Лабораторная работа 1

Построение структурной модели телекоммуникационной системы с помощью пакета PragmaDev Studio

Цель: Изучить этапы создания проекта в пакете PragmaDev Studio на примере построения заданной системы и формирования структурной диаграммы взаимодействия элементов этой системы.

Задание:

1. Создать проект в пакете PragmaDev Studio и выполнить пример из п.2.2 и 2.3 данных методических указаний. Сделанный проект сохранить для использования при выполнении заданий лабораторной работы 2 и3.
2. Выполнить индивидуальное задание (п.3), в котором по номеру варианта (по последней цифре пароля, если 0, то вариант 10) следует:
   - создать в пакете PragmaDev Studio заданную структуру системы, отразить указанные взаимосвязи между элементами системы, а также связи с окружением;
   - средствами языка SDL построить структурную диаграмму взаимодействия элементов заданной системы;
   - результаты отразить в отчете.

1. Методические указания к выполнению лабораторной работы

1.1. Общее описание моделируемой системы

При выполнении всего цикла лабораторных работ с помощью пакета PragmaDev Studio в качестве примера будем рассматривать сравнительно простую систему связи, которая состоит из центрального блока и нескольких оконечных устройств. Программная поддержка рассматриваемой системы включает в себя:

1) процесс pCentral, который размещается в центральном блоке и реализует функции этого блока;
2) процесс pLocal, алгоритм работы которого обеспечивает непосредственное обслуживание поступающих вызовов; этот процесс размещается в каждом

оконечном устройстве, т.е. существует в нескольких экземплярах.

Оконечным устройствам присвоены определенные номера (будем называть их списочными), и эти номера используются абонентами для связи друг с другом. Кроме того, на стадии инициализации системы, когда процесс pCentral порождает необходимое количество экземпляров процесса pLocal и загружает их в оконечные устройства, каждому экземпляру автоматически присваивается уникальный внутренний идентификатор PId (Process Identifier). Параллельно формируются данные о соответствии между списочными номерами и идентификаторами PId. В дальнейшем эти данные хранятся в памяти центрального блока, и процесс pCentral использует их при своей работе.

Если абоненту A нужно связаться с другим абонентом B, то он должен использовать списочный номер вызываемого абонента B. Экземпляр процесса pLocal, закрепленный за оконечным устройством абонента A, получает этот номер и отправляет его процессу pCentral. В ответ процесс pCentral должен сообщить PId другого экземпляра процесса pLocal, который управляет оконечным устройством абонента B. Затем указанные экземпляры процесса pLocal самостоятельно взаимодействуют друг с другом, обеспечивая логическую последовательность действий на всех этапах обслуживания поступившего вызова с учетом разнообразия ситуаций, возникающих на этих этапах.

1.2. Диаграмма взаимодействия объектов в составе системы

Для описания в более строгом (формализованном) виде последовательности происходящих событий, которые относятся к объектам в составе моделируемой системы, воспользуемся языком диаграмм взаимодействия (Message Sequence Charts – MSC). Основным элементом при построении такой диаграммы (часто её называют «стрелочная диаграмма») является *трасса* объекта – отдельная вертикальная ось времени между двумя прямоугольниками, которые называют стартовым (вверху) и конечным (внизу). Вдоль этой оси откладываются события, имеющие отношение к конкретному объекту, имя которого указывается в стартовом прямоугольнике. Взаимодействие между двумя объектами (или меж-

ду объектом и окружением системы) осуществляется только при помощи передачи некоторых сообщений, и каждое событие обозначается горизонтальной стрелкой с указанием названия сообщения.

