УДК 004.4
DOI: http://dx.doi.org/10.21686/1818-4243-2017-4-13-20

А.С. Бождай, Ю.И. Евсеева, А.А. Гудков
Пензенский государственный университет, Пенза, Россия

Применение методов интеллектуального анализа данных для реализации рефлексивной адаптации в системах e-learning[1]

В последние годы технологии электронного обучения (e-learning) стремительно набирают обороты в своем развитии. В связи с этим актуальными становятся вопросы, связанные с повышением качества программного обеспечения виртуальных образовательных систем: увеличение срока непрерывной эксплуатации программ, повышение их надежности и гибкости. Перечисленные характеристики напрямую зависят от способности программной системы адаптироваться к изменениям в предметной области, условиям внешней среды и особенностям пользователей. В ряде случаев такая способность сводится к своевременной оптимизации программой собственных интерфейсов и структур данных. В настоящее время известно несколько подходов к созданию механизмов самооптимизации программных систем, но все они отличаются недостаточной степенью формализованности и, как следствие, универсальности. Целью данной работы является разработка основ технологии самооптимизации программных систем в составе e-learning. В основе предлагаемой технологии лежит сформулированный и формализованный принцип рефлексивной адаптации программного обеспечения, применимый к широкому классу программных систем и основанный на выявления новых знаний в поведенческой продукции системы.
Для решения поставленной задачи применялись методы интеллектуального анализа данных. Интеллектуальный анализ данных позволяет находить закономерности в функционировании программных систем, которые могут быть неочевидными на этапе их разработки. Нахождение таких закономерностей и последующий их анализ позволят реорганизовать структуру системы более оптимальным образом и без вмешательства человека, что позволит продлить жизненный цикл программного обеспечения и снизить затраты на его сопровождение. Достижение данного эффекта имеет важность для систем e-learning, поскольку они являются достаточно дорогостоящими.
К основным результатам работы следует отнести: предложенную классификацию механизмов адаптации программного обеспечения, учитывающую новейшие тенденции в IT-сфере в целом и сфере e-learning в частности; формулирование и формализацию принципа рефлексивной адаптации в программных системах, применимого к широкому классу прикладных программ; разработку универсального архитектурного шаблона программной системы, ориентированной на реструктуризацию в процессе эксплуатации; алгоритм самооптимизации пользовательского интерфейса программной системы на основе методов интеллектуального анализа данных.
Разработка теоретических основ автоматической реорганизации программного обеспечения e-learning позволит повысить гибкость виртуальной образовательной среды и увеличить срок ее непрерывной эксплуатации. В отличие от существующих аналогов, предлагаемые в статье методы являются универсальными и применимы к широкому классу прикладных программ. Данное обстоятельство является актуальным для систем электронного обучения, поскольку входящие в их состав подсистемы могут иметь различный тип и назначение (например, компонентами одной системы могут быть виртуальные тренажеры и информационно-библиотечное обеспечение).

Ключевые слова: *адаптация, виртуальная образовательная среда, e-learning, информационные системы.*

A.S. Bozhday, Y.I. Evseeva, A.A. Gudkov
Penza State University, Penza, Russia

Data mining methods application in reflexive adaptation realization in e-learning systems

In recent years, e-learning technologies are rapidly gaining momentum in their evolution. In this regard, issues related to improving the quality of software for virtual educational systems are becoming topical: increasing the period of exploitation of programs, increasing their reliability and flexibility. The above characteristics directly depend on the ability of the software system to adapt to changes in the domain, environment and user characteristics. In some cases, this ability is reduced to the timely optimization of the program's own interfaces and data structure. At present, several approaches to creating mechanisms for self-optimization of software systems are known, but all of them have an insufficient degree of formalization and, as a consequence, weak universality. The purpose of this work is to develop the basics of the technology of self-optimization of software systems in the structure of e-learning. The proposed technology is based on the formulated and formalized principle of reflexive adaptation of software, applicable to a wide class of software systems and based on the discovery of new knowledge in the behavioral products of the system.
To solve this problem, methods of data mining were applied. Data mining allows finding regularities in the functioning of software systems, which may not be obvious at the stage of their development. Finding such regularities and their subsequent analysis will make it possible to reorganize the structure of the system in a more optimal way and without human intervention, which will prolong the life cycle of the software and reduce the costs of its maintenance. Achieving this effect is important for e-learning systems, since they are quite expensive.
The main results of the work include: the proposed classification of software adaptation mechanisms, taking into account the latest trends in the IT field in general and in the field of e-learning in particular; Formulation and formalization of the principle of reflexive adaptation in software systems applicable to a wide class of applied programs; The development of a universal architectural template of the software

