Идеационные дискурсные функции и структура задачи при многоязыковой генерации текстов инструкций

Соколова Е.Г., Болдасов М.В.

sokolova@aha.ru boldasov@mail.ru

Аннотация

Основным вопросом при построении системы многоязыковой генерации текстов является вопрос о языковой независимости исходного представления текста. Часто под представлением содержания текста подразумевается семантическое или смысловое языковообусловленное представление. В статье на материале проекта AGILE(пользовательские инструкции к программе графического редактора) предварительно обсуждаются элементы структуры дискурса и их языковая реализация. Предлагается рассмотреть понятие идеационная дискурсная функция и модель стандартизованного интерфейса в качестве возможных средств обеспечения языковой независимости исходного представления и качественного выходного текста при многоязыковой генерации.

Введение

В исследованиях по структуре текста и моделях генерации текста вопросы моделирования контекста и представления содержания текста рассматриваются отдельно друг от друга. Контекст моделируется в виде сценариев, фреймов, риторических принципов. С другой стороны содержание текста моделируется в виде древесной структуры, в узлах которой находятся понятия. Понятия и отношения между ними составляют формальный язык представления содержания будущего текста. При генерации текста на одном языке элементы этого формального языка оказываются по объему и свойствам чрезвычайно близкими словам и пропозициям выходного естественного языка (ЕЯ). Тексты инструкций моделируются в виде структуры задачи, в узлах которой находятся пропозиции, связывающие понятия предметной области (Moore & Paris 1994), (Hartley & Paris 1996).

В системе AGILE также используется формальная спецификация – структура задачи (Task Structure), которая разбивает текст инструкции на цели и наборы методов по их решению, состоящие из последовательностей шагов (Power 1998), (Kruijff и др. 1999). Это представление приближено к структуре софтверного руководства, которое описывает действия пользователя по достижению цели, например, создания графического объекта конкретного типа с помощью конкретных программных средств в конкретной операционной системе. Содержание этого текста представляет собой целеполагание и действия пользователя по активизации функций объектов, созданию графических объектов (указание параметров графического объекта), дезактивации объектов, изменению параметров объектов графического интерфейса и даже действия, которые не отражаются на экранных объектах – например, сохранить

объект в файле, распечатать некоторое графическое отображение объекта, создать|уничтожить, активировать| дезактивировать диалоговые окна, палитры и т.д.). Пример фрагмента текста инструкции, генерируемого в системе AGILE:

Создание стиля мультилинии

Откройте диалоговое окно Multiline Styles одним из следующих способов:

Windows В панели инструментов Object Properties или в меню Data выберите пункт Multiline Style.

DOS uUNIX В меню Data выберите пункт Multiline Style

1 Нажмите кнопку Element Properties, чтобы добавить элементы в стиль.

2 В диалоговом окне Element Properties введите смещение первого элемента линии.

3 Нажмите кнопку Add, чтобы добавить этот элемент.

4 Выберите пункт Color. Затем выберите цвет элемента в диалоговом окне Select Color.

5 Выберите пункт Linetype… .

В многоязыковой системе генерации текстов инструкций AGILE понятия формального языка подчиняются двум онтологиям. С одной стороны, они классифицируются в соответствии с некоторыми понятиями предметной области, например, “экранный объект”, “графический объект”, “данные-объект”, с другой стороны, они включаются в языковую онтологию Upper Model (Bateman и др. 1995) и отражают преимущественно свойства слов и грамматических конструкций английского языка. В связи с ориентацией системы на генерацию текстов языковая онтология оказывается ведущей и возникают противоречия между различными способами концептуализации понятий в выбранном формальном языке и в выходных ЕЯ (болгарский, русский, чешский). В статье предлагается новый теоретический концепт – идеационная дискурсная функция (ИДФ), который по мнению авторов позволит развить онтологию понятий предметной области и в будущем рассмотреть возможность моделирования полностью языконезависимого исходного представления для генерации текстов.