Рассмотрим для примера MSC-диаграмму на рис. 1.1. Здесь показан процесс обслуживания одиночного вызова сразу после процедуры инициализации (запуска) системы, которая выбрана как объект изучения в цикле лабораторных работ. Эта диаграмма иллюстрирует следующую последовательность событий:

1) Объект pCentral с помощью сигнала sReady сообщает во внешнюю среду (объект RTDS_Env), что система инициализирована и готова к работе.
2) Для системы в целом фиксируется исходное глобальное состояние Disconnected.
3) Абонент (представлен как объект RTDS_Env), посылает левому объекту pLocal сигнал sCall, в котором содержится запрос на вызов другого абонента с номером 2.
4) Левый объект pLocal с помощью сигнала sGetId запрашивает у центрального блока pCentral значение внутреннего идентификатора (PId) для телефона с номером 2 и получает в ответ это значение как параметр сигнала sId.
5) Левый объект pLocal использует полученное значение идентификатора PId для отправки правому объекту pLocal запроса на соединение (сигнал sCnxReq).
6) Правый объект pLocal, находящийся в состоянии Disconnected, посылает ответный сигнал sCnxConf для подтверждения запроса на соединение.
7) С помощью аналогичного сигнала левый объект pLocal сообщает объекту RTDS_Env (внешняя среда), что поступивший вызов успешно обработан и соединение установлено.
8) Для системы в целом фиксируется глобальное состояние Connected. На этой фазе функционирования системы осуществляется сеанс связи с передачей соответствующих данных между абонентами.
9) Вызывающий абонент, представленный объектом RTDS_Env, с помощью сигнала sHangUp дает отбой.

10) Левый объект pLocal отправляет правому объекту pLocal сигнал sDisReq с запросом разъединения.
11) Правый объект pLocal посылает в обратном направлении сигнал sDisConf, подтверждающий разъединение.
12) Левый объект pLocal с помощью сигнала sHangUpConf сообщает объекту RTDS_Env (абонент A во внешней среде), что разъединение подтверждено.
13) Процесс обслуживания поступившего вызова закончен, и система в целом опять возвращается в глобальное состояние Disconnected.

