ЛЕКЦИЯ № 4.2. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА КОНЦЕПТУАЛЬНОГО ПРОЕКТИРОВАНИЯ

1. Среды быстрой разработки приложений.

2. Компонентно-ориентированные технологии.

3. Пример реализации компонентно-ориентированной технологии в САПР.

1. Среды быстрой разработки приложений.

CASE-системы часто отождествляют с инструментальными средами разработки ПО, называемыми *средами быстрой разработки приложений* (RAD — Rapid Application Development). Примерами широко известных инструментальных сред RAD являются VB (Visual Basic), Delphi, PowerBuilder фирм Microsoft, Borland, Power Soft соответственно. Применение инструментальных сред существенно сокращает объем ручной работы программистов, особенно при проектировании интерактивных частей программ.

Большое практическое значение имеют инструментальные среды для разработки 1111, предназначенных для работы под управлением операционных систем Windows, в связи с широкой распространенностью последних.

Простейшая система для написания Windows-программ на языке C++, позволяющая сократить объем кода, создаваемого пользователем вручную, основана на библиотеке DLL (DynamicLinkLibrary), которая содержит модули, реализующие функции API (ApplicationProgrammingInterface) для связи прикладных программ с ОС Windows.

Эта система получила развитие в MFC (MicrosoftFoundationClasses), представляющей собой библиотеку классов для автоматического создания каркасов ПО многоуровневых приложений. В библиотеке имеются средства для поддержки оконного интерфейса, работы с файлами и др.

В средах быстрой разработки приложений RAD обычно реализуется способ программирования, называемый *управлением событиями*. При этом достигается автоматическое создание каркасов программ, существенно сокращается объем ручного кодирования. В этих средах пользователь может работать одновременно с несколькими экранами (окнами). Типичными являются окна из следующего списка.

1. Окно меню с пунктами “file”, “edit”, “window” и т. п., реализующими функции, очевидные из названия пунктов.
2. Окно формы, на котором собственно и создается прототип экрана будущей прикладной программы.
3. Палитра инструментов — набор изображений объектов пользовательского интерфейса, из которых можно компоновать содержимое окна формы.

4. Окно свойств и событий, с помощью которого ставятся в соответствие друг другу объекты окна формы, события и обработчики событий. *Событием* в прикладной программе является нажатие клавиши или установка курсора мыши в объект формы. Каждому событию должна соответствовать событийная процедура (обработчик события), которая проверяет код клавиши и вызывает нужную реакцию. В RAD имеются средства для удобства разработки обработчиков событий.
5. Окно редактора кода, в котором пользователь записывает создаваемую вручную часть кода.
6. Окно проекта — список модулей и форм в создаваемой программе.

Для написания событийных процедур в VisualBasic используется язык и текстовый редактор одноименного языка, в Delphi — язык и редактор языка ObjectPascal. В CASE-системе фирмы IBM, включающей части VisualAge (для клиентских приложений) и VisualGen (для серверных приложений), базовым языком выбранSmallTalk. В среде разработки приложений клиент-сервер SQLWindows оригинальные фрагменты программ пишутся на специальном языке SAL. Нужно заметить, что для реализации вычислительных процедур и, в частности, для написания миниспецификаций используется обычная для 3GL технология программирования.

Обычно после написания ПП на базовом языке компилятор системы переводит программу на промежуточный p-код. Вместе с интерпретатором p-кода эта программа рассматривается, как EXE- файл. В некоторых развитых средах компилируется обычный EXE-файл, не требующий интерпретации для своего исполнения.

Помимо упрощения написания пользовательского интерфейса, в средах RAD предусматриваются средства для реализации и ряда других функций. Так, в наиболее развитой версии VisualBasic к ним относятся средства выполнения следующих функций:

— поддержка ODBC, что дает возможность работы с различными СУБД;
— разработка баз данных;
— разработка трехзвенных систем распределенных вычислений;.
— интерактивная отладка процедур на SQLServer;
— управление версиями при групповой разработке ПО;
— моделирование и анализ сценариев распределенных вычислений.

Для создания сред RAD в случае *сетевого программирования* требуется решить ряд дополнительных проблем, обусловливаемых многоплатформенностью в гетерогенных сетях, обилием применяемых форматов данных, необходимостью защиты информации и т.п. Решение этих проблем достигнуто в объектно-ориентированных технологиях на базе языка сетевого программирования Java. Кроме того, с помощью Java удается решить еще одну актуальную для Internet и Intranet задачу — сделать Web-страницы интерактивными.

