ОБЪЕКТНАЯ МОДЕЛЬ РАСПРЕДЕЛЁННОЙ КОМПЬЮТЕРНОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Лаздынь С.В., Землянская С.Ю.
Донецкий национальный технический университет, г. Донецк
кафедра автоматизированных систем управления
E-mail: slazd@ukr.net, zsaa@ya.ru

Abstract
Lazdyn S.V., Zemlianskaya S.Y. The object model of the distributed computer information system. *It is proposed to use the object-oriented approach to the model construction of the distributed computer information system (DCIS) at the stage of their designing or reconfiguration. The approach proposed consists in the DCIS components object models development and DCIS dynamic model construction*

Общая постановка проблемы проектирования РКИС

В условиях современного общества конкурентоспособность предприятия или организации определяется, в первую очередь, их возможностями по доступу, хранению, качественной и быстрой обработке информации, поэтому огромную важность приобретает грамотное использование передовых достижений в сфере информационных технологий. Любое предприятие в настоящее время имеет возможность создать распределенную информационную систему с выходом в вычислительные сети Украины и в международную сеть Internet и использовать её для организации информационного обмена о результатах деятельности, рекламирования достижений производственной и исследовательской деятельности и, самое главное, – для оперативного принятия управляющих решений при условии большой территориальной распределённости.

Потребность в информации для целей планирования и управления, совершенствования профессиональной деятельности предприятий (организаций), а также бурное развитие информационных процессов требует проектирования и внедрения распределенных компьютерных информационных систем (РКИС).

Наиболее сложно решаемыми являются задачи синтеза оптимальной структуры и построения адекватной модели РКИС, так как моделирование и проектирование РКИС относится к классу трудноформализуемых, многофакторных задач [1], при решении которых приходится решать задачи структурирования целей, разрабатывать аналитические модели, оперировать со статистическими данными, а также применять подходы описания параметров задач проектирования на качественном уровне. Таким образом, задачи моделирования и проектирования оптимальных РКИС относятся к числу сложных научно-исследовательских задач. В рамках существующих исследований и решений этих задач существует ряд нерешённых проблем.

С одной стороны, проблемы связаны с проектированием оптимальной организационной структуры РКИС, которая должна удовлетворять критерию масштабируемости как ключевому требованию с точки зрения экономии вложений. Это гарантирует, что не придется перестраивать систему по мере роста объема обрабатываемой информации и одновременно работающих пользователей. РКИС должна удовлетворять критерию расширяемости, сущность которого состоит в возможности наращивания; функциональных возможностей системы, не выходя за рамки принятой изначально концепции развития и технологической базы, в соответствии со специфическими потребностями пользователей.

С другой стороны – проблемы, связанные с удовлетворением критериев эффективности РКИС, таких как время доставки информации, достоверность передачи информации, надежность, и стоимость. Учет многокритериальности задачи проектирования РКИС можно будет свести к формированию некоторых ограничений в виде комплексного критерия эффективности [2].

Если проблемы, связанные с проектированием организационной структуры РКИС решены, тогда наличие выделенных уровней в технологической структуре делает возможным варьирование аппаратных и программных средств для реализации структурных составляющих информационно-технологической структуры компьютерной распределённой информационной системы: выбор операционных систем, СУБД, интерфейсов пользователей, серверов, рабочих станций и сетевого оборудования.

Целесообразно выбирать аппаратно-программное обеспечение, опираясь на требования к производительности конкретной информационной системы. Кроме того, необходимо реализовать наиболее оптимальную (двух- или трёхуровневую) клиент-серверную архитектуру, определиться с размещением приложений, используемых в системе.

Таким образом, задача определения эффективных параметров РКИС, необходимых для достижения максимальной производительности системы с минимальными финансовыми затратами требует проведения дальнейших исследований. Указанную задачу можно решить путём моделирования работы РКИС [3].

Краткий анализ проведенных исследований по моделированию РКИС.

Построение модели распределённой компьютерной информационной системы – сложная задача, решением которой занимаются с момента возникновения простейших распределённых систем обработки информации. В настоящее время в качестве РКИС выступают информационные системы крупных территориально распределённых предприятий (организаций) – корпоративные информационные системы. Моделированию корпоративных информационных систем (КИС) посвящён ряд работ, например [3-6]. В основном, в предлагаемых работах КИС моделируется либо как корпоративная компьютерная сеть, либо как распределённая база данных, либо как совокупность информационных процессов в корпоративной информационной системе. При таких подходах невозможно комплексно оценить влияние программных и технических компонентов на производительность системы.

