УДК 681.3.06

С.В.Смирнов

ОБ ОДНОМ АСПЕКТЕ УПРАВЛЕНИЯ ДАННЫМИ В ЗАДАЧАХ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

(г. Куйбышев)

Становление современных информационных технологий в программировании – функционального, логического и т.д. программирования, естественно, сопровождается изменением содержания ранее изученных и появлением новых задач управления данными. Это относится не только к управлению данными в проблемно-ориентированных ППП (пакет прикладных программ), построенных на основе той или иной технологии (см. /1, 2/), но и к соответствующей поддержке собственно средств программирования (примеры см. в работах /3, 4/).

Завоевывающий все большую популярность объектно-ориентированный стиль программирования /4/–/6/ также предполагает реализацию адекватных механизмов использования основной (ОП) и внешней памяти (ВЗУ). Предлагаемая статья посвящена вопросам организации архива программных объектов во внешней памяти, реализации функций записи, хранения объектов в ВЗУ, чтения объектов из ВЗУ в ОП.

Мир задачи. Как известно /3–6/, программный объект является семантической единицей, которая выражает свойства и(или) функции явно выделенных понятий предметной области и представляется в виде адресуемой программно-технической единицы в смысле абстрактных типов данных. Предметом определения в объектно-ориентированном языке является класс объектов (экземпляры классов – собственно объекты, могут порождаться и уничтожаться в ходе выполнения программы). Например, модифицируя абстрактное описание графа /4/, можно следующим образом определить класс "Граф" (GRAPH) в рамках модуля-концепта "Графы" (GRAPHS):

```
MODULE  GRAPHS
  CLASS  GRAPH  IS
    OPERATIONS:
      VERTICES :   GRAPH → SET(VERTEX)
      ARROWS :     GRAPH → SET(ARROW)
      ADD-NODE:    VERTEX x GRAPH → GRAPH
```

```
    ADD-ARROW: ARROW x GRAPH → GRAPH
    DEL-NODE : VERTEX x GRAPH → GRAPH
    DEL-ARROW: ARROW x GRAPH → GRAPH
END GRAPH
CLASS VERTEX IS
  OPERATIONS:
    GREATE:   IDENTIFIER → VERTEX
    CHILDREN: VERTEX → SET(VERTEX)
    ARROWS :  VERTEX → SET(ARROW)
    IDENTITY: VERTEX x VERTEX → BOOLEAN
END VERTEX
CLASS ARROW IS
  OPERATIONS:
    GREATE:      DESCRIPTION x VERTEX x VERTEX → ARROW
    SOURCE :     ARROW → VERTEX
    DESTINATION: ARROW → VERTEX
END ARROW
END GRAPHS
```

Концепт *GRAPHS* предоставляет комплект базовых понятий для решения задач в области графов. При этом, как правило, каждую конкретную задачу можно свести к исследованию одного экземпляра класса *GRAPH* , который составит предмет, или мир, данной задачи. Это обстоятельство иллюстрирует принципиально важный для нас факт: всегда существует класс (можно определить такой класс), интегрирующий все знания о предметной области задачи. Естественно, при решении задачи будет порождаться единственный экземпляр такого класса, т.е. то, что можно назвать объектом-миром задачи.

Проблема контекста. В общем случае любой программный объект содержит разнообразные ссылки (адресные величины) на другие объекты и в том смысле "связан" структурой адресного пространства выполняемой программы. Если разместить подобный объект для хранения в ВЗУ, то возвращение его в ОП, в адресный контекст другой программы, по меньшей мере проблематично.

Для объекта-мира драматизм положения несколько ниже - по крайней мере отсутствуют "внешние" ссылки. Семантика же "внутренних"

ссылок подобных объектов прежде всего представляет структуру объекта-мира. Например, объект *GRAPH* включает множества ссылок на составляющие его вершины и дуги - соответственно *SET(VERTEX)* и *SET(ARROW)*.Преодолеть адресную контекстную зависимость внутреннего представления объекта-мира для сохранения этого объекта в архиве, по-видимому, можно лишь разрушив ее, помещая в архив не слепок ОП, а план воссоздания структуры объекта в рамках другого адресного контекста. Такой подход реалистичен по той причине, что описание класса, экземпляр которого представляет мир задачи, обычно содержит операции "редактирования" структуры объекта (например, операции типа *ADD* и *DEL* в *GRAPH*) и, во всяком случае, операцию порождения объекта (в примере для *GRAPH* она отсутствует именно из-за наличия стандартного генератора объектов-миров; это не исключает случаев описания оригинальных генераторов, подобных *GREATE* у вершин и дуг графа).

