УДК 004.55

А.Н. Алфимцев, В.В. Девятков

НЕДЕТЕРМИНИРОВАННОЕ ПРОЕКТИРОВАНИЕ ИНТЕЛЛЕКТУАЛЬНЫХ МУЛЬТИМОДАЛЬНЫХ ИНТЕРФЕЙСОВ

Рассмотрен методологический подход к недетерминированному проектированию интеллектуальных мультимодальных интерфейсов на основе процессных и логических моделей. Процессные модели используются для описания процессов проектирования, а логические — для описания свойств, которому эти процессы должны удовлетворять. Недетерминированность проектирования заключается в выборе из совокупности альтернативных проектов того, который в наибольшей степени соответствует требуемым свойствам. Приведены различные подходы к осуществлению этого выбора.

E-mail: alfim@bmstu.ru

***Ключевые слова*:** *недетерминированное проектирование, интеллектуальный мультимодальный интерфейс (ИМИ), теория процессов, модальная логика.*

Введение. При проектировании интеллектуальных мультимодальных интерфейсов (ИМИ) могут учитываться различные критерии качества созданного проекта: экономический, эргономический, вычислительный, интеллектуальный, мультимедийный, мультимодальный и др. [1—3]. При этом принятие проектировщиком решения о выборе окончательного варианта проекта проводится не только в условиях многокритериальности, но и в условиях неопределенности, причинами которой являются:

— сложность и трудоемкость определения точных значений ряда показателей, вызванные небольшим объемом выделяемых ресурсов на предварительное исследование и анализ целей заказчика. В результате дополнительные требования и критерии качества, не заявленные прямо заказчиком, могут в итоге составить 95 % от первоначально сформулированных требований [4, 5];

— необходимость прогнозирования изменения первоначальных требований (число модальностей, типы графического интерфейса, правила бихевиоризации элементов интерфейса и т. д.) вследствие значительного временного интервала между началом проектирования и принятием решения по его внедрению. Ошибка в прогнозе более чем на 20…25 % приводит к переписыванию кода заново [6];

— уникальность проектируемых пользовательских интерфейсов, являющихся сложными человекомашинными системами. Программное обеспечение таких систем, протестированное с покрытием кода до 90 %, может содержать до 35 % дефектов, вызванных логическими ошибками, и до 40 % уникальных комбинаций дефектов [7].

Поскольку задача проектирования решается в условиях неопределенности, то возрастает вероятность того, что созданный проект не

будет удовлетворять начальным требованиям, что приведет к новым итерациям проектирования, рефакторингу, провалу или срыву отведенных сроков проектирования. По данным компании Standish Group только 16 % проектов по разработке программного обеспечения завершаются в срок [8]. Поэтому необходимо, с одной стороны, изначально формализовать процесс проектирования и представление требований к проекту, а с другой, обеспечить возможность выбора наилучшего варианта проекта из множества альтернативных. Все это может быть выполнено с помощью использования формальных моделей по определенной методологии проектирования ИМИ [3].

В настоящей работе в качестве формальных моделей для описания альтернативных проектов предложено применять процессные модели, а в качестве языка представления требований к проекту (свойств проекта) — язык модальной логики. Выбор проекта, в наибольшей степени удовлетворяющего требуемым свойствам, осуществлен с помощью логического вывода.

Формальные процессы. Каждый процесс имеет ***алфавит восприятий и реакций*** $Act = \{a_1, a_2, ..., a_m\}$. Каждый символ «$a$» алфавита именует некоторый объект, получаемый (воспринимаемый) процессом из внешней среды (***восприятие процесса***), выдаваемый процессом во внешнюю среду (***внешняя реакция процесса***) или объект, используемый процессом для внутренних нужд (***внутренняя реакция процесса***). Процессы действуют воспринимая, порождая для внутреннего употребления или выдавая во внешнюю среду объекты с соответствующими именами. Чтобы различать типы действий будем использовать следующие обозначения: ?a — восприятие; !a — внешние реакции; ta или t — внутренние реакции. Внутренняя реакция может быть скрытым процессом, структура которого не интересует. Тогда внутренняя реакция — просто имя или метка места, где соответствующий процесс вызывается. Имя места также можно рассматривать как задержку выполнения некоторого скрытого процесса или состояние процесса, которому принадлежит реакция и в которое процесс перешел после достижения этого места. В зависимости от уровня абстракции бывает целесообразным вообще не использовать внутренних реакций или, наоборот, после каждого действия вводить внутреннюю реакцию. Применим оба указанных приема. Последний из них аналогичен разметке регулярных выражений [9].