Мы также представляем часть предметной области, связанную с пользовательским интерфейсом в виде фрейма. Фрейм позволяет объяснить последовательность команд в структуре задачи. Наконец мы сопоставляем анализ содержания текста инструкции в терминах ИДФ и стандартные способы выражения стандартных действий, задаваемых шаблонами.

2. Составляющие идеационного дискурса и программный интерфейс с пользователем

Термином “дискурс” обычно обозначается связный текст в совокупности с экстралингвистическими (прагматическими, социокультурными, психологическими и др.) факторами, взятый в событийном аспекте и рассматриваемый как целенаправленное социальное действие во взаимодействии людей и механизмах их сознания, “жизненный контекст” (Лингвистический энциклопедический словарь 1990). В соответствии с традициями системно функциональной грамматики мы рассматриваем дискурс как композиционное образование, в этом случае текст является результатом взаимодействия дискурсных составляющих различной природы.

Мы основываемся на системно функциональной лингвистике Хэллидэя (Halliday 1985), которая является теоретической базой системы AGILE, см. (AGILE Project 1997), (Соколова, Шаров 1988) и определяем область дискурса, которую, пользуясь терминологией Хэллидэя, называем идеационной. Эта область отображает реальные объекты, вовлеченные в дискурс инструкций и изменения их статуса в процессе развития текста, т.е. то, что относится к идеационной метафункции по Хэллидэю. Мы отличаем от нее коммуникативную ситуацию и стороны, связанные со взаимодействием коммуникантов, которая относится к интерперсональной метафункции. Мы иллюстрируем изложение примерами из русского и английского языков и рассматриваем грамматические явления этих языков также пользуясь терминологией Хэллидэя.

Границы понятий идеационного дискурса определяются объектами, вовлеченными в деятельность по использованию графического редактора, и изменениями этих объектов. Область идеационного дискурса отражает объективно существующие объекты и их превращения независимо от других дискурсных составляющих, в частности, от наличия действующего объекта – пользователя, который является элементом интерперсональной составляющей. В зависимости от природы вовлеченных объектов идеационная составляющая дискурса в свою очередь разделяется на непересекающиеся области. Классификация объектов основывается их свойствах, определенных в предметной области, с учетом их отражения при языковой реализации.

(1) Область графических объектов. Целью использования графического редактора всегда является создание графического объекта – фигуры, чертежа, рисунка. Графические объекты состоят из элементов, которые также являются графическими объектами. Структура данной области могла бы быть представлена описанием фигур и элементов, их составляющих, т.е. является энциклопедической (знания о строении графических объектов). Это знание косвенно отражается в строении текста инструкции, например, цель - *Задание стиля мультилилинии* развертывается в последовательность действий по определению графических свойств линий, составляющих данную мультилинию, т.е. предполагает знание о том, что мультилиния состоит из нескольких параллельных линий, имеющих одну и ту же конфигурацию. Мы не рассматриваем структуры графических объектов, ограничиваясь их классификацией, несколько более подробной, чем в системе AGILE.

Данные (Data) – представляет один из следующих объектов предметной области: Графический объект (GO – Graphical Object) – *линия, фигура, рисунок*; Графический параметр (GP – Graphical Parameter) – параметр, который задается показом на экране - *цвет*; Символьный параметр (SP - Symbolic Parameter) – параметр, который задается в виде символов, букв и цифр - *угол, имя;* Сложный параметр (CP – Complex Parameter)– *стиль (линии)*,

(2) Область объектов интерфейса. Сам графический редактор являет собой средство реализации цели, т.е. создания графического объекта. В тексте инструкции представлен интерфейс редактора с пользователем. Классификация объектов пользовательского интерфейса связана с функциями, которые данные объекты выполняют, например, открытие диалогового окна, подтверждение выбора параметра или завершение редактирования параметров и т.п.

