Уфа: УГАТУ, 2002 Вестник УГАТУ Т. 3, № 2. С. 164–171

УДК 681.51

А. И. ФРИД, Д. И. КАРДАШ

НЕЙРОСЕТЕВЫЕ МЕТОДЫ КОНТРОЛЯ ХОДА ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА В ИНФОРМАЦИОННО-УПРАВЛЯЮЩИХ СИСТЕМАХ

Рассматриваются вопросы построения адаптируемого программного обеспечения информационно-управляющих систем и особенности их контроля и диагностики. Рассматриваются виды адаптации программного обеспечения, определяется класс адаптируемых программных систем. Особое внимание уделяется вопросам применения искусственных нейронных сетей как для контроля и диагностики, так и для построения основного программного обеспечения. *Адаптация; программное обеспечение; информационно-управляющие системы; контроль; диагностика; самоорганизация; мультимасштабный анализ; нейронные сети; распознавание образов*

Жизнь происходит от «неустойчивых равновесий». Если бы равновесия везде были устойчивы, не было бы жизни. Но неустойчивое равновесие – тревога, «неудобно мне», опасность. Мир вечно тревожен и тем живет.

В.В. Розанов. *Опавшие листья. Короб первый*

ВВЕДЕНИЕ

Внедрение компьютерных и телекоммуникационных технологий во все сферы жизни даёт возможность рассматривать многие компоненты реального мира в виде объектов управления информационно-управляющих систем (ИУС). В этой ситуации понятие «информационно-управляющая система» может давать универсальное представление для широкого спектра форм человеческой деятельности.

Отличительной особенностью процесса построения и функционирования современных ИУС является воздействие на них ряда феноменов, качественно изменяющих деятельность компьютеризованных компонентов ИУС. Это — персональные вычисления (возможность широкого использования персональных ЭВМ); кооперативные технологии (компьютерная поддержка совместной согласованной работы группы работников); компьютерные коммуникации (увеличение возможностей обмена любой информацией). Они составляют технологическую основу обеспечения параллельного вычислительного процесса в распределенной компьютерной сети, ставшей, де-факто, основой построения современных ИУС. Все это, с одной стороны, дает разработчику новые технологические возможности, а с другой — целый ряд новых проблем, до этого не известных.

Важнейшей характеристикой ИУС производственных предприятий является конкурентоспособность. Эта характеристика вмещает в себя широкий спектр качеств ИУС (гибкость, учет требований реального мира, сроки разработки или модернизации, их стоимость, надежность, отказоустойчивость и др.). Для их обеспечения все шире начинают применять эволюционные методы разработки и сопровождения ИУС. Они составляют довольно широкий круг информационных технологий, затрагивающих самые разнообразные области применения и решаемые задачи. В качестве примера можно привести способы построения эволюционных компьютеров, использующих средства, имитирующие механизмы естественной эволюции [1]; программирование с применением прототипов [2]; CASE-технологии; программирование на адаптируемых языках программирования [3] и им подобные. Все указанные способы лежат в русле предложенного Эдвардсом Демингом бизнес-принципа, названного «Непрерывное усовершенствование процессов» (Continuous Processes Improvement) [4].

Его информационно-технологическая составляющая заключается в такой организации работ, при которой модернизация программного продукта идет в течение всего времени эксплуатации. Его применение дает возможность предприятиям сохранять конкурентоспособность в условиях непрерывной изменчивости внешней среды.

В конечном итоге процесс разработки и реинжиниринга ИУС и, в частности, их программного обеспечения (ПО) происходит на основе процессов самоорганизации [5]. Совместное функционирование отдельных рабочих станций распределенных ИУС можно рассматривать с точки зрения взаимодействия элементов мультиагентной системы – некоторых сущностей (агентов), способных к активным самостоятельным действиям. Основным качеством, определяющим агентов, является активность, возможность самостоятельного выполнения каких-либо действий, а также возможность изменения своего внутреннего состояния при взаимодействии с окружающей средой.