Таким образом, в архиве предлагается хранить протоколы создания объектов-миров с помощью их собственных операций редактирования. При этом из всех операций редактирования в протоколе будут отмечены лишь "конструктивные", необходимые для восстановления структуры объекта,существовавшей на момент записи в архив.По этой причине в протоколе скорее всего не будут фигурировать операции переименования, уничтожения и др. (например *DEL* - операции в *GRAPH*).

Архивные операции записи и чтения в рассматриваемой ситуации оказываются оригинальными для данного класса объектов-миров. Операция записи выполняет "сканирование" объекта-мира и формирование в архиве во внешней памяти протокола, где фиксируется порядок выполнения, тип и параметры операций редактирования для последующего воссоздания объекта. Операция чтения представляет собой интерпретатор архивного протокола с "конструктивными" операциями редактирования в качестве семантических процедур. В этом духе приведенное выше описание класса *GRAPH* можно предложить дополнить операциями

WRGRAPH: GRAPH → PROGRAPH

RDGRAPH: PROGRAPH → GRAPH

где *PROGRAPH* специальный для *GRAPH* тип протокола в архиве.

Разумеется, говоря об операциях чтения архива, следует добавить, что код описания соответствующих классов объектов-миров должен быть скомпонован с любой программой, где предполагается ввод объектов из архива.

Завершая обсуждение, необходимо подчеркнуть, что рассмотренный метод не позволяет в общем случае сохранить в архиве вместе с объектом какую-либо информацию кроме той, что необходима "конструктивным" операциям редактирования. Наиболее неприятной может оказаться невозможность хранения наряду с объектом определенной части результатов решения прикладных задач (прежде всего - ссылочных значений), аргументом которых является объект-мир.

Файлы. Предполагаемая последовательная интерпретация протокола восстановления объекта предопределяет последовательный способ организации архивных файлов в ВЗУ. При этом должна быть решена задача спецификации файла-протокола.

Вопрос об идентификации устройства (носителя), по-видимому, всегда может быть решен путем стандартизации. Типы файлов в архиве будут различны и должны однозначно соответствовать классам регистрируемых объектов-миров. Предпочтительным решением вопроса об именовании файлов следует считать выбор в качестве имени файла, содержащего протокол воссоздания некоторого объекта, идентификатора этого объекта. Поэтому каждый вновь генерируемый объект-мир наряду с обычной в объектно-ориентированном программировании идентификацией ссылочным значением целесообразно идентифицировать еще и по правилам именования файлов используемой системы программирования.

Рассмотренный способ хранения объектов формирует отношение "один объект - один файл". Дальнейшая структуризация множества хранимых объектов данного класса, группы родственных классов и т.д. возможна путем создания локальных каталогов или библиотечных файлов. Отметим, что предложенный подход к организации архива объектов успешно реализован автором в ППП сетевого планирования и управления специального вида, целиком выполненном как объектно-ориентированный проект на языке Модула-2 /7/.

Библиографический список

I. Будячевский И.А., Кораблин М.А., Смирнов С.В. Управление

данными при обработке на ЭВМ результатов научного эксперимента // Автоматика и вычислительная техника. 1977. № 2. С. 54–58.

2. Автоматизация проектирования АСУ с использованием пакетов прикладных программ /Ю.М.Черкасов, В.А.Гриншейн, Ю.Б.Радашевич и др. М.: Энергоатомиздат, 1987. 328 с.

3. Технология системного моделирования /Е.Ф.Аврамчук, А.А.Вавилов, С.В.Емельянов и др. М.: Машиностроение. Берлин: Техник, 1988. 520 с.

4. Фути К., Судзуки Н. Языки программирования и схемотехника СБИС. М.: Мир, 1988. 224 с.

5. Андрианов А.И., Бычков С.П., Хорошилов А.И. Программирование на языке Симула-67. М.: Наука, 1985. 288 с.

6. Wilson R. Object-oriented languages reorient programming techniques // Computer Design. 1987. November 1. p. 52–62.

7. Вирт Н. Программирование на языке Модула-2. М.: Мир, 1987. 224 с.

УДК 681.518.2:620.16

Ш.Ю.Исмаилов, М.И.Рева, О.В.Шакирова

ОЦЕНКА ПОГРЕШНОСТЕЙ КВАНТОВАНИЯ В ИЗМЕРИТЕЛЬНО-ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЕ ДЛЯ УСТАЛОСТНЫХ ИСПЫТАНИЙ В МАШИНОСТРОЕНИИ

(г. Ленинград)

В настоящее время ведутся интенсивные исследования по созданию измерительно-вычислительных систем (ИВС) для измерения накопленных усталостных повреждений (НУП) сложных машиностроительных конструкций в процессе их эксплуатации. Внедрение систем данного класса должно способствовать существенному снижению аварийности и повышению ресурса эксплуатируемой техники, например в авиации до 45–60тыс. летных часов /1/.

В работе рассматриваются вопросы, связанные с построением вычислительной подсистемы многоканальной ИВС, реализующей группу так