Загальні питання

УДК 004.8

И.Е. Бибичков, В.В. Сокол, А.Ю. Шевченко

Харківський національний університет радіоелектроніки, Харків

МОДЕЛЬ САМОАДАПТИВНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ОСНОВАННАЯ НА ЗНАНИЯХ ИЗ КОРПОРАТИВНОЙ ПАМЯТИ

Проведен анализ методов управления корпоративной памятью с целью создания модели самоадаптируемого программного обеспечения (ПО). Предложена концепция разработки ПО, которое адаптируется под индивидуальные потребности пользователя для систем, в которых данные содержатся в корпоративной памяти в онтологическом виде. Предложена специализированная структура, основанная на онтологии для разработки самоадаптивного ПО. Предложен механизм персонализации для самоадаптивной модели, который фильтрует данные и выдает пользователю актуальную информацию. В результате исследований была разработана модель самоадаптивного ПО, построенная на корпоративной памяти, представленной в виде онтологии. Разработанная модель адаптируется в соответствии с изменениями, которые были внесены в онтологию.

Ключевые слова: *база знаний, внешний вид, самоадаптация программного обеспечения, онтология, корпоративная память.*

Введение

Постановка проблемы. Одной из актуальных проблем при создании программного обеспечения является минимизация затрат связанных с изменениями в программном обеспечении. Необходимость в таких изменениях возникает, когда происходят уточнения из-за изменившегося со временем представления о предметной области. Вариантом решения такой проблемы является минимизация затрат на создание программного обеспечения, которое могло бы изменять внешний вид без изменения исходного кода. В таком случае достаточно сделать изменения в базе знаний и приложение может адаптироваться для работы с новыми данными.

В настоящее время широкое распространение получила онтологическая модель описания предметной области. Эта тенденция обусловлена, прежде всего, универсальностью и гибкостью данного подхода.

Применение онтологий как основного хранилища информации в программных системах позволяет создавать адаптивные приложения, в основе которых лежит динамически изменяемая модель знаний. По сравнению с базами данных, применение онтологий обеспечивает легкость в моделировании сложных связей и отношений. Кроме того, использование онтологий в программных системах открывает возможность для динамического изменения работы программы: смена интерфейса, функциональных компонентов, за счет внесения изменений исключительно в модель знаний.

В рамках данного исследования была предложена модель для разработки самоадаптивного программного обеспечения (ПО), которая основана на использовании корпоративной памяти. Для представления корпоративной памяти в рамках данного исследования, применяются онтологии.

Ключевую роль в разрабатываемой системе играет предложенная онтологическая модель представления корпоративной памяти [1]. Данная модель предполагает создание нескольких онтологий, объединенных в одну базу знаний (БЗ). Такой подход позволит разделить структурные элементы системы и элементы предметной области для достижения максимальной гибкости и стабильности функционирования ПО.

Стандартные методы, которые применяются в разработке приложений в настоящее время имеют ограничения, связанные с достаточно жесткой структурой приложений, наличием обязательных компонентов, которые регламентируют набор элементов и внешний вид. Предлагаемый метод, благодаря возможностям «OWL 2» [2] позволяет описать собственную модель, и регламентировать элементы из которых будет состоять программный продукт.

Внедрение предложенной модели было произведено в рамках кастомизации «Web-ресурса»: «Портал Обеспечения Качества Высшего Образования» (http://portal.dovira.eu) [3].

Анализ последних исследований и публикаций. Среди основных направлений разработки ПО, способного, адаптироваться в процессе эксплуатации в

соответствии с пользовательскими требованиями, можно выделить:

– системы самообучения [4–7];

– системы с поддержкой персонализации [8–9];

– системы, расширяющие существующий функционал ПО [10].

В [4] предложен метод, построенный на применении интеллектуальных агентов в онтологии для создания адаптивного ПО. В основе исследования [4] лежит идея создания приложения с адаптивным персонализированном интерфейсом. В качестве примера представлена разработанная на базе этого метода «e-learning» система (e-learning – от англ. electronic learning – системы электронного обучения). Недостатками исследования [4] является недостаточная универсальность разработанного метода (метод, описанный в [4] ориентирован на работу с иерархическими системами).

