Лабораторная работа № 7
Диаграмма сотрудничества Collaboration

Назначение диаграммы

Второй, уже упоминавшийся тип диаграмм взаимодействия, — это Collaboration Diagram. С английского языка этот термин переводится как «сотрудничество». Действительно, эта диаграмма отличается от предыдущей тем, что она не акцентирует внимание на последовательности передачи сообщений, она отражает наличие взаимосвязей вообще, то есть на этой диаграмме отражается наличие сообщений от клиентов к серверам. Так как временная шкала не участвует в демонстрации сообщений, то эта диаграмма получается компактней и как нельзя лучше подходит для того, чтобы окинуть одним взглядом взаимодействие всех объектов.

Однако необходимо понимать, что диаграмма показывает взаимодействие между объектами, а не классами, то есть является мгновенным снимком объектов системы в некотором состоянии. Ведь объекты, в отличие от созданных на этапе проектирования классов, создаются и уничтожаются на всем протяжении работы программы. И в каждый момент имеется конкретная группа объектов, с которыми осуществляется работа. В связи с этим появляются такие понятия, как время жизни и область видимости объектов, которые будут рассмотрены далее.

Создание диаграммы

Разработчики Rational Rose заложили удобную возможность создания на основе диаграммы Sequence диаграммы Collaboration и наоборот.

А так как диаграмма Sequence у нас уже есть, то создадим на ее основе Collaboration. Для этого, находясь в диаграмме, сделаем следующее: Menu: Browse=>Create Collaboration Diagram.

При этом для создания необходимого типа диаграммы в диалоговом окне (рис.50) выберите тип диаграммы Collaboration.

Однако в этом случае будет создана пустая диаграмма, и уже созданные объекты не будут туда перенесены. Поэтому для нас лучше подходит первый вариант.

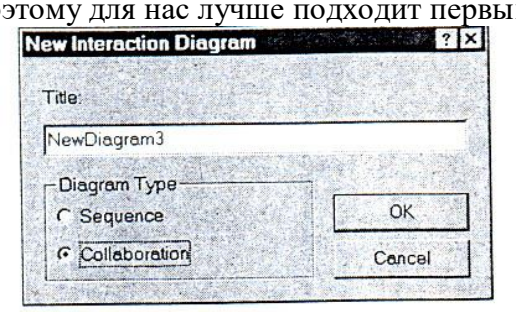

Рис.50.Окно выбора создания Collaboration диаграммы

Будет создана диаграмма, представляющая на первый взгляд, нагромождение значков. Но после того как вы ее «растянете» мышкой, диаграмма приобретет вполне читаемый вид (рис.51).

На этой диаграмме объект Controller центральный и все сообщения поступают к нему или исходят от него, что совершенно правильно отражает наше представление о системе. Взаимодействия между контроллером и остальными устройствами изображаются линиями с добавленными стрелками, аналогичными тем, которые изображаются на Sequence Diagram. Но нетрудно заметить, что все сообщения одного направления собираются вместе и даются как подпись к одной стрелке, таким образом получаем полную картину взаимодействия

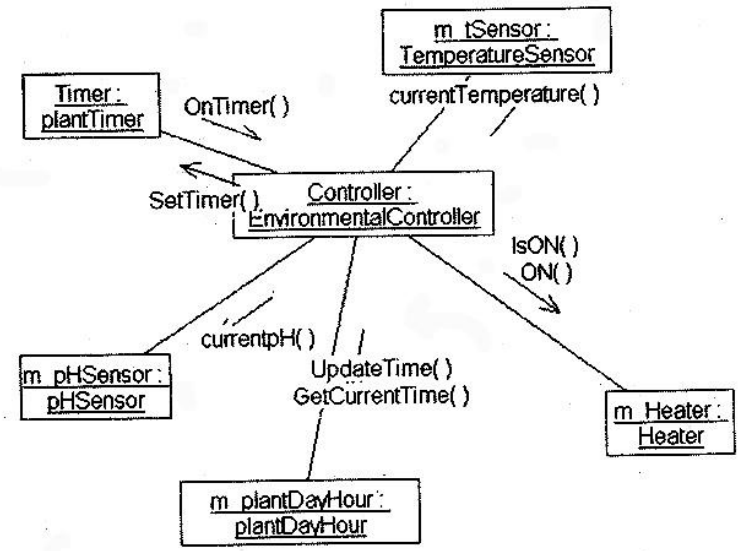

Рис.51. Автоматически созданная Collaboration диаграмма

Строка инструментов

При активизации диаграммы строка инструментов приобретает следующий вид (рис.52).

