АКАДЕМИЯ НАУК СССР
СИБИРСКОЕ ОТДЕЛЕНИЕ
А В Т О М Е Т Р И Я

№ 4 1990

ОРГАНИЗАЦИЯ ИНТЕРФЕЙСА С ПОЛЬЗОВАТЕЛЕМ

УДК 519.681.3.06

И. М. ШАМАЕВА
(Москва)

СИСТЕМЫ УПРАВЛЕНИЯ ПОЛЬЗОВАТЕЛЬСКИМ ИНТЕРФЕЙСОМ

Введение. Система поддержки диалога с пользователем — UIMS (User Interface Management System) — принятый термин для обозначения программной системы, позволяющей разработчику интерактивной прикладной системы строить интерфейс с конечным пользователем.

В работе приводятся основные определения и перечисляются задачи в области создания UIMS. Предлагается схема конкретной UIMS, ориентированной на применение в области геометрического моделирования. Выбор схемы основан на нашем опыте разработки пользовательского интерфейса в системах дизайна поверхностей кузовов.

1. **Что такое UIMS?** Термин UIMS был введен в 1982 г. Д. Касиком [1] применительно к его системе TIGER. С этого времени термин применялся к различным программным системам, однако и сейчас «значение этого термина составляет само по себе предмет исследований» [2]. Обсуждения UIMS, имеющиеся в литературе,— это либо описание конкретных систем, либо перечисление свойств, которыми должна обладать такая система [3].

В [2] указывается, что основное назначение UIMS — освободить разработчика интерактивной системы от программирования деталей диалога компьютера с конечным пользователем. Одновременно применение UIMS должно гарантировать легкое и эффективное взаимодействие конечного пользователя с прикладными модулями системы.

Другими словами, назначение UIMS — выделить разработку диалога в самостоятельную задачу, при этом пользовательский интерфейс становится не зависящим от приложения. Таким образом, интерактивная система разбивается на две подсистемы — прикладной пакет и подсистема диалога с пользователем, реализуемая средствами UIMS.

Характерная черта UIMS — наличие следующих двух подсистем: проектирования диалога на стадии создания конечной системы и ведения диалога с пользователем в процессе работы с этой системой.

2. **Проблематика.** В области разработки UIMS имеются отдельные подзадачи. Первая — это определение (на этапе проектирования) и воплощение (на этапе выполнения) примитивов диалога, как правило, из стандартного набора: «всплывающие» (pop-up) и «свешивающиеся» (pull-down) меню, «hot keys», командные строки и т. д. Для систем, работающих с текстом и графикой, этот вопрос хорошо изучен. Для систем геометрического моделирования вопрос определения удобного набора специфических примитивов диалога остается открытым.

Вторая — это описание и реализация процесса диалога в целом. Каждое элементарное действие пользователя переводит систему в некоторое новое состояние. Во-первых, система реагирует на ввод, выполняя какие-

то функции, во-вторых, в системе предопределяется реакция на последующие варианты пользовательского ввода. Распространено описание диалога в целом с использованием модели состояний-переходов. Задается граф, в котором вершины — это состояния, т. е. моменты ожидания ввода, дуги — переходы. Выбор перехода зависит от действия пользователя, переход может сопровождаться выполнением прикладной функции. По-видимому, эта модель является устаревшей (хотя бы потому, что описание какой-либо реальной интерактивной системы на этом языке является слишком трудной задачей), однако понятие состояния системы остается полезным.

Третья подзадача, связанная со второй, состоит в следующем: какой должна быть архитектура конечной системы с точки зрения связи прикладной и диалоговой компонент? Следует определить, как происходит ровь; полная универсализация здесь вряд ли возможна, так как различные приложения требуют существенно различной формы диалога. Ниже мы приводим схему UIMS, которую назовем Системой диалога. Система диалога ориентирована на геометрические приложения. Интерактивную геометрическую систему, построенную с использованием Системы диалога, назовем конечной системой.