[1]Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 15-07-01553.

system, oriented to restructuring in the process of operation; Algorithm for self-optimization of the user interface of the software system based on methods of data mining.
The development of the theoretical basis for the automatic reorganization of e-learning software will increase the flexibility of the virtual educational environment and increase the period of its exploitation. Unlike existing analogues, the methods proposed in the article are universal and applicable to a wide class of applied programs. This is relevant for e-learning systems, because their may have a different types and purposes (for example, virtual simulators and information library software may be components of one system).

Keywords: *Adaptation, virtual educational environment, e-learning, information systems.*

Введение

Очень востребованные и широко используемые на сегодняшний день системы электронного обучения(e-learning) являются достаточно дорогостоящими. По этой причине необходимо, чтобы они имели длительный жизненный цикл, а следовательно, могли приспосабливаться к изменениям в предметной области и окружающей среде, а также способствовали экономии времени и ресурсов, затрачиваемых на сопровождение. Выбор методов для реализации адаптивности в системах электронного обучения – первоочередная задача, решение которой позволит в дальнейшем разработать более совершенные технологии проектирования таких систем.

Проведенный анализ адаптивного программного обеспечения позволил выделить 4 основных механизма адаптации в программных системах. Характеристики механизмов представлены в табл. 1.

Runtime-адаптация осуществляется программной системой в процессе ее функционирования и характеризуется относительной быстротой реструктуризации программы. Наиболее часто адаптацией по runtime-типу характеризуются системы-тренажеры: собирая данные об обучаемом в процессе выполнения, они без существенных задержек в функционировании программы формируют индивидуальную траекторию обучения. Другая область возможного применения моделей времени выполнения – построение так называемых самовосстанавливающихся систем [1, 2]. Такие системы должны контролировать собственную надежность и безопасность, а также быть способными автоматизировать задачи, которые зачастую приводят к сбоям системы и требуют внимания специалистов.

Реализация крупных образовательных сред (виртуальных, университетов, лабораторий, технопарков) требует повышенной отказоустойчивости, а следовательно, и включения в них механизмов самовосстановления.

Успешной реализацией runtime-адаптации в e-learning можно считать проект Smart Sparrow, используемый рядом австралийских университетов и школ [3].

Адаптация предметной области используется преимущественно в информационных системах и системах поддержки принятия решений (СППР). От runtime-адаптации она отличается, во-первых, обязательным участием пользователя (проектировщика или эксперта) при внесении изменений в модели предметной области, а во-вторых, необходимостью временного вывода из эксплуатации модифицируемой системы. Впервые данный тип адаптации был применен в ERP-системах, называемых также системами управления ресурсами предприятия [4, 5].

Адаптация предметной области активно применяется в системах e-learning. К числу таких систем относится, например, используемая в немецком Институте Бизнес-информатики платформа электронного обучения OpenUSS [6].

Рефлексивная адаптация характеризуется наличием и сочетанием черт механизмов адаптации предыдущих рассмотренных типов. Как и runtime-адаптация, она не требует существенного участия эксперта для самомодификации системы, а также вывода модифицируемой системы из эксплуатации. Однако, как и в случае с адаптацией предметной области, ей необходимо определенное время для анализа текущего состояния программы и подготовки рекомендаций о ее последующей реструктуризации. В отличие от runtime-адаптации, рефлек-

Таблица 1

Характеристики механизмов адаптации в программных системах

Механизм адаптации	Используемые модели	Используемые методы
Адаптация предметной области	Модели предметной области	Методы инженерии знаний
Рефлексивная адаптация	Модели самонаблюдения	Методы интеллектуального анализа данных, динамического анализа программного кода
Runtime-адаптация	Модели времени выполнения	Методы искусственного интеллекта, работающие с малыми объемами данных и характеризующиеся относительным быстродействием
Адаптация на основе наблюдения за информационной средой	Модели наблюдения за внешней средой	Методы работы с большими данными (Big Data)

сивная адаптация не приводит к автоматической самомодификации в процессе выполнения. Ее основным назначением является “offline”-анализ поведения системы за счет использования информации о ее внутреннем устройстве и формирование на его основе решений о возможной реструктуризации.