Нитью a^* будем называть кортеж (конечный или бесконечный) действий $<a_0a_1a_2 \ldots a_{m-2}a_{m-}>$, т. е. $a^*=<a_0a_1a_2 \ldots a_{m-2}a_{m-}>$. ***Выполнение нити*** — последовательность действий в порядке их записи в нити слева направо, т. е. осуществление в порядке слева направо по порядку восприятия или реакции. Символом «e» обозначим пустое действие. Нить, состоящая из единственного пустого действия, называется ***пустой нитью***.

Процесс P — множество нитей $Th(P)$, которые может выполнять процесс. ***Поведение процесса*** P — правила выполнения нитей, при-

надлежащих множеству $Th(P)$. Эти правила записываются на какомлибо ***языке описания поведения*** процесса. Воспользуемся языком регулярных выражений [10].

Если нить $a^*=<a_0a_1a_2 \dots a_{m-2}a_m>$ конечна, то ее описание на языке регулярных выражений определяется как $P' \triangleq {}_0.a_1.a_2 \dots a_{m-}2.a_m.0$, где P' — имя такого процесса, при котором множеству $Th(P')$ принадлежит нить a^* и все ее начала; $\triangleq$ — знак, обозначающий равенство по определению; все действия разделены точками; 0 — пустой процесс, $Th(0) = \varnothing$. Наличие пустого процесса после действия a_m означает конец нити a^* (после действия a_m не следует никаких действий). Если процесс слишком длинный, например,

$$P' \triangleq a_1^a.\overset{1}{a_2^a}....a_{m_a}^a.a_1^b.\overset{1}{a_2^b}....a_{m_b}^b....a_1^t.\overset{1}{a_1^t}.a_2^t....a_{m_t}^t.0,$$

то он может быть представлен как кортеж процессов

$$P' \triangleq P^a\,P^b \dots P^t,$$

где $P^a \triangleq a_1^a.\overset{1}{a_2^a}....a_{m_a}^a.$; $P^b \triangleq a_1^b.\overset{1}{a_2^b}....a_{m_b}^b.$; …; $P^t \triangleq a_1^t.\overset{1}{a_2^t}....a_{m_t}^t.0$.

Если процессы

$$P^a = a_0.a_1.a_2....a_{m-2}.a_m.a_1^a\overset{1}{a_2^a}....a_{m_a}^a.0.;$$

$$P^b = a_0.a_1.a_2....a_{m-2}.a_m.a_1^b\overset{1}{a_2^b}....a_{m_b}^b.0.; \dots;$$

$$P^t = a_0.a_1.a_2....a_{m-2}.a_m.a_1^t\overset{1}{a_2^t}....a_{m_t}^t.0.$$

имеют общее начало $P' \triangleq {}_0.a_1.a_2 \dots a_{m-}2.a_m.$, то они могут быть представлены процессным выражением

$$P'' \triangleq P'(P^a|P^b...|P^t),$$

$$P^a \triangleq a_1^a.\overset{1}{a_2^a}....a_{m_a}^a.0.;\ P^b \triangleq a_1^b.\overset{1}{a_2^b}....a_{m_b}^b.0.;\ ...;\ P^t \triangleq a_1^t.\overset{1}{a_2^t}....a_{m_t}^t.0.$$