3. **Система диалога.** Описывая Систему диалога, уделим основное внимание архитектуре конечной системы и взаимодействию ее прикладной и диалоговой компонент. Опишем информацию, которой разработчик снабжает Систему диалога в процессе создания конечной системы, не конкретизируя способы ввода этой информации (интерактивный ввод, создание программ на специальном языке или заполнение таблиц).

3.1. *Назначение Системы диалога.* Система диалога представляет собой инструментальную среду, которая позволяет с небольшими затратами организовать качественный диалог в интерактивной графической системе, работающей с геометрическими объектами. Система диалога предназначена для использования разработчиком прикладной подсистемы и служит для превращения набора прикладных алгоритмов в систему для конечного пользователя. Создание конечной системы состоит в настройке компонент подсистемы диалога на работу в составе конкретной конечной системы.

3.2. *Формирование диалога в конечной системе.* Форма диалога в конечной системе определяется в несколько этапов. Первый этап, основной, выполняется разработчиком и состоит в привязке объектов будущей системы к их внешним представлениям и установлении соответствия между прикладными программами и командами будущей системы. Этот этап назовем генерацией конечной системы. Он выполняется один раз, и его результатом является конкретная работоспособная интерактивная система.

Следующий этап — настройка системы. Он может быть выполнен разработчиком или квалифицированным пользователем. Сюда входит повторное задание атрибутов диалога (получивших на этапе генерации значения по умолчанию): распределение окон на экране, задание конкретных цветов, текстов подсказок и меню, кодов кнопок. На этапе настройки возможно расширение базового набора команд, образованного при генерации, с помощью аппарата макрокоманд. Этот этап служит, в частности, для учета уровня квалификации конкретного пользователя.

Наконец, во время работы с конечной системой пользователь может динамически менять атрибуты диалога. На наш взгляд, динамической настройке должно подвергаться минимальное число атрибутов: для конечного пользователя удобно работать с привычным видом экрана и стилем диалога.

3.3. *Система диалога для конечного пользователя.* Работа готовой системы, построенной с помощью Системы диалога, представляет собой последовательное выполнение команд; выбор команд осуществляет пользователь. Часть команд направлена на преобразование прикладной модели, что и является целью сеанса работы с интерактивной системой. Остальные команды вспомогательные. Они служат для ввода-вывода объектов модели, оценки качества преобразованной модели, настройки параметров диалога, протоколирования сеанса работы.

Команды имеют входные и выходные аргументы. Конечный пользователь общается с системой, выбирая команды, вводя входные аргументы и анализируя выходные аргументы по их изображению (подчеркнем, что деятельность конечного пользователя только в этом и состоит). При этом аргументы представляют собой семантически значимые для пользователя величины: геометрические объекты, атрибуты этих объектов, множества геометрических объектов и т. д.

3.4. *Система диалога для разработчика конечной системы.* Конечная система представляет собой объединение диалоговой и прикладной компонент. Система диалога снабжает разработчика средствами создания диалоговой компоненты. Прикладная компонента — это пакет подпрограмм, не использующих структур данных диалоговой компоненты; прикладные программы способны работать автономно.

3.4.1. Что предоставляется разработчику. Средствами Системы диалога обеспечиваются визуализация прикладной модели (т. е. геометрических объектов) и примитивов диалога (т. е. меню, подсказок и т. д.), организация запроса у пользователя команд и входных аргументов и вывода выходных, слежение за корректностью последовательности выполняемых команд.

Управление в конечной системе лежит на диалоговой компоненте, разработанной средствами Системы диалога. Она инициализирует выполнение прикладных функций и снабжает эти функции значениями параметров в формате, заданном разработчиком прикладной компоненты. Дисциплина взаимодействия прикладной и диалоговой компонент требует, чтобы прикладные функции не обращались к графическому вводу-выводу. Таким образом, прикладные функции оказываются независимыми от внешнего представления объектов.