Элементы рефлексивной адаптации успешно реализованы в системе электронного обучения WebCT, разработанной и впервые внедренной в Университете Британской Колумбии [7]. В WebCT реализован механизм Web Mining – оптимизации интерфейса в соответствии с запросами пользователя.

Адаптация на основе наблюдения за информационной средой является пока что слабо проработанным механизмом реализации адаптивного поведения в программных системах, и большая часть работ в данном направлении посвящена в основном перспективам использования технологии Big Data в построении адаптивных систем подобного плана [8]. Основная идея, лежащая в основе этого механизма адаптации, заключается в использовании различных методов сбора и анализа большого количества данных, относящихся к предметной области программной системы, и последующей реструктуризации системы на основе полученных в результате анализа выводов Вопрос об успешной реализации данного механизма адаптации в системах электронного обучения в настоящее время является дискуссионным [9].

Существует определенное количество научных работ, посвященных практической реализации тех или иных адаптации. Так, runtime-адаптации посвящены работы [10, 11, 12, 13], рефлексивной адаптации – работы [14, 15, 16, 17] и адаптации предметной области – работы [18, 19, 20]. Однако все они объединены одним серьезным недостатком: отсутствием фундаментальных математических моделей, методов и принципов, характеризующих тот или иной адаптивный механизм. По этой причине до сих пор не были созданы методы адаптации, применимые к широкому классу программного обеспечения. Эта проблема особенно актуальна для систем электронного обучения, поскольку в их состав могут входить подсистемы различного типа и назначения (например, виртуальные тренажеры и информационные системы). Целью данной работы является формулирование и формализация принципа рефлексивной адаптации, а также разработка метода рефлексивной самооптимизации пользовательского интерфейса, применимого к широкому классу систем. Дальнейшим развитием идей, заложенных в данной работе, будет создание и обобщение новых универсальных моделей и методов реализации адаптивного поведения в системах e-learning.

1. Рефлексивная адаптация виртуальной образовательной среды

Рефлексивная адаптация может быть реализована в обучающих системах, в частности, виртуальных тренажерах. Протоколирование поведения различных пользователей на протяжении определенного промежутка времени и последующий анализ полученных данных позволит устранить из поведения тренажера проблемные моменты. Примером может послужить виртуальный медицинский тренажер для проведения операции лапароскопии. Ход работы подобных тренажеров обычно разделен на несколько этапов (наложение пневмоперитонеума, ввод троакара для оптического инструмента, осмотр органов по строго отработанной схеме и т.д.), каждый из которых характеризуется собственным уровнем сложности. Протоколирование того, как различные пользователи проходят различные этапы операции, позволит определить, не является ли сложность некоторых отдельных этапов неадекватно завышенной или заниженной, а затем автоматически внести исправления в структуру тренажера.

Другое назначение рефлексивной адаптации в виртуальных образовательных средах – предоставить обучаемому максимально комфортные условия для усвоения учебного материала. Примером рефлексивной адаптации такого типа может служить оптимизация пользовательского интерфейса. В качестве данных, используемых механизмом адаптации, выступают протоколы взаимодействия обучаемого с интерфейсами обучающей системы. Например, в состав некоторой виртуальной образовательной среды включена информационная библиотечная система. Главное меню этой системы включает в себя пункты, соответствующие группам литературных источников по основным направлениям обучения (математика, машиностроение, юриспруденция и т.д.). Тем не менее, протоколирование поисковых запросов пользователей в информационной системе может выявить отсутствие ссылок на группы источников по отдельным направлениям (например, робототехнике). Механизм рефлексивной адаптации способен выявить этот недостаток и автоматически исправить его.

Как показывает практика, гибкость архитектуры оказывает огромное влияние на способность программы адаптироваться к изменениям и усложнениям. Гибкая архитектура – это, как правило, результат детального моделирования и тщательного выбора проектных решений.

Построение универсального шаблона гибкой архитектуры для абсолютно всех возможных в реализации программных систем вряд ли является выполнимой задачей. Однако, если рассматривать организацию системы только с точки зрения интересующих свойств (например, адаптивности), можно создать архитектурный шаблон, позволяющий проектировать программы, оптимальные именно с позиции рассматриваемого свойства.

