УДК 004.4’41

О.В. КОНЮХОВА, Э.А. КРАВЦОВА

O.V. KONYUHOVA, E.A. KRAVTSOVA

**МОДЕЛИРОВНИЕ ЛОГИКИ ДИАЛОГА ГРАФИЧЕСКОГО**

**ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА:**

**ПОДХОДЫ К ТРАНСЛЯЦИИ ВЫРАЖЕНИЙ**

**С ЯЗЫКА АНАЛИЗА ЗАДАЧ НА ЯЗЫК ДЕЙСТВИЙ ПОЛЬЗОВАТЕЛЯ**

**modeling the logic of a graphical user interface**

**dialog: THE WAYS TO TRANSLATe EXPRESSIONS FROM THE TASK ANALYSIS LANGUAGE TO THE USER'S ACTIONS LANGUAGE**

*В статье приводится описание возможных подходов к трансляции спецификации графического пользовательского интерфейса с языка анализа задач на язык действий пользователя при моделировании логики диалога.*

*Ключевые слова: графический пользовательский интерфейс, язык анализа задач, язык действий пользователя, трансляция выражений, правила трансляции.*

*In this paper there is description of possible ways to translate the graphical user interface specification from the language of task analysis to the language of user actions when modeling dialog logic.*

*Keywords and Phrases: graphical user interface, task analysis language, user action language, expression translation, translation rules.*

**ВВЕДЕНИЕ**

Пользовательский интерфейс, в том числе, графический пользовательский интерфейс (ГПИ), является неотъемлемой частью интерактивных программных систем. Дружественность ГПИ во многом зависит от качества его диалоговой компоненты. Для описания ГПИ на разных этапах его разработки используются различные нотации. Согласно методике создания ГПИ [1], на этапе сбора и анализа требований к ГПИ применяется язык анализа задач (ЯАЗ) на базе нотации GOMS (Goals, Operators, Methods, Selection rules) [1], на этапе проектирования ГПИ – язык действий пользователя (ЯДП) на базе нотации UAN (User Action Notation) [1, 2]. Описание ГПИ на ЯАЗ отражает пользовательское видение (с учётом собственных знаний, умений, опыта, пожеланий) решения поставленной задачи в ходе диалога с программной системой. По сути, это является спецификацией требований к ГПИ. ЯДП описывает ГПИ в виде совокупности взаимодействующих задач, а также реакции системы на действия пользователя в процессе диалога.

По описанию ГПИ на ЯАЗ может быть получена первоначальная спецификация ГПИ на ЯДП, которая в дальнейшем может быть дополнена и подкорректирована с учётом особенностей ЯДП. Компилятор с ЯАЗ на ЯДП позволит ускорить процесс преобразования одной спецификации ГПИ в другую.

Важным компонентом компилятора являются правила трансляции [3], выполняющие перевод выражений с одного языка в выражения другого языка; в нашем случае, выражения с ЯАЗ в выражения на ЯДП. О правилах трансляции с ЯАЗ на ЯДП и возможных подходах их применения, пойдёт речь ниже.

**1 НОТАЦИЯ ДЕЙСТВИЙ ПОЛЬЗОВАТЕЛЯ**

Синтаксис и семантика ЯАЗ описаны в работах [4, 5].

Прежде чем рассматривать правила трансляции выражений с ЯАЗ на ЯДП, остановимся на терминах и условных обозначениях нотации действий пользователя UAN.

Нотация UAN (User Action Notation)[1, 2] была разработана в 80-х годах XX века. ЦельюUAN является описание взаимодействия между пользователем и системой в терминах действий пользователя, реакций системы и состояний интерфейса.

Основной организационной единицей в UAN является задача, которая может включать в себя как отдельные действия пользователя (выбор пиктограммы), так и другие задачи. Действие пользователя является атомарным элементом задачи. Таким образом, в нотации UAN задачи могут состоять или только из подзадач (верхний уровень абстракции), или из подзадач и действий пользователя (нижний уровень абстракции).

Задачи верхнего уровня абстракции записываются вне таблиц в виде равенств, в левой части которых записаны задачи, а в правой – соединенные операторами UAN подзадачи, необходимые для их выполнения. Перечень операторов приведен в таблице 1.

