граммирования агентов (TeleScript); языки сценариев (Tcl/Tk); символьные языки и языки логического программирования (Oz).

Одно из самых главных свойств агента – это интеллектуальность. Интеллектуальный агент владеет определенными знаниями о себе и об окружающей среде, и на основе этих знаний он способен определять свое поведение. Интеллектуальные агенты являются основной областью интересов агентной технологии. Важна также среда существования агента: это может быть как реальный мир, так и виртуальный (компьютерный), что является важным в связи с всеобщим распространением сети Internet.

От агентов требуют способности к обучению и даже самообучению. Поскольку обучение обуславливает наличие знаний у обучаемого, то обучаемым или самообучаемым может быть только интеллектуальный агент. В зависимости от наличия свойства планирования агенты подразделяются на регулирующие и планирующие. Если умение планировать не предусмотрено (регулирующий тип), то агент будет постоянно переоценивать ситуацию и заново вырабатывать свои действия на окружающую среду. Планирующий агент имеет возможность запланировать несколько действий на различные промежутки времени. При этом агент имеет возможность моделировать развитие ситуации, что дает возможность более адекватно реагировать на текущие ситуации. Однако агент должен учитывать не только свои действия и реакцию на них, но и сохранять модели объектов и агентов окружающей среды для предсказания их возможных действий и реакций.

Агент может иметь доступ к локальным и глобальным ресурсам. При этом агентов, которые имеют доступ к локальным ресурсам (ресурсы, к которым имеет доступ пользователь, в том числе и сетевые), называют персональными помощниками, они автоматизируют работу текущего пользователя, помогая ему в выполнении некоторых операций. Соответственно, сетевой агент самостоятельно получает доступ к информации, не доступной пользователю напрямую либо доступ к которой не был предусмотрен. Важным свойством является мобильность – возможность менять свое местонахождение в окружающей среде. Для программного агента под мобильностью понимается возможность передвигаться по сети от компьютера к компьютеру. Переходя от одного компьютера к другому, такой агент может обрабатывать данные и передавать по сети только результаты своей работы. Система, в которой несколько агентов могут общаться друг с другим, передавать друг другу некоторую информацию, взаимодействовать между собой, и называется многоагентной (МАС) [1].

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Трахтенгерц Э.А.* Компьютерная поддержка принятия согласованных решений. // Информационные технологии (приложение к журналу), 2002, №3.
2. http://dll.botik.ru/nut/searchagents.html.

И. Ю. Косов, В.В. Марков

ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЙ САПР С ИСПОЛЬЗОВАНИЕМ ГРАФИЧЕСКОЙ СРЕДЫ РАЗРАБОТКИ НА ОСНОВЕ ПРАВИЛ ЯЗЫКОВ ВЫСОКОГО УРОВНЯ

Введение. Уровень сложности программного обеспечения современных САПР достаточно высок, и эффективная разработка прграммных решений проектных задач в используемой операционной среде иногда серьезно затруднена.

Упростить и облегчить разработку приложений возможно с помощью таких сред программирования, как Borland C++ Builder или Microsoft Visual C++. Так, Borland C++ Builder является достаточно удобным инструментом при создании стандартных приложений, имеющих графически сложный интерфейс, за счет расширенного набора стандартных компонентов [2]. Достоинством этой среды программирования является также краткость записи кода; недостатком - ограниченная гибкость в использовании библиотек и довольно «размытое» описание правил и методик их использования [5]. В свою очередь, гибкость разработки приложений может быть обеспечена в среде Microsoft Visual C++, на базе использования библиотеки Microsoft Foundation Class Library (MFC), упрощающей и ускоряющей процесс программирования под Windows [1]. Недостатком в использовании этой среды программирования является необходимость достаточно подробного описания программных кодов, что существенно увеличивает временные затраты на разработку программных приложений.

Однако при создании проекта независимо от среды разработки, получение «чистого» кода проекта, даже сравнительно малого объема, сопровождается проблемой понимания исходного текста программы. Особенно зримо эта проблема проявляется в случаях разработки программного продукта командой разработчиков. Проблема не снимается использованием, например, комментариев.

Таким образом, достаточно важной и актуальной проблемой является разработка среды программирования, сочетающей простоту использования и гибкость при создании приложений и, в то же время, обеспечивающей эффективность работы как отдельных программистов, так и коллективов разработчиков, за счет имеющихся средств повышения наглядности и идентификации программного кода. Такой средой может быть графическая среда разработки приложений (ГСРП).

Предлагаемая статья содержит описание структуры новой графической среды разработки программных приложений, структуры хранения и обработки информации, а также средств визуального представления проектируемых приложений на основе среды проектирования “NI LabView 8.0” [4].

1. Структура и описание графической среды разработки приложений. ГСРП предназначена для разработки программных приложений с использованием графически представляемых программных структур. Среда дает возможность создавать графическое описание проекта, преобразовывать его в семантике языка C++ и имитировать исполнение проекта.