Платформенная инвариантность в Java достигается, благодаря введению виртуальной метамашины с системой команд, максимально приближенной к особенностям большинства машинных языков. Любой Web-

сервер при наличии запроса на Java-программу со стороны клиента транслирует (компилирует) эту программу на язык метамашины. Скомпилированный модуль, называемый байт-кодом, пересылается клиенту. Клиент должен выполнить интерпретацию байт-кода. Соответствующие интерпретаторы в настоящее время имеются в браузерах всех основных разработчиков Web-технологий.

Java используется двояким образом. Во-первых, как средство “оживления” Web-страниц. В этом случае программныйJava-компонент называют *аплетом,*аплет встраивается в страницу с помощью специального тега, имеющегося в языке HTML. Во-вторых, Java — универсальный язык программирования и может быть использован для написания любых приложений, не обязательно привязанных к Web-технологии.

Хотя и ранее были известны технологии на базе промежуточных p-кодов, именно технология Java, оказалась наилучшим образом приспособленной для использования в гетерогенной сетевой среде. Она последовательно отражает принципы объектно-ориентированного программирования и обеспечивает приемлемую эффективность (производительность) исполнения программ. Эту эффективность можно еще более повысить, если в браузерах заменить интерпретацию на компиляцию.

Для разработки ПОна языке Java создан ряд инструментальных средств. Основной средой является JDK (JavaDeveloper’sKit). В ней имеются: 1) библиотеки классов, в том числе библиотеки основных элементов языка, часто используемых оболочек (wrapper), процедур ввода-вывода, компонентов оконного интерфейса и др. 2) инструментальные средства такие, как компилятор байт-кодов, интерпретатор, просмотрщикаплетов, отладчик, формирователь оконных форм и т.п. Развитую RAD- среду — PowerJ — предлагает фирма Sybase.

Значительное внимание уделяется разработке инструментальных сред для создания Web-узлов, примером такой среды может служить HAHTSite фирмы HAHTSoftware. Для разработки Java-программ из готовых компонентов можно использовать среду IBMVisualAgeforJava, в которой имеются (как и в среде VB) версии учебная, профессиональная и общецелевая (Enterprise), и др.

Наряду с самостоятельными RAD-системами имеются и RAD-системы в составе САПР. Это прежде всего упомянутая выше система CAS.CADE фирмы MatraDatavision.

2. Компонентно-ориентированные технологии.

Появление компонентно-ориентированных технологий вызвано необходимостью повышения эффективности разработки сложных программных систем, являющихся в условиях использования корпоративных и глобальных вычислительных сетей распределенными системами. Компонентно-ориентированные технологии основаны на использовании предварительно разработанных готовых программных компонентов.

Компиляция программ из готовых компонентов — идея не новая. Уже первые шаги в области автоматизации программирования были связаны с созданием библиотек подпрограмм. Конечно, для объединения этих подпрограмм в конкретные прикладные программы требовалась ручная разработка значительной части программного кода на языках третьего поколения. Упрощение и ускорение разработки прикладного ПО достигается с помощью языков четвертого поколения (4GL), но имеющиеся системы на их основе являются специализированными и не претендуют на взаимодействие друг с другом.

Современные системы интеграции ПО построены на базе объектной методологии.

Так, выше приведены примеры библиотек классов, применяя которые прикладные программисты могут создавать субклассы в соответствии с возможностями наследования, заложенными в используемые объектно-ориентированные языки программирования. При этом интероперабельность компонентов в сетевых технологиях достигается с помощью механизмов, подобных удаленному вызову процедур RPC. К библиотекам классов относятся MFC, библиотеки для доступа к реляционным БД (например, для встраивания в прикладную программу драйверов ODBC) и др.

Преимущества использования готовых компонентов обусловлены тщательной отработкой многократно используемых компонентов, их соответствием стандартам, использованием лучших из известных методов и алгоритмов.

В то же время в компонентах библиотек классов спецификации интерфейсов не отделены от собственно кода, следовательно, использование библиотек классов не профессиональными программистами проблематично. Именно стремление устранить этот недостаток привело к появлению CBD — компонентно-ориентированных технологий разработки ПО. Составными частями таких технологий являются унифицированные способы интеграции программного обеспечения.

Возможны два способа включения компонентов (модулей) в прикладную программу — модернизация (reenginering) или инкапсуляция (encapsulation или wrapping).