Здесь следует упомянуть появившуюся уже более 20 лет тому назад классификацию ПО, названную SPE-классификацией [6]. Согласно ей, ПО делится на три класса, причем принадлежность тому или иному классу зависит от возможности программы настраиваться на требования реального мира. В частности, последний (самый высокий) класс ПО, *E*-программы, характеризуется возможностью адаптации к изменяющимся требованиям и в конечном итоге эволюционным характером всего жизненного цикла ПО. В свое время данная модель не получила широкой известности, видимо, из-за того, что программы той поры выполнялись как жесткие и неизменяемые модули, определяемые, в первую очередь, требованиями системных спецификаций. Особенности же исполнения современного ПО ИУС придают новый смысл и значение упомянутой классификации.

Приведенные до этого рассуждения являются попыткой анализа особенностей ИУС и их ПО, обеспечивающих гибкость, учет требований реального мира, сроков разработки, но не вопросов обеспечения надежности и отказоустойчивости. Между тем именно связанные с ними вопросы контроля и диагностики выходят на первый план при реализации ПО в виде взаимодействующих между собой программных агентов, выполненных в виде эволюционирующих программ. Так, если на самом низком уровне SPE-классификации вопросы контроля и диагностики заключаются в контроле системных спецификаций и ошибок программистов, то при повышении уровня приходится контролировать уже не только саму программу на семантическом уровне, но и ее взаимодействие с окружающей средой, а на самом верхнем уровне — и процесс ее эволюции.

Несмотря на широкие современные возможности реализации агентов ИУС, в большинстве своем они будут определяться теми же факторами, на основании которых развивались до настоящего момента информационные технологии: последовательный детерминированный (алгоритмический) процесс, объектно-ориентированная технология программирования, фон-неймановская компьютерная архитектура, традиционная вычислительная математика. Но в последнее время быстрыми темпами идет развитие средств искусственного интеллекта (СИИ) и нейрокомпьютеров, применение которых, по мнению академика *А.С.Нариньяни* [7], рано или поздно вытеснит руководящий принцип алгоритма и традиционную вычислительную математику. В этом случае на смену существующим алгоритмам контроля и диагностики ПО с известным и неизменным алгоритмом работы должны прийти методы на принципиально новой основе.

Развивающееся использование СИИ в ПО ИУС все чаще оказывается связанным с сетевыми технологиями и, в частности, с глобальной компьютерной сетью Интернет. Интернет рассматривается как среда развития искусственного интеллекта, в которой «живут» автономные обучаемые нейросетевые агенты, процесс обучения которых никогда не заканчивается [8]. При этом процесс обучения и дообучения нейронных сетей (НС) уже сам по себе является объектом контроля и диагностики, тогда как в случае применения классического подхода исследованию подверглись бы только программы, реализующие искусственные НС.

Таким образом, структуры ПО современных ИУС представляют собой системы распределенной обработки данных, постоянно пребывающие в процессе настройки или расширения функциональных возможностей. В силу этого возникает потребность осуществления контроля и диагностики адаптируемого к внешней среде ПО, обладающего чертами интеллектуального поведения.

1. ОБЕСПЕЧЕНИЕ АДАПТИВНОСТИ ПО ИУС

Для решения этих задач, имеющих преимущественно глобальный характер, в первую очередь необходимо определить состав самого понятия «адаптация ПО». Его можно разделить на два основных процесса. Первый процесс можно охарактеризовать как «параметрическая адаптация» или «настройка». Например, уже готовая система изменяет характеристики тех или иных собственных компонентов, при этом их взаимосвязи остаются неизменными. Физически это может заключаться в широком спектре действий, например, от перемещения движков потенциометров, изменения тех или иных программных констант или текстовых строк до изменения синаптических весов в нейроподобных структурах. Второй процесс может быть охарактеризован как «функциональная адаптация», или, если речь идет об изменении словарного состава программных систем, «лингвистическая адаптация» [9]. Он предполагает изменение структурных связей или создание новых программных модулей, новых функций в программной сфере. При этом следует отметить логическую связанность двух процессов адаптации. Хотя они и могут быть представлены двумя технологически раздельными действиями, но в совокупности они представляют собой одно явление, подчиненное единственной цели осуществления конкурентоспособности объекта как технической, так и иной природы.