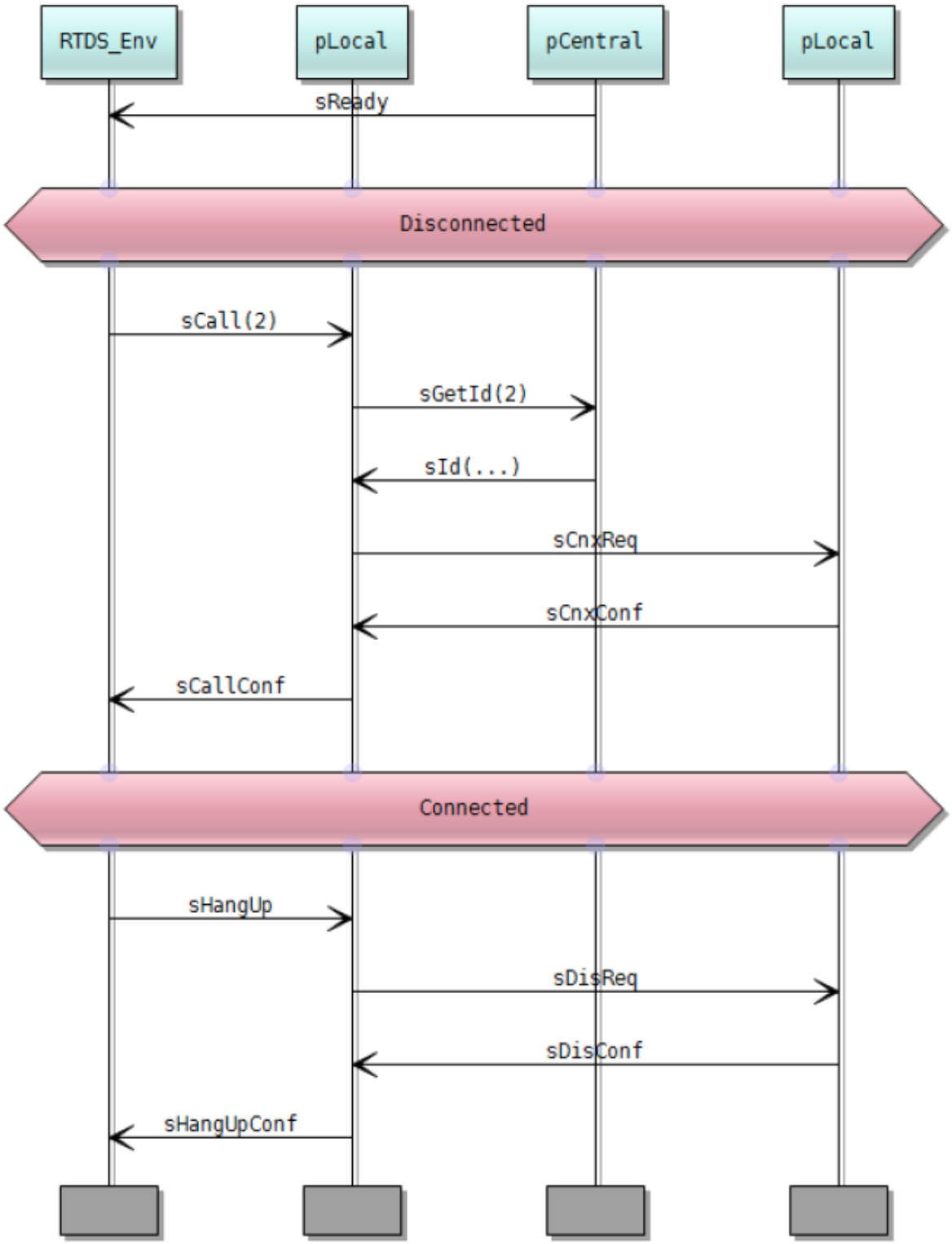

Рис. 1.1 – MSC-диаграмма функционирования системы связи

Подобные диаграммы довольно часто используются для графического представления требований, предъявляемых к алгоритмам функционирования телекоммуникационных систем, а также для удобного и наглядного описания таких алгоритмов.

1.3. Структурная модель системы

Для большей наглядности структурную модель системы разделим на 2 основных раздела: 1) объявление новых типов данных и используемых сообщений (раздел Declarations); 2) представление архитектуры системы на уровне отдельных процессов (раздел Architecture). При этом важно, что такой подход к моделированию системы согласуется с возможностями графических средств пакета PragmaDev Studio, который позволяет проводить декомпозицию (разложение) громоздкого и объемного описания сложной программной системы на единичные описания фрагментов системы.

В графической версии языка SDL раздел Declarations выглядит так, как показано на рис. 1.2. Здесь присутствует графический символ Text, который обозначается прямоугольником с загнутым верхним правым углом.

