КОМП'ЮТЕРНІ ЗАСОБИ, МЕРЕЖІ ТА СИСТЕМИ

УДК 004.923

И.И. ВЕРЕЩАГИН

ФОРМАЛИЗАЦИЯ АВТОМАТИЗИРОВАННОГО РЕШЕНИЯ СИНТЕЗИРОВАННЫХ ЗАДАЧ ГИБКИХ ТРЕНАЖЁРОВ

Работа завершает изложение общей денотационной модели гибких автономных тренажёров для привития интеллектуальных навыков. Рассмотрены основы инвариантной к реализациям денотационной модели автоматизированного решения синтезированных тренировочных задач в произвольной многопроцессной операционной системе.

Актуальность и новизна научно-технической задачи разработки прикладной модели гибких, т.е. перепрограммируемых тренажёров обусловлена отсутствием в настоящее время систематического подхода к созданию универсальных автоматизированных систем для привития специалистам интеллектуальных навыков [1, 2]. Установлено, что инвариантная к реализациям прикладная денотационная модель автономных гибких тренажёров такого рода раскладывается на две части: модель автоматизированного синтеза собственно тренировочных задач и модель их решения в произвольной операционной системе на персональном компьютере [3]. Основы денотационной модели синтеза самих тренажей были рассмотрены в [4]. В данной работе излагаются основополагающие принципы второй части общей денотационной модели, которая формализована посредством языка VDM-SL Венского метода разработки систем [5 – 7].

Напомним, что тренировочная задача синтезируется в гипотетической системе визуальной разработки, функционирование которой формализует первая часть общей денотационной модели [4]. Такая система визуального создания тренажей по сути предоставляет разработчикам наглядный, интуитивно понятный язык, содержательная семантика которого вынуждает их подходить к задаче с позиций трёх сущностей: режиссёр (исполнитель тренировочного плана), стажер (тренируемый на предмет интеллектуальных навыков) и учитель“за кадром”, который со-

ветует, приказывает, выдаёт и проверяет задания. В ходе последующего автоматизированного решения тренировочной задачи перечисленным сущностям отвечают одноимённые потоки вычисления, интерпретирующие команду за командой (каждый поток по своему), – один и тот же текст плана тренировки, который служил заключительным этапом визуального синтеза тренажа. Хотя в действительности тренировочные задачи должны быть реализованы в терминах распределённых, конкурирующих, т. е. параллельных потоков (процессов), описываемая подсистема гибкого тренажёра смоделирована аппликативно (функционально), как потенциально недетерминированная система (в этом смысле придерживаемся подхода Бьёрнера к формальному специфицированию систем распределённого типа) [8]. При императивном (алгоритмическом) подходе вышеуказанный поток, в свою очередь, распадается на пару потоков, в которой первый читает команду плана тренировки и передаёт, если требуется, сообщение второму потоку по поводу её дальнейшей обработки. С другой стороны, в модели применяется терминология и обозначения, принятые Венской школой [9], а также её сторонниками и последователями, в частности Ирландской школой VDM [5], при формализации семантики языков программирования.

Как выяснилось, все три потока тренировочной задачи укладываются в довольно абстрактную схему единого потока вычисления, позволяющую исследовать формальными средствами специфические и существенные аспекты создания гибких автономных тренажёров. Рассмотрим эту схему вычисления. Ниже представлены в упрощённом виде главные предложения абстрактного синтаксиса плана тренировки (ПланТр):

$$\text{ПланТр} = \text{Команда} \tag{1}$$

$$\text{Команда} = \text{СписКмнд} \cup \text{БазСцена} \cup \text{Акт} \cup \text{Действие} \cup \text{Антракт} \cup \ldots \tag{2}$$

$$\text{СписКмнд} = \text{Команда}^{*} \tag{3}$$

$$\text{Поток} = \text{Команда} \times \text{НазвЦикл}^{+} \times \text{СфаИзобр}^{+} \times \text{НачУрЗнан}^{+} \times \text{ном\_цкл:Nat0} \tag{4}$$

Предложениям (1) – (4) абстрактного синтаксиса тренировочного плана соответствуют следующие основные рекурсивные семантические функции, интерпретирующие все циклы тренажа (НазвЦикл^{+}) тренировочной задачи (префикс **M** в названии семантических функций – от англ. meaning “смысл”):

```
MПоток : Поток → ИнфСрд → СтатСрд → ДинСрд,
MПоток[mk - Поток(_ком, …, _ном_цкл)] (_и_с) (_с_с) Δ=
  let дин_срд = mk - ДинСрд(…, _ном_цкл),
      рез_цкл = mk - MКоманда[_ком] (_и_с) (_с_с) (дин_срд),
      нов_цкл_ = нов_цкл(рез_цкл)
  in if   нов_цкл_ = 0 then рез_цкл
  else MПоток[(_ком, …, нов_цкл_)] (_и_с) (_с_с)                    (5)
```

