Лекция № 4. Разработка структуры диалога.

Обмен информацией между пользователем и компьютером (точнее, его программным обеспечением) по всем формальным признакам соответствует понятию «диалог». Для того чтобы диалог был конструктивным, должны соблюдаться следующие *правила*:

- участники диалога должны понимать язык друг друга;
- участники диалога не должны говорить одновременно;
- очередное высказывание должно учитывать как общий контекст диалога, так и последнюю информацию, полученную от собеседника.

Если собеседники обсуждают вопросы, относящиеся к какой-либо специальной области, они должны придерживаться единой терминологии; при объяснении следует сначала пояснить основные термины и понятия.

Очень краткие ответы и слишком большие паузы могут сбить собеседника с толку, а злоупотребление специальными терминами или жаргонизмами вообще может привести к преждевременному завершению беседы.

Перечисленные выше факторы существенно влияют на структуру диалога, т.е. на форму общения.

Т.о. при проектировании диалога, необходимо определить:

- вид диалога;
- структуру диалога;
- возможный сценарий развития диалога;
- содержание управляющих сообщений и данных, которыми могут обмениваться человек и приложение (семантику сообщений);
- временные характеристики, требования к темпу ведения диалога;
- визуальные атрибуты отображаемой информации (синтаксис сообщений).

Вид диалога.

В зависимости от того, кому принадлежит ведущая роль в диалоге, различают два вида диалога: диалог, управляемый системой и диалог, управляемый пользователем.

Диалог, управляемый системой – это диалог, в котором процесс жестко задает, какое задание можно выбрать и какие данные вводить.

Диалог, управляемый пользователем – это диалог, в котором инициатива принадлежит пользователю, т.е. он непосредственно подает команду на выполнение нужного на данном этапе задания.

Диалог, управляемый системой, более удобен, потому что он лучше подстраивается под пользователя, но при этом имеет больше ограничений, чем диалог, управляемый пользователем. Предполагается также, что задания структурированы (обычно иерархически) и диалог ведется в соответствии с этой структурой.

Выбор структуры диалога.

Рассмотренные ниже четыре варианта структуры диалога являются разновидностями структуры типа «вопрос-ответ», тем не менее, каждая из них имеет свои особенности и наиболее удобна для определенного класса задач.

Диалог типа «вопрос-ответ».

Структура диалога типа «вопрос-ответ» (Q&A) основана на аналогии с обычным интервью. Система берет на себя роль интервьюера и получает информацию от пользователя в виде ответов на вопросы. Это наиболее известная структура диалога; все диалоги, управляемые компьютером, в той или иной степени состоят из вопросов, на которые пользователь отвечает, однако в структуре Q&A этот процесс выражен явно.

В каждой точке диалога система выводит в качестве подсказки один вопрос, на который пользователь дает один ответ. В зависимости от полученного ответа система может решить, какой следующий вопрос задавать.

Структура Q&A предоставляет естественный механизм ввода как управляющих сообщений (команд), так и данных. Никаких ограничений на диапазон или тип входных данных, которые могут обрабатываться, не накладывается.

Эта структура может удовлетворить требования различных пользователей и типов данных. Такая структура особенно уместна при реализации диалога с множеством ответвлений (в тех случаях, когда на каждый вопрос предусматривается большое число ответов, каждый из которых влияет на то, какой вопрос будет задан следующим). По этой причине, структура Q&A часто используется в экспертных системах.

Данная структура имеет ряд существенных недостатков. Во-первых, не гарантирует минимального объема ввода, оцениваемого по количеству нажатий клавиш; во-вторых, возможны проблемы с анализом и интерпретацией вводимых данных; в-третьих, процедура ввода ответов набором их с клавиатуры достаточно утомительна для пользователя.

Диалог на основе меню.

Меню является наиболее популярным вариантом организации запросов на ввод данных во время диалога, управляемого компьютером.

Существует несколько основных форматов представления меню на экране:

- список объектов, выбираемых прямым указанием, либо указанием номера (или мнемонического кода);
- меню в виде блока данных;
- меню в виде строки данных;
- меню в виде пиктограмм.