Интеллектуальный метод обработки данных, для представления контекстно-ориетнтированной и персонализированной информации пользователю представлен в [5]. В основе метода [5] лежит контекстно-ориентированная и персонализированная оценка событий на основе онтологии и алгоритма распространения. Особенностями предложенного в [5] метода является применение интеллектуальных агентов. Функциями интеллектуальных агентов в [5] является сбор информации о пользователе и выявлении его интересов. Недостатком метода, описанного в [5] является ограниченность его применения специфическими системами: «Продвинутый Университет» [5] и подобные.

Описание адаптивной системы самообучения, построенной на основе онтологии, представлено в [6]. В основе, описанной в [6] системы, стоит разработанный фреймворк для создания адаптивных и высокоинтерактивных веб-ориентированных систем обучения. Недостатком данного метода является его узконаправленность, а именно ограниченность области применения системами веб-обучения.

В [7] предложен метод для создания систем адаптивного обучения информационной безопасности на основе онтологий. Суть предложенного в [7] метода состоит в анализе индивидуальных особенностей персонала компании (анализа корпоративной памяти) и, на основании полученных данных, динамическом изменении функционирования ПО. Недостатками метода, представленного в [7] является его узкая специализация и ограниченная направленность: преимущественно для обучающих систем.

Методика создания адаптивного сервиса для предоставления услуг описана в [8]. Адаптивность данного метода [8] достигается посредством применения онтологий в качестве хранилища данных. В онтологии содержится информация о службах, которые будут предоставлять услуги конечному клиенту. В описание службы входит перечень ее профессиональных особенностей и список предлагаемых услуг. Система подбирает наиболее оптимальную службу под конкретную услугу. К недостаткам данного метода относится слабая персонализация: в системе отсутствует функционал для адаптации приложения под индивидуальные потребности пользователя.

В [9] предложен метод самоадаптации приложений в виде системы персонализации веб-задач. Суть данного метода [9] в анализе запросов пользователя, выявлении в них персональных целей и помощи в достижении этих целей, посредством веб-систем. Особенностями данного метода [9] является использование онтологий в качестве основного хранилища информации. Благодаря использованию в [9] онтологий, становится доступным создание более структурированной модели данных, а так же открываются возможности для интеллектуального анализа и работы с данными в системе. Недостатком данного метода является его ограниченность в использовании. Применение такого подхода будет эффективно только для веб-систем.

Методика применения семантических технологий для создания самоадаптивных приложений представлена в [10]. Суть данного подхода в том, чтобы при помощи онтологий расширить функционал приложений и сделать их самоадаптивными (в том числе и приложения, которые изначально не проектировались для поддержки такого функционала). Применение онтологий позволяет системе динамически изменять ее внешний вид, адаптировать конфигурацию, а также вносить изменения в ход выполнения программы. Недостатком данной системы [10] является слабая персонализация. То есть отсутствует функциональность, которая анализирует в процессе эксплуатации личные предпочтения пользователя системы и предоставляет персонализированный контент.

Таким образом, результаты анализа позволяют сделать вывод о том, что единой системы, которая была бы способна решить все поставленные задачи в рамках разработки самоадаптивного ПО нету. Существующие же системы в той или иной степени способны решить только часть этих задач.

Целью статьи является разработка модели программного обеспечения с возможностью самоадаптации.

Для достижения цели были поставлены следующие задачи:

– разработать концепцию самоадаптивного ПО;

– разработать структуру онтологии для самоадаптивного ПО;

– разработать онтологическую модель корпоративной памяти для ПО с элементами самоадаптации;

– разработать механизм персонализации данных пользователя

– провести апробацию разработанной модели.

Изложение основного материала

Концепция самоадаптивного ПО. Предложенная в данном исследовании концепция самоадаптивного ПО построена на использовании онтологии в качестве основного хранилища не только пользовательской информации, но и элементов ПО, таких как UI (User Interface), и других функциональных программных элементов. Наряду с пользовательской информацией в онтологии хранится структура ПО и элементы программного интерфейса, а также функциональные элементы, то есть блоки (модули). Хранение программных модулей вместе с пользовательской информацией в базе знаний позволяет строить ПО с гибким адаптивным интерфейсом и изменяемой функциональностью.