Функциональный объект (FO – Functional Object) – элемент пользовательского интерфейса, реализующий конкретную функцию в пользовательском интерфейсе. Это либо символические (SFO – Symbolic Functional Object) - набор символов в управляющей строке, например, “л” – переход в режим рисования линиии. Либо экранные объекты - кнопки в диалоговом окне (BFO – Button Functional Object). Мы указали те, которые используются модели предметной области AGILE. Другие виды функциональных объектов перечислены в таблице .

Функциональные объекты всегда принадлежат определенному рабочему пространству (WS - Working Space) – это либо принадлежность определенному меню, показываемому системой в диалоговом окне, либо определенному программному объекту (ST - Software Tool) – *система, режим.*

(3) Компьютерная среда (физическая среда). Манипулятор мышь, экран монитора, клавиатура являются объектами, составляющими физическую среду компьютера. Они обеспечивают пользователю возможность манипулирования функциональными объектами.

3. Дискурсные идеационные функции

Идеационный дискурс представлен объектами реальность которых оправдывается либо энциклопедическими данными (типы графических объектов и их строение), либо свойствами компьютерной среды, либо конкретной программой, в нашем случае – графическим редактором. Основная цель введения ИДФ состоит в том, чтобы освободить узлы структуры задачи от языковоориентированных понятий. ИДФ обозначает процесс, но, в отличие от языкового предиката, не описывает действие в физическом мире, а определяет только изменение статуса объекта в системе понятий идеационного дискурса данной предметной области.

При выражении простой ИДФ языковым предикатом конструкция “функция + объект” может рассматриваться как ситуация с идеационным актантом или как ситуация с двумя актантами – идеационным и взятым из коммуникативной ситуации “интерперсональным” актантом – пользователем.

Во втором случае пользователь является адресатом в модели коммуникативной ситуации, что выражается эксплицитно в предложении в форме императива и одновременно является действующим лицом или каузатором процесса (Соколова 1999), что выражается в классификации процесса как DIRECTED-ACTION (Bateman и др. 1995) и выборе лексемы с соответствующими свойствами, например, *Откройте диалоговое окно Multiline styles*. *Open the dialog box Multiline styles.* В этом случае используется двухролевая грамматическая модель и объект оказывается грамматически оформленным на русском и английском языках

в роли пассивного участника – Direct complement, а процесс классифицируется как User-action.

В первом случае модель коммуникативной ситуации сохраняется ибо она является общей для текста инструкции, пользователь является адресатом. Но она уже не выражается эксплицитно языковыми средствами в предложении (индикатив). Кроме того актуализованное значение языкового предиката остается в рамках идеационного дискурса, пользователь как действующее лицо в смысл предиката не втягивается. В результате используется одноролевая грамматическая модель, в которой роль актанта в русском и английском языках выражается подлежащим, например, *На экране появится диалоговое окно Multiline styles. The dialog box Multiline styles appears.* Процесс классифицируется как NONDIRECTED-HAPPENING (Bateman и др. 1995). Выбирается лексема с соответствующими свойствами (непереходный гланол). В том случае, если процесс реализуется двуролевым глаголом, роль каузатора “гасится” средствами грамматической конструкции, например, *Откроется диалоговое окно Multiline styles*.*The dialog box Multiline styles opens.* В русском используется возвратная форма, не предполагающая указания на каузатора в творительном падеже, в английском лабильный глагол *to open* используется в активной конструкции. В обоих случаях объект оказывается грамматически оформленным в роли активного участника – Subject, а процесс может быть классифицирован как Object-function.

ИДФ являются отражением реальных изменений в предметной области, не являясь при этом прямым отражением каких-то элементарных действий, например, программных функций системы, сохраняя масштаб языковых единиц. Они не являются толкованиями лексем, т.е. не рассматриваются как языковые значения. Но мы настаиваем на их освобождении от коммуникативной перспективы и антропоцентрических свойств.