Пользователь диалогового меню может выбрать нужный пункт, вводя текстовую строку, которая идентифицирует этот пункт, указывая на него непосредственно или просматривая список и выбирая из него. Система может выводить пункты меню последовательно, при этом пользователь выбирает нужный ему пункт нажатием клавиши.

Меню *в виде строки данных* (Menu bar, Main menu) может появляться вверху или внизу экрана и часто остается в этой позиции на протяжении всего диалога. Таким образом, посредством меню удобно отображать возможные варианты данных для ввода, доступных в любое время работы с системой.

Если в системе имеется достаточно большое разнообразие вариантов действий, организуется иерархическая структура из соответствующих меню.

Дополнительные меню *в виде блоков данных* (PopUp menu) «всплывают» на экране в позиции, определяемой текущим положением указателя, либо «выпадают» (PullDown Menu) непосредственно из строки меню верхнего уровня. Эти меню исчезают после выбора варианта.

Меню *в виде пиктограмм* представляет собой множество объектов выбора, разбросанных по всему экрану; часто объекты выбора содержат графическое представление вариантов работы.

Меню можно с равным успехом применять для ввода как управляющих сообщений, так и данных. Приемлемая структура меню зависит от его размера и организации, от способа выбора пунктов меню и реальной потребности пользователя в поддержке со стороны меню.

Структура типа меню является наиболее естественным механизмом для работы с устройствами указания и выбора: меню представляет собой изображение тех объектов, которые выбираются пользователем. Если диалог состоит исключительно из меню, можно реализовать последовательный интерфейс, при котором пользователь применяет только устройства для указания; однако такое постоянство редко достижимо на практике. Следует отметить, что хотя работа с этими устройствами не требует профессионального

владения клавиатурой, для подготовленного пользователя это не самый быстрый способ выбора из меню. Вместо указания пользователь может сообщить о своем выборе вводом соответствующего идентификатора.

Меню – это наиболее удобная структура диалога для неподготовленных пользователей; жесткая очередность открытия и иерархическая вложенность меню может вызывать раздражение профессионала, замедлять его работу. Традиционная структура меню недостаточно гибка и не в полной мере согласуется с методами адаптации диалога, такими, например, как опережающий ввод, с помощью которого можно ускорить темп работы подготовленного пользователя.

Диалог на основе экранных форм.

Как структура типа «вопрос-ответ», так и структура типа меню предполагают обработку на каждом шаге диалога единственного ответа. Диалог на основе экранных форм допускает обработку на одном шаге диалога нескольких ответов.

На практике формы используются там, где учет какой-либо деятельности требует ввода стандартного набора данных. Человек работает с формой до тех пор, пока не заполнит ее полностью и не передаст системе. Система может проверять каждый ответ непосредственно при вводе или по окончании заполнения всей формы.

Сообщения об ошибках, выводимые непосредственно после ответа, могут отвлечь внимание, но могут оказать и положительное влияние. В тех случаях, когда информация для ввода выбирается из некоторого целостного документа, проверку лучше отложить до конца заполнения формы, чтобы не прерывать процесс ввода; если же такой целостности нет, то проверку следует выполнять сразу после ввода ответа (после заполнения очередного поля).

Если встретилась какая-либо ошибка, приложение не должно заново выводить пустую форму; выводится форма с предыдущими ответами и допущенными ошибками. Новый «бланк» выдается лишь в случае соответствующего запроса пользователя.

Такую структуру уместно применять там, где источником данных служит существующая входная («бумажная») форма документа.

Не обязательно, чтобы внешний вид этих форм совпадал, но все вводимые элементы данных должны располагаться в том же относительном порядке и иметь такой же формат, что и в исходном документе.

Часто все необходимые единицы ввода нельзя отобразить одновременно в пределах одного экрана (или окна), и их необходимо разделить на группы, которые отображаются на последовательности экранов (окон). Важно, чтобы это разбиение сохраняло логические связи и не приводило к разделению связанных частей документа.

Структура диалога на основе экранной формы обеспечивает высокий уровень поддержки пользователя: для каждой формы могут быть предусмотрены сообщения об ошибках и справочная информация. Пользователю можно также оказать помощь, включив некоторые элементы формата ответа в вопрос или в поле ответа.