Основная идея создания модели самоадаптивного ПО заключается в апробации интеллектуального метода программной разработки. Данный метод расширяет традиционные взгляды на разработку ПО и в будущем позволит создавать программные решения с меньшим количеством затраченных человекочасов, либо вовсе, без участия человека.

Структура онтологии для самоадаптивного ПО. В общем случае, корпоративная память (corporate / organizational memory) является одним из способов представления знаний и информации внутри организации об опыте решения производственных задач [1]. В данном случае, корпоративная память используется для хранения всей системной информации.

В рамках данного исследования, корпоративная память представлена в виде трех отдельных онтологий, которые сливаются в одну в процессе работы системы. Онтологическая модель представления знаний состоит из таких онтологий: сервисная онтология, онтология предметной области и онтология индивидов (или конкретных экземпляров классов). Такой подход обусловлен достижением гибкости и расширяемости системы в целом. Ниже рассмотрены подробно характеристики и назначения онтологий.

Центральная онтология является сервисной и содержит основные классы, через которые ядро системы «видит» всю остальную онтологию. Сервисная онтология хранит ключевые классы и свойства, чьими потомками являются классы предметной области. Данная онтология является корневой и определяющей функциональный состав системы. Любые изменения, которые были произведены в онтологии, отразятся в программном коде ядра системы. Сервисная онтология определяет структуру и задает ограничения для онтологии предметной области (рис. 1). Корме того, в сервисной онтологии находятся структурные элементы приложения: элементы пользовательского интерфейса и функциональные элементы.

Онтология предметной области (рис. 1) основывается на сервисной онтологии и содержит информацию про все классы объектов, которые можно зарегистрировать в системе. Данная онтология определяет структуру информации, которая будет представлена пользователю. В частности, онтология предметной области содержит набор функциональных элементов, которые представляются пользователю и, которые он может изменять.

Онтология экземпляров классов (индивидов) (рис. 1). Использование данной онтологии обусловлено упрощением процесса модификации пользовательской информации. Экземпляры всех классов добавляются лишь в эту онтологию. Выделение индивидов в отдельную онтологию обеспечивает дополнительную гибкость и универсальность для системы. В случае изменения предметной области, нет необходимости перестраивать структуру хранения данных. Это объясняется тем, что в онтологии информация хранится в виде объектов, а не в строго структурированном виде, как в классических системах управления базами данных (СУБД).

Важной функциональной характеристикой описанной выше модели представления знаний является ее универсальность. Под универсальностью, подразумевается возможность адаптирования модели знаний под любую предметную область, путем внесения необходимых изменений в сервисную онтологию.

Рис. 1. Схема онтологической модели корпоративной памяти

Так, благодаря внесению изменений исключительно в структуру сервисной онтологии, открывается возможность перепрофилирования системы в целом.

Моделирование самоадаптивного ПО. В рамках данного исследования был предложен новый подход

для разработки ПО, основанный на применении «ODD»-методологии, описанной выше. Разрабатываемое приложение предполагает реализацию самоадаптивности, что в данном контексте определяет способность ПО динамически изменяться в процессе работы. Все изменения, протекающие в ходе работы ПО можно условно разделить на 2 группы: изменения, которые происходят на уровне UI и изменения на уровне логики работы программы.

Изменения, внесенные в онтологию предметной области, отражаются на функционировании ПО (процесс адаптации).

Онтология предметной области может быть изменена вручную, или автоматически, самой системой, посредством ризонеров[11]. В первом случае, процесс внесения изменений заключается в создании резервной копии и ее редактировании посредством сторонних средств (таких как Protege[12]). Затем, отредактированная онтология предметной области объединяется с двумя другими: сервисной онтологией и онтологии экземпляров классов. После загрузки обновленной онтологии, система автоматически адаптируется в соответствии с внесенными изменениями.

Процесс автоматического внесения изменений в онтологию происходит при помощи специализированных инструментов – ризонеров[11]. Их основная функция – проверка онтологии на валидность. Кроме того, ризонеры могут вносить изменения в онтологию посредством анализа связей между узлами онтологии и логического вывода.

