УДК 004.424
DOI: http://dx.doi.org/10.21686/1818-4243-2019-3-33-41

А.М. Бершадский, А.С. Бождай, А.А. Гудков, Ю.И. Евсеева

Пензенский государственный университет, Пенза, Россия

Самоадаптация обучающих программных систем на основе наблюдения за информационной средой*

Цель исследования. *Целью проводимого исследования является разработка нового универсального метода самоадаптации прикладных программных систем, используемых в сфере электронного обучения (E-Learning). Под самоадаптацией понимается способность программного приложения изменять собственную структуру и поведение в зависимости от внешних обстоятельств, к числу которых можно отнести, например, личностные характеристики обучаемого, что особенно актуально для систем, используемых в сфере образования. Подобное самоадаптивное поведение должно быть достаточно гибким и не сводиться к выбору одного из множеств вариантов поведения, заранее определяемых разработчиком (также подобные варианты поведения должны генерироваться на протяжении жизненного цикла системы).*

Материалы и методы исследования. *В качестве исходных данных предложенный метод использует массив пользовательских отзывов о программном обеспечении, для последующей обработки которого применяются методы латентно-семантического и дистрибутивно-статистического анализа. Для представления обобщенной самоадаптивной структуры системы используются модели характеристик. Конфигурации модели характеристик представляют собой отдельные состояния самоадаптивной системы, их генерация осуществляется автоматически на протяжении жизненного цикла программы следующим образом: на основе массива пользовательских отзывов формируется семантическая сеть основных понятий, характеризующих программу, которая в дальнейшем сопоставляется с исходной моделью характеристик и личностными характеристиками пользователя, оставившего отзыв. Определение личностных характеристик пользователя может осуществляться множеством способов (например, с помощью психологического тестирования или с помощью анализа результатов обучения).*

Результаты. *Основными результатами проведенного исследования являются: 1) универсальные принципы построения самоадаптивной системы электронного обучения 2) способ представления самоадаптивной структуры программной системы в форме модели характеристик, актуальный для широкого круга программного обеспечения 3) новый универсальный метод самоадаптации прикладного программного обеспечения, используемого в сфере E-Learning, основные отличия которого от существующих заключаются, во-первых, в использовании мнений самих пользователей системы для настройки самоадаптивного поведения, во-вторых, в возможности генерации новых состояний системы на протяжении всего периода ее функционирования.*

Заключение. *Разработанный теоретический аппарат позволяет существенным образом индивидуализировать процесс обучения, учитывать мнения и склонности самих обучаемых, снизить роль педагогического работника в оценке знаний и умений. Помимо проблем исключительно образовательного характера, применение метода позволяет также успешно решить технические вопросы, связанные с разработкой программного обеспечения в целом. К числу таких проблем относится, например, проблема сложности программного обеспечения, когда программа, показывающая хорошие результаты в одних условиях функционирования, показывает недостаточно хорошую производительность в других. Также серьезной задачей, с решением которой может справиться предложенный метод, является задача увеличения жизненного цикла программной системы.*

Ключевые слова: *адаптация, виртуальная образовательная среда, E-Learning, самоадаптивные программные системы*

Alexander M. Bershadskiy, Alexander S. Bozhday, Aleksey A. Gudkov, Yulia I. Evseeva

Penza State University, Penza, Russia

Self-adaptation of e-learning software based on observing the information environment

Purpose of the research. *The purpose of the study is to develop a new universal method of self-adaptation of applied software systems used in the field of e-learning (E-Learning). Self-adaptation refers to the ability of a software application to change its own structure and behavior depending on external circumstances, which include, for example, the trainee's personal characteristics, which is especially important for systems used in education. Such self-adaptive behavior should be sufficiently flexible and not be reduced to the choice of one of the many behavioral options predetermined by the developer (such behaviors should also be generated throughout the system's life cycle).*

Materials and methods. *The method being developed uses an array of user reviews about software as initial data, for the subsequent processing of which the methods of latent-semantic and distributive-statistical analysis are used. To represent the generalized self-adaptive structure of the system, models of characteristics are used. The configuration of the model of characteristics is a separate state of the self-adaptive system, they are generated automatically during the program's life cycle as follows: based on an array of user reviews, a semantic network of basic concepts characterizing the program is formed, which is further compared with the original model of characteristics and personal characteristics of the user who left review. Determining a user's personal characteristics can be done in a variety of ways (for example, using psychological testing or by analyzing learning outcomes).*

Results. *The main results of the study are: 1) universal principles of building a self-adaptive e-learning system 2) a way of presenting the self-adaptive structure of a software system in the form of a characteristics model relevant to a wide range of software 3) a new universal method of self-adapting applied software used in E-Learning the main differences of which from the existing ones are, firstly, in using the opinions of the users of the system themselves to adjust with self-adaptive behavior, secondly, in the possibility of generating new states of the system throughout the entire period of its operation.*

Conclusion. *The developed theoretical apparatus makes it possible to significantly individualize the learning process, take into account the opinions and inclinations of the students themselves, reduce the*

* Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 18-07-00408.

role of the pedagogical worker in the assessment of knowledge and skills. In addition to problems of a purely educational nature, the application of the method also allows you to successfully resolve technical issues related to the development of software in general. Such problems include, for example, the problem of software complexity, when a program that shows good results in some operating conditions shows insufficient performance in others. Also a serious task, which the proposed method can cope with, is the task of increasing the life cycle of a software system.