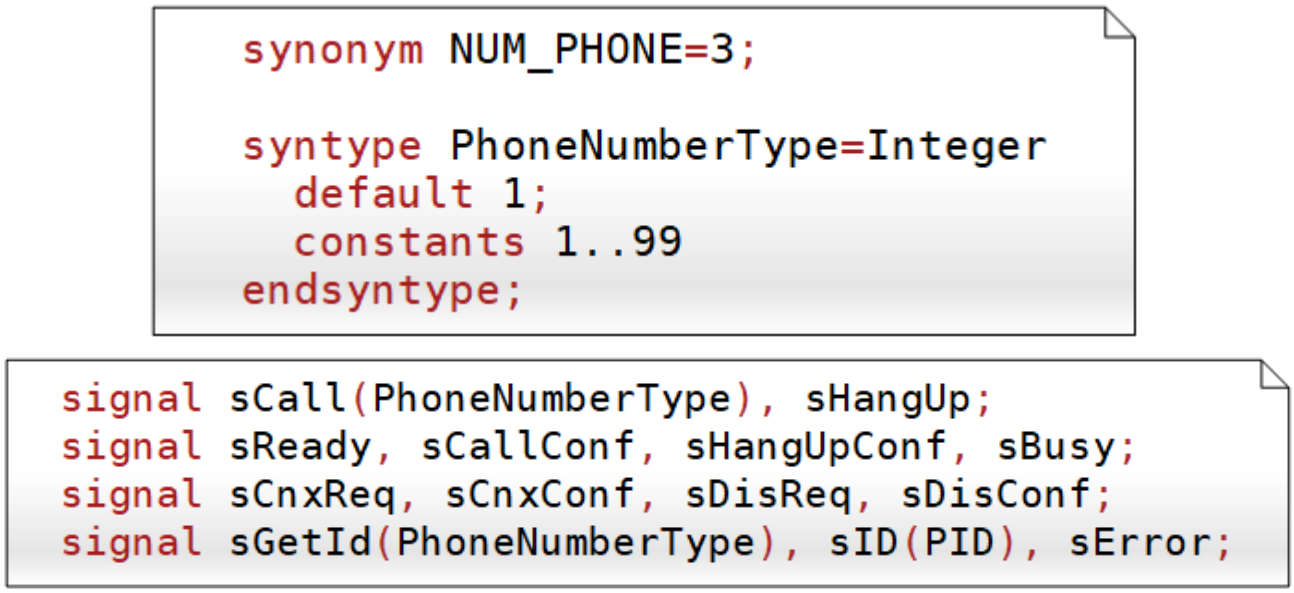

Рис. 1.2 – Объявление новых типов данных и используемых сообщений

Верхний символ Text содержит:

1) Объявление целочисленной константы NUM_PHONE, которая задает максимальное количество телефонов (в данном случае 3).
2) Объявление специального типа данных PhoneNumberType. Этот тип включает в себя целые числа из диапазона от 1 до NUM_PHONE, значение по умолчанию равно 1.
3) Объявление типа данных pLocalArrayType для организации одномерного

массива элементов, которые относятся к типу PID[1]. Выборка конкретного элемента из этого массива осуществляется по индексу, принадлежащему к типу данных PhoneNumberType.

Следующий символ Text содержит объявление сигналов, которые необходимы для функционирования рассматриваемой системы связи. Многие из этих сигналов уже использовались при построении MSC-диаграммы на рис. 1.1, здесь к ним добавлены вспомогательные сигналы sBusy и sError. Следует отметить, что сигналы sCall, sGetId и sId имеют по одному параметру. Смысл этого параметра заключается в том, чтобы указать тип элемента данных, который транспортируется при передаче сигнала.

С точки зрения архитектурного описания система имеет достаточно простую организацию, поэтому её структурную модель можно построить сразу на уровне процессов без использования таких структурных элементов, как блоки. Как показано на рис. 1.3, система состоит из двух процессов – pCentral и pLocal. Особенность обозначения процесса pLocal заключается в том, что после его названия в круглых скобках указано сначала число экземпляров этого процесса (0) в момент физического запуска системы, а затем (после запятой) – максимальное число экземпляров (NUM_PHONE), которые могут одновременно существовать (параллельно действовать) в системе. При обозначении процесса pCentral такие параметры не указаны, поэтому он (по умолчанию) всегда присутствует в единственном экземпляре.

Взаимодействие процессов pCentral и pLocal друг с другом и с окружающей средой осуществляется с помощью каналов. При обозначении каждого канала указывается (помимо его уникального имени в системе) список сигналов (в квадратных скобках), которые могут передаваться по конкретному каналу. В случае двухстороннего (дуплексного) канала такой список нужно обязательно указать для каждого направления передачи.

---

[1] Базовый тип в языке SDL, используемый для представления идентификаторов, которые имеют все процессы, возникающие при функционировании системы (уникальное значение идентификатора присваиваются каждому экземпляру процесса в момент его порождения).