Эта структура позволяет повысить скорость ввода данных по сравнению со структурой типа «вопрос-ответ» и манипулировать более широким диапазоном входных данных, нежели меню; кроме того, с ней могут работать пользователи любой квалификации.

Поскольку эта структура имеет последовательную, а не древовидную организацию, она в меньшей степени подходит для работы в режиме выбора вариантов.

Еще одной областью применения экранных форм является задание параметров запроса в базах данных. Этот механизм иногда называют запросом по образцу (Query by Example).

Одним из типов заполнения форм являются также многовариантные меню. В таких меню пользователю предоставляется список вариантов, и он не ограничен возможностью единственного выбора; можно указать несколько вариантов.

Диалог на основе командного языка.

Структура диалога на основе командного языка столь же распространена, что и структура типа меню. Она очень часто используется в операционных системах. Исторически это первая из реализованных структур диалога.

При такой организации диалога система не выводит ничего, кроме постоянной подсказки (приглашения на ввод команды), которая означает готовность системы к работе. Каждую команду вводят с новой строки и обычно заканчивают нажатием клавиши «ввод». Ответственность за правильность задаваемых команд ложится на пользователя. Система информирует о невозможности выполнения неверной команды, не поясняя, как правило, причин.

Подобно меню, диалог на базе команд удобен для ввода управляющих сообщений, однако он обеспечивает более широкие возможности выбора в любой точке диалога и не требует иерархической организации обслуживающих его программ.

Программная система может поддерживать достаточно большое количество команд, но на практике следует ограничивать их число, чтобы не перегружать память пользователя.

Структура на базе командного языка не отличается хорошей поддержкой пользователя и пригодна в основном для подготовленных специалистов. Более того, поскольку системе неизвестно, что намеревается делать пользователь, трудно представить какую-либо реальную помощь в процессе работы, кроме выдачи справок общего характера.

Поскольку данная структура предполагает большой объем запоминаемого материала, имена команд следует выбирать так, чтобы они несли смысловую нагрузку и легко запоминались.

Диалог должен управлять данными. В интерфейсах на основе командного языка это достигается с помощью составных командных строк, где команда предшествует списку параметров (входным данным). Параметры в списке можно задавать в одной из двух форм – позиционной и ключевой.

Назначение *позиционного параметра* определяется по его месту в командной строке. Позиционные параметры уменьшают объем вводимой информации, но их существенным недостатком является то, что вводимые значения должны указываться в строго определенном порядке, нарушение которого может повлечь непредсказуемые последствия. Задание позиционных параметров осложняется, если их список достаточно велик. Этот недостаток стремятся компенсировать путем пропуска неизменяемых параметров, вводя два разделителя друг за другом.

В случае *ключевых параметров* каждое значение предваряется определенным идентификатором, который определяет его назначение. Ключевые параметры уменьшают нагрузку на память пользователя в том отношении, что отпадает необходимость в запоминании порядка их следования; можно опускать необязательные параметры. Недостатком является то, что пользователю приходится запоминать множество ключевых слов, а разработчику – подбирать для них «осмысленные» имена. Этот подход требует большего времени работы системы, чтобы распознать ключевые слова, заданные в произвольном порядке.

Структура на основе языка команд по своим возможностям самая быстрая и гибкая из всех структур диалога. Большинство пользовательских интерфейсов на базе естественного языка реализуется с помощью языков команд с очень большим набором ключевых слов.

Подготовленный пользователь испытывает удовольствие от ощущения того, что он управляет системой, а не наоборот. Однако эта структура не обеспечивает пользователя поддержкой, поэтому даже подготовленные пользователи считают, что очень сложно использовать все заложенные в ней возможности. Большинство пользователей хорошо

знакомы только с весьма ограниченным набором средств, с которыми работают регулярно.

Для выбора подходящей структуры диалога пользуются таблицами выбора. Рассмотрим одну из таких таблиц. Использовать ее можно как для выбора оптимального типа диалога, так и для проверки соответствия выбранного типа диалога рассматриваемым критериям.

Выбор подходящей структуры диалога на основе таблицы выполняется следующим образом.