Keywords: *adaptation, virtual educational environment, E-Learning, self-adaptive software*

1. Введение

На современном этапе развития общества в образовательной среде все большее внимание уделяется вопросам применения технологий адаптивного обучения. Адаптивный подход основывается на максимальном использовании потенциала личности как субъекта образовательного процесса и предполагает самостоятельные учебные действия со стороны такого субъекта, что в дальнейшем ведет к формированию личности, готовой к непрерывному самообразованию. Поскольку электронные образовательные технологии в настоящее время являются неотъемлемой составляющей любого образовательного процесса, актуальным становится вопрос создания адаптивных программных систем электронного обучения [8–10].

Основное внимание, уделяемое заказчиками программного обеспечения (ПО) в процессе разработки, направлено на оптимизацию общего объема временных и материальных затрат [11]. Неоправданное занижение этого уровня может привести к возникновению программных систем с укороченным жизненным циклом – когда недостаточная гибкость системной структуры и функционала не позволяет использовать ПО в ситуациях, которые не были прямо предусмотрены на этапе их разработки. Различные методологии разработки ПО имеют собственный взгляд на решение данных вопросов.

Зародившееся исторически первым структурное программирование сделало возможным создание крупномасштабных программных систем. Необходимым условием для этого являлось наличие строгих спецификаций, которые не должны были претерпевать серьезных изменений в ходе работы над проектом. Появившаяся десятилетие спустя концепция объектно-ориентированного программирования позволила реорганизовывать компонентный состав программ при внесении изменений в спецификацию. Однако подобные операции по-прежнему являются трудозатратными и дорогостоящими [12].

Обсуждаемое в настоящее время адаптивное программирование направлено на создание такого ПО, которое может легко адаптироваться к изменениям в потребностях и пожеланиях пользователей, а также к условиям выполнения. В адаптивных программах четко представлены действия, которые нужно выполнить, и цели, которые нужно достичь. Это позволяет пользователю изменять цели функционирования системы без необходимости переписывания исходного кода программы [13].

На сегодняшний день не существует какой-либо универсальной техники адаптивного программирования. Более того, для решения различных задач могут использоваться различные методологии. Если попытаться дать какое-либо конкретное определение адаптивному ПО, то правильнее всего будет сказать, что это программы, использующие сведения об изменениях в окружающей информационно-вычислительной среде для улучшения своей работы.

Немало исследований проводилось в области создания специализированных архитектурных решений для адаптивных программных систем. К подобным работам относится прежде всего ряд публикаций о создании адаптивных компьютерных игр и виртуальных тренажеров [14–16]. Серьезным недостатком подобных подходов является отсутствие у них универсальности даже для четко определенного класса программных систем – систем электронного обучения.

Существует также ряд подходов, сконцентрированных не на разработке специфической архитектуры, а на создании специализированного математического аппарата синтеза адаптивного программного обеспечения. К числу таких методов можно отнести, например, основанный на применении конечных автоматов с расширенными свойствами подход из работы [17] или метод адаптации программ на основе аппарата нечеткой логики из работы [18], однако и их существенными недостатками является отсутствие универсальности. Системы, построенные на таких принципах, также не являются достаточно гибкими и надежными. Известная концепция автономных вычислений (autonomic computing), подробно изложенная в работе [19], также не универсальна. Принципы автономных вычислений применимы в основном для задач конфигурирования в режиме реального времени крупных программно-аппаратных комплексов. Специфика набравших популярность в последние десятилетия агентных технологий такова, что их можно применить далеко не к каждому типу прикладных программных систем [20].

Основной задачей данного исследования является разра-

ботка такого теоретического аппарата самоадаптации прикладных программных систем (в частности, систем E-Learning), который будет лишен недостатков методов, технологий и подходов, приведенных выше. В первую очередь, такой подход должен быть универсален (поскольку системы электронного обучения могут относиться к достаточно широкому классу программного обеспечения), но прежде всего должен акцентировать мнение и ключевые характеристики самих пользователей. Предлагаемый метод концентрируется на анализе пользовательских отзывов и реконфигурировании в соответствии с ними программного обеспечения.