Структура графической среды разработки приложений, показанная на рис.1, опирается на следующие основные блоки:

- графический интерфейс ввода проекта;
- блок считывания элементов программных структур;
- библиотека правил формирования кодов элементов программных структур;
- блок формирования кода проекта на языке высокого уровня (C++);
- библиотека правил преобразования графических программных структур в коды на языке высокого уровня;
- блок имитации исполнения работы программных структур;
- библиотека правил имитации работы элементов;
- блок проектирования интерфейса конечного пользователя.

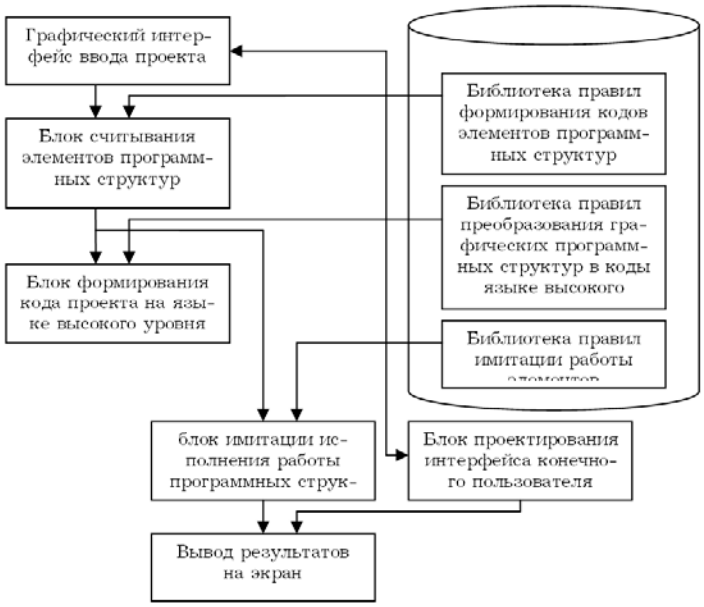

Рис.1. Структурная схема ГСРП

В соответствии с предложенной структурой ГСРП, назначение элементов, в неё входящих, может быть кратко сформулировано следующим образом:

- графический интерфейс ввода проекта представляет возможность пользователю достаточно просто, в понятной и доступной графической форме, описать проект разрабатываемого приложения (алгоритма, программы);
- блок считывания элементов программных структур предоставляет возможности обработки графического изображения проекта и его представления во внутренней табличной структурированной форме (структурированная таблица представления данных программы), реализуемые на основе использования библиотеки правил формирования кодов элементов программных структур, хранящей информацию о соответствии условного графического отображения программной структуры и её логической структуры;
- блок формирования программного кода проекта на языке высокого уровня позволяет, используя библиотеку правил преобразования графических программных структур в коды языка высокого уровня (здесь хранится информация о соответствии логической структуры данных с ее реализацией на языке высокого уровня), провести трансляцию структурированной таблицы представления данных программы в код языка C++, получая на выходе текстовое описание разрабатываемого приложения в кодах указанного языка;
- блок имитации исполнения работы элементов программных структур, работающий в связке с библиотекой правил имитации, предназначен для проверки и оценки работоспособности проектируемого приложения без его обязательного перевода в программный код;
- интерфейс конечного пользователя представляет возможность разместить на экране поля данных, корректно и удобно отображающие результаты работы проектируемого приложения.

Таким образом, реализация проекта подразумевает построение разрабатываемого приложения в графическом интерфейсе, при этом одновременно возможно формировать поля вывода результатов работы спроектированного приложения (интерфейс конечного пользователя). Представление проекта в графическом виде в последующем может быть преобразовано в программный код языка С++, либо передано программному имитатору для визуализации его функционирования в качестве приложения Windows [3].

2. Структура хранения данных и их индексация. Все поставленные задачи создания ГСРП прямо или косвенно сводятся к возможности обработки С++ кода. Опираясь на официальные документы языка С++, можно сделать вывод о необходимости привлечения следующих структур хранения данных: структура одного данного (рис.2), общая структура хранения всех данных и описание их связей.