Если процесс P''' зацикливается, то для его описания используем регулярное выражение $\{P'''\}$. Если зацикливание происходит не с начала, а после некоторой нити a^*, то для описания применяется регулярное выражение $P'' = a^*.\{P'''\}$. Если $P''' \triangleq \left(P^a \middle| P^b \dots \middle| P^t\right)$, то $\{P'''\} = \left(\{P^a\} \middle| \{P^b\} \dots \middle| \{P^t\}\right)$. Если $P''' = P'\left(P^a \middle| P^b \dots \middle| P^t\right)$, то $\{P'''\} =$

$= \left(\{P'P^a\} \middle| \{P'P^b\} \ldots \middle| \{P'P^t\} \right)$. Процесс P''', состоящий из процессов $P^a, P^b, \ldots, P^t$, которые выполняются параллельно, обозначим как $P''' \triangleq \left(P^a \parallel P^b \ldots \parallel P^t \right)$.

Описание свойств процессов проекта ИМИ. Моделью каждого проекта ИМИ является некоторый процесс. Частный случай описания процессов — описание их на языке графов переходов в виде совокупности параллельно функционирующих и взаимодействующих автоматов. С помощью такого описания требуемые свойства ИМИ могут быть сформулированы на языке модальной логики [11]. Для этого используются различные модальные операторы, например: $\bigcirc\xi$ — формула ξ будет истинной в ближайшем будущем; $\Diamond\xi$ — формула ξ станет истинной когда-нибудь в будущем; $\blacklozenge\xi$ — формула ξ была истинной когда-то в прошлом.

В соответствии с идеологией временной модальной логики символы времени в формулах не применяются. Согласно теореме Габбэя, любая формула временной модальной логики может быть переписана в логически эквивалентную форму с учетом правил вида: *прошлое* $\supset$ *будущее* [12].

Описание требуемых свойств ИМИ на языке модальной логики легко применимо к случаю, когда проект ИМИ представляется в виде процессных выражений. Так, свойство обязательной реакции в терминах процессных понятий на естественном языке формулируется в виде высказывания: если в прошлом процесс P выполнял последовательность действий $a_0.a_1.a_2....a_{m-2}.a_m.$, то в будущем он должен обязательно выполнить последовательность действий $b_0.b_1.b_2....b_{n-2}.b_n.$ [11].

На языке модальной логики приведенному выше высказыванию удовлетворяют, например, следующие, но не единственные формулы:

$$f(a_0.a_1.a_2....a_{m-2}.a_m.) \supset \bigcirc \varphi(b_0.b_1.b_2....b_{n-2}.b_n.).,$$

$$\blacklozenge f(a_0.a_1.a_2....a_{m-2}.a_m.) \supset \Diamond \varphi(b_0.b_1.b_2....b_{n-2}.b_n.).$$

Здесь $f(a_0.a_1.a_2....a_{m-2}.a_m.)$ и $\varphi(b_0.b_1.b_2....b_{n-2}.b_n.)$ — предикаты, истинные на последовательностях $a_0.a_1.a_2 \ldots a_{m-2}.a_m.$ и $b_0.b_1.b_2 \ldots b_{n-2}.b_n.$ соответственно.

Пример описания альтернативных процессов. Рассмотрим введенные понятия на примере процесса ИМИ телевизора. Процесс ИМИ предполагает наличие у каждого пользователя нескольких модальностей, с помощью которых он управляет работой телевизора (выключить или не выключить): $m1$ — статические жесты; $m2$ — кнопочный пульт; $m3$ — мимика лица; $m4$ — голосовые команды; $m5$ — динамические жесты. Вокруг телевизора выделим три зоны, в которых могут быть распознаны перечисленные модальности: $m1$,

$m3$, $m5$ — только в зоне $z1$ (расположена перед телевизором); $m2$ — в зонах $z1$ и $z2$ (включает в себя пространство слева и справа от телевизора); $m4$ — в зонах $z1$, $z2$ и $z3$ (находится сзади телевизора). Символ с черточкой наверху, например $\bar{z}1$, будет обозначать любой символ, кроме $z1$.