На рис. 1 приведена упрощенная архитектурная модель адаптивной системы. Основная идея предлагаемой модели заключается в том, что компоненты каждого из существующих архитектурных уровней будут собственным образом реагировать на необходимость в реорганизации.

Интерфейсный уровень архитектуры содержит объекты-интерфейсы, предназначенные для взаимодействия с конечным пользователем. Уровень данных объединяет используемую в работе системы информацию (например, информацию из баз данных). Уровень функционалов включает в себя 3 уровня: уровень функционалов предметной области (программные операции, выполняемые отдельными объектами предметной области); уровень операционных функционалов (распределяют задачи между объектами предметной области); уровень инфраструктурных функционалов (обеспечивают непосредственную техническую поддержку для верхних уровней).

Наибольший интерес представляют модели предметной области. Благодаря специализированным предметно-ориентированным языкам, конечный пользователь системы видит их как цельные понятия предметной области. Однако окружающей архитектурной средой эти понятия представляют собой комбинацию

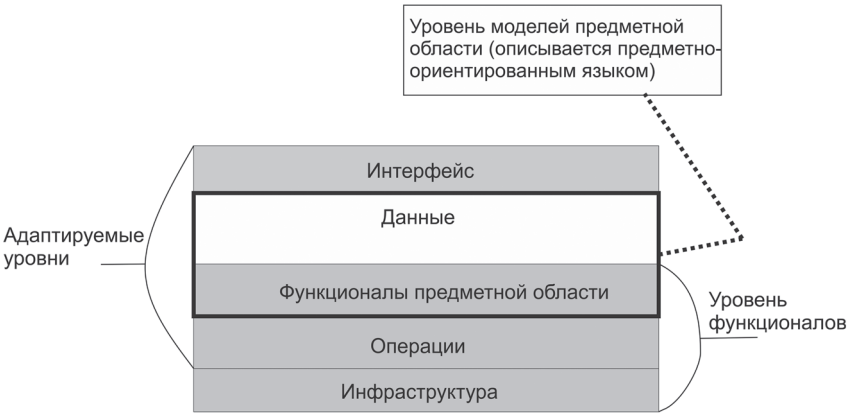

Рис. 1. Архитектурные уровни адаптивной информационной системы

компонентов 2 смежных уровней – уровня данных и уровня функционалов. С этой точки зрения, модели предметной области являются аналогами классов в объектно-ориентированном проектировании, однако в предлагаемой модели разделение между функциональным уровнем и уровнем данных акцентируется и в том случае, когда речь идет о цельном понятии предметной области. Это связано с тем фактом, что функционалы и данные различным образом реагируют на необходимость в изменениях.

Для рассмотренной ориентированной на самомодификацию архитектуры механизм рефлексии определяется следующим соотношением:

$$S_A = A(S, L(S, V)) \qquad (1)$$

где S_A – модифицированная структура системы; A – функция анализа поведенческой продукции системы и модификации ее первоначальной структуры; L – функция вычисления поведенческой продукции системы; S – исходная структура системы (включает совокупность объектов адаптируемых уровней архитектуры); V – совокупность внешних воздействий на адаптируемые объекты.

Применительно к S также справедливо

$$S = \{I, D, F_A, M\},$$

где I – множество объектов уровня интерфейсов; D – множество объектов уровня данных; $F_A = F \backslash F_I = \{F_D, F_O\}$ – множество адаптируемых функционалов; M – множество объектов предметной области. К адаптируемым функционалам относятся функционалы уровня предметной области (F_D) и функционалы операционного уровня (F_O). Функционалы уровня инфраструктуры (F_I) не являются адаптируемыми объектами.

Множество объектов предметной области представляет собой неориентированный гиперграф

$$M = \{W_D, W_F, E\},$$

где W_D – множество вершин, отображающих используемые структуры данных (точки на уровне данных рис. 2); W_F – множество вершин, отображающих функционалы предметной области; E – семейство непустых подмножеств множества $W_D \cup W_F$ (цельных объектов предметной области).

Следует отметить, что адаптация уровня целостных объектов предметной области не сводима к адаптации отдельных компонентов этих объектов (функционалов и данных). Помимо оптимизации функционалов и данных, она включает в себя также изменение структуры целостного объекта. Изменение структуры объектов реализуется через операции над гиперграфом M.

