start State и состояние Idle. Соединим их связью State Transition и назовем ее New Planting.

Чтобы не вводить еще раз эти значки, можно выделить их на уже созданной диаграмме Statechart и перенести во вновь созданную.

Теперь вспоминаем, что перед тем, как перейти в состояние ожидания, контроллер должен установить значение таймера, продолжительность выращивания, время ожидания между опросами датчиков. В этом случае можно сразу разделить функции, выполняемые таймером и контроллером, для чего создадим отдельные разделители Swimlane для этих объектов.

#### **Редактирование спецификаций Swimlanes**

Добавим новые разделители Swimlanes для таймера и контроллера. Эти разделители будут названы программой Rational Rose автоматически NewSwimlane и NewSwimlane1, что, конечно же, необходимо исправить. Для этого активизируем окно свойств Swimlane. Для доступа к окну свойств необходимо выбрать заголовок линии и далее RClick=>OpenSpecification. Откроется окно, показанное на рис. 27.

Здесь можно изменить название и имя класса, из которого получен объект. В нашем случае это plantTimer. И конечно же, можно документировать полученную Swimlane.

Введем информацию, как она представлена на рисунке. Аналогично изменим информацию для следующей линии, которая представляет Controller, полученный из класса EnvironmentalController.

Теперь необходимо «перетащить» введенные значки в колонку Controller, для того чтобы показать, что основным действующим лицом в нашем случае является именно он. Если вы проделали все действия, то должно получиться то, что показано на рис. 28.

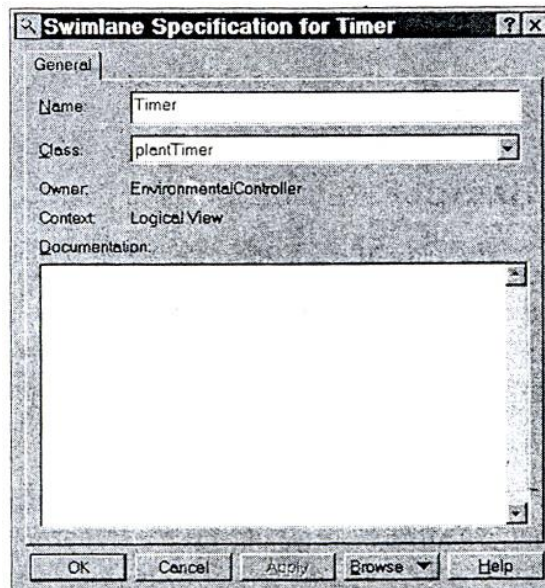


Рис.27. Редактирование свойств Swimlanes

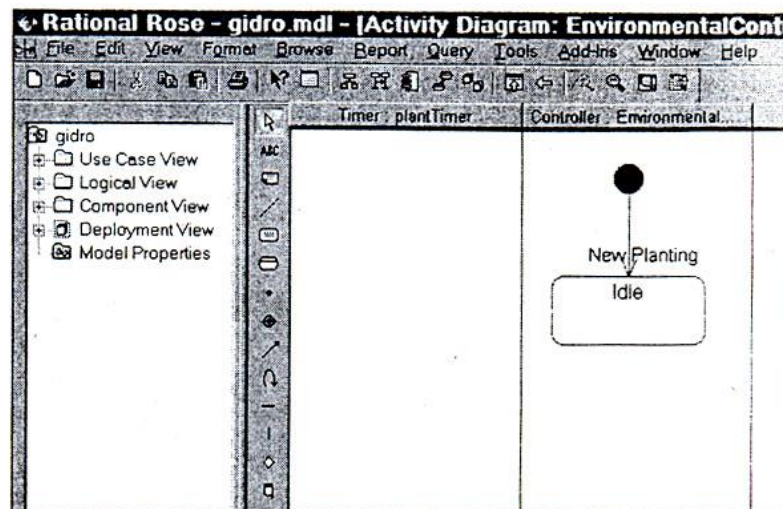


Рис.28. Добавленные в диаграмму активности Swimlanes

Однако для установки таймера необходимо произвести действие установки. Отразим этот момент добавлением нового значка активности.

#### **Настройка спецификаций значка активности**

Для значка активности доступно следующее окно изменения свойств, показанное на рис. 29.

Изменим наименование на SetTimer. Откроем вкладку Actions и добавим новое действие RClick=>Insert. Перейдем в окно свойств действия и получим окно, показанное на рис.30.

Возможные варианты:

- On entry — действие происходит при входе данный тип активности;
- On exit — действие происходит при выходе из данного типа;
- Do — действие происходит между действиями входа и выхода;
- On Event — действие происходит в ответ на определенное событие.

Если выбрано действие в ответ на определенное событие, то открываются поля ввода аргументов этого события.