Первый процесс:

$$P_1 \triangleq P_{11} \| P_{12} \| P_{13} \| P_{14} \| P_{15},$$

$$P_{11} \triangleq ?m1.(?\bar{z}1.G \| ?z1.D) \| ?\bar{m}1.D,$$

$$P_{12} \triangleq ?m2.(?z1.(?z2.G \| ?\bar{z}2.D) \| ?\bar{z}1.D) \| ?\bar{m}2.D,$$

$$P_{13} \triangleq ?m3.(?z1.G \| ?\bar{z}1.D) \| ?\bar{m}3.D,$$

$$P_{14} \triangleq ?m4.(?z1.(?z2.(?z3.G \| ?\bar{z}3.D) \| ?\bar{z}2.D) \| ?\bar{z}1.D) \| ?\bar{m}4.D,$$

$$P_{15} \triangleq ?m5.(?z1.G \| ?\bar{z}1.D) \| ?\bar{m}5.D,$$

$$G \triangleq !\textit{выключить}.0,$$

$$D \triangleq !\textit{продолжить}.0.$$

Второй процесс:

$$P_2 \triangleq S_1 \| S_2,$$

$$S_1 \triangleq ?m1.!a.0 \| ?m2.!b.0 \| ?m3.!a.0 \| ?m4.!c.0 \| ?m5.!a.0,$$

$$S_2 \triangleq ?a.M_1 \| ?b.M_2 \| ?c.M_3,$$

$$M_1 \triangleq ?z1.G \| ?\bar{z}1.D,$$

$$M_2 \triangleq ?z1.(?z2.G \| ?\bar{z}2.D) \| ?\bar{z}1.D,$$

$$M_3 \triangleq ?z1.(?z2.(?z3.G \| ?\bar{z}3.D) \| ?\bar{z}2.D) \| ?\bar{z}1.D,$$

$$P_{15} \triangleq ?z1.G \| ?\bar{z}1.D,$$

$$G \triangleq !\textit{выключить}.0,$$

$$D \triangleq !\textit{продолжить}.0.$$

Два процесса должны выполнять идентичные функции, но описание их будет различным. Первый процесс не имеет параллельно выполняющихся подпроцессов, а второй — содержит такую пару подпроцессов S_1, S_2.

Проверка свойств процессов. Проверка, или доказательство свойств проектных процессов, может осуществляться с помощью различных процедур. Одной из таких процедур является редукция процессных выражений по определенным правилам [13]. Рассмотрим ее суть на примере процессов ИМИ телевизора. Для этих процессов свойства обязательной реакции с точки зрения пользователя формулируются просто: если пользователь выполнил управляющее воздействие с помощью некоторой модальности в зоне, в которой модальность может быть распознана, то телевизор должен быть выключен. В противном

случае телевизор продолжит работать. Более точно свойство обязательной реакции для процессов ИМИ телевизора можно перефразировать так: если в прошлом какой-либо из этих процессов выполнял последовательности $?m1.?z1.$, $?m2.?z1.?z2.$, $?m3.?z1.$, $?m4.?z1.?z2.?z3.$, $?m5.?z1.$ действий, то в будущем соответственно управляющее воздействие, переданное с помощью модальностей $m1, m2$, $m3$, $m4$, $m5$, должно быть выполнено, т. е. процессы должны обязательно получить реакцию $!выключить$.

Свойства обязательной реакции $!выключить$ на языке модальной логики

$$\blacklozenge\ f_1(?m1.?z1.) \supset \Diamond\ \varphi_1(выключить),$$
$$\blacklozenge\ f_2(?m2.?z1.?z2.) \supset \Diamond\ \varphi_2(выключить),$$
$$\blacklozenge\ f_3(?m3.?z1.) \supset \Diamond\ \varphi_3(выключить),$$
$$\blacklozenge\ f_4(?m4.?z1.?z2.?z3.?, m5.?z1.) \supset \Diamond\ \varphi_4(выключить),$$
$$\blacklozenge\ f_5(?m1.?z1.) \supset \Diamond\ \varphi_5(выключить).$$