2. Принципы построения самоадаптивной системы электронного обучения

Ни одна из рассмотренных технологий построения адаптивного программного обеспечения не предусматривает в полной мере возможностей автоматической реорганизации системы в ответ на изменения в потребностях (или характеристиках) пользователей. Для обеспечения таких возможностей предлагается дополнительный набор принципов:

В качестве базовой самоадаптивной единицы системы выступает так называемая «характеристика». Характеристика может представлять собой как отдельный компонент системы (например, менеджер ресурсов или отдельную трехмерную модель), так и системный параметр (например, уровень сложности, на котором пользователь проходит обучение).

Каждая из характеристик имеет множество допустимых значений (например, может быть несколько уровней сложности обучающей программы и несколько уровней детализации трехмерной модели).

Все характеристики должны быть объединены в рамках единой модели. Модель также должна включать в себя отношения между различными характеристиками (например, должна учитываться совместимость конкретного значения одной характеристики с выбранными значениями других).

Основная информация о том, какие именно значения характеристик должны быть выбраны и в какой ситуации, должна поступать через рефлексивный контур обратной связи непосредственно от пользователей в специальных формах для отзывов и рецензий.

Для возможности реконфигурирования программной системы в процессе выполнения из пользовательской информации должны быть извлечены основные характеристики системы, а также их возможные значения, и представлены в структурированной форме. В дальнейшем на основе извлеченных характеристик будет подбираться необходимая конфигурация системы.

Таким образом, для реализации метода самоадаптации программной системы на основе наблюдения за информационной средой необходимо решить следующие задачи:

1. Разработать универсальное и легко формализуемое представление иерархии характеристик программной системы.
2. Разработать способ извлечения характеристик и их возможных значений из текстовой информации.
3. Разработать универсальную форму представления извлеченных из текста характеристик в структурированном виде.
4. Разработать способ автоматической генерации состояний программной системы на основе извлеченных из текстовой информации характеристик.

Таким образом, основные проблемы, связанные с разработкой метода самоадаптации, имеют отношение к анализу неструктурированных данных и представлению их в структурированном виде. Решение данных проблем следует искать в комплексом использовании ряда существующих техник обработки текстовой информации.

3. Метод самоадаптации на основе наблюдения за информационной средой

Первой задачей, требующей решения при разработке предлагаемого метода самоадаптации, является выбор оптимального представления программной системы с помощью основных единиц изменчивости – характеристик. Для решения данной задачи авторы статьи обратились к инженерии линеек программных продуктов (SPLE) – концепции повторного использования компонентов программного обеспечения, помогающей разрабатывать семейства (линейки) продуктов с сокращением времени выхода на рынок и повышением качества [1–5].

Центральным понятием концепции SPLE выступает понятие модели изменчивости. Модель изменчивости – это некоторое формализованное описание множества возможных конфигураций программной системы. Наиболее близким по смыслу понятием в технических науках является понятие морфологического множества. По сути, модель изменчивости и представляет собой морфологическое множество программной системы, дополненное некоторыми ограничениями и правилами. Данные ограничения и правила касаются вопросов совместимости отдельных компонентов потенциальной системы друг с другом, однако могут иметь и иной смысл (в зависимости от типа используемой модели изменчивости).

В SPLE используются следующие типы моделей изменчивости [3]:

Модели характеристик – модели, которые наиболее часто используется на практике. Графически отображаются в форме модифицированного И/ИЛИ-дерева, называемого диаграммой характеристик (рис. 1), могут иметь различное формализованное представление (гиперграф [6], пропозициональная формула, алгебраическая нотация и др.).

Ортогональные модели изменчивости. Схожи с моделями характеристик, графически также отображаются в форме диаграмм. Основное отличие заключается в том, что ортогональные модели показывают только наличие изменчивости в рассматриваемой программной системе, в то время как модели характеристик предоставляют более конкретное описание как предметной области, так и точек изменчивости.

Модели решений. Модель решений включает в себя следующие компоненты: вопросы из предметной области, на которые нужно получить ответы в процессе разработки программного продукта; множества возможных ответов на вопросы; ссылки на используемые компоненты (активы) или другие решения; описание последствий принятия решения (ответа на определенный вопрос или выбор определенного актива).

Для создания обобщенного описания самоадаптивной структуры программной системы было принято решение использовать модель характеристик. Преимуществами использования модели характеристик являются наглядность ее визуального представления и простота формализации [6].

Второй важной задачей является выбор структурированной формы представления характеристик, извлеченных из пользовательских отзывов и рецензий. В качестве такой формы была выбрана семантическая сеть – структурированная модель предметной области, имеющая вид ориентированного графа, вершины которого соответствуют объектам предметной области, а дуги задают отношения между ними. Объектами могут быть понятия события, свойства процессы.