Рассматривая программные средства, реализующие параметрическую адаптацию, следует отметить достаточную простоту технологии её осуществления. В этой области основу возникающих задач составляет необходимость построения алгоритмов определения новых значений параметров, а не способы непосредственного их изменения. Можно сказать, что вопросы параметрической адаптации более относятся к теории искусственных нейронных сетей, чем к технологиям программирования. Другое дело – лингвистическая адаптация. Ряд существующих программных систем обладает в той или иной мере подобными возможностями. В свое время фирмой Microsoft было заявлено об оснащении серверных реализаций операционной системы Windows 2000 интерпретаторами сценариев, язык которых будет определяться самим пользователем [10].

Можно предложить определение класса ПО, допускающего лингвистическую адаптацию, это – лингвистически адаптируемые программные системы (ЛАПС). ЛАПС – это такие языки программирования и операционные системы (ОС), словарный запас которых (множество поименованных модулей, выполняющих те или иные функции) может быть изменен пользователем без общей перекомпиляции ядра программной системы, а изменения вступают в силу немедленно после их внесения. При этом в ЛАПС не исключается возможность проведения и параметрической адаптации. Попытка ввести описание подобного класса программных систем уже предпринималась в [3], однако не было дано точной формулировки критериев принадлежности объектов к классу адаптируемых языков программирования. Предлагаемый авторами настоящей статьи класс ЛАПС включает в себя уже существующие языки программирования и программные оболочки. Необходимость его введения обусловлена тем, что в настоящее время становится актуальным развитие именно адаптивных свойств ПО, и с этой точки зрения приведенный класс оригинален.

Под это определение подпадает ряд языков программирования и ОС. Общераспространенные языки программирования, предполагающие процесс либо только компиляции, либо интерпретации (такие, как C, C++, Pascal, PERL и им подобные), к классу ЛАПС не относятся и, следовательно, в меньшей степени соответствуют процессу адаптации ПО. Для их использования в адаптируемых программных системах потребуется дополнительная разработка механизмов расширения состава функций. Особенностью языков программирования, относящихся к ЛАПС (Форт, LISP, ДССП, язык среды MatLab и др.), является сочетание ими функций интерпретатора и компилятора. Одним из основных представителей ЛАПС является язык программирования Форт. Процесс расширения словарного состава является его основной идеей, в силу чего он и получил название «адаптируемого языка» [3]. По мнению [11], язык Форт воплощает в себе стремление к простоте и оптимальному взаимодействию программы и аппаратуры. Он обладает такими преимуществами, как простота доступа к аппаратным средствам вычислителя, простота аппаратной реализации Форт-машины, сочетание в себе возможностей интерпретатора и компилятора, черт среды моделирования с чертами среды программирования и др. В силу этого Форт рассматривался нами как база для построения ЛАПС и организации процессов её контроля и диагностики. Однако здесь не постулируется исключительная роль Форта в создании ПО

ИУС. В качестве адаптируемого программного ядра может выступать любое программное средство, имеющее для этого соответствующие функции. Представленные в работе результаты могут быть перенесены и на другие языки программирования класса ЛАПС.

Применение ЛАПС при построении ПО ИУС позволяет сократить общий срок разработки (сопровождения или эволюции) за счет исключения операции перекомпиляции операционного ядра программной системы, сократить число проектных ошибок за счёт раннего и полного учета практических ограничений и требований. Это создаёт предпосылки для построения эволюционирующих, гибких ИУС, обеспечивающих конкурентоспособность в условиях динамично меняющейся окружающей среды. Однако возможность динамического изменения программных систем (кодов программ или переобучения НС) ведет к увеличению вероятности возникновения неисправностей. Поэтому весьма актуальными являются задачи обеспечения «корректной адаптации», а, по существу, задачи контроля и диагностики процессов функциональной и параметрической адаптации ПО.