Структуру системы, которая реализована на рис.1.3 можно представить в виде матрицы (таблица 1):

Таблица 1 – Матрица взаимодействия процессов

Взаимодействующие процессы	pCentral	pLocal	Env
pCentral		sID, sError	sReady
pLocal		sCnxReq, sCnxConf, sDisConf, sBusy	sCollConf, sHangUpConf, sBusy
Env	sGetId	sCall, sHangUp	

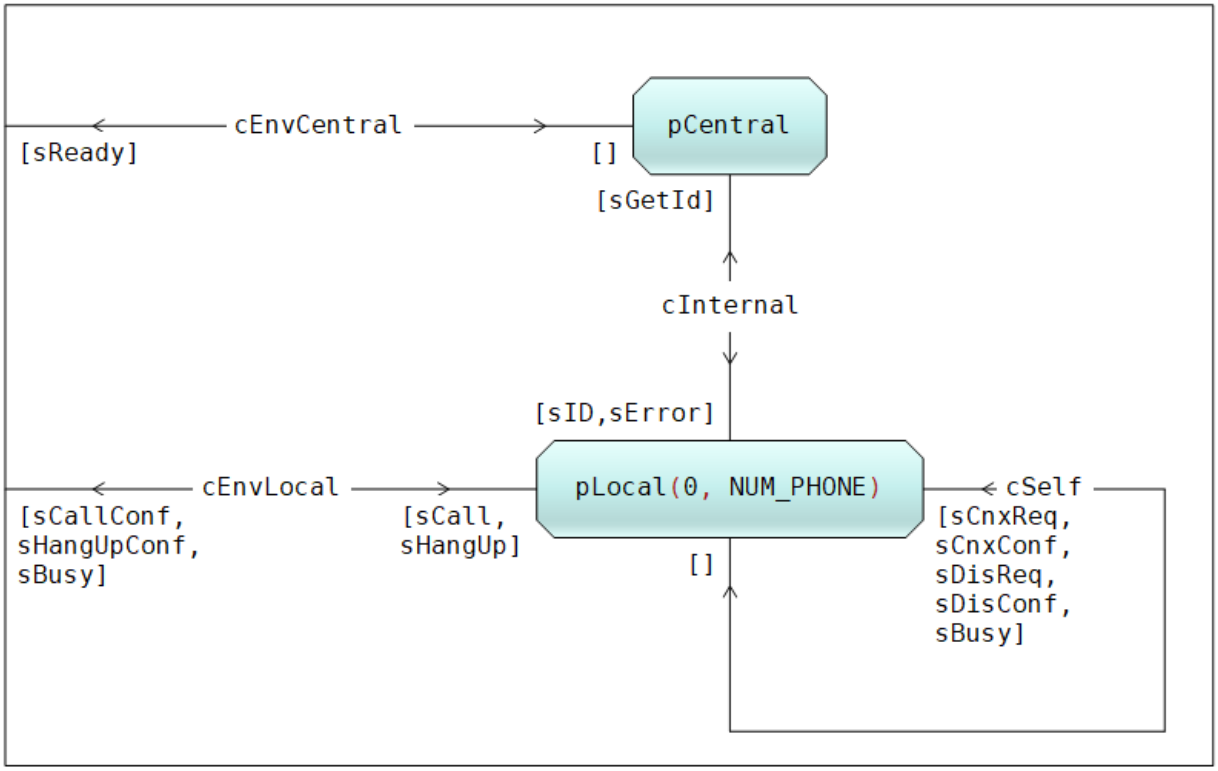

Рис. 1.3 – Архитектура системы

2. Выполнение лабораторной работы

2.1. Установка пакета PragmaDev Studio

При выполнении всего цикла лабораторных работ потребуется пакет PragmaDev Studio. Перед установкой этого пакета нужно зайти на сайт компании PragmaDev (http://www.pragmadev.com/downloads/) и скачать файл STUDIOV5-5-1.zip. Дальнейшие действия по установке пакета осуществляются

стандартным способом. Если собираетесь сразу же начать работу с пакетом, то нужно перезапустить свой компьютер.

В нижней части указанной страницы (раздел «PRAGMADEV STUDIO MANUALS») размещаются ссылки, которые позволяют скачать документацию по работе с пакетом PragmaDev Studio (все на англ. языке). После установки пакета на своем компьютере эти же документы можно найти в папке PragmaDev\doc.