1. Закрыть графы «Тип диалога».
2. В графе «Выбор пользователя» пометить критерии, относящиеся к рассматриваемому применению.
3. Для каждого типа диалога подсчитать число случаев, когда помечены соответствующие пункты в графе «выбор пользователя», и в графе «тип диалога».
4. Подсчитать число совпадений для каждого типа диалога.

В таблице знак * означает, что использование этого типа диалога данной категорией пользователей требует наличия системы помощи; знак ** означает, что использование средств системы возможно только в ограниченном объеме.

Критерии	Выбор пользователя	Тип диалога			
		Меню	Вопрос-ответ	Язык команд	Экранные формы
Цель:					
Запрос		+	+	+	+
Вычисления		+	+	+	+
Сложный выбор			+	+	
Ввод данных			+	+	
Ввод данных (большой объем)		+	+	+	+
Тип пользователя:					
Программист				+	+
Непрограммист с опытом работы		+	+	+	+
Непрограммист без опыта работы		+	+	*	*
Время обучения:					
Очень малое		+	+		
Менее 1 дня		+	+	**	**
Более 1 дня				+	+
Результат оценки					

При работе с таблицей, если предполагается, что одни пункты боле важны, чем другие, можно брать их с разными весовыми коэффициентами. Можно также указать, какие пункты должны рассматриваться как выполняемые безусловно; типы диалога, не соответствующие хотя бы одному из таких пунктов, должны немедленно отвергаться, сколько бы очков они ни набрали по остальным пунктам.

Разработка сценария диалога.

Развитие диалога во времени можно рассматривать как последовательность переходов системы из одного состояния в другое. Ни одно из этих состояний не должно быть тупиковым, т.е. пользователь должен иметь возможность перейти из любого текущего состояния диалога в требуемое (за один или несколько шагов). Для этого в ходе разработки интерфейса необходимо определить все возможные состояния диалога и пути перехода из одного состояния в другое – разработать *сценарий диалога*.

Цели разработки сценария:

- выявление и устранение возможных тупиковых ситуаций в ходе развития диалога;
- выбор рациональных путей перехода из одного состояния диалога в другое (из текущего в требуемое);
- выявление неоднозначных ситуаций, требующих оказания дополнительной помощи пользователю.

Сложность разработки сценария определяется в основном двумя факторами – функциональными возможностями создаваемого приложения (т.е. числом и сложностью реализуемых функций обработки информации) и степенью неопределенности возможных действий пользователя.

Степень неопределенности действий пользователя зависит от выбранной структуры диалога. Наибольшей детерминированностью обладает диалог на основе меню, наименьшей – диалог типа «вопрос-ответ», управляемый пользователем.

Таким образом, сценарий диалога можно упростить, снизив степень неопределенности действий пользователя. Возможные способы решения этой задачи:

- использование смешанной структуры диалога (применение меню с целью «ограничения свободы» пользователя там, где это возможно);
- применение входного контроля вводимой информации (команд и данных).

Дополнительные возможности по снижению неопределенности действий пользователя предоставляет объектно-ориентированный подход к разработке интерфейса, при котором для каждого объекта заранее устанавливается перечень свойств и допустимых операций. Наиболее эффективен такой подход при создании графического интерфейса.

Сокращая число возможных состояний диалога, разработчик должен помнить о необходимости отражения в его сценарии работы средств поддержки пользователя (что усложняет сценарий).

Способ описания сценария диалога зависит от степени его сложности. Методы описания сценариев делятся на две группы: неформальные и формальные методы.

Главное достоинство формальных методов состоит в том, что они позволяют автоматизировать как проектирование диалога, так и его модификацию (адаптацию) в соответствии с характеристиками пользователя.

В настоящее время наиболее широко используются формальные методы описания сценариев на основе сетей Петри и их расширений, а также на основе систем представления знаний (фреймовые модели и продукционные системы).

Независимо от способа описания сценария его основной структурной единицей является *шаг диалога*, соответствующий одному акту взаимодействия пользователя с системой (рис.1).