ИДФ, таким образом, отличаются и от лексических функций Жолковского-Мульчука (Жолковский, Мельчук 1965) по многим параметрам. Основным является следующий - ИДФ входят в систему отношений между состояниями объектов предметной области и не имеют отношения к закрепленным в языке структурным (синтаксическим) способам кодирования ситуации. Лексические функции, наоборот, принадлежат системе языка. В том случае, если содержание ИДФ связано с общими понятиями бытия или функциональности, можно представить, что ИДФ эквивалентна формуле лексической функции, например, допустим, FOActivation = CausFunc0 (кнопка), но ее лексическое выражение принадлежит терминологической системе данной предметной области. Например, если “нажать (кнопка)” – может считаться значением CausFunc0 слова “кнопка” в русском языке, то выражение “выбрать кнопку” используется только в инструкциях компьютерных руководств. Кроме того, лексическая функция в данном случае передает синтаксическую конструкцию переходности действия, чего не делает ИДФ. Реализацией данной ИДФ могжет быть, например, *нахождение кнопки в активном состоянии.*

Мы не рассматриваем способа перехода от ИДФ к языковым предикатам, указывая только соответствия между ними для русского и английского языков из наших текстов. Мы не ставили задачу дать сколько-нибудь полный список ИДФ для софтверных инструкций, скорее мы хотим обсудить их в роли источника конкретноязыковой реализации для представлении содержания текста в системе многоязыковой генерации.

Ниже перечислено несколько ИДФ. Под чертой для каждой группы функций даются примеры их естественно языковой реализации на английском и русском языках. Языковая реализация ИДФ похожа на значение лексической функции, но зависит не от конкретного слова, а от понятия предметной области.

1. **DataActualisation** – Актуализация (осуществление) объекта типа Data , предполагает использование средств графического редактора для реализации.

Object involved 1(Data)

Object involved 2 (WS)

_____

USER-ACTION (CP) – *a multiline style, an element of style*, *стиль мультилинии, элемент стиля* - (En) *create, define* - (Ru) *создать, определить;*

USER-ACTION (GO) – *a line, arc combination polyline, a line segment* (En) *draw* - (Ru) *нарисовать;*

USER-ACTION (SP) – *the offset, an angle, the start point, the name of the style* (En) *enter, specify* - (Ru) *ввести, задать;*

USER-ACTION (GP) – *the element's color* (En) *select* - (Ru) *выбрать;*

Рабочее пространство (WS) очевидно является обязательным концептаульным слотом, но не всегда реализуется в тексте в связи с контекстной заданностью. В случае реализации в тексте WS оформляется как обстоятельство места: (En) *In the Element Properties dialog box,* (Ru) *в диалоговом окне Element Properties.*

2. DataOperation – операции с объектами, не изменяющие их параметров. Это группа функций, которые различаются между собой как типами вовлеченных объектов, так и характером взаимодействия. Мы не предлагаем их формализации, ограничиваясь несколькими примерами, иллюстрирующими участие объектов различных классов. Характер изменения не указывается.

Object involved 1 (Data):

Object involved 2 (CADCAMObject) – самый общий класс объектов предметной области графических редакторов.

_____

USER-ACTION (GP- (экран)) – *background colour* (En) *display(-on)* - (Ru) *показать;*

USER-ACTION (CP-Data) – *the style of the multiline element* (En) *save(-to)* - (Ru) *сохранить;*

USER-ACTION (CP-CP) – *element(s) to the style* (En) *add-to* - (Ru) *добавить;*

USER-ACTION (GO-GO) – *the arc (snaps) to the endpoint* (En) *snap-to-* (Ru) *привязать;*

USER-ACTION (GO-GP) – display a line at the vertices of the multiline (En) display-at - (Ru) отобразить; и т.д.

Формализация характера взаимодействия выходит за рамки данной статьи. Поэтому мы не будем подробно останавливаться на этом вопросе. Отметим лишь, что характер изменения

можно отображать как атрибут ИДФ мнемонично. Например, «добавление» может быть представлено значком «+».

3. FOActivation – функция предполагает физическое действие пользователя по воздействию на функциональный объект (FO), который назван по имени и находится в некоторым рабочем пространстве (WS).