Рис.52.Строка инструментов Collaboration Diagram

Object (объект)

Object позволяет создавать объекты, которые имеют состояния, поведения и индивидуальны. Каждый объект на диаграмме показывает реализацию некоторого класса.

Class Instance (реализация класса)

Class Instance позволяет добавлять абстрактные реализации класса в диаграмму. В чем разница между объектом и абстрактной реализацией класса?

При добавлении значка на диаграмму внешне эти значки не отличаются. Отличие в их внутреннем содержании. Объект подразумевает настройку его времени жизни и других свойств, присущих только конкретному объекту. Абстрактная реализация класса не позволяет изменять эти свойства и предназначена только для показа взаимодействия.

Object Link (связь объекта)

Взаимодействия объектов отражается посредством показа их связей. Существование связей между двумя классами символизирует взаимодействие между их реализациями (объектами, созданными на основе этих классов). При этом один объект может посылать сообщение другому объекту.

Link To Self (связь с самим собой)

Так как объекты могут посылать сообщения самим себе, то данный значок показывает, что объект имеет обратную связь с самим собой.

Link Message (передача сообщения)

Link Message позволяет отразить связь, которая подразумевает обязательную передачу сообщения.

Reverse Link Message (обратная передача сообщения)

Reverse Link Message позволяет отразить связь, которая подразумевает обязательную передачу сообщения аналогично предыдущему пункту, но в обратном направлении.

Data Flow (поток данных)

Data Flow позволяет отразить связь, показывающую, что происходит передача данных от одного объекта к другому.

Reverse Data Flow (обратный поток данных)

Как и предыдущий значок, позволяет отразить связь, показывающую, что происходит передача данных от одного объекта к другому, но в обратном направлении.

Создание объекта

Добавим в диаграмму новый объект m_WaterTank.

Контекстное меню объекта

После добавления можно посмотреть, что нам предлагает контекстное меню (рис. 53).

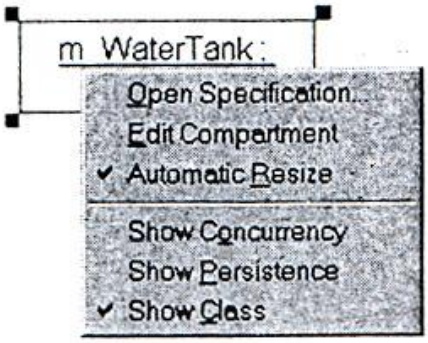

Рис.53. Контекстное меню объекта

Коротко перечислим возможности, предоставляемые посредством данного меню.

- Open Specification — редактирование спецификаций объекта;
- Edit Compartment активизирует диалоговое окно показа дополнительной информации об объекте. Содержание такой информации зависит от типа объекта;
- Automatic Resize позволяет устанавливать автоматическую настройку размера объекта по длине содержащегося в нем текста;
- Show Concurrency позволяет включить показ на данном значке типа согласования при создании многопотоковой программы. Данный тип определен в классе;
- Show Persistence позволяет показать на диаграмме время жизни объекта;
- Show Class позволяет показать на диаграмме имя класса.

Создание связи между объектами

Соединим новый объект с объектом Controller линией Object Link. Для того чтобы показать, что от контроллера будут поступать сообщения нажмем Link Message. После того как курсор приобретет вид креста, укажем на линию. Рядом с ней появится стрелка сообщения, для которой можно ввести свойства, аналогичные свойствам сообщения на Sequence Diagram, при помощи контекстного меню.

Для того чтобы показать, что объект m_WaterTank возвращает данные по запросам контроллера IsON (включен ли) и IsOFF (выключен ли), создадим указатель Reverse Data Flow. Для этого нажмем на указанный значок, и после того, как курсор примет форму креста, укажем на стрелку сообщения, но не на линию Link Message. После заполнения свойств сообщения должно получиться примерно следующее (рис.54).