В основу построения модели РКИС как распределённой базы данных положены: объектно-реляционные базы данных, дискреционная модель разграничения доступа к данным, отношения подчиненности и вложенности между объектами и репликация данных [4]. Этот подход не позволяет оценить работоспособность (производительность) технических компонентов РКИС.

В работе [5] РКИС рассматривается как совокупность информационных процессов в специализированной (экономической) информационной системе масштабов предприятия. Недостатком такого подхода является то, что невозможно исследовать распределённую информационную систему комплексно, не вдаваясь в тонкости предметной области, в которой используется РКИС, моделируются только информационные процессы, протекающие в системе.

В статье [6] предлагается объектная модель приложений в трёхзвенной архитектуре клиент-сервер. При создании объектной модели распределённых корпоративных информационных систем объекты приложения (приложений) распределяются по соответствующим пакетам. Пакеты – это основные семантически связанные группы объектов. Вводится такой объект как классификатор (справочник). Этот объект имеет свою реализацию и в интерфейсе пользователя, и в сервере приложений, и в сервере базы данных. Классификатор (справочник) – является списком возможных значений того или иного

реквизита некоторого объекта приложения. Клиент вызывает функцию контроллера уровня интерфейса, где передает номер открываемого классификатора. Полученная информация обрабатывается контроллером сервера приложений. После расшифровки информации контроллер ищет указанный классификатор в своем списке открытых классификаторов, с помощью соответствующего метода, реализованного в объекте контроллер. Недостатком этой модели является то, что основное внимание уделяется моделированию программной составляющей КИС, без учёта распределения данных в системе и используемых технических компонентов.

В предложенном ранее авторами подходе к построению модели компьютерной информационной системы [3] в результате структурного анализа выделяются типовые компоненты компьютерной информационной системы (КИС), для которых разрабатываются объектные модели в виде классов. КИС как объект моделирования представляет собой совокупность соединенных друг с другом и взаимодействующих типовых компонентов. В данной статье предлагается развитие указанного подхода применительно к РКИС.

Выбор типовых компонентов для модели РКИС

Проведенный структурный анализ показал, что в составе РКИС можно выделить следующие группы типовых компонентов, параметры которых влияют на производительность системы: компоненты технического обеспечения (компьютерное и сетевое оборудование), программное обеспечение, данные.

Компоненты технического обеспечения:

- Рабочая станция – место запуска пользовательских приложений и формирования запросов;
- Сервер приложений – сервер, хранящий и выполняющий сложные приложения, производительность сервера оценивается в условных единицах –JOP – количество тестовых операций, выполняемых в секунду [7];
- Сервер БД – используется для обработки и чтения/записи информации, производительность такого сервера измеряется в условных единицах TPC – количество транзакций в секунду;
- Узел компьютерной системы – совокупность серверов и /или рабочих станций и сетевых устройств, территориально расположенных в одном месте;
- Сетевое устройство - коммутатор, маршрутизатор;
- Канал передачи - физическая среда передачи данных, может представлять собой коммутируемый канал связи, выделенную телефонную линию, оптоволоконный кабель, беспроводный канал связи и др.

Компоненты программного обеспечения:

- Приложение – пользовательское приложение, запускаемое с определённой периодичностью на рабочей станции, характеризуется набором формируемых запросов. Если приложение находится на сервере приложений, то запускается оно по запросу с рабочей станции;
- Запрос (на модификацию, чтение, обновление данных). Запрос формируется к конкретному набору данных; место обработки запроса определяется распределением набора данных между серверами. Каждый запрос может инициировать последовательность других запросов (подзапросов).

Данные:

- Набор данных (таблица) – логический набор данных, который имеет название и характеризуется набором серверов, на которых он физически размещается.

Рассмотрим классы, описывающие выделенные компоненты. На рисунке 1 приведена UML-диаграмма иерархии классов объектной модели РКИС. Все компоненты, составляющие

конкретную РКИС, относятся к этим классам и инкапсулируют свойства конкретного типа серверов, рабочих станций, операционных систем и сетевого оборудования.