Изменения в онтологии предметной области приводят к изменениям в функционировании ПО. Так, модифицирование существующих классов онтологии, которые описывают функциональные элементы UI приведет к изменениям в работе этих элементов при их редактировании и просмотре. Например, изменение типа данных элемента в онтологии приведет к изменениям во внешнем виде соответствующего элемента в UI

Схематическое описание работы корпоративной памяти. В основе предложенного подхода к разработке самоадаптивного ПО лежит концепция применения специализированной базы знаний, которая состоит из трех взаимосвязанных онтологий (веток): сервисной онтологии, онтологии предметной области и онтологии экземпляров классов. Все три онтологии связаны между собой отношением наследования и используются для формализации корпоративной памяти.

Схема взаимодействия системы с корпоративной памятью представлено на рис. 2.

Добавление информации. В данном случае, предполагается расширение онтологии за счет добавления пользователем новой информации. Вновь поступившая информация приводится к онтологическому виду и добавляется в базу знаний.

Извлечение накопленной информации – один из наиболее трудоемких этапов работы с онтологией. От успешности выполнения данного этапа зависит дальнейшая жизнеспособность всей системы. Здесь происходит выборка из базы знаний необходимой информации и ее фильтрация с последующей записью в кэш. Использование кэша в процессе эксплуатации ПО необходимо для ускорения работы.

Обработка информации и подготовка для записи в БЗ. На данном этапе происходит обработка новой информации и приведение ее к онтологическому виду (так называемая предобработка).

Ризонинг данных. Применение ризонеров обеспечивает управление знаниями, которые содержатся в БЗ (рис. 2). При помощи ризонеров происходит корректировка формализованных знаний, а именно, их добавление и обновление. Кроме того в процессе обслуживания знаний происходит удаление устаревшей или невалидной (некорректной) информации, а так же автоматическое добавление новых знаний в БЗ. Применеие ризонеров предполагает возможность автоматического расширения БЗ на основании логических выводов.

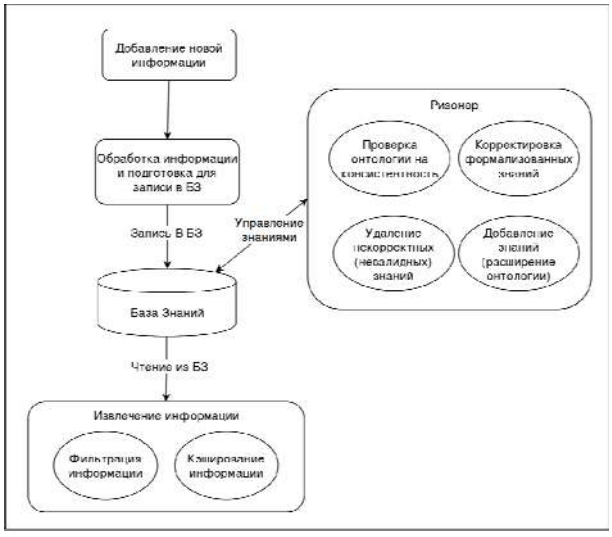

Рис. 2. Схема работы корпоративной памяти

Изменения, которые вносятся в корпоративную память, представленную в виде онтологии в процессе функционирования системы, затем приводят к изменениям в UI.

Описание UI. На рис. 3 представлена схема шаблонов UI: «Авторизация» «Главный экран», «Экран 1» и «Экран 2».

Экран «Авторизации» является стандартным и не может быть изменен не авторизованными пользователями системы.

После прохождения «Авторизации», пользователь попадает на «Главный Экран». При помощи механизма персонализации, пользователю в буду-

щем будет сгенерирован собственный набор элементов «Главного экрана» на основании его данных и предпочтений.

«Экран 1» и «Экран 2» представляют схему базового набора шаблонов UI. Данные шаблоны так же могут быть изменены в процессе работы ПО и персонализированы согласно предпочтениям пользователя.

Все сгенерированные в процессе использования системы экраны вместе с данными кэшируются для ускорения дальнейшей работы. При завершении работы происходит сохранение всех изменений в онтологию, в том числе сохраняются и сгенерированные во время работы UI, которые могут быть в последствии предоставлены не только пользователю, для которого они были сгенерированы, но и другим пользователям системы.