Таблица 1 – Операторы UAN для описания взаимодействия задач

|  |  |  |
| --- | --- | --- |
| **Название** | **Обозначение** | **Описание** |
| **1** | **2** | **3** |
| **Последовательная композиция** | A B | Задача *B* может начаться только после успешного завершения задачи *A*. |
| **Ожидание** | A (t=n) B | Задача *A* выполняется перед задачей *B*, но за время *n***.** |
| **Взаимодействие** | A | | B | Задачи *A* и *B* могут выполняться параллельно, синхронизируясь по одинаковым подзадачам. |
| **Чередование** | A ⇔ B | Задачи *A* и *B* могут многократно прерывать друг друга, чередуясь между собой. |
| **Однонаправленное чередование** | A ⇒ B | Задача *B*может прервать задачу *A*с последующим возобновлением задачи *А*. |
| **Независимый порядок** | A & B | Задачи *A* и *B* должны быть завершены, однако порядок их выполнения не важен. |
| **Прерывание** | A → B | Задача *B* может прервать задачу *A*для выполнения своих действий без последующего возобновления задачи *А*. |
| **Непрерывность** | <A> | Задача *A* не может быть прервана. |
| **Выбор** | A | B | Может выполняться либо задача *A*, либо задача *B*. |
| **Повторение** | A\* | Задача *A*может выполняться ноль или более раз. |
| A+ | Задача *A*может выполняться один или более раз. |
| An | Задача *A*может выполняться в точности *n* раз. |
| {A} | Задача *A*может выполняться ноль или один раз. |
| **Предусловие** | (Условие): A | Задача *A* может выполняться, если условие истинно. |

Задачи нижнего уровня абстракции описываются в виде таблицы, состоящей из трех колонок: «Действия пользователя», «Обратная связь» и «Состояние интерфейса», и читаются слева- направо сверху - вниз. В первой колонке записываются события в нотации UAN, представляющие собой последовательность действий пользователя с предусловиями или без них. Предусловие, может использовать только переменные, описанные в последней колонке таблицы и должно быть истинным, чтобы могли выполниться действия. При этом система реагирует на них посредством выполнения действий обратной связи, которые отражаются во второй колонке таблицы. В третьем столбце таблицы записываются выражения с переменными, которые отражают состояние интерфейса и выполняются, когда система реагирует на действия пользователя. Операторы UAN для обозначения действий пользователя приведены в таблице 2.

Нажатие отдельных клавиш клавиатуры удобно обозначать следующим образом: *K[Название клавиши]*. Тогда, например, нажатие и отпускание клавиши *Delete* запишется в виде строки *Kv[Delete] K^[Delete]*.

Таблица 2 – Операторы UAN для обозначения действий пользователя

|  |  |
| --- | --- |
| **Обозначение** | **Описание** |
| ~ | Перемещение курсора. |
| [X] | Контекст (окружение) объекта *X*. |
| ~[X] | Перемещение курсора в контекст объекта *X*. |
| [X]~ | Перемещение курсора вне контекста объекта *X*. |
| ~[x,y] | Перемещение курсора в произвольную точку с координатами *x* и *y*. |
| ~[x’,y’] | Перемещение курсора в конкретную точку с координатами *x* и *y*. |
| ~[x,y in A] | Перемещение курсора в произвольную точку с координатами *x* и *y* в пределах объекта *A*. |
| ~[X in Y] | Перемещение курсора к объекту *X* в пределах объекта *Y*. |
| Mv | Нажатие кнопки мыши. |
| M^ | Отпускание кнопки мыши. |
| Kv | Нажатие клавиши клавиатуры. |
| K^ | Отпускание клавиши клавиатуры. |

Ввод символов с клавиатуры удобно записывать в виде предусловий с регулярными выражениями в качестве условий: *(K = <Регулярное выражение>): (Kv K^)\**. Команды, в отличие от набора произвольных символов, вводимых с клавиатуры, будут представляться в виде строки, соответствующей имени команды. Например, ввод команды *Delete* запишется строкой *(K = ”Delete”):(Kv K^)\**, а ввод имени файла, состоящего из латинских строчных букв и арабских цифр: *(K = [a-z]([a-z]\*|[0-9]\*)): (Kv K^)\**. Для обозначения двойного нажатия кнопки мыши можно воспользоваться оператором ожидания: *MvM^ (t=n) MvM^*; время между кликами мыши не должно превышать установленного для двойного нажатия лимита времени. Кроме того, можно применять следующие операции и кванторы: квантор всеобщности (*∀*);квантор существования (*∃*); операторы сравнения: **=**, ≤, ≥, **≠**, <, >., логические операторы *AND*, *OR*, *NOT*.

Для обозначения обратной реакции системы в UAN также предусмотрен ряд операторов [1, 2]. Для описания состояния интерфейса можно использовать операторы произвольного языка программирования, псевдоязыка, алгоритмического языка. Для вопросов, обсуждаемых в данной статье они не актуальны, поэтому рассматриваться не будут.