Аналогично на языке модальной логики могут быть сформулированы свойства обязательной реакции $!продолжить$. Для доказательства этих свойств методом редукции процессных выражений по модальным формулам, задающим те или иные свойства проектных процессов, создается тестирующий процесс. Реакции тестирующего процесса представляют собой восприятия процесса, свойства которого проверяются. В этом случае процесс называется тестируемым. Если тестируемыми являются процессы P_1, P_2, то для свойства обязательной реакции $!выключить$ тестирующим процессом будет процесс $P_{тест}$:

$$P_{тест} \triangleq P^1_{тест} \left| P^2_{тест} \right| P^2_{тест} \left| P^3_{тест} \right| P^4_{тест} \left| P^5_{тест} \right.,$$
$$P^1_{тест} \triangleq !m1.!z1.?выключить,$$
$$P^2_{тест} \triangleq !m2.!z1.!z2?выключить,$$
$$P^3_{тест} \triangleq !m3.!z1.?выключить,$$
$$P^4_{тест} \triangleq !m2.!z1.!z2.?выключить,$$
$$P^5_{тест} \triangleq !m5.!z1.!z2.!z3.?выключить,$$

Таким образом, выполнение процесса $P_{тест}$ параллельно с процессом P_1 или P_2 для любого его подпроцесса, взаимно однозначно соответствующего реакциям одной из модальностей, всегда завершается пустым процессом, которому предшествуют реакция $!выключить$ со стороны процессов P_1 или P_2 и реакция $?выключить$ со стороны процесса $P^i_{тест}$, если его реакции допустимы. Следовательно, результатом редукции процессов $P^i_{тест} \parallel P_1$ и $P^i_{тест} \parallel P_2$ всегда будет вывод:

$$P^i_{тест} \,||\, P_1 \;\big|\!-\; ?выключить \,||\, !выключить \big|\!-\; 0,$$
$$P^i_{тест} \,||\, P_2 \;\big|\!-\; ?выключить \,||\, !выключить \big|\!-\; 0.$$

Докажем это, для процессов $P^1_{тест} \,||\, P_1$:

$$P^1_{тест} \,||\, P_1 =$$
$$P^1_{тест} \,||\, (P_{11} \big| P_{12} \big| P_{13} \big| P_{14} \big| P_{15}) =$$
$$!m1.!z1.?выключить. \,||$$
$$($$
$$\big|(?m1.(?z1.G \big| ?\bar{z}1.D) \big| ?\bar{m}1.D)$$
$$\big|(?m2.(?z1.(?z2.G \big| ?\bar{z}2.D) \big| ?\bar{z}1.D) \big| ?\bar{m}2.D)$$
$$\big|\big(?m3.(?z1.G \big| ?\bar{z}1.D) \big| ?\bar{m}3.D\big)$$
$$\big|(?m4.(?z1.(?z2.(?z3.G \big| ?\bar{z}3.D) \big| ?\bar{z}2.D) \big| ?\bar{z}1.D) \big| ?\bar{m}4.D)$$
$$) =$$
$$!z1.?выключить. \,||\, (?z1.G \big| ?\bar{z}1.D) \big| ?\bar{m}1.D) =$$
$$?выключить. \,||\, G =$$
$$?выключить. \,||\, !выключить.0$$
$$?выключить. \,||\, !выключить.0 =$$
$$0.$$

Недетерминированное проектирование. Недетерминированность проектирования ИМИ состоит в том, что проектируется несколько процессов для одного и того же ИМИ. Проектируемые процессы ИМИ должны удовлетворять определенным критериям, которые, в таком случае, являются свойствами ИМИ. На заключительном этапе проектирования следует выбрать тот процесс ИМИ, интегральная оценка совокупности свойств которого наилучшая. Для нахождения этой оценки необходимы детальная формулировка и выбор свойств процессов ИМИ, последующая их проверка и агрегирование результатов проверки для получения интегральной оценки. Недетерминированное проектирование процессов ИМИ на основе использования процедуры редукции требует наличия инструментальных программных средств, реализующих не только редукцию, но и проверку свойств с вычислением интегральных оценок процессов для альтернативных вариантов проекта. В настоящее время существует несколько реализаций таких программных средств для моделирования процессов [14, 15]. Однако эти реализации не позволяют вести недетерминированное проектирование ИМИ в указанном смысле альтернативного выбора наилучших проектов. Альтернативой этим средствам является традиционное логическое программирование с присущей ему недетерминированностью, которое может использоваться

для недетерминированного проектирования ИМИ. Рассмотрим использование логического программирования для недетерминированного проектирования ИМИ.