Кроме того, в онтологии хранится базовый набор шаблонов UI, который расширяется в процессе эксплуатации ПО.

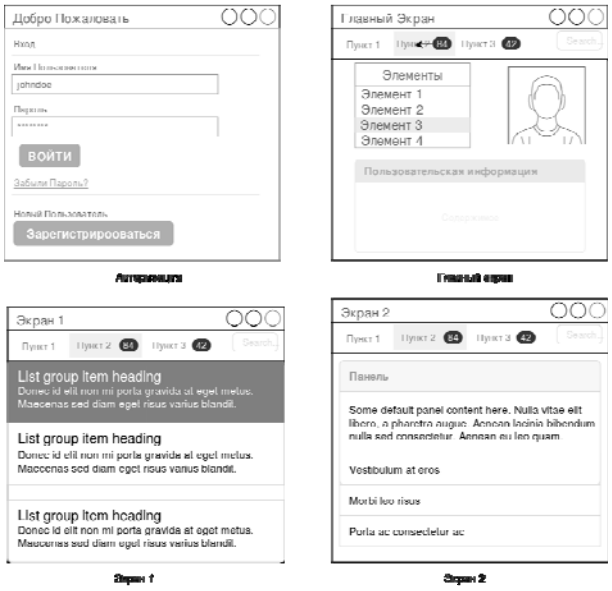

Рис. 3. Схемы шаблонов UI

UI элементы, которые хранятся в онтологии, могут быть адаптированы под конкретного пользователя благодаря механизму персонализации, который описан в следующем разделе.

Данные пользователя (общие и частные). Одним из основных механизмов, который используется в рамках разрабатываемой модели самоадаптации ПО, является механизм персонализации. Суть данного механизма в разграничении персональной и публичной информации. Объекты создаваемые пользователем могут быть как публичные, так и приватные. Такими объектами могут быть классы, свойства классов и экземпляры классов. Приватные объекты доступны только самому пользователю. Для него работа с этими объектами будет такая же, как и со всеми публичными объектами. При этом другие пользователи не только не могут получить доступ к приватным объектам этого пользователя, но и даже не знают о существовании таких приватных объектов.

Предложенный механизм персонализации построен на использовании персонального идентификатора пользователя в качестве «namespace» для ресурсов, которые должны быть доступны только пользователю, который их добавил.

Все существующие «namespace» разделяются на 3 группы:

1. общая группа – характерная для общепринятых элементов онтологии, таких как: «owl», «rdf», «rdfs», «xsd»;
2. группа публичных объектов – в рамках конкретной онтологической системы: «Trust_Domain» и «Trust_Service»;
3. группа персонализированных данных – элементы вида «User_<id>», где <id> – уникальный идентификатор пользователя.

За счет характерных особенностей «namespace» онтологий возможно фильтровать данные, считываемые из онтологического хранилища «SPARQL»-запросами. Элементами фильтрации будут наборы «namespace» из первой и второй группы и одного элемента третьей группы, соответствующий идентификатору пользователя, который залогинился в системе.

Модульный подход, по которому построено программное обеспечение управляемое онтологией, подразумевает, что вся необходимая информация для построения приложения считывается из онтологии. Это означает, что персонализировать можно не только данные пользователя, но и функциональность приложения и элементы UI.

Результаты исследований. Каждая из переменных трёх ветвей онтологии хранится в отдельном файле, который потом может быть отредактирован в любом редакторе онтологии и импортирован назад в портал. При таком подходе есть возможность изменить структуру онтологии без потери введенных данных и без внесения изменений в исходный код портала [3].

На примере модуля регистрации портала: «http://portal.dovira.eu» можно рассмотреть практическое применение самоадаптируемых систем, а так же особенности реализации. Модуль регистрации ресурсов создаёт все свои интерфейсы на основании анализа онтологии предметной области, используя ключевые классы и свойства сервисной онтологии. Таким образом, он адаптируется под текущее состояние онтологии.

Каждый пользователь получает возможность пересматривать информацию об объектах онтологии. После регистрации на портале, пользователь

так же получает возможность изменять и создавать новые объекты.