2.2. Создание нового проекта

1) Запустите пакет PragmaDev Studio.
2) Чтобы создать новый проект, нажмите кнопку «New project» ( ) на панели инструментов Менеджера проекта (Project manager). После выполнения стандартных действий, связанных с выбором папки для сохранения файлов нового проекта и вводом его имени «Phone», значок этого проекта появится в окне Менеджера проекта (рис. 2.1).

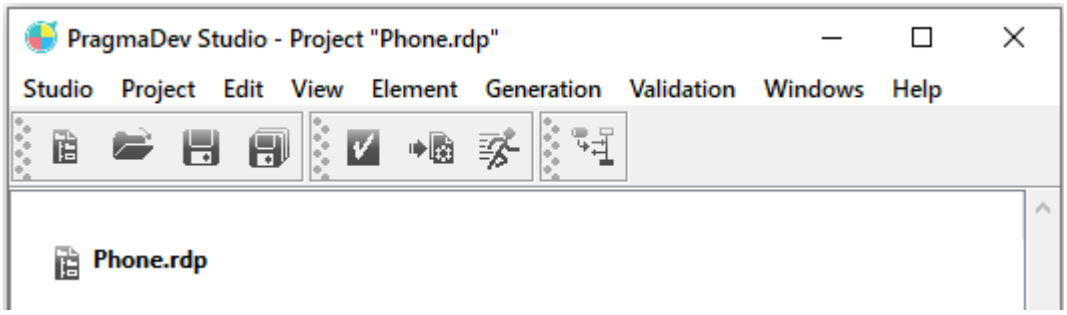

Рис. 2.1

2.3. Построение структурной модели системы

1) В окне Менеджера проекта щелкнуть правой кнопкой мыши на значке проекта «Phone» и в контекстном меню выбрать пункт «Add child element».
2) В левой части окна «Add child element» выделить категорию «Active architecture», затем отметить графический элемент «System». Диалоговое окно «Add child element» позволяет указать используемый язык моделирования, поэтому в поле «Language» следует выбрать SDL Z100 (рис. 2.2).