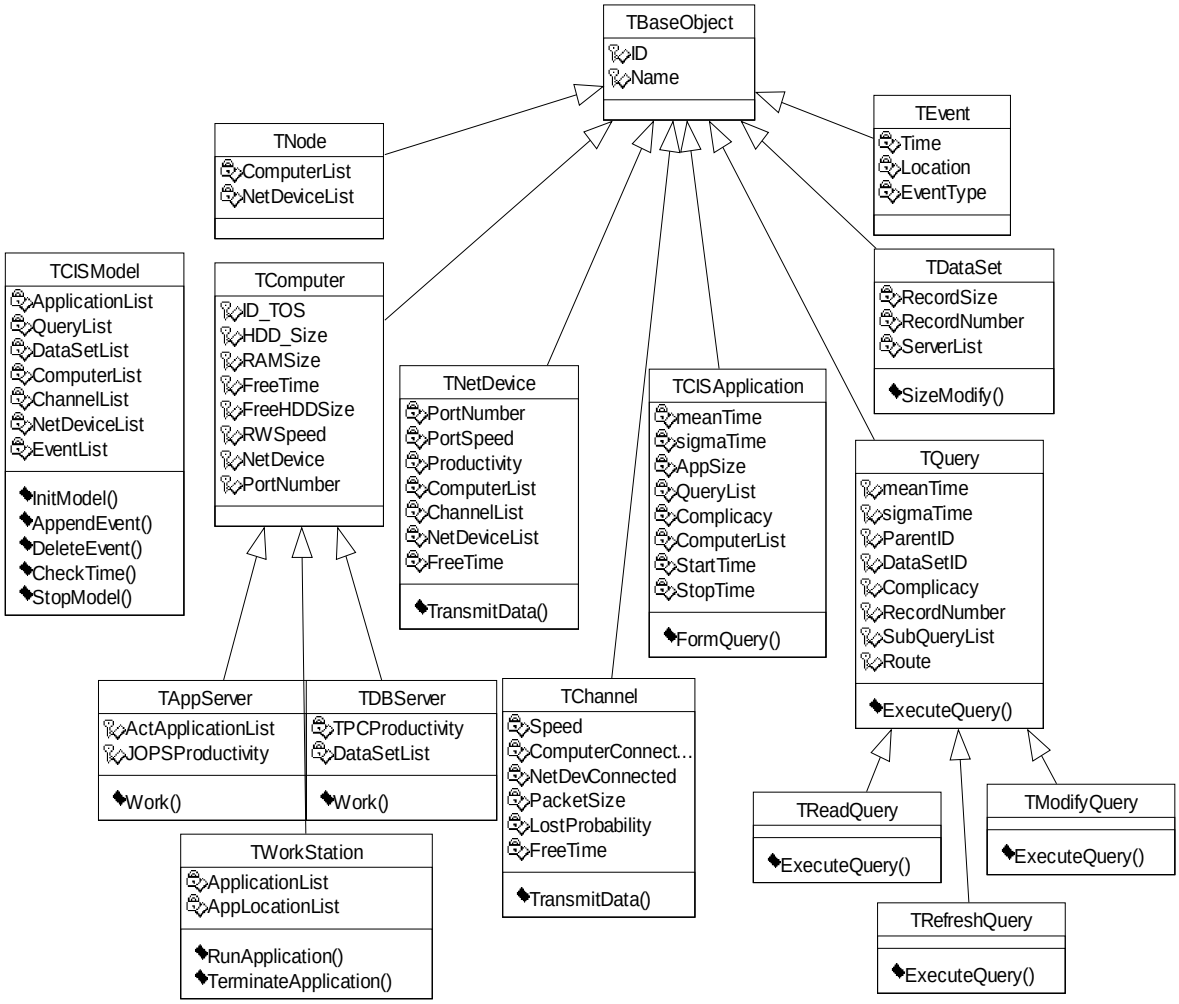

Рисунок 1 - UML-диаграмма иерархии классов объектной модели РКИС

Объектные модели типовых компонентов РКИС

Для унификации объектов модели введен абстрактный класс **TBaseObject**, включающий свойства, общие для всех объектов модели: идентификатор объекта (ID) и его наименование (Name). Все остальные классы модели являются производными от класса **TBaseObject** и наследуют эти свойства.

Объектные модели типовых компонентов технического обеспечения.