Object involved 1 (FO)

Object involved 2 (WS)

USER-ACTION (SFO) – *a* (здесь обозначает функциональный объект, запускающий режим рисования дуги (arc mode)), *l* (line mode) (En) *enter* - (Ru) *ввести.* WS – командная строка.

USER-ACTION (BFO) - *OK* (En) *press, choose* - (Ru) *нажать.*

USER-ACTION (BFO) – *Add – кнопка Add*, *Linetype – пункт Linetype* (En) *choose-from, select-from* - (Ru) *выбрать-в. .* WS реализуется в тексте.

4. WSActivation – функция предполагает физическое действие пользователя по воздействию на функциональный объект (FO), который “запускает” данное рабочее пространство, названое по имени.

Object involved (WS или ST)

USER-ACTION (WS) – *the Multiline Styles dialog box* (En) *open* - (Ru) *открыть*

OBJECT-FUNCTION (WS) – *the Multiline Styles dialog box* - (En) *appear*,
(Ru) *появиться (на экране).*

USER-ACTION (ST)– *the PLINE command* (En) start (), (Ru) запустить

5. WSDesActivation –дезактиваация рабочего пространства, реализуется обычно как следствие последнего шага фрагмента инструкции.

Object involved (WS или ST)

USER-ACTION (WS)– *the Multiline Styles dialog box* (En) *close* - (Ru) *закрыть*

OBJECT-FUNCTION (WS) – *the Multiline Styles dialog box* - (En) *disappear*,
(Ru) *исчезнуть (с экрана).*

USER-ACTION (ST)– the PLINE command (En) abort (Ru) прервать;

(En) exit, (Ru) выйти –из

6. NewWSActivation – переход в новое рабочее пространство

Object involved (WS)

_____

USER-ACTION – *the Arc mode* (En) *switch-to*, (Ru) *переключиться-в*

7. PrecedentWSActivation – возвращение в предшествующее рабочее пространство

Object involved (WS)

_____

USER-ACTION – *the Arc mode* (En) *return-to*, (Ru) *вернуться-в*

4. Фреймовая структура идеационного дискурса и структура задачи (Task structure)

В предыдущем пункте мы определили ИДФ, которые могут описывать предметную область языконезависимого представления текста софтверных инструкций для пользователя. Эти функции достаточно обще отражают очерченную предметную область графического редактора, но не образуют

конструкций, отражающих намерения повествователя, которые представлены в системе Agile структурой задачи (Task Structure). В системе Agile рассматривается содержание фрагментов инструкции, посвященных созданию графического объекта или его свойств, например, *Рисование дуги по трем точкам, Создание стиля мультилинии*. Способы достижения этих целей задаются техническим писателем в текстах инструкций.

Формальная спецификация Task Structure разбивает текст инструкции на цели и наборы методов по их решению, состоящие из последовательностей шагов, которые представлены в Task Structure как Methods. Рассмотрим конкретный пример разбора фрагмента текста инструкции спецификацией Task Structure:

<Goal>: Создание стиля (ACTEE) мультилинии (OWNER)

<Method>:

<Precondition>:

<*Goal*>:Сначала откройте диалоговое окно (ACTEE) Multiline Styles (LABEL)

<Methods>

<Method 1>

<Constraint>:

Windows

<Steps>: В панели инструментов Object Properties (LOCATION) или в меню Data (LOCATION) выберите пункт (ACTEE) Multiline Style (LABEL).

<Method 2>

<Constraint: DOS и UNIX

<Steps>: В меню Data (LOCATION) выберите пункт (ACTEE) Multiline Style (LABEL).

<Steps>:

<Goal>: добавить элементы (ACTEE) в стиль (RECIPIENT).