Таким образом, основная задача разрабатываемого метода самоадаптации будет заключаться в построении семантической сети характеристик на основе массива пользова-

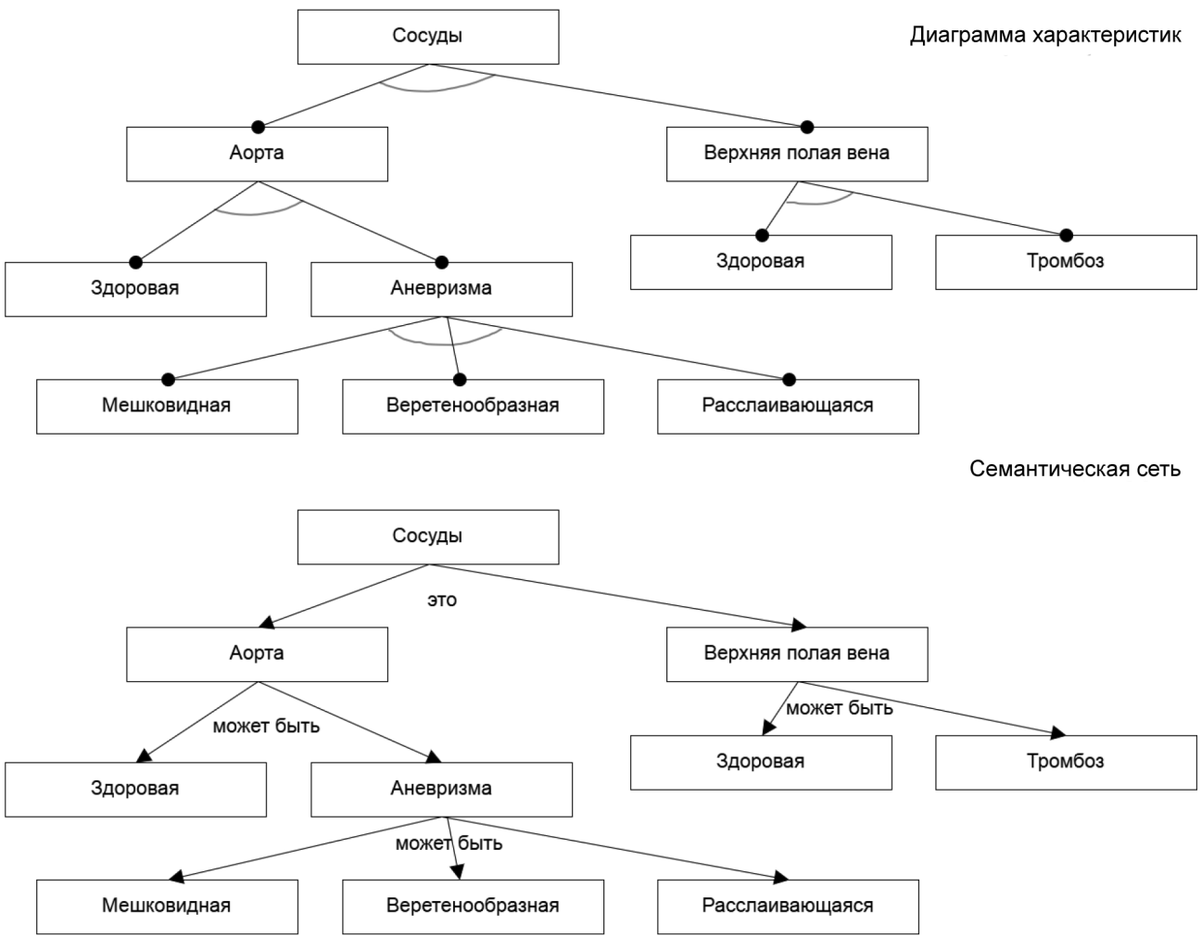

Рис. 1. Фрагмент модели изменчивости программной системы и соответствующий ей фрагмент семантической сети

тельских отзывов и составлении конфигурации диаграммы характеристик на основе полученной сети. На рисунке 1 представлен фрагмент диаграммы характеристик, описывающий предполагаемые варианты модели человеческого сердца. Данная диаграмма служит для описания поведения программы-тренажера в области медицины. Приведенный фрагмент диаграммы иллюстрирует возможные варианты повреждений органа, с которыми может столкнуться пользователь в процессе работы с тренажером. В зависимости от того, повреждена ли определенная часть сердечной мышцы или нет, а также в зависимости от того, какой тип повреждения имеет место быть, будут отображаться те или иные части трехмерной модели.

Метод самоадаптации программных систем на основе наблюдения за информационной средой включает в себя следующие этапы:

1. Сбор документов на естественном языке через специальный контур обратной связи, содержащих мнения пользователей о том, как должна работать система.