Модель объекта «Компьютер»

Для описания компьютерного оборудования введен базовый класс **TComputer**, имеющий общие для всех компьютеров свойства. Класс является производным от класса **TBaseObject**. Объект этого класса обобщённо описывает такие компоненты как серверы и рабочие станции и содержит следующие свойства: идентификатор – ID, название – Name, идентификатор операционной системы – ID_TOS, объем дискового накопителя HDDSize, объём свободного дискового простанства – FreeHDDSize, сетевое устройство – NetDevice и номер порта сетевого устройства, к котрому подключен комп'ютер – PortNumber, объем ОП – RamSize, объем свободной оперативной памяти (ОП) – FreeRamSize, время освобождения (время окончания выполнения запроса) – FreeTime.

Модель объекта «Рабочая станция»

Типовой компонент рабочая станция в распределённой информационной системе является источником запросов пользователя, связанных с получением или обновлением какой-либо информации из распределённой базы данных. Запросы являются продуктом работы определённых приложений, запускаемых на рабочей станции. Для описания объекта «рабочая станция» разработан класс **TworkStation**, являющийся производным от класса **TComputer** и наследующий его свойства. Кроме этого **TworkStation** характеризуется набором приложений, что описывается свойством «список приложений» - ApplicationList. Локализация запускаемых приложений описывается свойством AppLocatioList, в котором каждому приложению из списка ApplicationList соответствует сервер, на котором соответствующее приложение размещается. Для запуска и завершения приложения на рабочей станции используются методы RunApplication() и TerminateApplication(). Время запуска приложения – случайная величина. Экспериментальным путём было установлено, что эта величина распределена по нормальному закону, продолжительность работы приложения – тоже случайная величина с нормальным законом распределения.

Модель объекта «Сервер приложений»

Типовой компонент сервер приложений содержит приложения для обработки сложных запросов пользователей, требующие значительных затрат вычислительных ресурсов. Для описания объекта «сервер приложений» разработан класс **TAppServer**, являющийся производным от класса **TComputer** и наследующий его свойства. Кроме этого, **TAppServer** характеризуется такими свойствами как производительность – JOPSProductivity, список приложений - ApplicationList, список запущенных приложений – ActApplicationList. Выполнение на сервере вызванного приложения осуществляется при помощи метода Work(). При вызове этого метода объем свободной оперативной памяти сервера уменьшается на размер приложения:

$$FreeRAMSize=FreeRAMSize-AppSize, \quad (1)$$

где *FreeRAMSize* – объём свободной ОП сервера приложений, *AppSize* – размер приложения.

Сервер переходит в занятое состояние на время, которое определяется сложностью приложения и производительностью сервера. Время освобождения (*FreeTime*) сервера можно вычислить по формуле:

$$FreeTime=FreeTime+JOPSProductivity/Complicacy, \quad (2)$$

где *JOPSProductivity* – производительность сервера, Complicacy – сложность запроса

Модель объекта «Сервер баз данных»

Типовой компонент сервер баз данных используется для обработки запросов пользователя и выполнения операций чтения/записи. Для описания объекта «сервер баз данных» разработан класс **TDBServer** Кроме общих свойств, наследуемых от базового класса **TComputer**, объект класса **TDBServer** содержит такие свойства: производительность (TPC) – TPCProductivity, список расположенных на сервере наборов данных - DataSetList. Выполнение запросов пользователя осуществляется при помощи метода Work(). Этот метод моделирует занятость сервера на время, определяемое сложностью запроса – Complicacy и производительностью сервера – TPCProductivity. Время освобождения (FreeTime) сервера вычисляется по формуле:

$$FreeTime=FreeTime+TPCProductivity/Complicacy, \quad (3)$$

где *TPCProductivity* – производительность сервера, *Complicacy* – сложность запроса

Модель объекта «Сетевое устройство»

Типовой компонент «сетевое устройство» применяется для структуризации компьютерной сети РКИС, к нему могут подключаться серверы, рабочие станции и каналы для соединения между собой intranet-сетей. Для описания объекта «сетевое устройство»