2. Реализация рефлексивной адаптации методами интеллектуального анализа данных

В качестве примера рефлексивной адаптации рассмотрим самооптимизацию пользовательского интерфейса. Поскольку создание пользовательского интерфейса представляет собой задачу, решаемую человеком (разработчиком программного обеспечения или дизайнером), нет абсолютной гарантии того, что разработанный интерфейс будет оптимальным для всех категорий пользователей.

Рефлексию объектов уровня интерфейсов можно определить следующим образом:

$$I_A = A(I, L_I(S, V)), \qquad (2)$$

где I_A – адаптированная структура системы на уровне интерфейсов; L_I – функция сбора информации о функционировании интерфейсов на некотором временном промежутке.

На рис. 2 представлена схема пространства интерфейсов. Как видно из рисунка, пространство интерфейсов организовано иерархическим образом. Сплошными линиями изображены отношения агрегации между элементами, пунктирными – отношения вызова. Примеры отношения вызова – отображение дочернего окна при нажатии на пункт меню, переход по гиперссылке на другую страницу веб-сайта.

Таким образом, пространство интерфейсов можно представить как кортеж из 2 множеств:

$$I = (C, B).$$

Множество C описывает структуру пространства на уровне элементов:

$$C = \{C_1, C_2, C_3, ..., C_m\},$$

$$C_i = (T_i, P_i, DB_i, FB_i)\ \forall i = 1, ..., m,$$

где m-число элементов интерфейса; $T_i = 0|1$ – тип i-го интерфейса (элемент-контейнер, используемый только для отображения или вызова дочерних интерфейсов, или функционирующий элемент, связанный с определенными функционалами); P_i – множество параметров i-го интерфейса (например, ширина и высота окна); $DB_i \subseteq D$ – множество идентификаторов связанных с i-м интерфейсом данных; $FB_i \subseteq F$ – множество идентификаторов связанных с i-м интерфейсом функционалов.

Для оценки оптимальности создаваемых интерфейсов следует воспользоваться вычислением и анализом пользовательских шаблонов навигации. Пользовательский шаблон навигации представляет собой одну из наиболее часто используемых траекторий перемещения пользователя программной системы в пространстве интерфейсов. Для сбора информации о траекториях можно использовать запись пользовательских действий в лог-файл, для анализа и выявления навигационных шаблонов – методы нахождения последовательных шаблонов.

Для анализа поведенческой продукции системы будем использовать алгоритм AprioriALL. На вход алгоритму AprioriALL поступает лог-файл, протоколирующий действия пользователя с системой. В качестве выходных данных выступает список пользовательских шаблонов навигации.

Предлагаемый алгоритм самомодификации пользовательского интерфейса, основанный на технологии интеллектуального анализа данных, содержит следующие шаги:

1. Считывание списка исходного набора данных из файла.

2. Формирование списка пользовательских шаблонов навигации с помощью алгоритма AprioriALL. Список шаблонов навигации будет иметь вид

$$A = \{A_1, A_2, ..., A_z\},\ z = 1,..f,$$

$$A_i = \{A_{i1}, A_{i2}, ..., A_{ik}\},$$

$$A_{ij} = (id_j, idF_j, idD_j, v_j)\forall j = 1, ..., k_i,$$

где f – число шаблонов; k_i – число элементов i-го шаблона; $id_j \in P_j$ – идентификатор интерфейса j-го элемента i-го шаблона; $idF_j \in FB_j$ – идентификатор задействованного в операции над j-м интерфейсом i-го шаблона функционала; $idD_j \in DB_j$ – идентификатор задействованного в операции над j-м интерфейсом i-го шаблона объекта данных; $v_j \in V_I$ – связанная с воздействием на j-й интерфейс i-го шаблона информация (пользовательские действия).

3. Цикл по шаблонам, $s = 1, ..., f$.

4. Внутренний цикл по шаблонам, $r = s + 1..., f$.

5. Просмотр очередного шаблона A_r и проверка на то, является ли он схожим с шаблоном A_s. Для определения схожести шаблонов определяются совпадающие в обоих шаблонах подпоследовательности и оценивается порядок их следования. В схожих шаблонах порядок следования идентичных подпоследовательностей должен совпадать. Шаблоны можно считать достаточно схожими, если число элементов, не входящих в идентичные подпоследовательности, не превышает определенного порога.