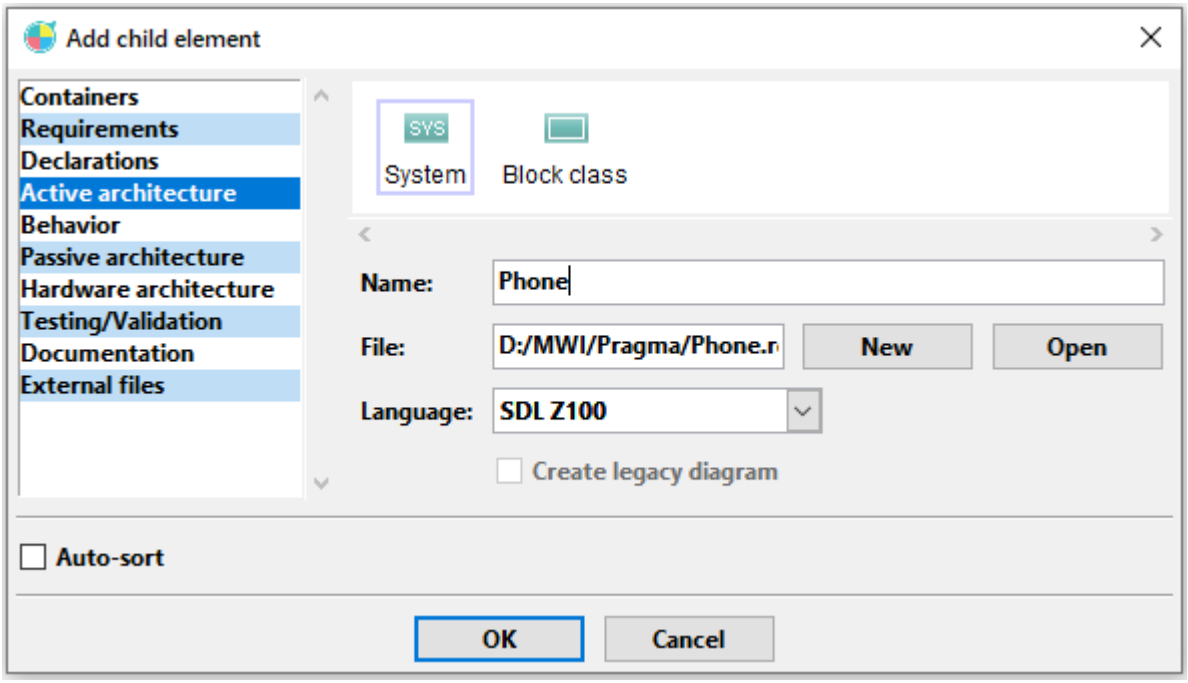

Рис. 2.2

3) Нажмите кнопку OK в нижней части окна «Add child element», и значок компонента проекта для архитектуры системы появится в окне Менеджера проекта. При двойном щелчке мышью на этом значке откроется окно редактора для работы с диаграммами на языке SDL. С помощью графических средств этого редактора предстоит построить модели (структурную и функциональную) системы связи, поведение которой ранее было описано в разделах 1.1 и 1.2, а также представлено в виде MSC-диаграммы (рис. 1.1).

4) Для новой диаграммы всегда открывается раздел (partition) с номером 1, и по умолчанию он получает название «Part.0». Это название отображается на панели справа от окна редактора при нажатой кнопке (Partitions). Для управления разделами диаграммы предусмотрена также специальная панель инструментов (рис. 2.3).

Рис. 2.3

5) Щелкнуть правой кнопкой мыши на названии «Part.0» и в контекстном меню выбрать пункт «Partition attributes». В открывшемся окне с таким же названием (рис. 2.4) указать новое имя раздела – Declarations.

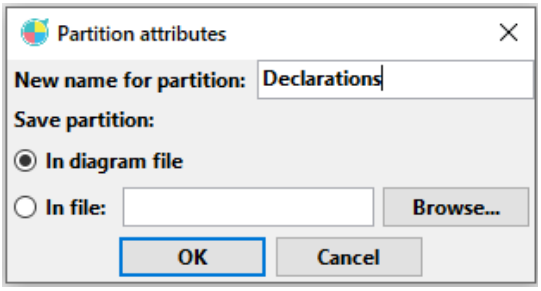

Рис. 2.4

6) Раздел Declarations заполнить графическими элементами, показанными на рис. 1.2.
7) С помощью кнопки «New partition after current one» на панели инструментов для управления разделами (рис. 2.3) создать новый раздел и присвоить ему имя Architecture.
8) Раздел Architecture заполнить графическими элементами, показанными на рис. 1.3. Чтобы нарисовать канал cSelf, который предназначен для обмена сообщениями между разными экземплярами процесса pLocal, держите нажатой клавишу Shift и щелкайте левой кнопкой мыши в тех точках, где линия канала должна менять свое направление.
9) Сохраните построенную диаграмму.
10) Чтобы проверить соответствие этой диаграммы правилам языка SDL, нужно раскрыть список команд для пункта Diagram в главном меню редактора и выбрать там команду «Check syntax/semantics».
11) Результаты проверки отображаются в окне Messages (рис. 2.5). На данном этапе этот результат является вполне адекватным: модель системы еще полностью не построена и функциональные диаграммы для процессов pLocal и pCentral пока отсутствуют. Важнее всего, что никаких других ошибок не обнаружено.