разработан класс **TNetDevice**. Этот класс является производным от базового класса **TBaseObject** и наследует от него свойства идентификатор – ID и наименование – Name. Кроме этого, объект класса **TNetDevice** содержит такие свойства: количество портов – PortNumber; вектор, хранящий скорости портов - PortSpeed, производительность (количество пакетов, передаваемых сетевым устройством в секунду) - Productivity, списки подключенных компьютеров (серверов, рабочих станций) – ComputerList, каналов – ChannelList и других сетевых устройств – NetDeviceList, время окончания передачи очередной порции данных - FreeTime. Для моделирования передачи данных через сетевое устройство разработан метод TransmitData(). Этот метод моделирует занятость сетевого устройства на время, соответствующее объёму передаваемых данных и скорости передачи порта (i), через который эти данные передаются. Время освобождения сетевого устройства определяется по формуле:

$$FreeTime=FreeTime+PortSpeed_i/(RecordNumber*RecordSize), \quad (4)$$

где $PortSpeed_i$ – скорость передачи порта, через который происходит передача данных, *RecordNumber* – количество передаваемых записей, *RecordSize* – размер одной записи.

Модель объекта «Узел РКИС»

Типовой компонент «Узел РКИС» объединяет территориально размещёные в одном месте серверы, рабочие станции и сетевые устройства. Для описания объекта «Узел РКИС» разработан класс **TNode**. Этот класс является производным от базового класса **TBaseObject** и наследует от него свойства идентификатор – ID и наименование – Name. Кроме этого, объект класса **TNode** содержит такие свойства: список компьютеров – ComputerList и список сетевых устройств – NetDeviceList.

Модель объекта «Канал передачи данных»

Типовой компонент «канал передачи данных» применяется для объединения отдельных intranet-сетей в общую информационную сеть. Для описания объекта «канал передачи данных» разработан класс **TChannel**. Этот класс является производным от базового класса **TBaseObject** и наследует от него свойства идентификатор – ID и наименование – Name. Кроме этого, объект класса **TChannel** содержит такие свойства: пропускная способность(кБит/с) – Speed, перечень подсоединённых компьютерных узлов - ComputerConnected, перечень подсоединённых сетевых устройств – NetDeviceConnected, размер передаваемого пакета данных – PacketSize, вероятность потери пакета – LostProbability, время освобождения – FreeTime. Для моделирования передачи данных по каналу разработан метод TransmitData(). Этот метод моделирует занятость канала, причём время занятости канала зависит от объёма передаваемых данных, пропускной способности канала, размера пакета данных и вероятности потери пакета. Время освобождения канала вычисляется по формуле:

$$FreeTime=FreeTime+(RecordNumber*RecordSize+LostPackets*PacketSize)/Speed, \quad (5)$$

где *LostPacket* – количество потерянных пакетов, вычисляемое по формуле:

$$LostPacket=\begin{cases} LostPacket+1, & Q>Lost\,\mathrm{Pr}obability \\ Lost\,\mathrm{Pr}obability, & Q\le Lost\,\mathrm{Pr}obability \end{cases}, \quad (6)$$

где Q – случайное число, равномерно распределённое в диапазоне от 0 до 1

Объектные модели типовых компонентов программного обеспечения

Модель объекта «Приложение»

Типовой компонент «приложение» является источником запросов рабочей станции, на которой оно запущено. Для описания объекта «приложение» разработан класс **TCISApplication**. Этот класс является производным от базового класса **TBaseObject** и наследует от него свойства идентификатор – ID и наименование – Name. Кроме этого, объект класса **TCISApplication** содержит такие свойства: математическое ожидание – meanTime и

среднеквадратичное отклонение времени запуска – sigmaTime, размер приложения – AppSize, список запросов, формируемых приложением – QueryList, сложность приложения (количество JOPS) – Complicacy, время запуска – StartTime, время окончания – StopTime, список компьютеров (серверов и рабочих станций), на которых может размещаться приложение - ComputerList. Для моделирования генерирования приложением запросов разработан метод FormQuery(). Для каждого запроса из списка запросов приложения формируется массив моментов времени запуска в соответствии со средним значением интервалов времени запуска запроса и среднеквадратичным отклонением, учитывая, что интервалы времени запуска запроса – случайная величина, распределённая по нормальному закону. Результатом работы метода является массив запросов с определённым временем запуска: $\{Query_i, time_i\}$.