Модернизация требует знания содержимого компонента, интероперабельность достигается внесением изменений собственно в сам модуль. Такой способ можно назвать способом “белого ящика”. Очевидно, что модернизация не может выполняться полностью автоматически, требуется участие профессионального программиста.

Инкапсуляция выполняется включением модуля в среду с помощью интерфейса — его внешнего окружения (оболочки — wrapper). При этом компонент рассматривается как “черный ящик”: спецификации, определяющие интерфейс, выделены из модуля, а детали внутреннего содержимого скрыты от пользователя. Обычно компоненты поставляются в готовом для использования виде скомпилированного двоичного кода. Обращения к модулю возможны только через его интерфейс. В специфи-

кации интерфейса включаются необходимые для интероперабельности сведения о характеристиках модуля — *модульная абстракция.* В состав этих сведений могут входить описания всех входных и выходных для модуля данных (в том числе имеющихся в модуле интерактивных команд), структура командной строки для инициализации процедур, сведения о требуемых ресурсах.

Компонентно-ориентированные системы построены на основе инкапсуляции компонентов. В архитектуре этих систем можно выделить следующие части: 1) прикладная программа (клиент), создаваемая для удовлетворения возникшей текущей потребности; 2) посредник (брокер или менеджер), служащий для установления связи между взаимодействующими компонентами и для согласования их интерфейсных данных; 3) множество компонентов, состоящих каждый из программного модуля, реализующего некоторую полезную функцию, и оболочки (интерфейса). В спецификации интерфейса могут быть указаны характеристики модуля, реализуемые методы и связанные с модулем события (например, реакции на нажатие клавиш).

Собственно интерфейс представляет собой обращения к функциям модуля, называемым в CBD- технологиях методами. Эти обращения переводятся в двоичный код, что обеспечивает при их использовании независимость от языка программирования. Один и тот же модуль может реализовывать несколько разных функций, поэтому у него может быть несколько интерфейсов или методов. Каждый новый создаваемый интерфейс обеспечивает доступ к новой функции и не отменяет прежние возможно еще используемые интерфейсы.

Схематично взаимодействие компонентов можно представить следующим образом. Клиент обращается с запросом на выполнение некоторой процедуры. Запрос направляется к посреднику. В посреднике имеется предварительно сформированный каталог (реестр или_репозиторий) интерфейсов процедур с указанием компонентов-исполнителей. Посредник перенаправляет запрос соответствующему исполнителю. Исполнитель может запросить параметры процедуры. После выполнения процедуры полученные результаты возвращаются клиенту.

При этом пользователь оперирует удобными для его восприятия идентификаторами компонентов и интерфейсов, а с помощью каталога эти идентификаторы переводятся в указатели (ссылки), используемые аппаратно-программными средствами и которые однозначно определяют интерфейс в распределенной сети из многих компьютеров.

В большинстве случаев реализуется синхронный режим работы, подразумевающий приостановку процесса клиента после выдачи запроса до получения ответа.

Наиболее популярными в настоящее время являются следующиеCBD-технологии.

— OpenDoc — технология, основанная на спецификациях CORBA, разработанных в начале 90х г. г. специально созданным консорциумом OMG,

в который вошли представители ведущих компьютерных фирм. В OpenDoc реализуется технология распределенных вычислений на базе программ-посредников ORB.

— COM (CommonObjectModel) — технология, развиваемая корпорацией Microsoft на базе механизма OLE. Сетевой вариант этой технологии (для систем распределенных вычислений) известен под названием DCOM (DistributedCOM). Объекты DCOM (в частности, объекты, которые можно вставлять в HTML-документы или к которым можно обращаться из Web-браузеров) известны под названием компонентов ActiveX. BCOM/DCOM, как и в OpenDoc, можно использовать компоненты, написанные на разных объектно-ориентированных языках программирования. Но в отличие от OpenDocвCOM/DCOM остается естественная для Microsoft ориентация только на операционные системы Windows (реализация DCOM предусмотрена в ОС WindowsNT 4.0). Технология ActiveX (прежнее название OLEAutomation) обеспечивает интерфейс для управления объектами одного приложения из другого. В общем плане ActiveX — технология интеграции программного обеспечения фирмы Microsoft. Например, используя эту технологию, можно в среде VBA организовать доступ к объектам AutoCAD.

— JavaBeans — сравнительно новая технология, в которой используются компоненты, написанные на языке Java.

Рассмотрим подробнее основные CBD-технологии.