3.4.2. Что требуется от разработчика. Разработчик на этапе генерации задает, во-первых, внешнее представление объектов модели и аргументов команд на основе предлагаемой библиотеки (состоящей из графических представлений базовых объектов), во-вторых, соответствие прикладных функций и команд системы, основанных на выполнении этих функций, и описание типов аргументов функций, в-третьих, дисциплину диалога. Под дисциплиной диалога в нашей системе понимается следующее: а) объединение команд в подменю, для которых автоматически выполняются заданные разработчиком инициализирующие или терминирующие подпрограммы; б) задание условий динамического запрета и разрешения выполнения команд.

Все эти данные задаются средствами Системы диалога.

3.5. *Модули диалоговой компоненты.* Диалоговая компонента состоит из нескольких модулей, каждый из которых участвует в конечной системе, выполняя определяемую ниже роль. Генерация конечной системы состоит в задании для каждого из модулей данных, определяющих его поведение в этой системе.

3.5.1. Функциональное назначение модулей диалоговой компоненты. 3.5.1.1. Управление диалогом. Управление диалогом — модуль, организующий работу остальных модулей диалоговой компоненты и прикладных функций. Он формирует текущий список доступных команд, осуществляет выполнение команд, снабжает внутренние функции, реализующие команды, значениями параметров нужных типов, ведет глобальный контроль ошибок ввода и выполнения (обработку ошибочных ситуаций, зарегистрированных другими компонентами).

3.5.1.2. Интерпретатор. Интерпретатор — модуль, осуществляющий ввод с эхоотображением и вывод имен команд и значений аргументов при работе конечной системы. Интерпретатор связывает внешнее представление этих данных для конечного пользователя (графическое представление и ввод с внешних устройств) с внутренним представлением, принятым в системе и используемым функциями системы. Кроме того, интерпретатор управляет различными формами внешнего представления команд и аргументов (например, вводом с разных устройств).

3.5.1.3. Визуализатор. Визуализатор — модуль, обеспечивающий изображение геометрической модели на экране, а также селекцию и индикацию геометрических объектов.

3.5.2. Описание данных. В этом разделе рассматриваются модули диалоговой компоненты с точки зрения тех данных, с которыми они работают.

3.5.2.1. Данные управления диалогом. Данные управления диалогом, задаваемые при генерации,— это описания внутренних, в частности прикладных, функций системы; выражение команд через внутренние функции; данные, определяющие дисциплину диалога.

Внутренняя функция — это предписание, внутри которого нет обращений к системе диалога; такими являются все прикладные функции. Задаваемое при генерации описание внутренних функций включает разделение их параметров на входные-выходные, а также описание типов параметров (структуру их внутреннего представления и соответствие с запросами интерпретатору на ввод-вывод).

Команда описывается как вызов одной функции или нескольких функций, связанных управляющими конструкциями. На порядок выполнения функций влияют коды возврата. Эти коды во время работы конечной системы поступают к управлению диалогом от прикладных функций и функций ввода (интерпретатора). При генерации описываются правила передачи параметров от одной функции (внутри одной или последовательно заказываемых разных команд) к другой.

Перечислим данные управления диалогом, заполняемые во время работы с конечной системой. Во-первых, это значения параметров функций. Эти значения требуются для передачи от одних функций к другим и для протоколирования сеанса. Во-вторых, это данные, определяющие условия динамического запрета команд.

3.5.2.2. Данные интерпретатора. Это, во-первых, данные, формируемые на этапе генерации системы (внешнее представление аргументов и команд на основе библиотеки базовых внешних представлений). Во-вторых, эта база данных диалога, заполняемая при работе конечной системы, в которой хранятся объекты — аргументы функций, не являющиеся объектами модели.

3.5.2.3. Данные визуализатора. Визуализационная база данных состоит из данных, необходимых для изображения и идентификации при указании объектов модели и вводе данных диалога.