Модель объекта «Запрос»

Типовой компонент «запрос» является основным средством передачи информации в модели распределённой компьютерной информационной системы. Для описания объекта «запрос» разработан базовый класс **TQuery**. Этот класс является производным от базового класса **TBaseObject** и наследует от него свойства идентификатор – ID и наименование – Name. Кроме этого, объект класса **TQuery** содержит такие свойства: математическое ожидание интервалов времени запуска– meanTime и среднеквадратичное отклонение интервалов времени запуска – sigmaTime, идентификатор приложения, породившего запрос – ParentId, идентификатор узла-источника запроса - SourceId, идентификатор набора данных – DataSetId, сложность запроса (в TPC или JOPS) – Complicacy, маршрут запроса (список каналов передачи) – Route, количество считываемых (модифицируемых) записей набора данных – RecordNumber. Для определения маршрута прохождения запроса разработан метод MakeRoute(). Этот метод формирует список каналов, по которому будет передаваться запрос к ближайшему серверу, содержащему требуемый набор данных. Общее время передачи запроса к серверу можно вычислить по формуле:

$$Time = \sum_{i=1}^{CN} ChannelTime_i + \sum_{i=1}^{NDN} NetDeviceTime_i \qquad (7)$$

где CN – количество каналов, по которым передается запрос; $ChannelTime_i$ – время передачи по i-ому каналу; NDN – количество сетевых устройств, через которые проходит запрос; $NetDeviceTime_i$ – время передачи запроса через i-ое сетевое устройство.

Можно выделить три основных вида запросов: запрос на выборку информации (чтение данных), запрос на модификацию (изменение данных) и запрос на обновление данных. Для описания этих видов запросов разработаны классы, производные от класса **TQuery: TReadQuery, TModifyQuery** и **TRefreshQuery.**

Модель объекта «Запрос на чтение»

Для описания объекта «запрос на чтение» разработан класс **TReadQuery**. Этот класс является производным от базового класса **TQuery** и наследует от него свойства идентификатор – ID, наименование – Name, математическое ожидание интервалов времени запуска – meanTime и среднеквадратичное отклонение интервалов времени запуска – sigmaTime, идентификатор приложения-родителя – ParentId, идентификатор узла-источника запроса - SourceId, идентификатор набора данных – DatasetId, сложность запроса (в TPC) – Complicacy, маршрут запроса (список каналов передачи) – Route. Для моделирования выполнения запроса разработан метод ExecuteQuery(). Результатом выполнения метода являются данные, передаваемые в качестве ответа на компьютер, пославший запрос. Объем данных вычисляется по формуле:

$$DataSize=RecordNumber*RecordSize, \qquad (8)$$

где $RecordNumber$ – количество запрашиваемых записей, $RecordSize$ – объём одной записи.

Модель объекта «Запрос на изменение»

Для описания объекта «запрос на чтение» разработан класс **TModifyQuery**. Этот класс является производным от базового класса **TQuery** и наследует от него свойства: идентификатор – ID, наименование – Name, математическое ожидание интервалов времени запуска – meanTime и среднеквадратичное отклонение интервалов времени запуска – sigmaTime, идентификатор приложения-родителя – ParentId, идентификатор узла-источника запроса - SourceId, идентификатор набора данных – DatasetId, сложность запроса (в TPC) – Complicacy, маршрут запроса (список каналов передачи) – Route. Для моделирования выполнения запроса разработан метод ExecuteQuery(). В результате выполнения этого метода вызывается метод изменения размера набора данных, к которому предназначен этот запрос.

Модель объекта «Запрос на обновление»

Для описания объекта «запрос на обновление» разработан класс **TRefreshQuery**. Этот класс является производным от базового класса **TQuery** и наследует от него свойства: идентификатор – ID, наименование – Name, математическое ожидание интервалов времени запуска – meanTime и среднеквадратичное отклонение интервалов времени запуска – sigmaTime, идентификатор приложения-родителя – ParentId, идентификатор узла-источника запроса - SourceId, идентификатор набора данных – DatasetId, сложность запроса (в TPC) – Complicacy, маршрут запроса (список каналов передачи) – Route. Для моделирования выполнения запроса разработан метод ExecuteQuery(). В процессе работы этого метода вызываются методы Work() для серверов, на которых находятся копии обновляемого набора данных.

Объектная модель типового компонента «Данные».