Указание передачи данных является справочной информацией и показывает нам, что как минимум одно из сообщений должно возвращать значение, обрабатываемое контроллером

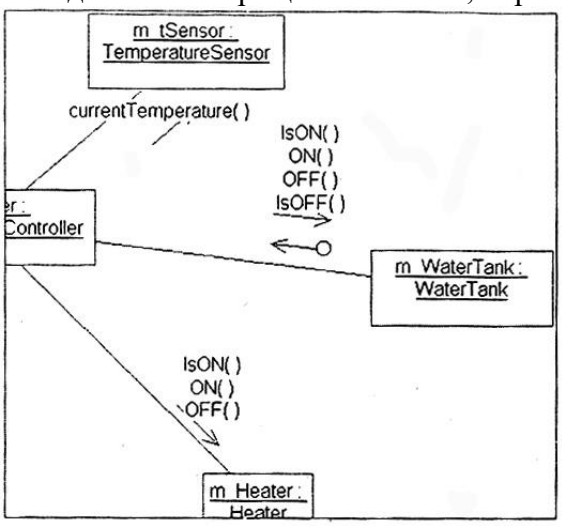

Рис.54. Указание передачи данных

Автоматический перенос данных в диаграмму Sequence

Если вы переключитесь в диаграмму Sequence, то увидите рис.55. Это Rational Rose автоматически перенес введенные нами изменения в диаграмму Sequence, избавив нас тем самым от монотонного труда по переносу данных из одной диаграммы в другую. Однако необходимо расставить сообщения в нужном порядке, о котором программа естественно, не догадывается и имеет в виду, что в каком порядке сообщения введены, в таком порядке они и выполняются.

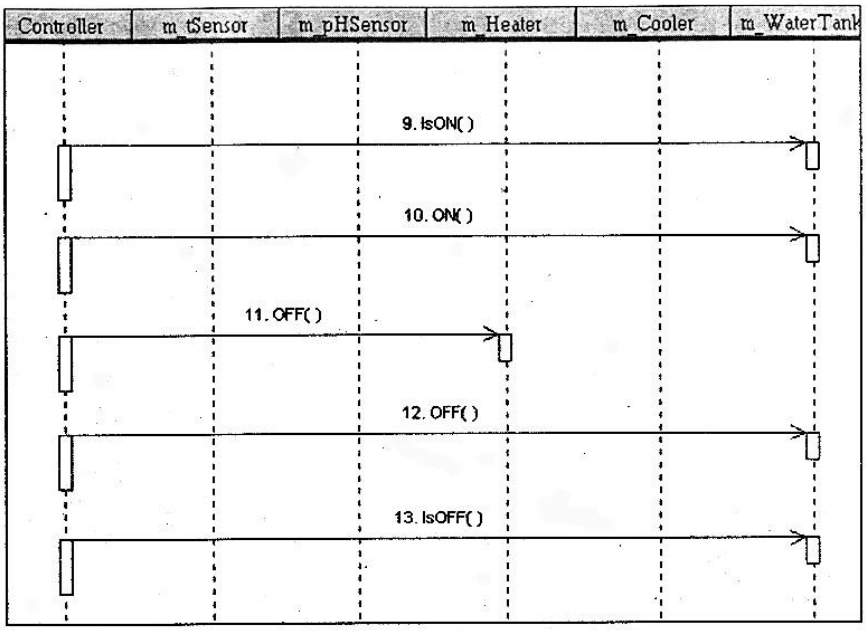

Рис. 55. Автоматическое заполнение Sequence Diagram

Настройка области видимости объектов

У каждой связи Link Message есть соответствующие свойства, которые позволяют настроить область видимости для связанных объектов.

Для начала разберемся с понятием «область видимости».

В любой, даже небольшой программе, могут существовать переменные и функции, принадлежащие разным классам и имеющие одинаковые имена. В том случае, когда

пользуются классы, написанные разными людьми, которые возможно никогда не видели друг друга, например, при использовании библиотек классов, трудно добиться уникальности имен. Читатель, знакомый с языком программирования СУБД FoxPro, знает, как трудно найти ошибку при изменении незначительной переменной в подпрограмме. Для тех, кто не знаком с этим языком поясним, что в этом языке область видимости простирается от определения переменной и далее.