Использование логического программирования для недетерминированного проектирования ИМИ. Недетерминированный выбор состоит в неизвестном заранее поиске пути определения варианта из числа тех возможных альтернатив, которые приводят к успешному вычислению, удовлетворяющему требуемым свойствам. Реализация недетерминированного выбора в полном объеме невозможна без некоторых ограничений на процесс вычисления, в частности, так, как это выполнено в языке логического программирования «Пролог» (Visual Prolog 5.2) [16]. В этом языке недетерминированность заключается в реализации принципа «найти и проверить»: сначала генерируется множество предполагаемых альтернативных решений задачи, а затем осуществляется проверка тех из них, которые соответствуют требуемым свойствам. То есть с помощью описания того или иного процесса можно найти последовательности действий, которые он может выполнять и которые интересуют разработчика ИМИ, а затем проверить, удовлетворяют ли эти действия требуемым свойствам. Рассмотрим применение принципа «найти и проверить» в языке на примере проверки свойств процесса P_1. Множество процессных выражений процесса P_1 сведем к одному процессному выражению с использованием подстановки в процесс подпроцессных выражений

$$P_{11} \triangleq$$

$$(?m1.(?z1.!\textit{выключить}.0|?\bar{z}1.!\textit{продолжать}.0)|?\bar{m}1.!\textit{продолжать}.0)$$

$$|(?m2.(?z1.(?z2.!\textit{выключить}.0|?\bar{z}2.!\textit{продолжать}.0)?\bar{z}1.!\textit{продолжать}.0)|?\bar{z}2.!\textit{продолжать}.0)$$

$$|(?m3.(?z1.!\textit{выключить}.0|?\bar{z}1.!\textit{продолжать}.0)|?\bar{m}3.!\textit{продолжать}.0)$$

$$|(?m4.(?z1.(?z2.(?z3.!\textit{выключить}.0|?\bar{z}3.!\textit{продолжать}.0)|?\bar{z}2.!\textit{продолжать}.0)?\bar{z}1.!\textit{продолжать}.0)?\bar{m}4.!\textit{продолжать}.0)$$

$$|(?m5.(?z1.!\textit{выключить}.0|?\bar{z}1.!\textit{продолжать}.0)|?\bar{m}5.!\textit{продолжать}.0).$$

Введем внутренние реакции в этот процесс. Реакцию *продолжить* обозначим буквой «*n*», а реакцию *выключить* — буквой «*в*». В результате получим

$$P_1 \triangleq t0.(?m1.t11.(?z1.t12.!в.t13.0|?\bar{z}1.t14.!n.t15.0)|?\bar{z}1.t16.!n.t17.0)$$

$$|t0.(?m2.t21.(?z1.t22.(?z2.t23.!в.t24.0|?\bar{z}2.t25.!n.t26.0)?\bar{z}1.t27.!n.t28.0)|?\bar{m}2.t28.!n.t29.0)$$

$$|t0.(?m3.t31.(?z1.t32.t33.!в.t34.0|?\bar{z}1.t35.!n.t36.0)|?\bar{m}3.t37.!n.t38.0)$$

$$|t0.(?m4.t41.(?z1.t42.(?z2.t43.(?z3.t44.!в.t45.0|?\bar{z}3.t46.!n.t47.0)|?\bar{z}2.t48.!n.t49.0)?\bar{z}1.t41.0.!n.t411.0)$$

$$|?\bar{u}4.t412.!n.t413.0)$$

$$|t0.(?m5.t51.(?z1.t52.!в.t53.0|?\bar{z}1.t54.!n.t53.0)|?\bar{m}5.t56.!n.t57.0).$$