Для описания объекта «набор данных» разработан класс **TDataSet**. Этот класс является производным от базового класса **TBaseObject** и наследует от него свойства: идентификатор – ID и наименование – Name. Для описания распределения набора данных между узлами РКИС добавлено свойство «список компьютерных узлов, содержащих фрагменты набора» - ServerList. Набор данных характеризуется количеством находящихся в нём записей – RecordNumber и размером одной записи - RecordSize. Методом этого класса является изменение размера - SizeModify(), входными данными для которого выступает количество модифицируемых записей, передаваемых запросом, а результатом – изменённый размер набора данных. Объем набора данных увеличивается (уменьшается) на количество записей, содержащееся в запросе:

$$DataSet.RecordNumber = DataSet.RecordNumber \pm Query.\ RecordNumber \quad (9)$$

где *DataSet.RecordNumber* – количество записей в наборе данных DataSet; *Query.RecordNumber* – количество добавляемых (удаляемых) записей, содержащееся в запросе.

Построение объектной модели РКИС и организация взаимодействия объектов

Модель РКИС представляет собой совокупность взаимодействующих объектов разработанных классов. В модели реализован алгоритм моделирования, управляемый событиями. Это наиболее распространенная реализация последовательного моделирования, при которой в качестве следующего значения часов модельного времени выбирается минимальное время события из списка событий [8]. Каждый объект работает по собственному алгоритму до тех пор, пока он не изменяет свое локальное время при выполнении некоторого события.

Контроль состояния объектов системы производится при помощи объектов класса «Событие». Эти объекты создаются в процессе моделирования, когда один из объектов системы меняет своё состояние, то есть изменяет своё модельное время.

Для описания объектов «событие» разработан класс **TEvent**, который является производным от класса **TBaseObject** и содержит информацию о событии и месте наступления этого события в системе, то есть включает такие свойства: время наступления события – Time, наименование события – Name, указатель на объект, инициировавший событие – Location, тип события – EventType. Тип события определяется местом и характером его возникновения (1 – появление запроса, 2 – освобождение сетевого устройства, 3 – освобождение канала, 4 – окончание выполнения запроса сервером, 5 – получение ответа на запрос рабочей станцией).

Для описания структуры РКИС и управления процессом моделирования в модель добавлен объект класса, который содержит методы, обеспечивающие синхронизацию работы всех объектов моделируемой системы, – **TCISModel**. Этот объект содержит списки всех объектов системы как логической, так и физической архитектуры. Класс **TCISModel** имеет такие свойства: список приложений – ApplicationList, список (очередь) запросов – QueryList, список наборов данных – DatasetList, список компьютерных узлов – ComputerList, список сетевых устройств – NetDeviceList, список каналов связи – ChannelList, список событий системы - EventList. Класс TCISModel также содержит следующие методы: инициализация модели РКИС – InitModel(); метод обработки событий; метод добавления и удаления события в список – AppendEvent(), DeleteEvent(); поиск события с минимальным временем наступления – CheckTime(); завершение моделирования – StopModel().

В начале моделирования формируется модель РКИС со структурой, соответствующей конкретной РКИС, функционирование которой необходимо промоделировать. Информация о структуре РКИС, обо всех компонентах логической и физической архитектуры хранится в базе данных. Из базы данных считывается информация о серверах, каналах связи и об остальных компонентах модели.

Работа модели начинается с инициализации очередей запросов. Для каждого запроса с учетом частоты его возникновения создается очередь вызовов в течение всего времени моделирования. В ходе выполнения запросов моделируются процессы считывания и записи данных в наборы данных. На рисунке 2 приведена диаграмма кооперации объектов при выполнении запроса.