Организация связи клиента с серверными компонентами и в CORBA, и в DCOM происходит с помощью разновидностей языка IDL (InterfaceDefinutionLanguage). Язык IDL в CORBA позволяет описывать интерфейсы создаваемых компонентов. Описание, называемое метаданными, представляется в виде модуля, состоящего из заголовка, описаний типов данных, интерфейсов и операций. В заголовке указывается идентификатор модуля. В части типов данных перечисляются атрибуты, возвращаемые значения, исключительные ситуации. Примерами типов данных могут служить типы базовые (например, float, double, char, boolean, struct), конструируемые пользователем (например, записи и массивы) и объектные ссылки, указывающие на интерфейсы компонентов. Описание интерфейсов начинается с ключевого слова interface, за которым следуют идентификаторы данного интерфейса и возможно наследуемых интерфейсов. Далее описываются операции (методы) в виде идентификаторов операций с возможными перечислениями параметров операций и указанием их принадлежности к входным или выходным данным.

Далее классы объектов (программные модули) должны быть реализованы в CORBA-среде. Для этого компилятор IDL выполняет следующие действия. Во-первых, метаданные для каждого класса объектов помещаются в специальную базу данных, имеющуюся в ORB, — репозитарий интерфейсов. Во-вторых, компилятор создает для каждого определенного на IDL метода клиентский и серверный стабы - специальные программные модули, обеспечивающие доступ к компонентам.

Основное назначение стабов - выполнение маршалинга и организация передачи данных через сеть. Маршалингом называют упаковку параметров в стандартный формат для пересылки. Маршалингнеобходим по той причине, что представление данных в разных компьютерных средах может быть различным (например, различия в кодировке символов, в изображении чисел с плавающей запятой). Клиентский стаб будет использоваться для передачи вызовов и данных от клиента в сеть, а серверный стаб, называемый также скелетоном, будет вызывать метод уже в среде сервера и возвращать результаты.

На серверной стороне данные о каждом новом классе объектов, поддерживаемом конкретным сервером, заносятся в репозитарий реализаций. Эту операцию выполняет объектный адаптер. Обычно в ORB имеется несколько объектных адаптеров, обслуживающих разные группы компонентов (так, возможны объектные адаптеры, ориентированные на библиотеки, на базы данных, на группу отдельных программ и т.п.).

Объектные адаптеры выполняют также ряд других функций, например, таких как интерпретация объектных ссылок, активация и дезактивация компонентов, вызов их методов через скелетоны. Возможны разные способы активации и дезактивации компонентов. В первом из них для обслуживания каждого клиентского запроса создается своя копия компонента. В других способах копии не создаются, компонент обслуживает все запросы с разделением или без разделения во времени.

При реализации запроса брокер через объектный адаптер активирует соответствующий компонент. Далее клиент- серверное взаимодействие происходит через стабы.

Изложенную схему клиент-серверного взаимодействия называют статической. В CORBA предусмотрены также динамические вызовы. Для их осуществления не требуется предварительного формирования стабов с помощью компилятора языка IDL. Вместо стабов, специфических для каждого вызываемого метода, в CORBA используются специальные программы динамического взаимодействия, инвариантные к вызываемым методам. При этом необходимые данные для обращения к компоненту должны быть представлены в клиентской программе, в частности, они могут быть предварительно получены из репозитария интерфейсов. Динамические вызовы обеспечивают большую гибкость при программировании, но выполняются они значительно медленнее

В CORBA предусмотрен ряд унифицированных сервисов, работающих под управлением ORB. В частности, это сервисы:

— именования — присваивает объектам уникальные имена, в результате пользователь может искать объекты в сети по имени;

— жизненного цикла - обеспечивает создание, перемещение, копирование и удаление объектов (документов) в системе, в том числе составных объектов вместе со всеми ссылками и ассоциированными объектами;

— обработки транзакций — осуществляет управление транзакциями (блокировка, фиксация и откат транзакций) из приложений или из ОС, что позволяет многим объектам в сети использовать одни и те же серверы;
— событий — обеспечивает асинхронное распространение и обработку сообщений о событиях, происшедших при реализации процессов, что позволяет заинтересованным объектам координировать свои действия;
— обеспечения безопасности — поддерживает целостность данных.