<Methods>

<Method 1>

<Steps>: Нажмите кнопку (ACTEE) Element Properties (LABEL)

Эта схема описывает последовательность шагов по достижению цели пользователя, которые представлены в Task Structure как Methods. Предлагаемая нами фреймовая структура есть ни что иное, как другой уровень абстракции понятия Methods: если в Text Structure оно расшифровывалось, отталкиваясь от особенностей представления последовательности шагов в тексте инструкции, то здесь мы пытаемся выделить последовательность не шагов, а стандартных этапов работы пользователя с системой.

Инструкция для пользователя строится в соответствии со следующими риторическими принципами - изложение от простого к сложному, причем количество шагов в одном разделе инструкции не должно быть слишком большим. Поэтому начальным материалом может быть даже не материал, представляющий ценность как нечто, что может быть использовано для достижения какой-то самостоятельной цели пользователя. Он может излагаться как некоторая последовательность шагов, которая часто используется при описании разных разделов инструкции.

Для того, чтобы можно было автоматически управлять глубиной раскрытия этапа, фреймовая структура должна иметь иерархическую структуру. Иерархичность структуры отражает, насколько детально описывается тот или иной этап. Если пользователь не обладает достаточными знаниями, чтобы осмыслить и реализовать очередной этап работы (например, в инструкции сказано «Задайте параметры мультилинии», но пользователь не знает, как), необходимо разбить данный этап на под этапы, чтобы разъяснить, как достичь указанной в этапе подцели.

При создании современного программного продукта большое внимание уделяется реализации стандартного программного интерфейса с пользователем. Это облегчает пользователю переход к использованию нового программного продукта в уже известной пользователю области его применения или освоение программного продукта в новой для него области. Любую современную систему программисты пытаются сконструировать из стандартных и по возможности наиболее крупных компонент. Так, например, в любой

системе, работающей с файлами, имеется меню в верхней части окна со стандартными пунктами «Файл», «Правка». Под этими пунктами находятся подменю, содержащие обязательные пункты «Создать», «Открыть», «Сохранить» и «Выход» в пункте «Файл»; и «Вставить», «Вырезать» и «Копировать» в пункте «Правка». Более того, дальнейший интерфейс с пользователем по этим пунктам также является стандартным: например, в пункте «Открыть» появится стандартное диалоговое окно «Открытие документа» и т.д. (Тихомиров Ю. 1998).

Таким образом, общение пользователя с системой состоит из набора стандартных шагов. Благодаря стандартизованному интерфейсу, можно выделить общую схему достижения пользователем своих общих целей. Рассмотрим фреймовую структуру по уровням иерархии, «сверху вниз». Верхний уровень обычно специфичен для конкретного программного продукта. Он отражает концептуальное деление работы пользователя на этапы, которые не связаны с общением с функциональными объектами, а представляют собой макро действия. Например, «сохранить файл», «задать стиль мультилинии». Причем, первый пример обычно подразумевает от пользователя стандартные действия, которые могут и выходить за рамки рассмотрения инструкции: тут мы обычно имеем дело со стандартизованным Windows-интерфейсом с пользователем. Второй же требует дальнейших пояснений.

Ниже приведена структура этого уровня (Method и Макро действие). Описание верхнего уровня (уровня целей) рекурсивно: макродействие может раскрываться через само себя. Таким образом, уровень целей может иметь довольно глубокую иерархию раскрытия.

Если программный продукт составлен в соответствии с существующими в настоящее время стандартами создания интерфейса с пользователем, то все остальные дейтсвия пользователя могут быть описаны с использованием стандартной терминологии. Набор средств представляется ему системной библиотекой среды или же находится в стандартной библиотеке по разработке программного обеспечения данной области применения, которая опять же использует библиотеку операционной системы. Тем не менее на следующем уровне можно выделить неатомарные состояния, требующие расшифровки. На следующих схемах, составленных в соответствии с (Баженова И.Ю. 1999) они выделены серым фоном. Горизонтальной штриховкой показаны исходные и конечные состояния. Незакрашенные овалы представляют атомарные состояния.

В результате мы получили конечный набор стандартных этапов, в которых пользователь может находиться, работая с системой. Описания этих этапов войдут в софтверную инструкцию.