Интерпретируя каждую внутреннюю реакцию как состояние процесса, можно представить его в виде множества переходов между состояниями в результате совершения действий. Для простоты, не

снижая общности, ограничимся двумя модальностями, т. е. сократим процесс P_1 до процесса P_1':

$$P_1' \triangleq t0.(?m1.t11.(?z1.!t12.в.t13.0)\,|\,?\overline{z}1.t14.!n.t15.0)\,|\,?\overline{m}1.t16.!n.t17.0)$$
$$|\,t0.(?m2.t21.(?z1.t22.(?z2.t23.!в.t24.0\,|\,?\overline{z}2.t25.!n.t26.0)?\overline{z}1.t27.!n.t28.0)\,|\,?\overline{m}2.t28.!n.t29.0).$$

Программа, написанная на языке «Пролог», по принципу «найти и проверить» для процесса P_1' содержит множество фактов, задающих переходы процесса. Чтобы представить в языке «Пролог» множество переходов процесса P_1', которые перечислены ниже, используем предикат *transition* (*state*, *action*, *state*). Здесь *state* — состояние процесса; *action* — действие. Состояния обозначим теми же именами, что и в процессных выражениях, а действия переименуем ($?m1$ в $m1$; $?\overline{m}1$ в *notm*1; $?m2$ в $m2$; $?\overline{m}2$ в *notm*2; $!n$ в *cont*; $?z1$ в $z1$; $?\overline{z}1$ в *notz*1; $?z2$ в $z2$; $?\overline{z}2$ в *notz*2; $!в$ в *turnoff*):

$t0.?m1.t11.$	$t0.?\overline{m}1.t16.$	$t16.!n.t17.$	$t11.?z1.t12.$	$t12.!в.t13.$	$t11.?\overline{z}1.t14.$
$t14.!n.t15.$	$t0.?m2.t21.$	$t21.?z1.t22.$	$t22.?z2.t23.$	$t22.?\overline{z}2.t25.$	$t25.!n.t26.$
$t0.?\overline{m}2.t28.$	$t28.!n.t29.$	$t23.!в.t24.$	$t21.?\overline{z}1.t27.$	$t27.!n.t28.$	

Кроме перечисленных фактов, задающих переходы, введем двуместный предикат *accessible* ($S1$, [X], $S2$). С помощью этого предиката добавим следующие правила, обусловливающие условия достижимости состояния $S2$ из состояния $S1$ после передачи процессу, находящемуся в состоянии $S1$ последовательности входных символов, представленных списком [X]:

нерекурсивное правило достижимости:
accessible ($B1$, X, $B2$)):- *transition*($S1$, X, $B2$);
рекурсивное правило достижимости:
(*accessible* ($S1$, [X | *Rest*], $B2$):- *transition* ($S1$, X, $S3$), *accessible* ($S3$,[*Rest*], $S2$)).

Если разработчика ИМИ интересует только проверка свойства $\blacklozenge f_1(?m1.?z1.) \supset \Diamond \varphi_1(turnoff)$, то на языке «Пролог» для этого формулируется цель: *accessible*($t0$, X, $t13$), *nl*. Указанное свойство будет удовлетворено, если из состояния $t0$ в состояние $t13$ ведет единственная нить $m1.z1.turnoff$. Программа, написанная на языке «Пролог» с использованием введенных фактов и правил, приведена ниже:

```
domains
state=symbol
action= symbol
list=action*

predicates
nondeterm transition(state, action, state)
nondeterm accessible(symbol, list, symbol)
nondeterm member(action,list)
```

```
clauses

transition(t0, m1, t11).
transition(t0, notm1, t16).
transition(t16, cont, t17).
transition(t11, z1, t12).
transition(t12, turnoff, t13).
transition(t11, notz1, t14).
transition(t14, cont, t15).
transition(t0, m2, t21).
transition(t21, z1, t22).
transition(t22, z2, t23).
transition(t22, notz2, t25).
transition(t25, cont, t26).
transition(t25, cont, t26).
transition(t0, notm2, t28).
transition(t28, cont, t29).
transition(t23, turnoff, t24).
transition(t21, notz1, t27).
transition(t27, cont, t28).

accessible(S1, [X], S2) :- transition(S1, X, S2).
accessible(S1, [X|Rest], S2) :- transition(S1, X, S3),
accessible(S3, Rest, S2).

member(Name, [Name|_]).
member(Name, [_|Tail]):-member(Name,Tail).

goal
accessible(t0, X, t13), nl.
```