2. КОНТРОЛЬ И ДИАГНОСТИКА ХОДА ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА

В условиях перманентного процесса изменения программ становится весьма проблематичным осуществление мероприятий по их контролю и диагностике, опирающихся на знание алгоритмов функционирования. Вслед за изменением объекта должны изменяться и алгоритмы контроля и диагностики. Проблему частично решает использование встроенных программ самодиагностики, которые изменяются одновременно с основным ПО. Однако существует ряд случаев, когда самодиагностика программ недостаточна. Примером такого случая может служить ситуация исполнения в многозадачной среде программ, негативно влияющих друг на друга. Причинами такого влияния являются неисправности, внесенные на различных этапах разработки ПО.

Если же данные программы исполнены различными коллективами разработчиков, то построение единой для них системы самоконтроля весьма затруднительно в силу того, что два или более коллектива должны изучить программные продукты друг друга. При высокой сложности современных программных систем возникновение подобных кооперации малоэффективно. В такой ситуации выгоднее рассматривать ПО как «черный ящик», имеющий связи с общими ресурсами операционной системы. В этом случае возможно осуществление контроля и диагностики процесса функционирования программ по характеру использования ими ресурсов системы. Примерами этих характеристик могут служить: интенсивность запросов этих ресурсов, объем занимаемой памяти, наблюдаемые через регистры MSR характеристики процессора класса Pentium, события типа обращений к прерываниям, каналам прямого доступа к памяти, кэш-памяти и многое другое.

При этом актуальной становится задача распознавания во временных рядах диагностических данных образов, соответствующих различным штатным режимам работы или ситуациям проявления каких-либо ошибочных действий в программной системе. Для решения этой задачи видится наиболее эффективным использование искусственных НС с их способностью к обучению.

Для решения этой задачи необходимо, в первую очередь, определить ассортимент образов, распознаваемых с помощью НС. Пусть задан ряд следующих друг за другом во времени (не обязательно через равные интервалы) значений наблюдаемой переменой $W = v_0, v_1, v_2, ..., v_\infty$. Из этого ряда выбирается фрагмент конечной длины N: $W_0 = v_i, v_{i+1}, v_{i+2}, ...v_{i+N}$. Рассматривается движение фрагмента конечной длины по ряду большей длины. Для W определена функция многомасштабного анализа, состоящая в построении новых, соразмерных W_0, фрагментов, элементы которых определены как взвешенные комбинации (в простейшем случае осредненные) элементов исходного ряда:

$$W_p^0 = \left\{ v_{p,i}^0, ..., v_{p,k-1}^0, v_{p,k}^0, v_{p,k+1}^0, ..., v_{p,i+N}^0 \right\},$$

$$v_{p,k}^0 = \frac{100}{pN} \left(v_k^0 + \sum_{j=k-\left[\frac{pN}{200}\right]}^{k-1} v_j^0 + \sum_{j=k+1}^{k+\left[\frac{pN}{200}\right]} v_j^0 \right), \quad (1)$$

где p — уровень осреднения (масштаб данных), определяющий, какое количество последовательных данных из общего числа будет использовано при осреднении, %; $[\cdot]$ — операция выделения целой части выражения в скобках.

В качестве базового набора распознаваемых образов выберем функции, принадлежащие пяти группам. Фразеологически можно обозначить эти группы как: «монотонно воз-