MКоманда : Команда → ИнфСрд → СтатСрд → ДинСрд → ДинСрд
MКоманда[**mk -** СписКмнд(_ком)] (_и_с) (_с_с) (_д_с) **Δ=**
let рез_ком = **M**Команда[**hd** _ком] (_и_с) (_с_с) (_д_с)
in if len _ком = 1 **then** рез_ком
else MКоманда[**mk -** СписКмнд(**tl** _ком)] (_и_с) (_с_с) (рез_ком) (6)

В связи с содержанием семантических функций (5) и (6) напомним, что циклом тренажа называется решение тренировочной задачи по одному и тому же плану тренировки, причём в каждом таком цикле стажёр решает задание одного типа, но, как правило, в более сложных условиях, скажем, при более жёстких строках, определяемых командами “антракт”:

(ИСЭтапы ⊆ ИнфСрд) =
запр_акт : ИмАкта — m → НазвЦикл ← m → BOOL ×
запр_дей : ИмДейств — m → НазвЦикл ← m → BOOL ×
антракты : ИмАнтр — m → НазвЦикл ← m → Длит (7)

Изображающее пространство тренажа складывается из сфер изображения (СфаИзобр$^{+}$), и для режиссёра, реализующего план тренировки, стажёра и учителя “за кадром” закрепляется своя, отдельная сфера. В каждой сфере представлена сцена (БазСцена), состоящая из картины (фона) с размещёнными в ней деталями картины (**M**Детал), например, персонажами и реквизитом (под персонажем понимается деталь картины, наделённая качествами человека). Например, в сфере режиссёра может быть отображён интерьер кабинета директора предприятия (**mk–**БазСцена). В такой обстановке тренируемый имеет возможность поместить курсор на рисунок рабочей папки, и на экране возникнет её содержимое:

(АктОбл — m → Метод) × (Знание ← m → Обл ⊆ СфаИзобр),

а может “нажать” мышью кнопку звонка и в кабинет войдёт секретарь:

(АктОбл — m → **mk–**ДинСцена[_им]) × (Метод$^{+}$ ← m → Обл).

В сфере изображения стажёра, вероятно, в это время будет находиться сцена с инструментами (деталями картины), позволяющими перемещаться по базовым сценам и возбуждать сцены динамические (**mk–**ДинСцена). Тогда в сфере изображения учителя может быть представлена сцена с элементами взаимодействия с гипертекстовой подсистемой информационной среды (ИнфСрд), и эти элементы будут выступать в качестве деталей картины в такой базовой сцене. В итоге, вышеизложенное поддерживается приведенными ниже элементами статической среды (СтатСрд) и среды информационной, в которую погружены потоки вычисления тренировочной задачи:

ССДтСц = (ИмДетал ∪ ИмБазСцен) ← m →
(Метод **- set** × обл : НазвАктОбл — m → Метод) (8)
ССБзСц = ИмБазСцен — m → ИмДетал **- set** (9)

$$\text{ССДинСц} = \text{ИмДинСцен} \text{ — m} \rightarrow \text{Метод}^{+} \quad (10)$$

$$\text{ИСДтСц} = (\text{ИмДетал} \cup \text{ИмБазСцен}) \leftarrow \text{m} \rightarrow \text{НазвЦикл} \leftarrow \text{m} \rightarrow \text{НазвАктОбл} \text{ — m} \rightarrow \text{АктОбл} \quad (11)$$

$$\text{ИСЗн} = \text{ИмДетал} \text{ — m} \rightarrow \text{НазвЦикл} \leftarrow \text{m} \rightarrow \text{НазвУрЗнан} \leftarrow \text{m} \rightarrow \text{Знание} \quad (12)$$

$$\text{ИСВыВвод} = \text{ИмДетал} \text{ — m} \rightarrow \text{НазвЦикл} \leftarrow \text{m} \rightarrow \text{вы} : \text{Обл} \times \text{вв} : \text{Обл} \quad (13)$$

$$\text{ИСВншВид} = \text{ИмДетал} \text{ — m} \rightarrow \text{НазвЦикл} \leftarrow \text{m} \rightarrow \text{Лицо} \times \text{ИмБазСцен} \text{ — m} \rightarrow \text{НазвЦикл} \leftarrow \text{m} \rightarrow \text{Картина} \quad (14)$$

$$\text{СтатСрд} = \text{ССДтСц} \cup \text{ССБзСц} \cup \text{ССДинСц} \cup \ldots \quad (15)$$

$$\text{ИнфСрд} = \text{ИСЭтапы} \cup \text{ИСДтСц} \cup \text{ИСЗн} \cup \text{ИСВыВвод} \cup \text{ИСВншВид} \cup \ldots \quad (16)$$