В COM/DCOM все объекты сгруппированы в классы и каждый класс имеет свой идентификатор CLSID, а каждый интерфейс (метод) класса - свой идентификатор. Для создания объекта (экземпляра класса) клиент обращается к серверу библиотеки COM с указанием CLSID и идентификаторов всех требуемых интерфейсов. Сервер библиотеки COM находит в таблице-реестре по CLSID адрес удаленной машины, на которой размещен запрошенный компонент, и передает ей запрос клиента. На серверной стороне создается объект (копия компонента), он активируется и возвращает клиенту указатели-ссылки на требуемые интерфейсы. Теперь клиент может многократно обращаться к методам объекта, указывая в своих запросах имена интерфейса, методов и их параметров. Обычно объект исполняется на той машине, на которой размещен компонент, но можно выбрать и другую машину, указав в запросе, например, ее IP- адрес.

Появление технологии JavaBeans обусловлено успехом языка программирования Java. Технологию JavaBeans отличают от COM/DCOM две особенности. Во-первых, Java — единственный в JavaBeans язык программирования. Единственность языка и притом объектно-ориентированного обусловливает сравнительную легкость освоения и применения технологии JavaBeans. Во-вторых, технология JavaBeans является платформенно-независимой.

Компоненты в JavaBeans являются классами Java. Для их создания, модификации и объединения в прикладные программы имеются специальные средства в составе инструментальной среды JDK (JavaDevelopmentKit). С их помощью компоненты JavaBeans могут быть встроены в Java-аплеты, приложения или другие более крупные компоненты. В качестве Java-аплетов компоненты JavaBeans поддерживаются большинством имеющихсяWWW-браузеров.

Развитие CBD-систем возможно в направлении дальнейшего упрощения программирования и, следовательно, сокращения сроков разработки ПО, однако это происходит за счет снижения степени универсальности соответствующих инструментальных средств. Такие более специализированные средства представляют собой группу компонентов, взаимосвязанных некоторым зависящим от приложения образом, и входят в системные среды САПР.

В общем случае компоненты системной среды объединены в несколько сценариев (потоков процедур или маршрутов), в которых выделяются точки входа для вставки специфичных пользовательских фрагментов и расширений. Имеются возможности не только вставки новых фрагментов, но

и замены исходных компонентов в потоках процедур на оригинальные с сохранением интерфейса. Собственно многие системы, основанные на применении языков четвертого поколения (4GL), относятся именно к таким системным средам, в которых последовательности инкапсулированных модулей образуются с помощью операторов 4GL.

3. Пример реализации компонентно-ориентированной технологии в САПР.

Основные идеи компонентно-ориентированной (объектной) технологии с созданием расширенных специализированных библиотек компонентов реализованы в системе CAS.CADE (ComputerAidedSoftware/ ComputerAidedDesignEngineering) фирмы MatraDatavision.

Система CAS.CADE состоит из нескольких частей. Основными частями являются библиотеки классов и инструментальная среда для создания программного обеспечения (ПО) технических и научных приложений.

Библиотеки (ObjectLibraries) в CAS.CADE представляют собой специализированные наборы заранее разработанных компонентов на языке C++. Совокупность библиотек имеет иерархическую структуру. Базовые компоненты соответствуют классам объектной методологии. Примерами компонентов являются строки, списки, точки, матрицы, линии, поверхности, деревья, решатели уравнений, операторы сортировки, поиска на графах и т.п. Классы группируются в пакеты (Packages), пакеты - в наборы (Toolkits), наборы - в домены (ResourseDomains).

В CAS.CADE выделено несколько библиотек. Во-первых, это библиотеки 2D и 3D моделирования, включающие компоненты для определения, создания и манипулирования геометрическими моделями. Во-вторых, ряд библиотек предназначен для связи с ОС и управления данными, для обмена данными с внешними CAD системами, для создания сеточных моделей и др. Так, в состав библиотеки обмена данными входят конверторы данных из формата CAS.CADE в Express- файл прикладного протокола AP214 стандарта STEP и обратно. Аналогичные конверторы имеются для взаимного преобразования данных из формата CAS.CADE в другие популярные в САПР форматы IGES и DXF/SAT.

Необходимо отметить, что основные приложения, на которые ориентированаCAS.CADE, — это приложения машинной графики и геометрического моделирования, поэтому в системе наиболее развиты библиотеки графических и геометрических компонентов.

Геометрическое моделирование и визуализация в CAS.CADE поддерживаются соответствующим ПО. В это ПО входят библиотечные наборы “Геометрия”, “Топология”, “Визуализация” и др. Для тестирования и демонстрации компонентов перед их встраиванием в проектируемую прикладную САПР используются специальные язык, интерпретатор и просмотрщик, составляющие подсистему “Тестирование”.

Набор “Геометрия” включает пакеты канонических геометрических элементов и массивов (множеств) этих элементов.