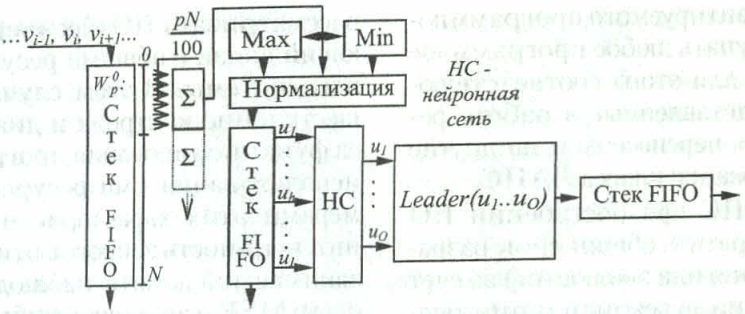

Рис. 1. Движение нейронной сети по фрагменту данных

растающие функции», «монотонно убывающие функции», «константы», «всплеск», «отрицательный всплеск». Задаче распознавания компонентов этого функционального базиса обучается НС – персептрон. Для подобных сетей с I входами и O выходами можно определить функцию $Leader(U_1 \ldots U_O)$ от выходов нейронной сети, определяющую номер лидирующего нейрона, и процедуру движения нейронной сети по W_0 (рис. 1). Работа ядра приведенной структуры НС-классификатора проиллюстрирована на рис. 2.

НС-классификатор совместно с функцией *Leader* определяет номер функции из выбранного базиса, наиболее адекватной поданным на ее входы значениям. На рис. 2, *а* и *б* показаны случаи распознавания НС данных, точно соответствующих функциям базиса. На рис. 2, *в* показано распознавание данных, которые могут быть ассоциированы с двумя из элементов базиса. В этом случае *Leader* выбирает наиболее «похожую» на данные функцию (f_6).

Приведенная структура позволяет исследовать несколько масштабов данных путем изменения значения p и выбрать тот масштаб данных, в котором исследуемое нарушение вычислительного процесса проявляется наиболее заметно. По результатам анализа входного стека (рис. 1) строятся вариационные ряды частоты появления в выходном стеке номеров функций базиса. В случае возникновения отклонений в ходе вычислительного процесса вариационные ряды изменяются и характер их изменения характеризует источник нарушения работы. Подробное описание примера функционального базиса и многомасштабного анализа диагностических данных приведено в [12, 13].

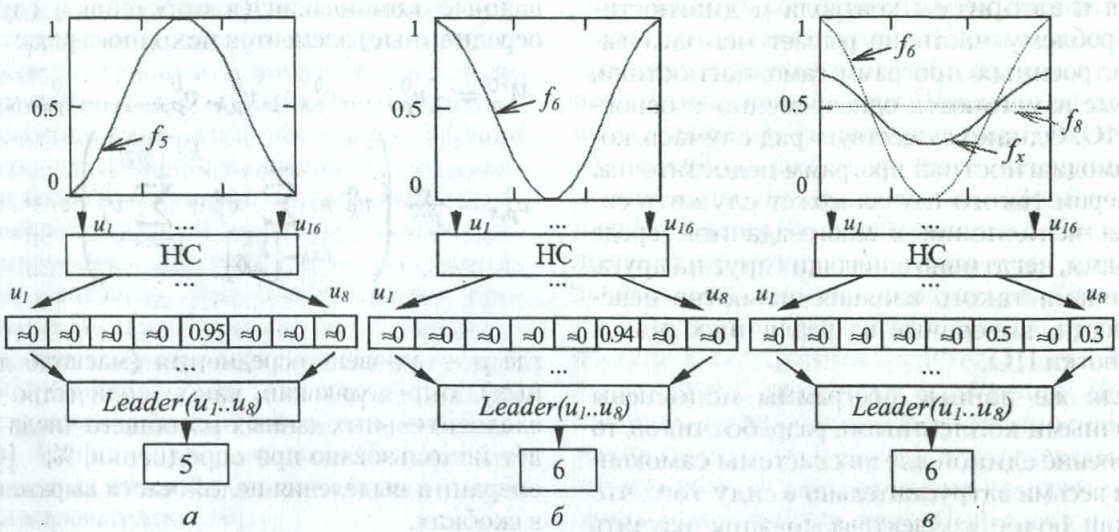

Рис. 2. Изменение значения функции *Leader* в зависимости от входных сигналов НС: *а*, *б* – распознавание данных, точно соответствующих функциям базиса; *в* – распознавание данных, которые могут быть ассоциированы с двумя из элементов базиса

3. КОНТРОЛЬ И ДИАГНОСТИКА ПАРАМЕТРИЧЕСКОЙ АДАПТАЦИИ НЕЙРОННОЙ СЕТИ

Современные нейросетевые технологии уже составляют некоторую конкуренцию принципам алгоритма. При этом применение последовательных алгоритмов для контроля и диагностики параметрически адаптируемого ПО далеко не всегда будет эффективно. Когда же для этого применяются нейросетевые методы, вся вычислительная система будет состоять из однородных объектов, которые в равной мере обладают адаптивными свойствами к изменению требований окружающей среды.