Зададим соответствие между узлами и стрелками с одной стороны, и ИДФ с другой стороны.

- Макро действие:
- DataActualisation
- DataOperation
- NewWSActivation
- PrecedentWSActivation
- WSActivation
- WSDesActivation
- Действие, меню, работа с диалоговым окном:
- FOActivation
- Стрелки
- WSActivation
- WSDesActivation

При описании каждого из выделенных этапов, начиная с уровня «действие», существуют устоявшиеся обороты и речевые выражения. Приведем их:

Узел:	**Шаблон:**
Активизация диалогового окно	Откройте диалоговое окно …
Меню	В меню … выберите пункт …
Внешнее меню	В меню … выберите пункт …
Подменю	(В открывшемся подменю) выберите пункт …
«Горячая» клавиша	Нажмите комбинацию клавиш …
Пиктограмма	Кликните левой кнопкой мыши по иконке … из строки пиктограмм
Работа с диалоговым окном	В диалоговом окне задайте необходимые параметры
Element	Выберите элемент (пункт) …

Edit Box	(В поле ввода) задайте …
Button	Кликните по кнопке …
Scroll Bar	Прокручивая …, используя … полосу скроллера
List Control	(Из списка элементов) … выберите …
Group	из группы …
Slider	Используя ползунок, задайте величину …
Tree Control	Выберите … (в дереве) …
Radio Button	Выберите альтернативу …
Spin	Задайте величину … используя наборный счетчик
Tab Control	Выберите закладку …
Check Box	Взведите флаг …
Progress	Запущенный процесс будет отображаться на индикаторе прогресса
Combo Box	Кликните на … из списка
List Box	Из списка … выберите …
Hot Key	Нажмите функциональную клавишу в поле ввода ...
Custom	< См. семантику программиста >

Содержание текста, описывающего реализацию конкретной цели пользователя, строится в порядке прохождения фрейма по по стрелкам между состоя-

ниями, пока не достигнет узла, в котором намеченная цель будет выполнена. Стрелками обозначены переходы между этими состояниями. Они тоже могут быть семантически нагружены. Стрелки отражают, что происходит в момент перехода между узлами, то есть результат выполнения функции в узле, откуда вышла стрелка.

Узел, куда указывает стрелка:	**Шаблон:**
Меню | «горячая» клавиша | пиктограмма	«В меню»
Подменю	«В открывшемся подменю»
Пиктограммы	«В строке пиктограмм»
Работа с диалоговым окном	«В открывшемся диалоговом окне»
Edit Box	В поле ввода
Scroll Bar	используя … полосу скроллера
List Control	Из списка элементов …
Slider	Используя ползунок
Tree Control	в дереве …
Spin	используя наборный счетчик
Hot Key	в поле ввода ...
Конец работы с диалоговым окном	Диалоговое окно … исчезнет с экрана

Подставив конкретные имена собственные (названия объектов) в шаблоны, идя по схеме в соответствии с семантикой программы, мы получим вполне связанный текст, описывающий производимые нами в программе действия.

Описанной в шаблонах части инструкции для пользователя соответствует только одна ИДФ - FOActivation. Все же остальные функции связаны с раскрытием понятия цели пользователя. Этот результат соответствует разделению goals vs. functions в (Hartley, Paris 1996), где подчеркивается их различная мотивация появления в тексте инструкции – появление первых обусловлено планом пользователя, вторых – функциональностью объектов интерфейса. Таким образом, мы приходим к идее совмещать ИДФ и шаблоны при моделировании содержания текста инструкции. Вместо слотов в шаблонах задаются параметры, которые представлены в таблице многоточиями. При разработке языконезависимой формализации текста инструкции шаблоны можно включать в формальный язык вместе с ИДФ, что позволит обеспечить адекватную языковую реализацию текста инструкции на каждом из выходных языков системы многоязыковой генерации.

5. Направления дальнейших исследований.