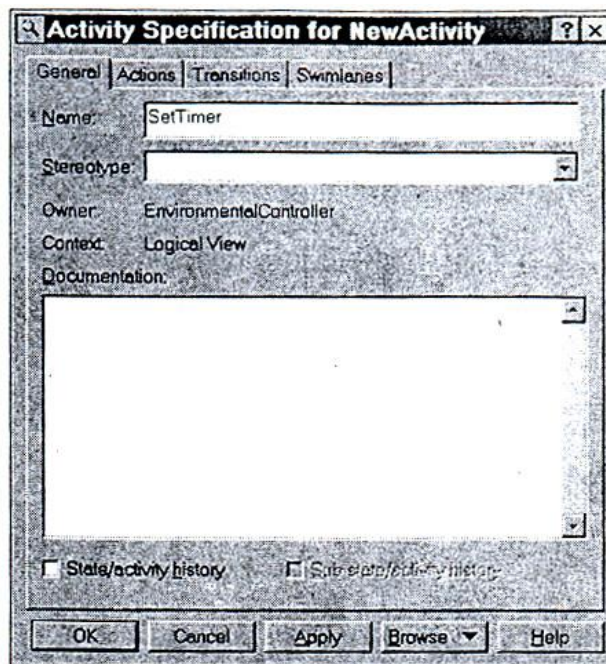


Рис.29. Свойства знача активности

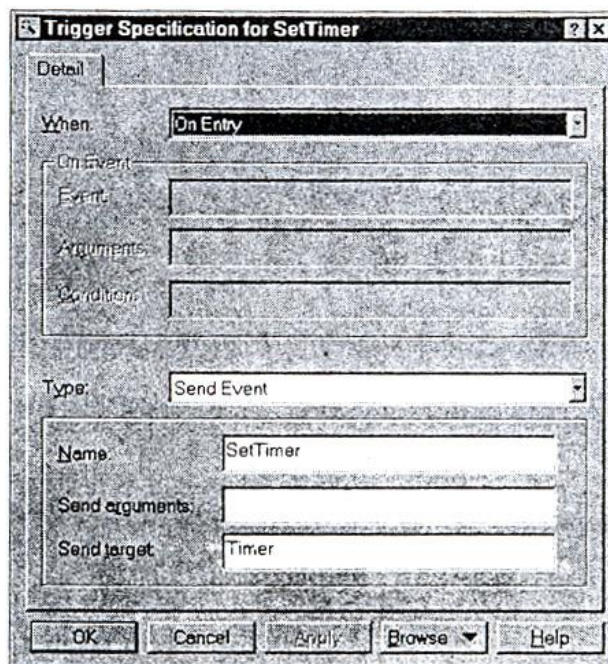


Рис.30. Настройка спецификаций

В поле ввода типа можно указать, какой тип активности предполагается использовать. Это может быть Action (действие) или Send Event (посылка сообщения).

- Action может использоваться для того, чтобы показать, что данный тип активности не взаимодействует с другим объектом, и действие происходит внутри объекта.
- Send Event показывает, что происходит взаимодействие с другими объектами посредством посылки сообщений. Здесь можно ввести имя сообщения, его аргументы и адресат сообщения, как сделано в нашем случае.



Вкладка Transitions показывает все входы и выходы из значка активности (рис.31).

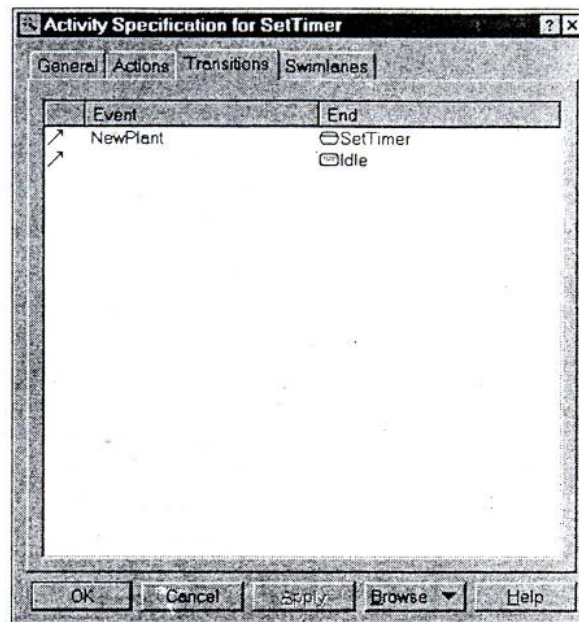


Рис. 31. Вкладка Transitions

Здесь можно подробно рассмотреть все входящие и выходящие события из данного состояния и изменить их спецификации прямо из этого окна.

Вкладка Swimlanes показывает, с какими Swimlane значок Activity или State взаимодействует.

Если все было заполнено правильно, то должно получиться состояние, показанное на рис. 32

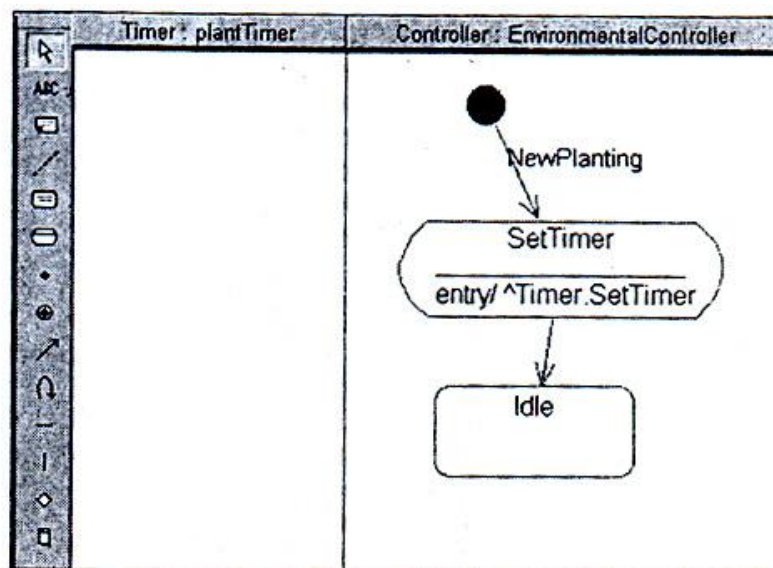


Рис.32.Диаграммы после заполнения значка активности