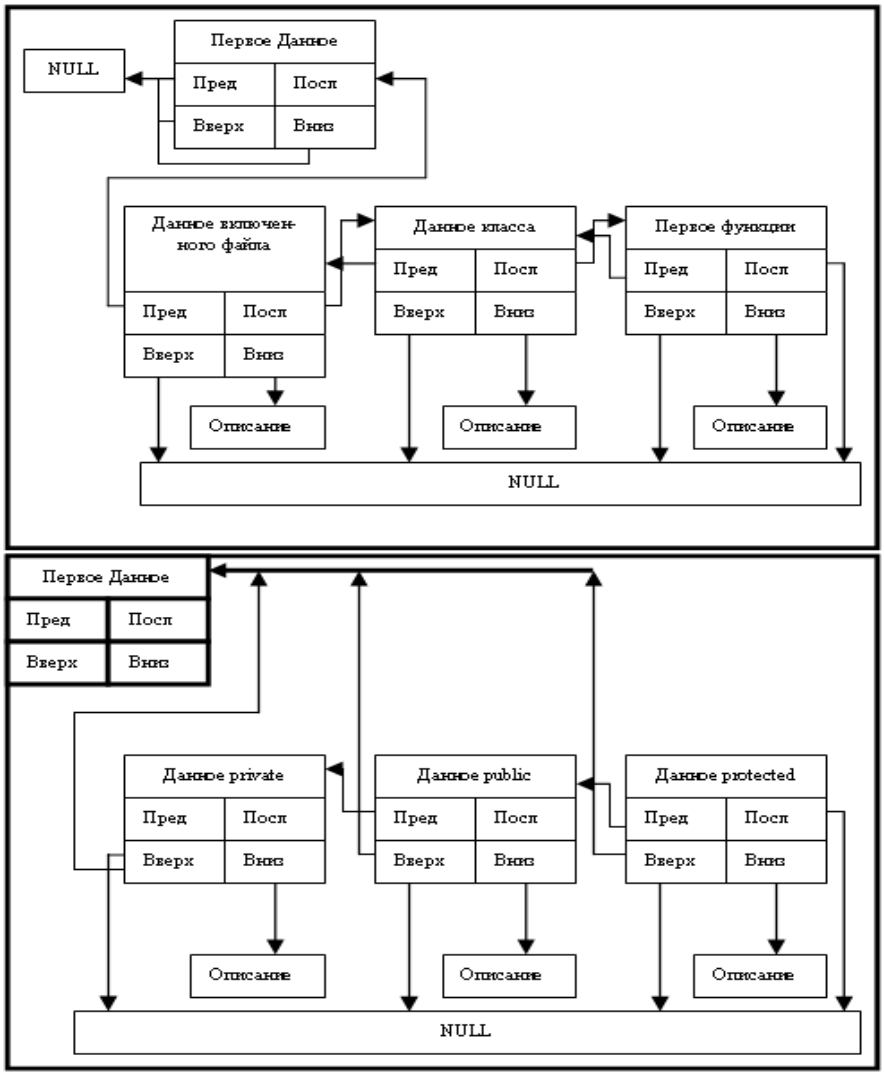

Рис.2. Схема формирования структуры данных

При формировании структуры данных предложено руководствоваться следующими правилами:

- каждая структура или класс представлен как данное;
- каждая структурная единица данных имеет зону видимости;
- каждая зона видимости имеет список данных и описание функций;
- каждая функция определяется списковой формой, куда заносятся, по определенным правилам, все данные и относительно которой формируется область видимости;
- каждая структурная единица данных имеет указатель на наследника;
- каждая структурная единица данных имеет список данных того же типа.

Схема формирования структуры данных (см. рис.2).

Структурная единица данных сопровождается шестью указателями: первые два на предыдущее и последующее данные; предусмотрен указатель отсутствия одного из них; третий указатель ссылается на таблицу объектов; четвертый – на типы и функции; пятый и шестой – на данные того же типа по иерархии вверх и вниз соответственно.

Таблица описания объектов хранит в себе название объекта, идентификатор типа объекта через ссылки на таблицу типов и функций, хранящийся в таблице типов, и указатель описания структурной единицы данных. Таблица типов и функций содержит три указателя на списки: типов, функций и операций. В свою очередь, список операций состоит из четырех указателей на все стандартные операции и на список подключаемых данных. Структура хранения данных одинакова для даных всех уровней , что позволяет обеспечить их хранение и индексацию без нарушения правил языка C++.

Заключение. Графическая среда разработки, позволяющая снизить остроту описанных выше проблем, по возможностям ничем не уступающая другим средам разработки, дает возможность увеличить наглядность процесса проектирования приложений и понимание кода, а также ускорить процесс создания стандартных приложений. Предлагаемая ГСРП обладает достаточной гибкостью за счет использования библиотек MFC. Используемый в ГСРП метод проектирования приложений обеспечивает возможность создания приложений, в том числе и приложений САПР, специалистам с недостаточно глубоким знанием языка программирования C++, что позволяет уделить большее внимание не задачам программирования, а решению консрукторско-технологических проблем

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Шилдт Г.* MFC: основы программирования. – Киев: BHV, 1997. – 560 с.
2. *Архангельский А.Я.* C++ Builder 6. Справочное пособие - Книга 2. Классы и компоненты. – М.: Бином-Пресс, 2002. – 528 с.
3. *Рихтер Дж.* Windows для профессионалов: создание эффективных Win32 приложений с учетом специфики 64-разрядной версии Windows. - 2000г.
4. *Суранов А.Я.* LabView 7: справочник по функциям. – М.: Изд-во ДМК, 2005. – 512 с.
5. *Hollingworth J., Butterfield D., and others.* C++ Builder 5 Developer`s Guide. - Willey, 2003.