В статье мы попытались предложить ИДФ как средство представления содержания текста исходя из текстов инструкций и модели предметной области проектаAGILE. Проведенное исследование показывает, что ИДФ зависят от вовлеченных объектов. Инвентарь ИДФ для конкретной области может быть построен только на основе описания объектного пространства этой области, т.е. построения подробной классификации объектов с описанием состояний, в которых они могут находиться. Например, файл принадлежит к классу данные-объект, при этом он может быть открыт и представлен на экране или закрыт и связан с некоторым объектом (в каталоге) рассматриваемом при языковой реализации как его вместилище. Таким образом описание объектного пространства является необходимым условием обеспечения языконезависимости исходного представления содержания текста.

Понятие ИДФ дает возможность обсуждать вопросы моделирования процессов семантисизации – выражения концептов средствами языка. При этом в различных случаях актуализованное значение в большей степени опирается на ИДФ, например, *создание, осуществление* или на физические аспекты действия, например, *нарисовать*. Разные языки могут выделять различные аспекты физических действий при реализации ИДФ - (Ru) *нажать кнопку в диалоговом окне* – (En) *choose button* – опираются на разные аспекты физических действий.

Дополнительных исследований достойна также проблема автоматизации стратегического планирования текста. При создании модуля интерфейса с пользователем, программисты все чаще пользуются специальным программным обеспечением, помогающим создавать стандартизованный интерфейс программы, промежуточным продуктом которого является спецификация интерфейса (Тихомиров Ю. 1998). С помощью визуальных средств программист создает «эскиз» интерфейса и связывает конечные функциональные объекты с конкретными функциями программы. Например, кнопку Ok в диалоговом окне «Create Multiline» с программной функциейMLineCreate. Зная, что делает функция MLineCreate, можно узнать, какими действиями она вызывается. В этом случае техническому писателю необходимо задать лишь план, содержащий цели (Hartley, Paris 1996) и связать их с программными функциями, а необходимое взаимодействие пользователя с программой будет выявлено системой порождения текстов инструкций автоматически.

Литература

AGILE Project (1997) INCO-COPERNICUS Programme of the European Commission, PL961104.

Bateman, J.A., R. Henschel, F. Rinaldi. (1995) *The Generalized Upper Model 2.0*, Technical Report, GMD/IPSI, Darmstadt, Germany.

Power R. (1998) Preliminary model of the CAD/CAM domain. AGILE deliverable 2.1, June 1998.

Kruijff G-J.M., Kruijff I., Bateman J. (1999) The text structuring module for intermediate prototype. AGILE deliverable, April 1999.

Halliday, M.A.K. (1985) *Introduction to Functional grammar*. London: Edward Arnold.

Hartley, A., Paris, C. (1996) *Two sources of control over the generation of software instructions // Proc.of the ACL Annual Meeting,* Santa Cruz, June 1996. pp. 192-199.

Баженова И.Ю. (1999) Visual C++ 6 (Visual Studio 98) Уроки программирования , “Диалог-МИФИ” 1999, с. 145-168, 333-340

Жолковский А.К., Мельчук И.А. (1965) О возможном методе и инструментах семантического анализа. НТИ, N6, с. 23-28.

Лингвистический энциклопедический словарь (1990) М.: Советская энциклопедия.

Соколова Е.Г., Шаров С.А. (1998) К многоязыковой генерации руководств пользователя: начальный этап проекта AGILE. // Труды Международного семинара Диалог’98 по компьютерной лингвистике и ее приложениям (ред. А.С. Нариньяни), Казань, с. 848-859.

Соколова Е.Г. О дискурсной структуре и стиле текстов инструкций на русском языке и категории повеления.// Труды Международного семинара Диалог’99 по компьютерной лингвистике и ее приложениям (ред. А.С. Нариньяни), Таруса, с. 298-305.

Тихомиров Ю. (1998) Visual C++ 6 , БХВ - Санкт-Петербург 1999, с. 160-168, 199-214