Таким образом, все определенные переменные а головной программе по умолчанию видны а подпрограммах.

Это удобно для того, чтобы не передавать параметры. Но представьте, что в некотором цикле по переменной i вызывается функция, которая написана другим программистом и тоже использует свой цикл по переменной i. В этом случае работа программы будет нарушена с непредсказуемыми последствиями.

К счастью, язык С значительно строже относится к области видимости и переменные, определенные в подпрограмме или даже выделенные блоком в самой программе, будут скрыты от изменения в других частях программы. Однако необходимо заботиться о передаче этих переменных как параметров.

При создании класса в С++ область видимости переменных имеет нечто промежуточное между описанными выше двумя вариантами. Переменные, определенные как члены класса, видны всеми его методами так, как если бы они были определены глобально, и, в то же время, они могут быть скрыты от внешнего случайного или даже намеренного изменения, если они определены в разделе private или protected.

Для того чтобы задать область видимости, перейдем в RClick=>Open Specification и попадем в диалоговое окно, представленное на рис. 56.

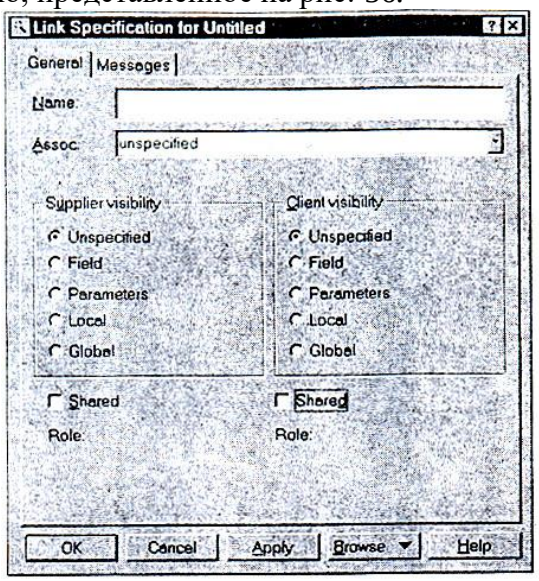

Рис.56. Задание области видимости для Link Message

Для задания области видимости объекта-сервера служит блок Supplier visibility, клиента — Client visibility.

В этих блоках доступны значения для выбора:

- Unspecified — не определено, это значение присваивается по умолчанию;
- Field — объект включен в другой объект;
- Parameters — объект передается параметром в другой объект;
- Local — объект локально определен в границах другого объекта;
- Global — объект глобален по отношению к другому объекту.

При изменении области видимости на концах соединяющей линии появляется квадратик с указанной областью видимости. Здесь же можно установить флажок, показывающий, что объект используется совместно (Shared).

Изменение свойств сообщений

Закладка Message (рис. 57) позволяет изменять свойства сообщений для выбранной связи. Причем можно добавлять методы прямо из этого окна как для клиента, так и для сервера. На рисунке показано окно списка сообщении с активизированным контекстным меню.

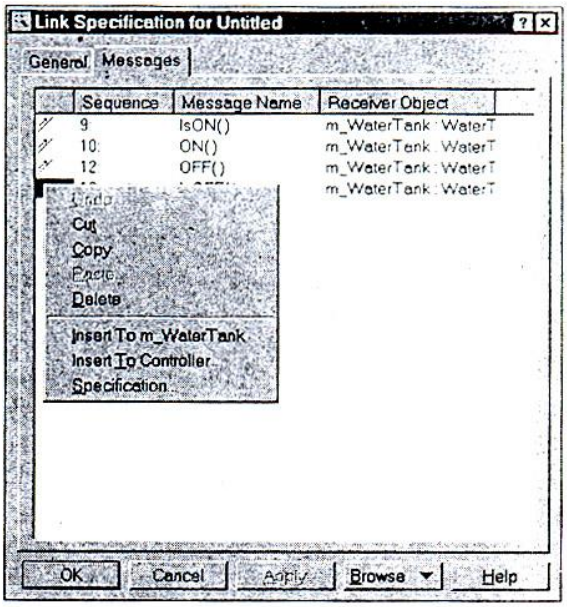

Рис. 57 Доступ к списку сообщений

Однако нужно помнить, что добавление методов в этом окне не добавит их в соответствующие классы. Это, с одной стороны, не совсем удобно, так как Нужно добавлять методы непосредственно в классе, но становится просто необходимо, если методы наследуются из базовых классов и добавлять их в дочерние вы не собираетесь.