Связь между всеми семантическими функциями осуществляется через так называемую динамическую среду (ДинСрд), объект которой создаётся в функции **М**Поток (см. (5)). Динамическая среда наряду с обеспечением связи функций служит также для моделирования операций ввода и вывода на экран персонального компьютера:

$$\text{ДинСрд} = \text{нов\_цкл} : \text{Nat0} \times \text{выв\_карт} : \text{Картина} \leftarrow \text{m} \rightarrow \text{Обл} \times \text{выв\_дет} : \text{Лицо} \leftarrow \text{m} \rightarrow \text{Обл} \times \text{интрф} : \text{АктОбл} \text{ — m} \rightarrow \text{Метод} \times \text{вы} : \text{Вопр} \leftarrow \text{m} \rightarrow \text{Обл} \times \text{вв} : \text{Обл} \leftarrow \text{m} \rightarrow \text{Отв} \times \ldots \quad (17)$$

Базовые сцены интерпретируются следующей семантической функцией:

$$\begin{aligned}
&\textbf{М}\text{Команда}[\textbf{mk}\text{-БазСцена}(_\text{им})]\ (_\text{и_с})\ (_\text{с_с})\ (_\text{д_с})\ \Delta= \\
&\quad \textbf{let}\ \text{нов_рез} = \text{разрушСцену}(_\text{д_с}), \\
&\qquad \text{карт} = \text{вывКартин}(_\text{и_с}, \text{нов_рез}, _\text{им}), \\
&\qquad \text{дет} = \text{вывДеталКартин}(_\text{и_с}, _\text{с_с}, \text{нов_рез}, _\text{им}), \\
&\qquad \text{а_обл} = \text{создАктОбл}(_\text{и_с}, _\text{с_с}, \text{нов_рез}, _\text{им}) \\
&\quad \textbf{in}\ \mu(\text{нов_рез}, \text{выв_карт} \mapsto \text{карт}, \text{выв_дет} \mapsto \text{дет}, \text{интрф} \mapsto \text{а_обл})
\end{aligned} \quad (18)$$

Следует отметить, что структуры информационной и статистической среды потоков вычисления обеспечивают полиморфизм на трёх уровнях: уровень деталей картин, уровень сцен и уровень плана тренировки. Структурный полиморфизм позволяет относительно легко модифицировать задания от цикла к циклу в пределах задачи, а также переходить к другой предметной области, используя наработки из иных областей. Например, детали картин, как правило, имеют “лицо” (внешний вид на экране) и знания в виде поименованных единиц текста (12), (14). Тогда персонаж с именем “Главный бухгалтер” будет иметь в каждом цикле тренажа (НазвЦикл) другое лицо. В каждом цикле тренажа можно автоматически менять также и его знания. Более того, в течение одного цикла уровнем знания (НазвУрЗнан) можно автоматически варьировать указанием дополнительных меток текста (см. (12)).

План тренировки разбит на именованные периоды времени, называемые актами. Каждый акт, в свою очередь, разделяется на именованные секции, которые называются действиями (см. (2)). С актами и действиями связаны переключатели – включить в план тренировки и исключить из плана (см. (7)). Следовательно, все циклы тренажа выполняются по одному и тому же плану тренировки, но при разных значениях переключателей для актов и действий. Разумеется, что лица деталей картин и внешний вид самих картин (см. (14)), а также знания их персонажей и реквизита (см. (12)), как следует из вышеизложенного, тоже подлежат модификации и изменению.

На заключительном этапе цикла тренажа стажёр выполняет контрольное задание, состоящее из ряда вопросов, ответы на которые характеризуют навык тренируемого принимать решения в сложных жизненных ситуациях заданного типа. На этом этапе поток вычисления учителя отображает базовую сцену (см. (18)) с деталями картины, обеспечивающими вывод на экран вопросов и ввод стажёром ответов на них (см.(13)), что формально можно записать таким образом:

$(\text{АктОбл} — m \rightarrow \text{Метод}) \times (\text{Вопр} \leftarrow m \rightarrow \text{Обл} \subseteq \text{СфаИзобр})$

$(\text{АктОбл} — m \rightarrow \text{Метод}) \times (\text{Обл} \subseteq \text{СфаИзобр} \leftarrow m \rightarrow \text{Отв})$