2. Извлечение терминов для определения похожих рецензий и для последующего извлечения из них характеристик программной системы. Данный этап включает следующие шаги: 1) Удаление слов, не имеющих достаточной смысловой нагрузки (предлогов, союзов и т. п.), а также специальных символов и знаков препинания из каждого документа; 2) лемматизация слов в документах (приведение различных форм слова к одной); 3) каждому слову в документе ставится в соответствие часть речи; 4) все слова, кроме существительных, прилагательных и глаголов, удаляются; 5) результатом работы алгоритма является матрица M размером $m \times n$, где m — количество терминов, n —количество документов. Элементами матрицы являются весовые коэффициенты, рассчитанные в соответствии с частотами, с которыми термины встречаются в документах.

3. Идентификация похожих документов. Как правило, матрица M является разреженной (содержит много нулевых элементов), что затрудняет ее анализ. Для устранения этого недостатка выполняют процедуру понижения размерности матрицы с использованием ее сингулярного разложения:

$$M = USV,$$

где матрицы U и V — ортогональные, а матрица S — диагональная. На главной диагонали матрицы S располагаются сингулярные числа матрицы M, в порядке убывания. Для снижения размерности матрицы M берут r первых сингулярных чисел (например, первые два). Соответственно, из матрицы U берутся r первых столбцов, а из матрицы V — r первых строк. Полученные таким образом матрицы обозначим как U', S', V'. Матрица $M' = U' S' V'$ будет являться приближением матрицы M с меньшим рангом r. Для определения похожих документов используют матрицу $D = S' V'$, имеющую размерность $r \times n$. При $r = 2$ столбцы матрицы D можно рассматривать как точки в двухмерном пространстве, соответствующие исходным документам. Далее следует фаза кластеризации документов по значениям матрицы D. Самым простым способом кластеризации является использование метода k-средних, основная задача которого сводится к оптимизации целевой функции (расстояния), задаваемой с помощью формулы

$$E = \sum_{i=1}^{c} \sum_{x \in C_i} d(x_i, m_i),$$

где m_i — центр кластера C_i, а $d(x_i, m_i)$ — евклидово расстояние между точками x_i и m_i. Вначале алгоритм разбивает все множество данных на c кластеров, затем случайным образом выбираются центры кластеров. Затем для каждой точки определяется ближайший к ней центр кластера и происходит повторное вычисление центров кластеров. Последние 2 шага повторяются до тех пор, пока центры кластеров не перестанут изменяться. Как альтернативный вариант можно рассматривать метод нечеткой кластеризации C-средних, который, по сути, является развитием метода k-средних. Метод включает в себя следующие этапы: 1) случайным образом выбираются центры кластеров; 2) инициализируется матрица принадлежности элементов к кластерам U; 3) значения U_j рассчитываются по формуле

$$U_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}}$$

4) вычисляются центры кластеров по формуле

$$C_k = \frac{\sum_{i=1}^{N} u_{ij}^m \cdot x_i}{\sum_{i=1}^{N} u_{ij}^m}$$

5) обновление значений U^K, U^{K+1}; 6) если $\|U^{(K+1)} - U^{(K)}\| < \varepsilon$, то кластеризация проведена оптимальным образом, в противном случае необходимо повторить действия, начиная с пункта 2.

4. Выделение характеристик в каждом кластере. В данном случае задача выделения характеристик будет сведена к задаче извлечения устойчивых словосочетаний. В данной работе для ее решения предлагается оценка взаимосвязанности слов в виде

$$c = \frac{t_1 + t_2}{2 \cdot f\left(t^1 t^2\right)},$$

где t_1 и t_2 отражает число различных пар с первым или вторым словом из рассматриваемого словосочетания, $f(t^1 t^2)$

отражает число появлений словосочетания типа «первое-второе слово вместе». Чем меньше значение c, тем лучшей считается пара слов (t^1, t^2).

5. Выявление взаимосвязей между выделенными характеристиками и построение семантической сети для каждого кластера. Для решения данной задачи предлагается использовать дистрибутивно-статистический метод. Основная гипотеза, лежащая в его основе, звучит следующим образом: значимые элементы языка, встречающиеся в пределах некоторого текстового интервала, семантически связаны между собой. Метод включает в себя два этапа: 1) расчет количественных (частотных) характеристик одиночной и совместной встречаемости значимых элементов языка; 2) расчет сил связей между элементами и составление матрицы связей между характеристиками. Центральным понятием в дистрибутивно-статистическом анализе является понятие контекста. Под контекстом понимается отрезок текста, последовательность (цепочка) синтагм. Синтагма – это отрезок предложения, состоящий из одного или нескольких слов, объединенных грамматически, интонационно и по смыслу. Минимальной длиной синтагмы следует считать простые словосочетания. Формально контекст определяется следующим образом:

$$T = C_1(T) + ... + C_q(T),$$

где $C_i(T) \cap C_j(T) = \varnothing$, i, j $(i \neq j) \in [1, q]$.Частотные характеристики рассчитываются следующим образом:

$$f_A = \frac{N_A}{N},\ f_B = \frac{N_B}{N},\ f_{AB} = \frac{N_{AB}}{N},$$

где N – общее количество контекстов, N_A и f_A – количество и частота контекстов, где встретилось только характеристика A, N_B и f_B – количество и частота контекстов, где встретилось характеристика B, N_{AB} и f_{AB} – количество и частота контекстов, в которых наблюдалась совместная встречаемость характеристик A и B. Следующим шагом является расчет коэффициентов силы связи, для которого можно использовать формулы, приведенные в табл. 1.