В результате работы этой программы получим X=["m1","z1","*turnoff*"], что означает следующее: единственная нить $m1.z1.turnoff$ ведет из состояния $t0$ в состояние $t13$. Аналогично с помощью языка «Пролог» могут быть проверены и другие более сложные свойства.

Заключение. В статье рассмотрен методологический подход к недетерминированному проектированию ИМИ на основе процессно-ориентированного описания проектов, использования модальной логики для описания свойств процессов и логического программирования как средства выбора из множества альтернативных процессов одного процесса, удовлетворяющего требуемым свойствам.

СПИСОК ЛИТЕРАТУРЫ

1. Гениатулина Е.В., Гриф М.Г. Методы генерации множества альтернатив в задачах оптимизации человеко-машинных систем // Научный вестник Новосибирского государственного технического университета. 2010. № 4. — С. 41—50.

2. Грибова В.В., Черкезишвили Н.Н. Развитие онтологического подхода для автоматизации разработки пользовательских интерфейсов с динамическими данными // Информационные технологии. 2010. № 10. — С. 54—58.
3. Девятков В.В., Алфимцев А.Н. Нечеткая конечно-автоматная модель интеллектуального мультимодального интерфейса // Проблемы управления. 2011. № 2. — С. 69—77.
4. Brooks Jr.F.P. The Design of Design: Essays from a Computer Scientist. — NY: Addison-Wesley, 2010. — 400 p.
5. Glass R.L. Building Quality Software. — NJ: Prentice Hall, 1992. — 369 p.
6. Гласс Р. Факты и заблуждения профессионального программирования / Пер. с англ. — СПб.: Символ-Плюс, 2007. — 240 с.
7. Макконнелл С. Совершенный код. Мастер-класс / Пер. с англ. — М.: Издательско-торговый дом «Русская Редакция»; СПб.: Питер, 2005. — 896 с.
8. Sutherland J. Managing the Iterative Process [Электронный ресурс]. URL: http://www.standishgroup.com/ (дата обращения: 28.06.2012).
9. Глушков В.М. Синтез цифровых автоматов. — М.: Физматлит, 1962. — 476 с.
10. Kleene S.C. Representation of Events in Nerve Nets and Finite Automata. In Shannon, Claude E.; McCarthy, John. Automata Studies. Princeton University Press, 1956. — P. 3—42.
11. Алфимцев А.Н., Девятков В.В. Необходимые и достаточные формальные свойства мультимодального интерфейса // Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение. 2011. Спец. вып. «Информационные технологии». — С. 159—166.
12. Gabbay D. Craig Interpolation Theorem for Intuitionistic Logic and Extensions // Journal of Symbolic Logic. 1977. 42 (2). — P. 269—271.
13. Milner R. A Calculus of Communicating Systems. — NY: Springer-Verlag New York, Inc. Secaucus, NJ, 1980. — 171 p.
14. BPML and Gentran Integration Suite Guide.Version 4.2. — NY: Sterling Commerce, 2006. — 35 p.
15. Oracle Fusion Middleware User's Guide for Oracle Business Process Management 11g. Release 1. [Электронный ресурс]. URL: http// docs.oracle.com/cd/E23943_01/ user.1111/e15175/bpmug_intro_bpm_suite.htm Дата обращения 28.06.2012.
16. Bratko I. Prolog Programming for Artificial Intelligence. — Boston: Addison Wesley, 2000. — 678 p.

Статья поступила в редакцию 4.07.2012