Поскольку вопросы автоматизации оценки принимаемых решений выходят за рамки рассматриваемой модели, тезисно обозначим нашу позицию по данной проблеме. Функция рассматриваемой архитектуры гибких тренажёров, в основном, заключается в генерировании (на основе тренировочных планов и знаний) виртуальных ситуаций, типичных для различных уровней оперативного управления социальными объектами и системами. Такие реальности диктуют постановку учебных вопросов качественного, количественного и качественно-количественного характера, соединённых в единое целое. В этом и заключается главное препятствие на пути создания сугубо автоматических оценивающих процедур. Формально ответ О стажёра на ряд связанных вопросов можно представить в виде списка $[o_1, \ldots, o_i, \ldots, o_n]$, где $o_i \in o_i$ **-set**. Поскольку в данном случае практически нереально создать всё множество ответов, решение тренируемого сравнивается с шаблоном (образцом) $Ш = [ш_1, \ldots, ш_i, \ldots, ш_n]$. При достигнутом к настоящему времени уровне формализации любая попытка “втиснуть” оценку абстрактной разности $|О – Ш|$ в “прокрустово ложе” формализации приведёт к двум главным негативным последствиям – чрезмерным затратам труда на формализацию и кодирование и к утрате процедурой оценки универсальности. Следовательно, усилия разработчиков и программистов должны быть сосредоточены на аспекте оказания помощи педагогу при анализе им решений тренируемых, а также на оказании помощи тренируемым при самооценке ими собственных решений – когда тренировочные задачи используются в самостоятельной работе. Оценочные процедуры должны только сигнализировать о промахах и о “больших” расхождениях $|o_i – ш_i|$ по каждому ответу из списка О, составляющего решение стажёра.

Основные выводы:

1. Подсистема визуального создания тренажей вынуждает разработчиков подходить к задаче с позиций трёх сущностей: режиссёр (исполнитель тренировочного плана), стажёр (тренируемый на предмет интеллектуальных навыков) и учитель "за кадром". В ходе последующего автоматизированного решения тренировочной задачи данным сущностям отвечают одноимённые потоки вычисления, интерпретирующие один и тот же текст плана тренировки.

2. Все три потока вычисления тренировочной задачи укладываются в абстрактную схему единого потока вычисления, позволяющую исследовать формальными средствами специфические и существенные аспекты создания гибких автономных тренажёров.

3. Подсистема автоматизированного решения тренировочных задач может быть смоделирована аппликативно, как потенциально недетерминированная система, наряду с описанием в терминах распределённых, конкурирующих потоков (процессов).

4. Рассмотрены основы инвариантной к реализациям денотационной модели автоматизированного решения синтезированных задач гибких тренажёров в произвольной операционной системе.

5. Достигнутый к настоящему времени уровень формализации в общем не позволяет использовать в гибких тренажёрах сугубо автоматические процедуры для оценки решений тренируемых. При типичном применении гибких тренажёров обозначенной архитектуры оценочные процедуры должны лишь сигнализировать о промахах и недопустимых расхождениях с образцами по каждому ответу из списка, составляющего решение тренируемого.

1. *Верещагін І.І., Динисовець В.Д.* Принципи архітектури гнучких тренажерів для вищих ланок управління // УкрІНТЕІ. – 2002. – № 1/2. – С. 10–13.
2. *Верещагин И.И.* Организация анимационных фильмов в гибких тренажёрах // Комп'ютерні засоби, мережі та системи. – 2003. – № 2. – С. 158–163.
3. *Верещагин И.И., Тарасов В.А.* Денотационная модель гибких тренажёров для автоматизированного синтеза тренировочных задач // Intern. Conf. TAAPSD'2004: Спец. вип. вісн. Київ. ун-ту. – 2004. – С. 257–260.
4. *Верещагин И.И.* Автоматизированный синтез тренировочных задач в гибких тренажёрах // Комп'ютерні засоби, мережі та системи. – 2004. – № 3.– С.158–163.
5. *Butterfield Andrew.* The VDM Reference. – University of Dublin; Trinity College, 1999. – 174 p.
6. *RAISE* Language Group. The RAISE Specification Language: BCS Practitioner Series, 1992. – 397 p.
7. *Петренко А.К.* Венский метод разработки программ. Неформальное введение // Программирование. – 1991. – № 6. – С. 3–23.
8. *Бьёрнер Д.* Формальное специфицирование как экспериментальная наука // Программирование. – 1991. – № 6. – С. 24 – 43.
9. *Jones C.B., Shaw R.C.F.*. Case Studies in Systematic Software Development.– 1986. – xvi + 392 p.

Получено 20.08.2005