Все формулы коэффициентов силы связи объединяет рассмотрение событий, связанных с появлением синтагм A и B, как системы случайных явлений. Для определения тематических связей между характеристиками длина контекста должна составлять 50–100 слов. На основе расчетов коэффициентов сил связи строится матрица семантической связности характеристик. На основе матрицы осуществляется построение семантической сети. Узлами сети будут являться характеристики, извлеченные из отзывов, дуги, связывающие характеристики, будут показывать наличие смысловой связи между ними.

6. Формирование конфигураций диаграммы характеристик. Для каждого кластера документов необходимо осуществить сопоставление полученной семантической сети с моделью характеристик системы и сформировать таким образом конфигурацию; при этом при формировании каждой конфигурации должно учитываться, от какого именно типа пользователя были получены отзывы (с этой целью необходимо предварительно определить усредненный портрет пользователей, чьи рецензии сформировали кластер).

7. Дальнейший процесс самоадаптации заключается в том, что пользователю, относящемуся к некоторому типу, будет ставиться в соответствие определенная конфигурация, т.е. система будет подстраиваться под конкретных пользователей. Предполагается, что отзывы могут оставлять только зарегистрированные в системе пользователи, профиль которых содержит достаточно полную информацию о них.

В качестве дальнейшего направления исследований можно рассматривать усовершенствование разработанного метода. В первую очередь, необходимо решить проблему синонимов, поскольку одна и та же характеристика может в различных пользовательских отзывах иметь различные названия. Второй проблемой является учет того, что пользователи могут допускать орфографические ошибки и опечатки при написании отзывов. Необходимо также реализовать автоматическое исправление орфографии. В текущем варианте метода в качестве характеристик рассматриваются устойчивые словосочетания из 2 слов, однако в определенных случаях компоненты и параметры программной системы

Таблица 1

Формулы расчета сил связи между синтагмами.

Формула	Кем предложена
$K_{AB} = \frac{N_{AB}}{N_A + N_B - N_{AB}}$	Т. Танимото, Л. Дойл [7]
$K_{AB} = \frac{N_{AB} - f_A \cdot f_B}{N}$	М. Мэйрон, Дж. Кунс [7]
$K_{AB} = \frac{f_A \cdot N}{f_A \cdot f_B}$	А.Я. Шайкевич, Дж. Солтон, Р. Куртис [7]
$K_{AB} = \frac{N_{AB} - N_A \cdot N_B}{\sqrt{N_A \cdot N_B}}$	С. Деннис [7]
$K_{AB} = \log_{10} \frac{\left[\left(f_{AB} \cdot N - f_A \cdot f_B\right) - \frac{N}{2}\right]^2 \cdot N}{f_A \cdot f_B \cdot (N - f_A) \cdot (N - f_B)}$	Х.Е. Стайлз [7]

могут иметь и более сложные составные названия, и для их корректного выделения необходимо усовершенствовать механизм поиска устойчивых словосочетаний.

5. Заключение

Адаптивные обучающие программы способны индивидуализировать процесс обучения и обеспечить более качественное усвоение материала пользователем. Существует множество техник построения таких систем, но ни один из них не применим к широкому классу систем.

Помимо этого, существует также проблема сложности программного обеспечения, которая, в свою очередь, порождает проблему снижения качества его функционирования. Заранее сложно учесть, как поведет себя система при различных внешних условиях, и на подобные оценки не всегда есть время. Программа, показывающая хорошие (например, в плане производительности) результаты в одних ситуациях, может недостаточно хорошо работать в других. Более того, разработка сложной программной системы, предусматривающей функционирование в различном программно-аппаратном окружении и при различных условиях (в том числе и пользовательских), может стать длительной и ресурсозатратной задачей.

Решение проблем лежит в области разработки специализированных методов самоадаптации программных систем. Существует немало подходов к разработке адаптивного программного обеспечения, однако ни один из них не решает важной задачи – не обеспечивает автоматическую реорганизацию системы в ответ на изменения в потребностях пользователей. Для ее решения авторами был предложен новый метод, согласно которому самоадаптируемая структура системы представляется с точки зрения основных единиц изменчивости – характеристик, а также их возможных значений. Для реализации самоадаптивного поведения используются массивы пользовательских отзывов, из которых извлекаются характеристики, представляются в форме семантической сети и в дальнейшем для формирования системной конфигурации сопоставляются с общесистемной моделью характеристик.