Сценарий диалога позволяет описать процесс взаимодействия пользователя с приложением на уровне решаемой им прикладной задачи. Однако для программной реализации интерфейса такое описание носит слишком общий характер. Поэтому на этапе реализации необходимо перейти на уровень описания соответствующих процессов с помощью используемых инструментальных средств разработки приложения.

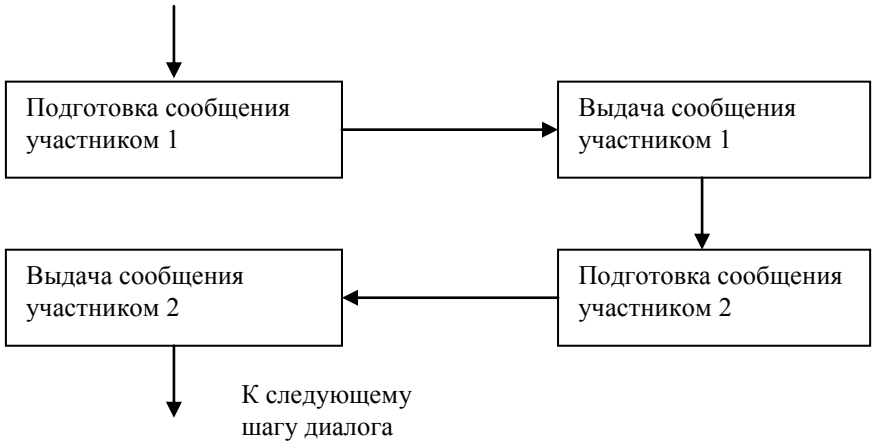


Рисунок 1 – шаг диалога

Описание структуры диалога с помощью сети переходов.

Развитие диалога можно рассматривать как последовательность переходов от одного состояния к другому. Диалог может находиться в особом состоянии ожидания ввода от пользователя и будет переходить в одно из нескольких возможных состояний в зависимости от характера принятой информации. В соответствии с этим диалог можно представить в виде сети переходов или диаграммы состояний. Каждое состояние представляется вершиной графа, помеченной соответствующим ей номером. Будем рассматривать вершину как некоторую точку, в которой диалог выводит сообщение пользователю или требует входного сообщения от пользователя. Связи между вершинами обозначаются направленными дугами, соединяющими две вершины, метка на дуге определяет условие, при выполнении которого возможен переход. Может существовать несколько дуг, соединяющих две вершины и определяющих, что переход может быть вызван несколькими условиями.

Выделяют три типа вершин.

1. Вершина, в которой выводится сообщение пользователю с запросом на ввод. Передача на соседнюю вершину зависит от контекста введенного сообщения.
2. Вершина, в которой выводится сообщение пользователю без запроса на входное сообщение; следует автоматическая передача на соседнюю вершину.
3. Вершина, в которой выводится сообщение пользователю с запросом на ввод, после которого осуществляется безусловный переход на соседнюю вершину.

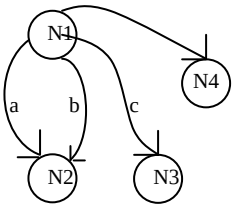
Рассмотрим различия между вершинами типа 2 и 3. В типе 2 переход осуществляется автоматически, т.е. не требуется никакого входного сообщения. В типе 3 перехода не произойдет до тех пор, пока не будет сделан ввод, но, независимо от контекста введенных данных, будет иметь место один и тот же переход.

Каждая вершина в сети переходов представляет собой отдельное состояние диалога. Она образует точку переключения в развитии диалога. Переключением можно управлять придерживаясь структуры, в которой устанавливается соответствие между назначением следующей вершины и каждым элементом из набора условий.

Условие	Следующая вершина
C(1) C(2) ….. C(k)	N(1) N(2) …… N(k) N(k+1)

Каждому условию C(i) соответствует вершина N(i), в которую должен быть осуществлен переход. Если никакое из условий не выполнено, то переход произойдет в вершину N(k+1) (например, в случае ошибки). Такое же обозначение используется и для безусловного перехода. Рассмотрим примеры.

Условие	Следующая вершина
	N2



Условие	Следующая вершина
a b c	N2 N2 N3 N4

Примеры графа и соответствующей таблицы переходов содержится указаниях к выполнению практической работы (ИДЗ).