Так же, в портале присутствует универсальный редактор ресурсов, реализованный на основании онтологий. Основной его особенностью является динамическое формирование формы редактирования, в зависимости от его типа, представленного в онтологии. Перечень элементов управления на форме управления и их поведение целиком определяется перечнем и типом свойств класса онтологии, экземпляром которой является редактируемый объект. Дополнительной возможностью редактора является способность отображать дополнительные свойства объекта, которые не сохраняются для него в модели, но могут быть вычислены при необходимости в режиме реального времени на основании других свойств. Например, если необходимо отобразить возраст, следует использовать поле «дата рождения» или «дата создания» и на его основании сформировать свойство «возраст», которое не может быть отредактировано.

Универсальный редактор «умных свойств» при инициализации связывается с процедурой пересоздания одного свойства в другое и на страницах детального просмотра автоматически отображает её поля, которые невозможно отредактировать. Примером такого свойства может быть «возраст объекта».

На основании анализа онтологии предметной области, существует возможность адаптации программных элементов, то есть динамическое формирование интерфейса программного обеспечения. Подобный принцип применялся в портале при регистрации нового модуля, а так же формировании системы ценностей. Модуль регистрации ресурса, используя ключевые классы и свойства сервисной онтологии, может адаптироваться под её текущее состояние. Таким же образом формируется интерфейс модуля формирования системы ценностей. Интерфейс данного модуля генерируется динамически, в зависимости от классов и свойств, которые были добавлены в онтологию предметной области.

Онтология предметной области содержит набор достижений, которые учитываются порталом. При выборе типа ресурса портал анализирует возможные достижения и выбирает те из них, которые могут касаться выбранного. Достижения могут быть оценены пользователем. Задать «ценность» можно посредством указания значение от 0 до 100 для каждой конкретной системы. Так же, в систему ценностей могут быть включены числовые параметры, характерные для данного типа достижений, их значения могут быть указаны в диапазоне от 0 до 100. Изменение численных характеристик «ценности» приведет к изменениям функционирования ПО.

Выводы

В рамках данного исследования была разработана модель самоадаптивного ПО. Данная модель основана на использовании корпоративной памяти, представленной в виде онтологии, в качестве основного хранилища пользовательской и служебной информации.

Был разработан механизм персонализации данных пользователя, который применяется для фильтрации данных в целях предоставления наиболее актуальной информации.

Была проведена апробацию разработанной модели на портале обеспечивания качества высшего образования.

Использование корпоративной памяти, представленной в виде онтологии для хранения информации ПО позволяет создавать более универсальные программные решения, при этом используя меньшее количество человекочасов на разработку.

Недостатком работы систем, основанных на онтологиях является высокая затратность вычислительных операций, при взаимодействии с БЗ. Для ускорения данных процессов, в рамках исследования был использован метод оптимизации быстродействия онтологических БЗ, описанный в [13].

Разработанная модель была апробирована в качестве концепции создания самоадаптивного ПО, основанного на применении корпоративной памяти в виде онтологии, в качестве интеллектуального хранилища информации. В будущем, развитие описанного в данной работе подхода для разработки самоадаптивного ПО, позволит создавать программные решения, способные не только изменяться в процессе эксплуатации, но и самостоятельно расширяться.

Список литературы

1. Гаврилова Т.А. Базы знаний интеллектуальных систем: моногр. / Т.А. Гаврилова, В.Ф. Хорошевский. – Спб.: Изд. Питер, 2000. – 384 с.

2. Bao, J. OWL 2 Web Ontology Language Document Overview / J. Bao, D. Calvanese, B.C. Grau та ін. // W3C. – 2012. [Электронный ресурс] – Режим доступа: https://www.w3.org/TR/owl2-overview/.

3. Шевченко А.Ю. Национальный Портал Обеспечения Качества Высшего Образования / А.Ю. Шевченко, В.В. Сокол, Е.Л. Шевченко // Инструкция для разработчиков. – 2011. [Электронный ресурс]. – Режим доступа: http://portal.dovira.eu/ img/TRUST%20Portal%20Developer%20Manuals.pdf.