Приведенный далее способ контроля и диагностики вычислительного процесса состоит из нескольких нейросетевых блоков, служащих как для распознавания функций базиса, так и анализа получаемых по данным выходного стека предложенной структуры вариационных рядов [12]. При использовании при построении ПО ИУС программных реализаций искусственных НС получаемый программный код характеризуется регулярностью. Выявить допущенные программные ошибки в этом случае достаточно просто. Однако в процессе обучения даже программно корректно реализованная НС может быть сконфигурирована таким образом, что в ходе работы она будет выдавать неверные значения. При этом процесс обучения будет проходить корректно. Но, как показывает практика, по характеристикам процесса обучения НС можно провести диагностику базы обучающих примеров и уже на фазе обучения отбраковать некорректные данные, тем самым обезопасив себя от некорректных решений НС на этапе функционирования.

Ниже приведен пример использования предложенного нейросетевого метода контроля хода вычислительного процесса при обучении нейронной сети. Следует отметить, что рассматриваемые способы применимы не только для программных, но и для аппаратных реализаций НС.

В качестве практического примера области, в которой требуется выполнение контроля базы обучающих примеров, можно указать задачу прогнозирования поведения биологических систем, представленную в [14]. Аутоассоциативная НС-предиктор обучалась распознаванию векторов данных биохимических исследований. На рис. 3 показано изменение числа итераций обучающей парадигмы для каждого из примеров задачника, необходимых до достижения удовлетворительной ошибки обучения. Данные получены на основании обучения НС в [15].

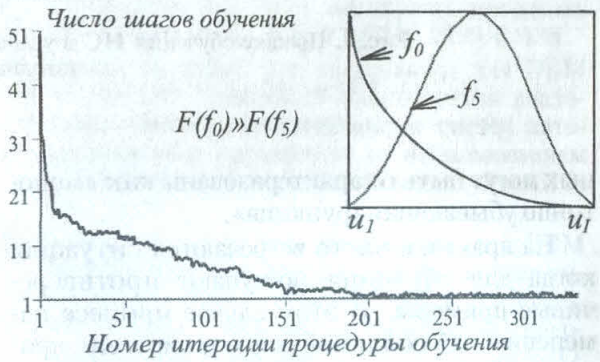

Рис. 3. Процесс обучения НС корректному задачнику ($F(f_x)$ — частота лидирования функции x)

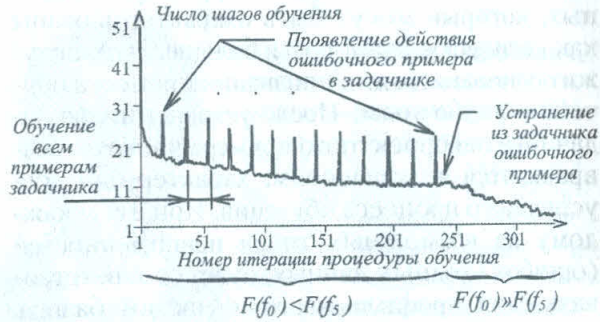

Рис. 4. Обучение НС задачнику с примером-ошибкой

Обучающие выборки часто могут содержать ошибочные, противоречащие друг другу примеры. На рис. 3 приведена ситуация, когда все примеры задачника корректны. С помощью структуры, показанной на рис. 1, можно произвести распознавание образов (функций базиса), с помощью которых может быть наиболее точно аппроксимирована экспериментальная кривая на различных масштабах данных. При этом происходит вычисление частоты лидирования каждой функции базиса. В приведенном примере для выявления некорректных данных в обучающей выборке из всех рассматриваемых функций (подробное их описание приведено в [12]) актуальными оказались функции f_0 и f_5 (см. рис. 3). По графику видно, что f_0 наиболее адекватно отражает характер изменения числа шагов обучения сети. Это подтверждается и результатами работы структуры на рис. 1. Частота $F(f_0)$ распознавания во входном потоке данных функции f_0 много больше частоты распознавания функции f_5 и других функций, т. е. подавляющее большинство фрагментов дан-

Рис. 5. Процесс обучения НС в условиях вероятного поступления ошибочных входных данных

ных могут быть охарактеризованы как «монотонно убывающая функция».