Теперь можно добавить объект Light (лампочка) и указать область видимости для остальных Link линий. Должно получиться следующее (рис.58).

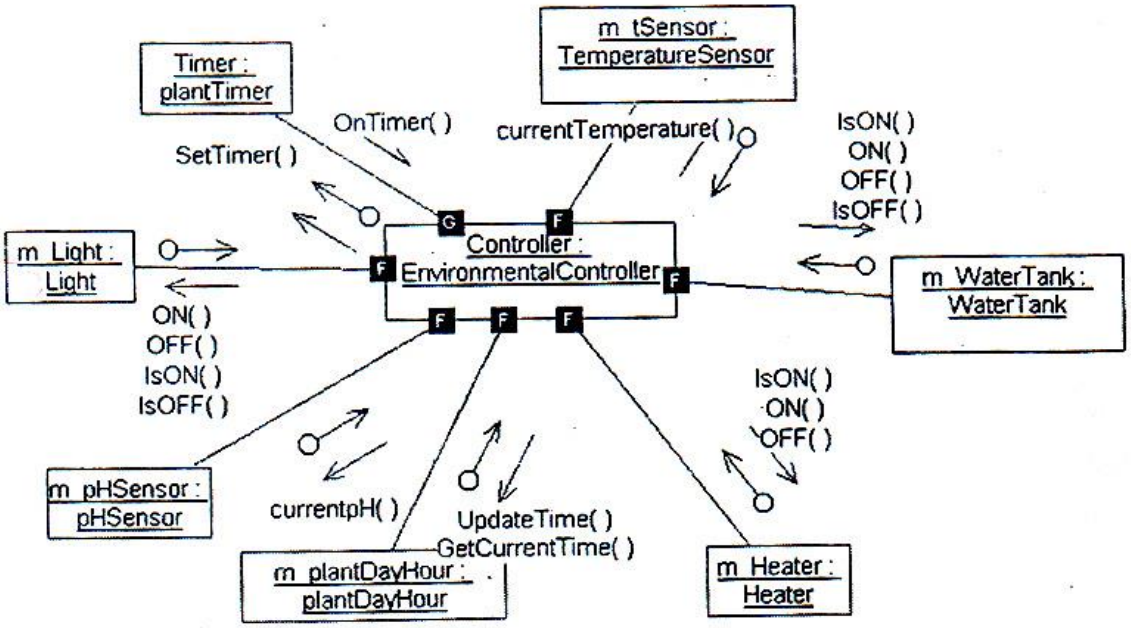

Рис.58. Итоговая Collaboration диаграмма

Еще раз об области видимости

Если внимательно присмотреться, то для всех объектов, кроме таймера, можно поставить область видимости Field, а на объекте Timer стоит область видимости Global. Это не случайно.

Область видимости конкретных объектов определяется при проектировании структуры программы и в данном случае все объекты сделаны полями в объекте контроллера, а таймер будет внешним по отношению к контроллеру поэтому его указывают как глобальный. Под глобальным объектом подразумеваю объект, созданный в объекте контейнере, в который входит и контроллер. Таким объектом может быть Application (объект приложения), или можно создать объект GreenHouse (теплица), который описывает физическое наличие в нашей теплице различных устройств, при этом все исполнительные и измерительные устройства входили бы в этот объект.

В последнем случае можно было бы использовать массивы устройств, для того чтобы пользователь системы мог указывать количество конкретных управляющих устройств (в том числе и контроллеров), находящихся в теплице. Но так как задача больше учебная, чем практическая, то используем классы минимальной сложности и наибольшей иллюстративности.