6. Если A_s и A_r являются схожими шаблонами, то осуществляется их «слияние» – формирование на первом общем для шаблонов интерфейсе-контейнере нового дочернего интерфейса, реализующего совпадающую функциональность шаблонов и использующего общие данные. В процессе выполнения операций, связанных с новым интерфейсом, осуществляется обращение к интерфейсам, входящим в неидентичные подпоследовательности, для получения необходимых данных. Если A_s не является схожим шаблоном, то осуществляется переход к пункту 8.

7. Поиск элементов из совпадающих подпоследова-

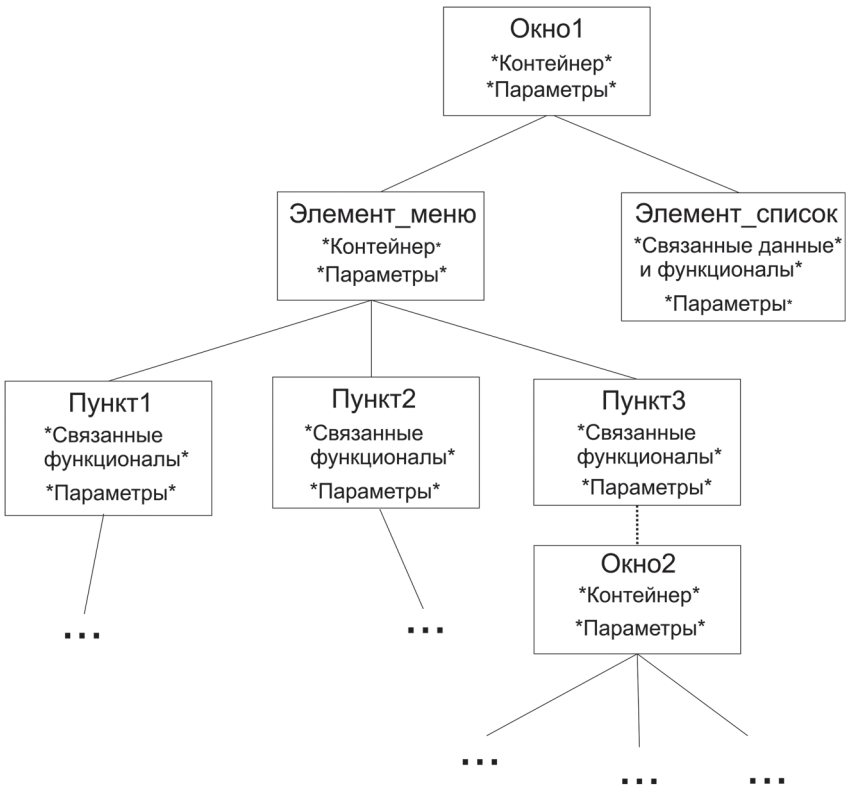

Рис. 2. Схема пространства интерфейсов

тельностей в других шаблонах списка. Если какой-либо элемент не встречается в других шаблонах, то принимается решение о том, что данный элемент является редко используемым. Редко используемый элемент становится скрытым элементом.

8. Завершение внутреннего цикла по шаблонам. Если еще есть непросмотренные шаблоны, переход к пункту 5, если нет – к пункту 9.

9. Если для шаблона A_s не были найдены схожие шаблоны, осуществляется оптимизация интерфейса на основе информации, содержащейся в A_s. На первом интерфейсе-контейнере шаблона формируется новый дочерний интерфейс, реализующий функциональность и работающий с данными всего шаблона.

10. Завершение внешнего цикла по шаблонам.

11. Завершение работы алгоритма.

Предлагаемый алгоритм, разработанный в рамках представлений о рефлексивной адаптации программных систем, позволяет решить 2 основные проблемы, возникающие в процессе создания и эксплуатации пользовательских интерфейсов: проблему недостаточности пространства интерфейсов (отсутствия элементов управления для наиболее часто выполняемых операций) и проблему избыточности (наличие элементов, реализующих «несамостоятельную» функциональность).