Передача информации между прикладной моделью и визуализационной базой данных осуществляется на основе процедур, написанных разработчиком на этапе генерации по определенным правилам. В начале работы конечной системы визуализатор, пользуясь этими предписаниями, заполняет свою базу данных ссылками на геометрическую информацию об объектах введенной модели, а в процессе работы корректирует эту базу, приводя в соответствие с моделью в случае редакции последней.

Кроме геометрической информации, в визуализационной базе данных содержатся графические атрибуты отдельных объектов, распределение объектов по сегментам и окнам, атрибуты сегментов и окон.

3.5.3. Связь диалоговой и прикладной компонент. В п. 3.5.3.1 описывается связь компонент в конечной системе по функциям, в 3.5.3.2 — по данным.

3.5.3.1. Выполнение прикладных функций. Выполнение прикладной функции во время работы системы инициируется управлением диалогом,

при этом функции передаются необходимые параметры. Прикладная функция возвращает управлению диалогом выходные параметры, в частности имена вновь созданных объектов модели, а также код завершения.

3.5.3.2. Визуализация прикладной модели. Визуализационная база данных имеет стандартный интерфейс с основной моделью. Наличие этого интерфейса позволяет при изменении модели получить новое изображение и при указании объекта на экране — имя этого объекта, которое затем передается прикладной функции.

3.5.4. Взаимодействие компонент в конечной системе. Конечный пользователь выбирает команду с помощью предназначенного для выбора диалогового примитива (например, «Графическое меню» или «Ввод команд с клавиатуры»). Для выполнения какой-либо внутренней функции, входящей в состав команды, управление диалогом, если требуется пользовательский ввод аргументов, обращается к интерпретатору. Затем управление диалогом вызывает внутреннюю функцию, передавая ей в качестве параметров введенные аргументы. Функция выполняется, и возвращаются выходные параметры к управлению диалогом. Если при этом требуется вывод на экран, то управление диалогом обращается к интерпретатору. Вновь полученные геометрические объекты на изображении выделяются, например цветом.

В промежутках между этими действиями управление диалогом формирует список допустимых команд, выполняет процедуры, которые обязательны до или после выполняемой команды, а также запоминает идентификаторы выполненных функций и значения их параметров, если ведется функциональный протокол.

Заключение. Описанная Система диалога имеет следующие характерные черты. Управление конечной системой возлагается на диалоговую компоненту; кроме того, прикладные функции лишены доступа к вводу-выводу. Такое решение выбрано из соображений удобства для разработчика, имеющего возможность отладки прикладных модулей вне системы.

Предлагаемая схема допускает макетирование конечной системы. В макете системы интерфейс разработан во всех деталях, а прикладной уровень заменен имитирующими подпрограммами.

В конечной системе пользователь после выбора команды получает от интерпретатора запросы на ввод необходимых аргументов. Порядок ввода аргументов фиксируется при генерации системы. Это накладывает определенные ограничения на стиль диалога. В систему несложно добавить команды, выполняемые в любой момент при нажатии определенных кнопок,— так называемые «hot keys», что делает диалог более гибким.

В системе явно предлагаются средства выделения полученных командами результатов, тогда как разработчики UIMS обычно основное внимание уделяют только обработке пользовательского ввода. Индикация полученных аргументов особенно существенна для конечного пользователя, если он имеет дело со сложным графическим представлением модели.

Не вызывает сомнений, что в настоящее время требуются исследования в области разработки UIMS с целью определения основных понятий и в перспективе выработки стандарта UIMS.

СПИСОК ЛИТЕРАТУРЫ

1. **Kasik D. J.** A user interface management system: Proc. SIGGRAPH'82 // Comput. Graph.— 1982.— **16**, N 3.— P. 99.
2. **Prime M.** User interface management systems — a current product review // Proc. Conf. EUROGRAPHICS'89, 1989.
3. **Proceedings** of the ACM SIGGRAPH Symposium on User Interface Software.— Banff, Alberta: ACM PRESS, 1988.

Поступила в редакцию 16 января 1990 г.