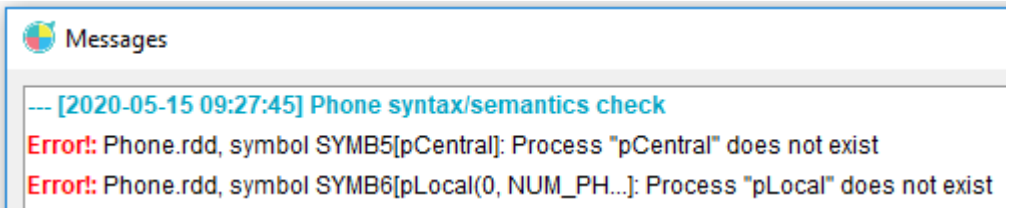

Рис. 2.5

3. Индивидуальное задание

Имеется система, состоящая из нескольких компонентов (блоки или процессы). Эти компоненты взаимодействуют друг с другом, а также с внешним окружением системы (**Env**). Общая картина взаимодействия описывается с помощью матрицы, в которой на пересечении i-й строки и j-го столбца указываются сигналы, передаваемые в направлении от i-го элемента системы и j-му элементу. По аналогии с рис.1.3 по заданной матрице построить структуру системы, задать каналы взаимодействия и указать передаваемые сигналы. Средствами языка SDL нужно построить структурную диаграмму взаимодействия элементов системы, реализовать эту диаграмму с помощью пакета PragmaDev Studio и провести ее синтаксическую проверку. Результаты выполнения задания оформить в виде отчета.

Варианты заданий

Вариант 1. Система P содержит блоки M1, M2 и M3.

	M1	**M2**	**M3**	**Env**
M1		S5	S3	S2
M2	S5; S6		S4	S1
M3		S7		S7
Env	S2	S1	S7	

Вариант 2. Система B содержит процессы K1, K2, K3.

	K1	**K2**	**K3**	**Env**
K1		S2	S4	S5
K2	S2; S3		S1	S3
K3	S6	S1		S7
Env	S5	S3		

Вариант 3. В системе L имеются блоки D1, D2 и процесс D3.

	D1	**D2**	**D3**	**Env**
D1		S1	S4	S2
D2	S7		S5	S3
D3	S4; S6	S5		S8
Env		S3		

Вариант 4. В системе M имеются процессы X1, X2 и блок X3.

	X1	**X2**	**X3**	**Env**
X1		S4	S2	S1
X2	S4		S5	S3
X3	S2; S6	S5		S7
Env				

Вариант 5. Система Z содержит блоки E1, E2, E3 и E4.

	E1	**E2**	**E3**	**E4**	**Env**
E1		S3	S2		S1
E2				S4	S5
E3	S6			S9	S7
E4		S4; S8			
Env	S1				

Вариант 6. Система E содержит процессы F1, F2, F3 и F4.

	F1	**F2**	**F3**	**F4**	**Env**
F1		S4; S5	S1		S2
F2				S3	

F3	S1; S6			S7; S9	S8
F4		S3			S7
Env	S2		S8		

Вариант 7. Система H содержит блоки J1, J2 и процессы J3, J4.

	J1	**J2**	**J3**	**J4**	**Env**
J1		S5	S3; S8		S2
J2	S5; S6			S4	S1
J3	S8			S7; S8	
J4					S4
Env	S2	S1			

Вариант 8. Система W содержит процессы V1, V2, V3 и блок V4.

	V1	**V2**	**V3**	**V4**	**Env**
V1		S2	S4		S5
V2	S2; S3			S1	S3
V3	S6			S4	S7
V4		S1; S8			S9
Env	S5	S3		S9	

Вариант 9. Система G содержит блоки B1, B2, B3 и процесс P1.

	B1	**B2**	**B3**	**P1**	**Env**
B1			S1; S8		
B2	S4; S5			S3	S6
B3	S8			S7	
P1		S9	S9		
Env	S2	S6			

Вариант 10. Система Q содержит процессы P1, P2 и блоки B1, B2.

	P1	**P2**	**B1**	**B2**	**Env**
P1		S2	S3		S3
P2	S2; S4			S1; S8	
B1	S5			S7	S6
B2		S8			
Env	S3				

4. Требования к отчету по лабораторной работе

1) В отчете кратко описать процесс построения структурной модели телекоммуникационной системы согласно рекомендациям из раздела 2 методических указаний.

2) Привести решение индивидуального задания из раздела 3.

3) К отчету приложить архивные файлы с проектами, которые были получены при работе с пакетом PragmaDev Studio.