На практике часто встречаются ситуации, когда для обучения поступают противоречивые примеры. В этом случае процесс изменения наблюдаемой характеристики процесса обучения примет вид, представленный на рис. 4. При анализе этих данных, частота распознавания f_5 становится больше частоты распознавания f_0, т. е. фрагментов данных, которые могут быть охарактеризованы как «всплеск», становится больше. Это и служит основанием для признания процесса обучения ошибочным. После устранения из задачника некорректного примера частоты возвращаются к значениям, характерным для успешного процесса обучения. При этом каждому из возможных типов неисправностей (ошибок входных данных) будет соответствовать свой профиль частот функций базиса, являющийся исходными данными для осуществления процесса диагностики. На рис. 5 проиллюстрировано прохождение контролируемого с помощью предложенного алгоритма процесса обучения НС.

ЗАКЛЮЧЕНИЕ

В статье описано состояние вопроса и некоторые подходы к построению систем контроля вычислительного процесса и ПО современных ИУС с использованием СИИ. Особенностью предложенного метода контроля и диагностики является то, что он предназначен для работы с изменяющимися, эволюционирующими программными модулями, в том числе и с построенными на основе СИИ. Данный метод реализует контроль и диагностику процесса параметрической адаптации нейросетевого агента, однако, как было показано во введении, этот вид адаптации тесно связан с функциональной или лингвистической адаптацией. Вопрос лингвистической адаптации ПО ИУС рассмотрен в [9, 12]. Таким образом, предложенный метод является составной частью процесса осуществления адаптации ПО ИУС в целом, процесса когерентного развития как параметрической, так и лингвистической адаптации, что является предметом дальнейшего исследования.

СПИСОК ЛИТЕРАТУРЫ

1. **Штрик А. А., Осовецкий Л. Г., Мессих И. Г.** Структурное проектирование надежных программ встроенных ЭВМ. Л.: Машиностроение, 1989. 296 с.
2. **Букатова И. Л.** Когнитивный эволюционный компьютер // Нейрокомпьютер. 1997. № 3, 4. С. 35–48.
3. **Редько В. Г.** Эволюционный подход к исследованию естественных и созданию искусственных «биокомпьютеров» // Нейрокомпьютер. 1994. № 1. С. 38–49.
4. **Eversheim W. et al.** Simultaneous Engineering. Erfahrungen aus der Industrie fuer die Industrie. Springer-Verlag, 1995. 264 p.
5. **Nevins J. L., Whithey D. E.** Concurrent Design of Products and Processes. New York: McGraw-Hill, 1989. 268 p.
6. **Смирнов А. В., Юсупов Р. М.** Технология параллельного проектирования: основные принципы и проблемы внедрения // Автоматизация проектирования. 1997. № 2. С. 15–20.
7. **Вендров А. М.** CASE-технологии. Современные методы и средства проектирования информационных систем // http://www.citforum.ru/database/case/index.shtml.
8. **Дьяконов В. П.** Форт-системы программирования персональных ЭВМ. М.: Наука, 1992. 352 с.
9. **Деминг В. Э.** Выход из кризиса. Тверь: Альба, 1994. 120 с.
10. **Виттих В. А., Скобелев П. О.** Мультиагентные системы для моделирования процессов самоорганизации и кооперации // http://www.kg.ru/Publish/ma_r.stm.htm.
11. **Шаповалов В. И.** Теоретические принципы, лежащие в основе моделирования простейшей самоорганизующейся системы // Проблемы самоорганизации и управления в сложных коммуникационных пространствах: Матер. Первой междунар. конф. М.: Изд-во МАИ, 1997. С. 68.
12. **Леман М. М.** Программы, жизненные циклы и законы эволюции программного обеспечения // ТИИЭР. 1980. Т. 68, № 9. С. 26–45.