Несмотря на недостатки [1, 2], UAN обладает богатым и компактным языком для описания действий пользователя, а также позволяет наиболее наглядным и естественным способом описать взаимодействие между пользователем и программной системой. На базе нотации UAN был разработан базовый синтаксис ЯДП [1].

**2 ПОДХОДЫ К ТРАНСЛЯЦИИ ВЫРАЖЕНИЙ С ЯЗЫКА АНАЛИЗА ЗАДАЧ**

**НА ЯЗЫК ДЕЙСТВИЙ ПОЛЬЗОВАТЕЛЯ**

ЯАЗ и ЯДП описывают ГПИ с разных позиций.

ЯАЗ представляет ГПИ в виде иерархии целей и методов, который желает достичь или выполнить пользователь в ходе решения поставленной задачи, взаимодействуя с программной системой и используя стандартные виды алгоритмов: линейный, ветвление, цикл.

На ЯДП ГПИ описывается в виде набора взаимодействующих между собой задач, а также действий пользователя, выполнение которых в процессе диалога пользователя с программной системой позволит ему достичь требуемого результата в решении проблемы.

Здесь можно провести аналогию с разработкой информационных систем, в частности, с описанием предметной области в виде бизнес-процессов и в виде взаимосвязанных сущностей базы данных [6]. В первом случае бизнес-процессы позволяют представить предметную область в динамике: в виде совокупности работ (активностей) и связывающих их потоков (финансовых, материальных, информационных и т.п.). Во втором случае предметная область описывается в статике: в виде совокупности взаимосвязанных структурированных данных, необходимых для осуществления бизнес-процессов.

Применительно к ГПИ, ЯАЗ отражает пользовательский взгляд на сценарий решения задачи: какими функциональными возможностями для этого должен обладать ГПИ. В свою очередь, ЯДП позволяет описать взаимодействие между функциональными возможностями ГПИ в процессе решения задачи пользователя.

Следует отметить, что правила взаимодействия между задачами на ЯДП намного разнообразнее, чем правила достижения целей (выполнения методов) на ЯАЗ [2], как было показано выше. Одному варианту достижения целей (выполнения методов) может соответствовать несколько вариантов взаимодействия между задачами. Например, для правила выбора на ЯАЗ имеется однозначное соответствие правила выбора между задачами на ЯДП. А оператору цикла на ЯАЗ можно поставить в соответствие четыре оператора повторения на ЯДП. Еще больше вариантов взаимодействия между задачами на ЯДП (первые семь строк таблицы 1) может соответствовать последовательному достижению целей (выполнения методов, операторов) на ЯАЗ. Однозначное соответствие можно установить между названиями целей (методов) ЯАЗ и названиями задач ЯДП. Названия целей и методов состоят из идентификаторов [4]; каждому идентификатору ЯАЗ соответствует идентичный идентификатор на ЯДП. Таким образом, названия целей (методов) на ЯАЗ соответствуют названиям задач на ЯДП.

Для операторов ЯАЗ тоже не всё так однозначно. Например, оператору перемещения курсора также могут соответствовать несколько действий пользователя, связанных с перемещением курсора в разные позиции (Таблица 2). Для остальных операторов ЯАЗ можно найти однозначное соответствие в виде действий пользователя на ЯДП.

Ещё больше вариантов перевода вызывают условия в операторах выбора и цикла ЯАЗ. Дело в том, что в ЯАЗ могут использоваться два вида условий:

1) условия, контролируемые программной системой, например, «Если объект выделен» или «Если выбрано рисование линии» и т.п.;

2) условия, контролируемые исключительно пользователем, например, «Пока пользователю требуется сохранение файла», «Пока работа с документом не завершена», «Если пользователю необходимо нарисовать окружность» и т.п.

Условиям первой группы ЯАЗ соответствуют предусловия на ЯДП, при этом, логические операторы ЯАЗ транслируются в соответствующие логические операторы на ЯДП: *И* в *AND*, *ИЛИ* в *OR*, *НЕ* в *NOT*. Формулировка самого условия на ЯАЗ представляет собой простое предложение, состоящее из набора идентификаторов [4]. В зависимости от интерпретации условия, оно может быть странслировано по аналогии с названием цели (метода) или же в более компактной форме с использованием операторов сравнения. Во избежание усложнения компилятора, интерпретация условий при трансляции возлагается целиком на пользователя. В принципе, компилятор можно наделить средствами анализа естественного языка, но данный вопрос выходит за рамки настоящей статьи.