4. Rani, M. An ontology-based adaptive personalized e-learning system, assisted by software agents on cloud storage / M. Rani, R. Nayak, O.P. Vyas // Knowledge-Based Systems. – 2015. – № 90. – С. 33-48. [Электронный ресурс]. – Режим доступа: https://doi.org/10.1016/j.knosys.2015. 10.002.

5. Régia, A.D. Agent-based architecture for context-aware and personalized event recommendation / A.D. Régia, Á.G. Marcos, C.G. Ralha // Expert Systems with Applications. – 2014. – № 41 (2). – С. 563-573. https://doi.org/10.1016/j.eswa.2013. 07.081.

6. Shimko, M. ALEF: A Framework for Adaptive Web-Based Learning 2.0. / M. Shimko, M. Barla, M. Bielikova // IFIP Advances in Information and Communication Technology. – 2010. – № 324. – С. 367-378. http://dx.doi.org/10.1007/978-3-642-15378-5_36.

7. Mangold, L.V. Using Ontologies for Adaptive Information Security Training / L.V Mangold // Availability, Reliability and Security (ARES), 2012 Seventh International Conference on. – 2012. – № 1. – С. 522-524. https://doi.org/10.1109/ARES.2012.52.

8. Kertesz, A. An interoperable and self-adaptive approach for SLA-based service virtualization in heterogeneous Cloud environments / A. Kertesz, G. Kecskemeti, I. Brandic // Future Generation Computer Systems. – 2014. – № 32. – С. 54-68. https://doi.org/ 10.1016/j.future.2012.05.016.

9. Castañeda, L. Self-Adaptive Applications: On the Development of Personalized Web-Tasking Systems / L. Castañeda, N.M. Villegas, H.A. Müller // SEAMS 2014 Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. – 2014. – № 1. – С. 49-54. https://doi.org/10.1145/2593929.2593942.

10. Poggi, F. An Application of Semantic Technologies to Self Adaptations / F. Poggi, D. Rossi, P. Ciancarini, L. Bompani // 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI). – 2016. – № 1. – P. 1-6. http://doi.org/10.1109/RTSI.2016.7740548.

11. Bibichkov, I. Ontological Knowledge Bases Productivity Optimization Through The Use Of Reasoner Combination / I. Bibichkov, V. Sokol, O. Shevchenko // Eastern-European Journal of Enterprise Technologies. – 2017. – № 2(89). – С. 49-54. https://doi.org/10.15587/1729-4061.2017.112347.

12. Noy, N.F. Ontology Development 101: A Guide to Creating Your First Ontology [Электронный ресурс] / N.F. Noy, D.L. McGuinnes // Ontology 101. – 201. [Электронный ресурс]. – Режим доступа: http://protege.stanford.edu/ publications/ontology_development/ontology101.pdf.

13. Бибичков И.Е. Оптимизация быстродействия онтологических баз знаний, построенных на основе «Virtuoso» / И.Е. Бибичков, В.В. Сокол, А.Ю. Шевченко // Восточно-Европейский Журнал Передовых Технологий. – 2014. – № 2 (71). – С. 4-8. https://doi.org/10.15587/1729-4061.2014.28553.

References

1. Havrylova, T.A. and Khoroshevskyj, V.F. (2000), "*Bazy znanij intellektual'nyh sistem*" [*Intelligent Systems Knowledge Bases*], Piter, St Petersburg, 384 p.

2. Bao, J., Calvanese, D. and Grau, B.C. (2012), "*OWL 2 Web Ontology Language Document Overview*", www.w3.org/TR/owl2-overview (accessed 11 October 2017).

3. Shevchenko, O., Sokol, V. and Shevchenko, O. (2011), "Nacional'nyj Portal Obespechenija Kachestva Vysshego Obrazovanija" [National Portal for Quality Assurance in Higher Education], *Instruction for developers*, portal.dovira.eu/img/TRUST%20Portal%20Developer%20Manuals.pdf (accessed 12 October 2017).

4. Rani, M., Nayak, R. and Vyas, O.P. (2015), An ontology-based adaptive personalized e-learning system, assisted by software agents on cloud storage, *Knowledge-Based Systems*, No. 90, pp. 33-48, https://doi.org/10.1016/j.knosys.2015.10.002.