Заключение

Предлагаемая в статье идея рефлексивной адаптации и способ ее реализации методами интеллектуального анализа данных позволяет достичь снижения ресурсозатрат на сопровождение информационных систем. Процесс сопровождения программы становится менее ресурсоемким за счет уменьшения в нем участия специалиста. Проблема снижения ресурсозатрат особенно актуальна для дорогостоящих систем, к числу которых относятся системы e-learning. Адаптация к требованиям конкретного пользователя (или группы пользователей) позволит не только снизить стоимость сопровождения электронной образовательной системы, но и повысить эффективность процесса обучения, предоставляя обучаемому более комфортные условия для усвоения учебного материала. Дальнейшее развитие идеи рефлексивной адаптации позволит существенно повысить гибкость электронных обучающих систем путем улучшения их способности к индивидуальной подстройке под различные категории обучающихся.

Литература

1. *Keromytis A.D.* Characterizing Software Self-healing Systems // Computer Network Security. Springer. 2007. P. 22–33.

2. *Shin M.E., Cooke D.* Connector-Based Self-Healing Mechanism for Components of a Reliable System // Workshop on the Design and Evolution of Autonomic Application Software (DEAS 2005). ACM, 2005. P. 1–7.

3. Smart Sparrow. URL: https://www.smartsparrow.com/ (дата обращения: 22.05.2017).

References

1. *Keromytis A.D.* Characterizing Software Self-healing Systems. Computer Network Security. Springer. 2007. P. 22–33.

2. *Shin M.E., Cooke D.* Connector-Based Self-Healing Mechanism for Components of a Reliable System. Workshop on the Design and Evolution of Autonomic Application Software (DEAS 2005). ACM, 2005. P. 1–7.

3. Smart Sparrow. URL: https://www.smartsparrow.com/ (accessed: 22.05.2017).

4. *Rajan C.A., Baral R.* Adoption of ERP system: An empirical study of factors influencing the usage of ERP and its impact on end user // IIMB Management Review. 2015. No. 2. P. 105–117.

5. The Next Evolution of ERP: Adaptive ERP // ERP the Right Way: Changing the game for ERP Cloud implementations URL: https://gbeaubouef.wordpress.com/2012/09/05/adaptive-erp/ (дата обращения: 14.05.2017).

6. WWU Munster // OpenUSS. URL: https://www.uni-muenster.de/studium/orga/openuss.html (дата обращения: 22.05.2017).

7. WebCT. URL: http://www.cuhk.edu.hk/eLearning/c_systems/webct6/ (дата обращения: 22.05.2017).

8. Заметки о Big Data // ЕС-Лизинг URL: http://www.ec-leasing.ru/public/publikatsii/index.php?ELEMENT_ID=39 (дата обращения: 28.04.2017).

9. Big Data in eLearning: The Future of eLearning Industry // eLearning Industry. URL: https://elearningindustry.com/big-data-in-elearning-future-of-elearning-industry (дата обращения: 22.05.2017).

10. *Ravindran K., Rabby M.* Software cybernetics to infuse adaptation intelligence in networked systems // IEEE International Conference on the Network of the Future (NOF). Washington: IEEE Computer Society, 2013. P. 1–6.

11. *Wang P., Cai K.Y.* Representing extended finite state machines for SDL by a novel control model of discrete event systems // Sixth IEEE International Conference on Quality Software (QSIC 2006). Washington: Ieee Computer Society, 2006. P. 159–166.

12. *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.

13. *Patikirikorala T., Colman A., Han J., Wang L.* A systematic survey on the design of self-adaptive software systems using control engineering approaches // 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (IEEE Press). Washington: IEEE Computer Society, 2012. P. 33–42.

14. *Lorenzoli, D., Mariani, L., Pezzè, M.* Automatic generation of software behavioural models // 30th international ACM conference on Software engineering. New York: ACM, 2008. P. 501–510.

15. *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.

16. *Liu L., Zhou Q., Liu J., Cao Z.* Requirements cybernetics: elicitation based on user behavioural data // J. Syst. Software. Amsterdam: Elsevier. 2016.

4. *Rajan C.A., Baral R.* Adoption of ERP system: An empirical study of factors influencing the usage of ERP and its impact on end user. IIMB Management Review. 2015. No. 2. P. 105–117.

5. The Next Evolution of ERP: Adaptive ERP. ERP the Right Way: Changing the game for ERP Cloud implementations URL: https://gbeaubouef.wordpress.com/2012/09/05/adaptive-erp/ (accessed: 14.05.2017).