Условия второй группы определяются только пользователем, исходя из решаемой им задачи, и не находят отражения в качестве предусловий на ЯДП, т.е. никак не транслируются.

Таким образом, в зависимости от принадлежности условия на ЯАЗ к первой или ко второй группе, оно либо транслируется в качестве предусловия на ЯДП либо нет. Исходя из этого, можно сделать вывод о том, что при наличии *n* условий в варианте достижения целей (выполнения методов) даже при однозначном соответствии варианта взаимодействия задач на ЯДП (без условий), возможно *2n* вариантов взаимодействия задач на ЯДП (с условиями).

Тогда возникает вопрос, каким образом организовать трансляцию выражений с ЯАЗ на ЯДП в системе анализа требований к ГПИ? Можно выделить два подхода.

1. *Трансляция по умолчанию*. Суть заключается в том, что, исходя из однозначности названий целей (методов) и задач, для каждого варианта достижения целей (выполнения методов) на ЯАЗ из всех возможных вариантов трансляции выбирается наиболее подходящий вариант взаимодействия между соответствующими задачами на ЯДП. Аналогично, для каждого оператора ЯАЗ выбирается наиболее подходящее действие пользователя на ЯДП. Что касается условий в операторах выбора и цикла на ЯАЗ, то во избежание потери условий, контролируемых программной системой, транслируются абсолютно все условия в идентичные предусловия на ЯДП, с учетом трансляции логических операторов. При последующей корректировке полученной черновой спецификации условия второй группы пользователем будут удалены, а операторы взаимодействия по умолчанию между задачами заменены наиболее подходящими.

Тогда, если *F* – это функция отображения объектов с ЯАЗ в объекты ЯДП, правила трансляции для операторов ЯАЗ (в соответствии с его синтаксисом и семантикой [4, 5]) в действия пользователя ЯДП можно представить следующим образом (Формула 1).

|  |  |
| --- | --- |
| F〚**Нажать\_кнопку\_мыши**〛 = Mv | (1) |
| F〚**Отпустить\_кнопку\_мыши**〛 = M^ |
| F〚**Дважды**\_**нажать\_кнопку\_мыши**〛 = MvM^ (t=n) MvM^ |
| F〚**Нажать\_клавишу\_клавиатуры** [<Название клавиши>]〛 = Kv **[**[Название клавиши]**]** |
| F〚**Отпустить\_клавишу\_клавиатуры** [<Название клавиши>]〛 = Kv **[**[Название клавиши]**]** |
| F〚**Переместить\_курсор** <Позиция>〛 = ~[Объект], где Объект = pick(x,y) |

Описание цели транслируется в выражение, где название цели соответствует имени задачи слева от знака равенства, а описание достижения цели – описанию взаимодействия подзадач на ЯДП (Формула 2). Аналогично для описания метода.

|  |  |
| --- | --- |
| F 〚**Цель:** <Название цели>**;** **[** <Описание достижения цели>**]** **Проверить\_результат.**〛 =  ¬ is\_name(F〚Название цели〛) ⇒ ошибка Название задачи = F〚Тело цели〛 | (2) |

Последовательно записанные выражения на ЯАЗ (подцели, методы, операторы) по умолчанию транслируются в выражения на ЯДП, соединенные оператором последовательной композиции (Формула 3).

|  |  |
| --- | --- |
| F〚Выражение1; Выражение2; … Выражениеn; 〛= ∃F〚Выражениеi〛.  ¬is\_name(F〚Выражениеi〛) ⇒ ошибка  ∃ F〚Выражениеi〛. ¬is\_op(F〚Выражениеi〛) ⇒ ошибка  F〚Выражение1〛F〚Выражение2〛 … F〚Выражениеn〛 | (3) |

Выбор между выражениями на ЯАЗ (целями, методами) по умолчанию транслируется с последовательность выражений на ЯДП с предусловиями, соединенных оператором выбора (Формула 4).

|  |  |
| --- | --- |
| F〚 **Выбор: [**Условие1: Выражение1; Условие2: Выражение2;…Условиеn:Выражениеn;**]**; 〛 =  ∃F〚Условиеi〛. ¬is\_bool(F〚Условиеi〛) ⇒ ошибка  ∃F〚Выражениеi〛. ¬is\_name(F〚Выражениеi〛) ⇒ ошибка  ∃F〚Выражениеi〛. ¬is\_op(F〚Выражениеi〛) ⇒ ошибка  (F〚Условие1〛): F〚Выражение1〛|  (F〚Условие2〛): F〚Выражение2〛 | … | (F〚Условиеn〛): F〚Выражениеn〛 | (4) |