Литература

1. Schobbens P. E., Heymans P., Trigaux J.C. Feature diagrams: a survey and formal semantics // 14th IEEE International Requirements Engineering Conference (RE'06). Washington: IEEE Computer Society, 2011. P. 139–148.
2. Kang K. C. et al. Feature-oriented domain analysis (FODA): feasibility study. Pittsburgh: Software Engineering Institute, 1990. 161 p.
3. Sinnema M., Deelstra S. Classifying variability modeling techniques // Information and software technology. 2007. №7. P. 42–54.
4. Dinkelaker T. et al. On goal–based variability acquisition and analysis // 14th IEEE International Requirements Engineering Conference (RE'06). Washington: IEEE Computer Society, 2010. P. 77–85.
5. Berger T. Variability modeling in the real: an empirical journey from software product lines to software ecosystems. Leipzig: University of Leipzig, 2012. 225 p.
6. Бершадский А.М., Бождай А.С., Евсеева Ю.И., Гудков А.А Математическая модель рефлексии самоадаптивных программных систем // Известия Волгоградского государственного технического университета. 2018. № 2 (218). С. 7–14.
7. Москович В.А. Информационные языки. М.: Наука, 1971.
8. Захарова О.А. Виртуальная образовательная среда в профессиональной подготовке и системе повышения квалификации: монография. Ростов н/Д.: Издательский центр ДГТУ, 2011. 146 с.
9. Захарова О.А. Интерактивное повествование и мультимедиа в системе профессионального обучения и повышения квалификации // Мир науки, культуры, образования. 2013. № 1(38). С. 21–24.
10. Ritke-Jones William. Virtual Environments for Corporate Education: Employee Learning and Solutions. Cybernations Consulting Group, 2010. 426 p.
11. Колоденкова А. Е. Задачи программного инжиниринга сложных систем на основе критерия жизнеспособности проекта // Проблемы управления и моделирования в сложных системах: Труды XII Международной конференции. Самара: Самарский НЦ РАН, 2010. С. 593–598.
12. Рамбо Дж, Блаха М. UML 2.0. Объектно-ориентированное моделирование и разработка. Изд. 2-е. СПб.: Питер, 2007.
13. Линда Дейли Полсон. Разработчики переходят на динамические языки // Открытые системы. 2007. № 2.
14. Wang A.L., Nordmark N. Software architectures and the creative processes in game development. In: Entertainment computing. Eds. K. Chorianopoulos et al. Cham: Springer International Publishing, 2015. P. 272–285.
15. Cornforth D.J., Adam M.T. Cluster evaluation, description and interpretation for serious games. In: Serious games analytics. Eds. C.S. Loh, Y. Sheng, D. Ifenthaler. Cham: Springer International Publishing, 2015. P. 135–155.
16. Carvalho M.B. et al. The journey: a service-based adaptive serious game on probability. In:

Serious games analytics. Eds. C.S. Loh, Y. Sheng, D. Ifenthaler. Cham: Springer International Publishing, 2015. P. 97–106.

17. Wang P., Cai, K. Y. Supervisory control of a kind of extended finite state machines // 24th IEEE Chinese Control and Decision Conference (CCDC). Washington: IEEE Computer Society, 2012. P. 775–780.

18. Yang Q., Lü J., Xing J., Tao X., Hu H., Zou Y. Fuzzy control-based software self-adaptation: A case study in mission critical systems // IEEE 35th Annual Computer Software and Applications Conference Workshops (COMPSACW). Washington: IEEE Computer Society, 2011. P. 13–18.

19. Ahuja K., Dangey H. Autonomic Computing: An emerging perspective and issues // IEEE International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT 2014). Washington: IEEE Computer Society, 2014. P. 471–475.

20. Sim K. M. Agent-based cloud computing // IEEE Transactions on Services Computing. Washington: IEEE Computer Society, 2012. P. 564–567.

References

1. Schobbens P. E., Heymans P., Trigaux J.C. Feature diagrams: a survey and formal semantics. 14th IEEE International Requirements Engineering Conference (RE'06). Washington: IEEE Computer Society; 2011: 139-148.

2. Kang K. C. et al. Feature-oriented domain analysis (FODA): feasibility study. Pittsburgh: Software Engineering Institute; 1990. 161 p.

3. Sinnema M., Deelstra: Classifying variability modeling techniques. Information and software technology. 2007; 7: 42-54.

4. Dinkelaker T. et al. On goal–based variability acquisition and analysis. 14th IEEE International Requirements Engineering Conference (RE'06). Washington: IEEE Computer Society; 2010: 77-85.

5. Berger T. Variability modeling in the real: an empirical journey from software product lines to software ecosystems. Leipzig: University of Leipzig; 2012. 225 p.

6. Bershadskiy A.M., Bozhday A.S., Evseyeva YU.I., Gudkov A.A Mathematical Model of the Reflection of Self-Adaptive Program Systems. Izvestiya Volgogradskogo gosudarstvennogo tekhnicheskogo universiteta = Bulletin of Volgograd State Technical University. 2018; 2(218): 7-14. (In Russ.)