13. **Нариньяни А. С.** Информационные технологии 21 века: На пороге революции // http:// members.spree.com/ SIP/ futurerussia/ build/ projects/ proj6.htm.
14. **Дебора де Во.** Распределенные агенты SRI обеспечивают гибкость // Computerworld: Междунар. компьютерный еженедельник. 1997. № 4. С. 40–45.
15. **Шумский С. А.** Нейросетевые агенты в Интернете // Компьютерра. 2000. № 4. С. 20–25.
16. **Kardash D. I., Frid A. I., Frid S. A.** Adaptation of Territorial-Outlying Neural Networks // Proc. of the 8th Int. Conf. on Neural Information Processing. November 14–18, 2001, Shanghai, China (ICONIP–2001). 2001. V. 2. P. 730–734.
17. **Андреев А. Г. и др.** Microsoft RWindows 2000: Server и Professional / Под общ. ред. А. Н. Чекмарева и Д. Б. Вишнякова. СПб: БХВ–С.-Петербург, 2000. 992 с.
18. **Чепыженко А.** Результаты исследований в области архитектуры микропроцессоров для встраиваемых применений // http:// www.forth.org.ru/ drom/index.html.
19. **Intel** Architecture Software Developer's Manual. Volume 3: System Programming Guide // http://www.intel.com/design/pentium/manuals/ 24319201.pdf.
20. **Кардаш Д. И.** Алгоритмы контроля и диагностики программного обеспечения информационно-управляющих систем на основе адаптируемых языков программирования и нейронных сетей: Дис. ... канд. техн. наук. Уфа, 2001. 161 с.
21. **Kardash D. I., Frid A. I.** Checking algorithm of computing process on the basis of the Forth-system // Proc. of the 2nd Int. Workshop on Computer Science and Information Technologies. Ufa: USATU Publ., 2000. P. 52–57.
22. **Горбань А. Н., Россиев Д. А.** Нейронные сети на персональном компьютере. Новосибирск: Наука. Сиб. изд. фирма РАН, 1996. 276 с.
23. **Фрид С. А.** Состояние углеводно-энергетического и липидного обмена при артериальной гипертонии: Дис. ... канд. мед. наук. Уфа, 2000. 231 с.
24. **Кардаш Д. И., Фрид С. А.** Прогнозирование поведения биологических систем с помощью фрагментарных нейронных сетей в условиях перманентного эксперимента // Интеллектуальные системы управления и обработки информации: Тез. докл. междунар. молодежн. науч.-техн. конф. Уфа: УГАТУ, 1999. С. 138.
25. **Свид.** об офиц. рег. программы для ЭВМ № 990180. Проблемно-адаптируемая диалоговая система моделирования систем автоматического управления с использованием нейронных сетей / Д. И. Кардаш. РосАПО, 09.4.1999.
26. **Свид.** об офиц. рег. программы для ЭВМ № 991016. Интеллектуальная диалоговая система прогнозирования состояния углеводного обмена у больных артериальной гипертонией / Д. И. Кардаш, С. А. Фрид. РосАПО, 30.12.1999.

ОБ АВТОРАХ

Фрид Аркадий Исаакович, профессор каф. вычисл. техники и защиты инф. УГАТУ. Дипл. инж.-электромеханик (УАИ, 1968). Д-р техн. наук по управлению в технических системах (УГАТУ, 2000). Исследования в области управления сложными системами в условиях неопределенности.

Кардаш Денис Иванович, ст. преп. той же кафедры. Дипл. инж.-системотехник (УГАТУ, 1995). Канд. техн. наук по математическому и программному обеспечению (УГАТУ, 2001). Исследования в области информационно-управляющих систем, адаптируемого программного обеспечения, искусственного интеллекта.