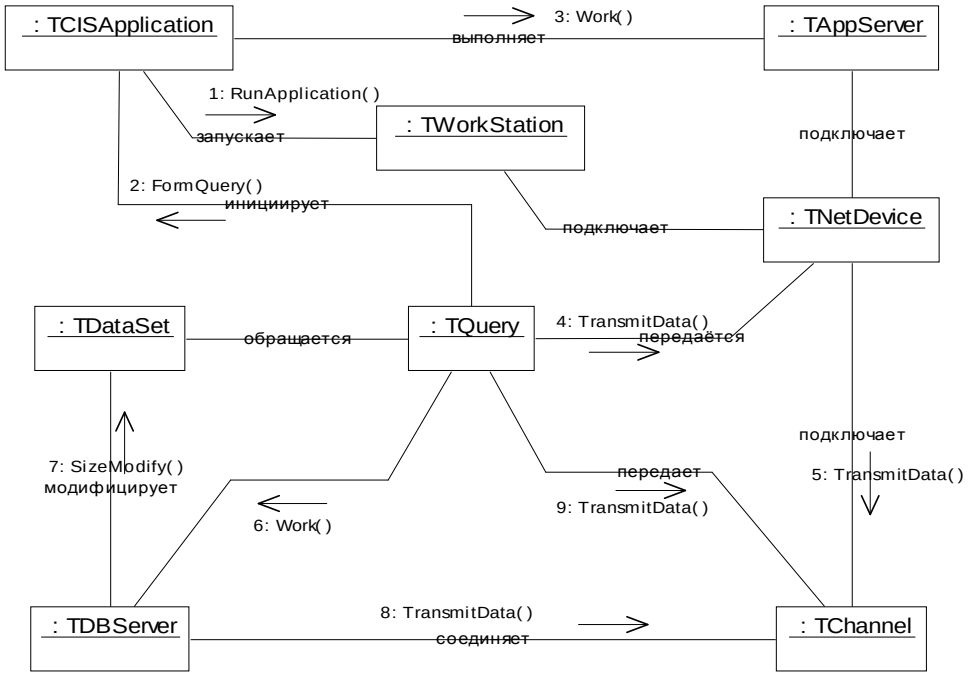

Рисунок 2 – Диаграмма кооперации объектов модели при выполнении запроса.

Когда наступает необходимость в обработке очередного события, происходит обращение к менеджеру объектов (TCISModel)

TCISModel управляет передачей данных, производит поиск маршрута между узлами для передачи данных с учетом текущей загруженности каналов связи и сетевых устройств. Задачей менеджера объектов является синхронизация модельного времени всех объектов, чтобы не допустить нарушения целостности имитации [8].

Менеджер объектов просматривает список событий, выбирает событие с минимальным временем наступления, удаляет его из очереди и инициирует запуск метода обработки соответствующего события. В результате работы метода изменяется модельное время объекта, событие которого обрабатывалось менеджером, и формируется новое событие. Таким образом, исключается возможность нарушения хронологического порядка событий при моделировании работы системы.

Заключение

Для построения модели распределённой компьютерной информационной системы предложено использовать объектно-ориентированный подход. Проведен структурный анализ РКИС и выделены её типовые компоненты технического и программного обеспечения. Разработаны объектные модели типовых компонентов РКИС в виде классов, описаны их основные свойства и методы. Организовано взаимодействие объектов модели в динамике, для чего предложен метод синхронизации объектов в модели с помощью дополнительного управляющего класса менеджера объектов. Разработанная модель может быть использована для оценки параметров и определения структуры как для проектируемых распределённых компьютерных информационных систем, так и для модернизируемых.

Литература

1. Ferscha A. Parallel and Distributed Simulation of Discrete Event Systems. Parallel and Distributed Computing Handbook, McGraw-Hill. 1996. PP. 1003-1041.

2. Лаздынь С.В., Землянская С.Ю. Многокритериальная оптимизация корпоративных информационных систем // Наукові праці ДонНТУ. Серія “Обчислювальна техніка та автоматизація” - Випуск 12(118). - Донецьк: ДонНТУ, 2007. - С. 74 - 82.

3. Лаздынь С.В., Землянская С.Ю. Разработка динамической модели компьютерной информационной системы // Наукові праці ДонНТУ. Серія “Обчислювальна техніка та автоматизація” - Випуск 74. - Донецьк: ТОВ Лебідь, 2004. - С. 152 - 159.

4. Агибалов Г.П., Скутин А.А. Математическая модель и технология разработки безопасных корпоративных информационных систем. Томский государственный университет/ URL: http://zhurnal.ape.relarn.ru/articles/2001/151.pdf

5. Гламаздин Е.С., Новиков Д.А., Цветков А.В. «Управление корпоративными программами: информационные системы и математические модели». – М: Компания Спутник+, 2003. - 159 с.

6. Войтиков К.Ю. Змеев О.А., Моисеев А.Н. Реализация классификаторов на сервере приложений в трехзвенной архитектуре «клиент/сервер».//Всероссийская научно-практическая конференция «Новые технологии и комплексные решения: наука, образование, производство»: Тезисы докладов. – Анжеро-Судженск: Изд-во филиала КемГУ в г. Анжеро-Судженске, 2004, Часть VI, с. 10-12.

7. Оценка производительности серверов / URL: http://www.sun.com.ua/win/news

8. Аверилл М.Лоу, Девид Кельтон. Имитационное моделирование. – СПб: Питер, 2004. – 846 с.