6. WWU Munster. OpenUSS. URL: https://www.uni-muenster.de/studium/orga/openuss.html (accessed: 22.05.2017).

7. WebCT. URL: http://www.cuhk.edu.hk/eLearning/c_systems/webct6/ (accessed: 22.05.2017).

8. Zametki o Big Data. ES-Lizing URL: http://www.ec-leasing.ru/public/publikatsii/index.php?ELEMENT_ID=39 (accessed: 28.04.2017).

9. Big Data in eLearning: The Future of eLearning Industry. eLearning Industry. URL: https://elearningindustry.com/big-data-in-elearning-future-of-elearning-industry (accessed: 22.05.2017).

10. *Ravindran K., Rabby M.* Software cybernetics to infuse adaptation intelligence in networked systems. IEEE International Conference on the Network of the Future (NOF). Washington: IEEE Computer Society, 2013. P. 1–6.

11. *Wang P., Cai K.Y.* Representing extended finite state machines for SDL by a novel control model of discrete event systems. Sixth IEEE International Conference on Quality Software (QSIC 2006). Washington: Ieee Computer Society, 2006. P. 159–166.

12. *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.

13. Patikirikorala T., Colman A., Han J., Wang L. A systematic survey on the design of self-adaptive software systems using control engineering approaches. 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (IEEE Press). Washington: IEEE Computer Society, 2012. P. 33–42.

14. *Lorenzoli, D., Mariani, L., Pezzè, M.* Automatic generation of software behavioural models. 30th international ACM conference on Software engineering. New York: ACM, 2008. P. 501–510.

15. *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.

16. *Liu L., Zhou Q., Liu J., Cao Z.* Requirements cybernetics: elicitation based on user behavioural data. J. Syst. Software. Amsterdam: Elsevier. 2016.

17. *Park J.S.* Essence-based, goal-driven adaptive software engineering // EEE/ACM 4th SEMAT Workshop on General Theory of Software Engineering (GTSE). Washington: IEEE Computer Society, 2015. P. 33–38.

18. *Liu C., Jiang C., Hu H., Cai K.Y., Huang D., Yau S.S.* Control-based approach to balance services performance and security for adaptive service based systems (ASBS) // 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC'09). Washington: IEEE Computer Society, 2009. P. 473–478.

19. *Zhou Y., Taolue C.* Software Adaptation in an Open Environment: A Software Architecture Perspective. Boca Raton: CRC Press, 2017.

20. *Lemos R.* Software Engineering for Self-Adaptive Systems. Berlin: Springer, 2009.

17. *Park J.S.* Essence-based, goal-driven adaptive software engineering. EEE/ACM 4th SEMAT Workshop on General Theory of Software Engineering (GTSE). Washington: IEEE Computer Society, 2015. P. 33–38.

18. *Liu C., Jiang C., Hu H., Cai K.Y., Huang D., Yau S.S.* Control-based approach to balance services performance and security for adaptive service based systems (ASBS). 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC'09). Washington: IEEE Computer Society, 2009. P. 473–478.

19. *Zhou Y., Taolue C.* Software Adaptation in an Open Environment: A Software Architecture Perspective. Boca Raton: CRC Press, 2017.

20. *Lemos R.* Software Engineering for Self-Adaptive Systems. Berlin: Springer, 2009.

Сведения об авторах

Александр Сергеевич Бождай
Д.т.н., профессор кафедры «Системы автоматизированного проектирования»
Пензенский государственный университет,
Пенза, Россия
Эл. почта: bozhday@yandex.ru

Юлия Игоревна Евсеева
К.т.н., ассистент кафедры «Системы автоматизированного проектирования»
Пензенский государственный университет,
Пенза, Россия
Эл. почта: shymoda@mail.ru

Алексей Анатольевич Гудков
К.т.н., доцент кафедры «Системы автоматизированного проектирования»
Пензенский государственный университет,
Пенза, Россия
Эл. почта: alexei.gudkov@gmail.com

Information about the authors

Alexandr S. Bozhday
Dr. Sci. (Eng.), professor of the department «Computer Aided Design»
Penza State University,
Penza, Russia
E-mail: bozhday@yandex.ru

Yulia I. Evseeva
Cand. Sci. (Eng.), assistant of the department «Computer Aided Design»
Penza State University,
Penza, Russia
E-mail: shymoda@mail.ru

Alexey A. Gudkov
Cand. Sci. (Eng.), associate professor of the department «Computer Aided Design»
Penza State University,
Penza, Russia
E-mail: alexei.gudkov@gmail.com