Циклическое выполнение выражения ЯАЗ (цели, метода) по умолчанию транслируется в выражение на ЯДП с предусловием, к которому применяется повторение: замыкание Клини (Формула 5).

|  |  |
| --- | --- |
| F〚 **Цикл: [**Условие: Выражение; **]**; 〛 = ¬is\_bool(F〚Условие〛) ⇒ ошибка  ¬is\_name(F〚Выражение〛) ⇒ ошибка  ¬is\_op(F〚Выражение〛) ⇒ ошибка  ((F〚Условие〛): F〚Выражение〛)\* | (5) |

Преимущество этого подхода заключается в большей скорости получения первоначальной спецификации, а недостаток – в большем времени на последующую корректировку.

2. *Трансляция с участием пользователя*. Суть заключается в том, что последовательно проходя по выражениям спецификации ГПИ на ЯАЗ, пользователь сам указывает наиболее подходящие операторы для взаимодействия задач, варианты действий пользователя, отмечает условия, необходимые для трансляции, указывает подходящие операторы сравнения. При этом, тратится, соответственно, больше времени на подготовку к трансляции, но в дальнейшем полученная спецификация на ЯПД потребует гораздо меньше уточнений и корректировок.

Исходя из указанных преимуществ и недостатков рассмотренных подходов к организации трансляции с ЯАЗ на ЯДП при моделировании логики диалога ГПИ, можно сказать, что оптимальным вариантом является предоставление системой анализа требований к ГПИ возможности использования обоих подходов.

**ЗАКЛЮЧЕНИЕ**

По синтаксически и семантически верному описанию ГПИ на ЯАЗ, можно получить первоначальную спецификацию ГПИ на ЯДП в двух вариантах: либо быстро с параметрами по умолчанию (Формулы 1-5), но с серьёзными последующими корректировками, либо с предварительной подготовкой, но с гораздо меньшими корректировками в последующем. Выбор наиболее подходящего варианта остаётся за пользователем.

**СПИСОК ЛИТЕРАТУРЫ**

1. Конюхова, О.В. Графический пользовательский интерфейс для автоматизированных систем раскроя изделий сложной формы [Текст]: дис. канд. техн. наук 05.13.06: защищена 27.06.206: утв. 13.10.2006/ О.В. Конюхова, Орёл, 2006.- 171 с.

2. User Action Notation (UAN) [Электронный ресурс].- Режим доступа: http://www.it.bton.ac.uk/staff/teaching/notes/UAN.html– Систем. требования: P IV; 64 Мб ОЗУ; Windows 98 и выше; SVGA 32768 и более цветов; 640×480; мышь; IE 4.0 и выше. – Загл. с экрана.

3. Ахо, Альфред, В. Компиляторы: принципы, технологии и инструменты/ Альфред. В. Ахо, Рави Сети, Джеффри Д. Ульман: Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 768 с.; ил.

4. Конюхова, О.В. Синтаксис языка анализа задач для спецификации графического пользовательского интерфейса интерактивных компьютерных систем [Электронный ресурс]/ О.В. Конюхова// Методологические и теоретические аспекты создания и развития информационных технологий: Материалы V международной научно-технической конференции «Информационные технологии в науке, образовании и производстве.- Орёл, 2012.- 1 эл.опт. диск (CD-ROM): цв.

5. Конюхова, О.В., Кравцова, Э.А. Компилятор с языка анализа задач на язык действий пользователя: семантический анализ [Текст]/ О.В. Конюхова, Э.А. Кравцова// Информационные технологии в науке, образовании и производстве: сб. тр. VII Международной научно-технической конференции, 17-19 октября 2018г. – Белгород: Изд-во. ООО «ГиК». – 2018. – С. 89 – 94.

6. Дейт, К., Дж. Введение в системы баз данных, 8-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 1328 с.: ил. – Парал. тит. англ.- ISBN 5-8459-0788-8

**Конюхова Оксана Владимировна**

Орловский государственный университет им. И.С. Тургенева, г. Орёл

Кандидат технических наук, доцент кафедры «Программная инженерия»

Тел. 8-961-624-22-25

E-mail: oks1112@yandex.ru

**Кравцова Эльвира Александровна**

Орловский государственный университет им. И.С. Тургенева, г. Орёл

Кандидат технических наук, доцент кафедры «Информационные системы»

Тел. 8-906-569-20-20

E-mail: elvira\_kravtsova@mail.ru