7. Moskovich V.A. Informatsionnyye yazyki = Informational languages. Moscow: Science; 1971. (In Russ.)

8. Zakharova O.A. Virtual'naya obrazovatel'naya sreda v professional'noy podgotovke i sisteme povysheniya kvalifikatsii: monografiya = Virtual educational environment in vocational training and advanced training system: monograph. Rostov n/D.: Publishing Center DGTU; 2011. 146 p. (In Russ.)

9. Zakharova O.A. Interactive narration and multimedia in the system of vocational training and professional development. Mir nauki, kul'tury, obrazovaniya = World of science, culture, education. 2013; 1(38): 21-24. (In Russ.)

10. Ritke-Jones William. Virtual Environments for Corporate Education: Employee Learning and Solutions. Cybernations Consulting Group; 2010. 426 p.

11. Kolodenkova A. E. Tasks of software engineering of complex systems based on the criterion of viability of the project. Problemy upravleniya i modelirovaniya v slozhnykh sistemakh: Trudy XII Mezhdunarodnoy konferentsii = Problems of control and modeling in complex systems: Proceedings of the XII International Conference. Samara: Samara Scientific Center of the Russian Academy of Sciences; 2010: 593-598. (In Russ.)

12. Rambo Dzh, Blakha M. UML 2.0. Ob"yektno-oriyentirovannoye modelirovaniye i razrabotka. Izd. 2-e. = UML 2.0. Object-oriented modeling and development. Ed. 2nd. Saint Petersburg: Piter; 2007. (In Russ.)

13. Linda Deyli Polson. Developers switch to dynamic languages. Otkrytyye sistemy = Open Systems. 2007; 2. (In Russ.)

14. Wang A.L., Nordmark N. Software architectures and the creative processes in game development. In: Entertainment computing. Eds. K. Chorianopoulos et al. Cham: Springer International Publishing; 2015; 272-285.

15. Cornforth D.J., Adam M; Cluster evaluation, description and interpretation for serious games. In: Serious games analytics. Eds. C: Loh, Y. Sheng, D. Ifenthaler. Cham: Springer International Publishing; 2015; 135-155.

16. Carvalho M.B. et al. The journey: a service-based adaptive serious game on probability. In: Serious games analytics. Eds. C: Loh, Y. Sheng, D. Ifenthaler. Cham: Springer International Publishing; 2015; 97-106.

17. Wang P., Cai, K. Y. Supervisory control of a kind of extended finite state machines. 24th IEEE Chinese Control and Decision Conference (CCDC). Washington: IEEE Computer Society; 2012; 775-780.

18. Yang Q., Lü J., Xing J., Tao X., Hu H., Zou Y. Fuzzy control-based software self-adaptation: A case study in mission critical systems. IEEE 35th Annual Computer Software and Applications Conference Workshops (COMPSACW). Washington: IEEE Computer Society; 2011; 13-18.

19. Ahuja K., Dangey H. Autonomic Computing: An emerging perspective and issues. IEEE International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT 2014). Washington: IEEE Computer Society; 2014; 471-475.

20. Sim K. M. Agent-based cloud computing. IEEE Transactions on Services Computing. Washington: IEEE Computer Society; 2012; 564-567.

Сведения об авторах

Александр Моисеевич Бершадский
д.т.н., профессор, заведующий кафедрой САПР
Пензенский государственный университет,
Пенза, Россия
Эл. почта: bam@pnzgu.ru

Александр Сергеевич Бождай
д.т.н., доцент, профессор кафедры САПР
Пензенский государственный университет,
Пенза, Россия
Эл. почта: bozhday@yandex.ru

Алексей Анатольевич Гудков
к.т.н., доцент кафедры САПР
Пензенский государственный университет,
Пенза, Россия
Эл. почта: alexei-ag@yandex.ru

Юлия Игоревна Евсеева
к.т.н., доцент кафедры САПР
Пензенский государственный университет,
Пенза, Россия
Эл. почта: shymoda@mail.ru

Information about the authors

Aleksander M. Bershadskiy
Dr. Sci. (Engineering), Professor, Head of the Department of CAD
Penza State University, Penza, Russia
E-mail: bam@pnzgu.ru

Aleksander S. Bozhday
Dr. Sci. (Engineering), Associate Professor, Professor of the Department of CAD
Penza State University, Penza, Russia
E-mail: bozhday@yandex.ru

Aleksey A. Gudkov
Cand. Sci. (Engineering), Associate Professor of the Department of CAD
Penza State University, Penza, Russia
E-mail: alexei-ag@yandex.ru

Yulia I. Evseeva
Cand. Sci. (Engineering), Associate Professor of the Department of CAD
Penza State University, Penza, Russia
E-mail: shymoda @ mail.ru