Пакеты gp, geom2d и geom включают 2D и 3D геометрические элементы (классы), используемые в качестве сущностей в вычислительных процедурах, в том числе в таких операциях, как поворот, отражение, масштабирование и т.п. Примерами элементов могут служить декартовы координаты, точки, векторы, линии, окружности, квадратичные кривые, сферические, тороидальные и конические поверхности, кривые и поверхности Безье, B-сплайнов и др.

Большое число пакетов разработано для выполнения геометрических построений и метрических расчетов. Пакеты gce, GC, GCE2d включают алгоритмы построения сущностей из элементов пакетов gp, Geom, Geom2d, например, построение прямых, дуг окружностей, кривых по заданным параметрам таким, как инцидентные точки, центральные точки и радиусы, параллельные или нормальные прямые и т.п.

Набор “Топология” определяет структуры данных, описывающих связи (отношения) между геометрическими сущностями - классами предыдущего набора “Геометрия”.К структурам топологических данных относятся вершины, ребра, линии каркасных моделей, участки поверхности, оболочки - совокупности связанных через ребра участков поверхности, тела - части пространства, ограниченные оболочкой, совокупности тел, в том числе простые конструкции вида частей цилиндра, конуса, сферы, тора. В наборе имеются также средства: 1) для скругления острых углов и кромок, т.е. формирования галтелей постоянного или переменного радиуса; 2) для поддержания непрерывности при сопряжении разных поверхностей; 3) для метрических расчетов - определения длин ребер, площадей участков поверхности, объемов тел, центров масс и моментов инерции.

В подсистему “Тестирование” входят командный язык TCL (TestCommandLanguage), на котором задается программа тестирования и просмотра библиотечных компонентов, интерпретатор TCL и 2D/3D визуализатор. В TCL имеются обычные для языков программирования команды, такие как присвоение значения переменной, организация цикла, условный переход, так и специальные команды. Среди последних выделяют базовые, геометрические и топологические команды. Примеры базовых команд: задержка при исполнении программы (например, при презентациях), обращение к файлу, вывод на экран координат и других параметров геометрических объектов, создание окон для различных видов, масштабирование изображения, его поворот, установка цвета, выделение на экране одного заданного объекта и т.п. С помощью геометрических команд выполняют создание и модификацию кривых, поверхностей, геометрические преобразования типа поворота или зеркального отражения, вычисления координат, кривизн, производных, нахождение точек пересечения линий и поверхностей. Аналогичные действия производят по отношению к топологическим объектам с помощью топологических команд.

Инструментальная среда CAS.CADE включает интегрированную оболочку, подсистему проектирования пользовательского интерфейса, а также ряд многократно используемых специализированных программ, таких как 2D и 3D моделеры, подсистема управления данными, прикладные программы анализа и т.п.

Интегрированная оболочка служит для управления версиями и параллельной работой многих пользователей.

Для проектирования пользовательского интерфейса в CAS.CADE имеются специальные языковые и программные средства. Язык проектирования диалога состоит из команд создания интерфейса и доступа к компонентам.

Создание интерфейса включает создание контейнеров и диалоговых элементов. Контейнер представляет собой экранное окно, в котором будут размещаться элементы. Элементы обеспечивают информирование пользователя создаваемого приложения о возникающих событиях, дают возможность пользователю задавать значения параметров, выбирать режим работы и т.п.

Различают ряд видов контейнеров. Среди них контейнеры для сообщений, предупреждающих об ошибке, запрашивающих от пользователя ответы типа “да/нет”, задания размеров или цвета, выбора файла и т.п.

Примерами команд проектирования диалоговых элементов могут служить команды определения позиции элемента в окне, выбора одного элемента из заданного множества, конструирования текстовой строки или меню, фиксации событий, вызванных выбором мышью позиции или пункта меню, и др.

В структуре прикладной программы, создаваемой в среде CAS.CADE, можно выделить диалоговый модуль (модуль пользовательского интерфейса GUI - GraphicUserInterface), модуль связи с прикладной частью и собственно прикладную часть, включающую отобранные компоненты и БД, зависящую от приложения

Объединение используемых в приложении компонентов в прикладную программу осуществляется на языке C++ или специальном языке описания интерфейсов, напоминающем язык IDL.. Следовательно, реализуются присущие C++ поддержка наследования и ограничение доступа (компоненты могут иметь статус защиты от несанкционированного доступа).

С помощью CAS.CADE создают специализированные приложения (прежде всего специализированные САПР) с сравнительно малыми затратами времени и средств.