5. Régia, A.D., Marcos, Á.G. and Ralha, C.G. (2014), Agent-based architecture for context-aware and personalized event recommendation, *Expert Systems with Applications*, No. 41(2), pp. 563-573, https://doi.org/10.1016/j.eswa.2013.07.081.

6. Shimko, M., Barla, M. and Bielikova, M. (2010), ALEF: A Framework for Adaptive Web-Based Learning 2.0, *IFIP Advances in Information and Communication Technology*, No. 324, pp. 367-378, https://dx.doi.org/10.1007/978-3-642-15378-5_36.

7. Mangold, L.V. (2012), Using Ontologies for Adaptive Information Security Training, *Availability, Reliability and Security (ARES), 2012 Seventh International Conference on*, No. 1, pp. 522-524, https://doi.org/10.1109/ARES.2012.52.

8. Kertesz, A., Kecskemeti, G. and Brandic I. (2014), An interoperable and self-adaptive approach for SLA-based service virtualization in heterogeneous Cloud environments, *Future Generation Computer Systems*, No. 32, pp. 54-68, https://doi.org/10.1016/j.future.2012.05.016.

9. Castañeda, L., Villegas, N.M. and Müller, H.A. (2014), Self-Adaptive Applications: On the Development of Personalized Web-Tasking Systems, SEAMS 2014 Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, No. 1, pp. 49-54, https://doi.org/10.1145/2593929.2593942.

10. Poggi, F., Rossi, D., Ciancarini, P. and Bompani, L. (2016), An Application of Semantic Technologies to Self Adaptations, *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, No. 1, pp. 1-6, http://doi.org/10.1109/RTSI.2016.7740548.

11. Bibichkov, I., Sokol, V. and Shevchenko, O. (2017), Ontological Knowledge Bases Productivity Optimization Through The Use Of Reasoner Combination, *Eastern-European Journal of Enterprise Technologies*, No. 2(89), pp. 49-54, https://doi.org/10.15587/1729-4061.2017.112347.

12. Noy, N.F. and McGuinnes, D.L. (2008), Ontology Development 101: A Guide to Creating Your First Ontology, *Ontology 101*, protege.stanford.edu/publications/ontology_development/ontology101.pdf (accessed 15 October 2017).

13. Bibichkov, I., Sokol, V. and Shevchenko, O. (2014), "Optimizacija bystrodejstvija ontologicheskih baz znanij, postroennyh na osnove "Virtuoso" [Optimizing the performance of ontological knowledge bases built on the basis of "VIRTUOSO"], *Eastern-European Journal of Enterprise Technologies*, No. 2(71), pp. 4-8, https://doi.org/10.15587/1729-4061.2014.28553.

Поступила в редколлегию 6.11.2017
Одобрена к печати 7.12.2017

Відомості про авторів:

Бібічков Ігор Євгенович
аспірант кафедри
Харківського національного
університету радіоелектроніки,
Харків, Україна
https://orcid.org/0000-0003-1424-6960
e-mail: bibi4kov@gmail.com

Сокол Вадим Вікторович
аспірант кафедри
Харківського національного
університету радіоелектроніки,
Харків, Україна
https://orcid.org/0000-0003-2461-3453
e-mail: sokol@sw-expert.com

Шевченко Олександр Юрійович
кандидат технічних наук
доцент кафедри
Харківського національного
університету радіоелектроніки,
Харків, Україна
https://orcid.org/0000-0002-0068-4698
e-mail: shevchenko@sw-expert.com

Information about the authors:

Igor Bibichkov
Postgraduate Student
Department of Kharkiv National
University of Radioelectronics,
Kharkiv, Ukraine
https://orcid.org/0000-0003-1424-6960
e-mail: bibi4kov@gmail.com

Vadym Sokol
Postgraduate Student
Department of Kharkiv National
University of Radioelectronics,
Kharkiv, Ukraine
https://orcid.org/0000-0003-2461-3453
e-mail: sokol@sw-expert.com

Oleksandr Shevchenko
PhD,
Senior Lecturer
Department of Kharkiv National
University of Radioelectronics
Kharkiv, Ukraine
https://orcid.org/0000-0002-0068-4698
e-mail: shevchenko@sw-expert.com

МОДЕЛЬ САМОАДАПТИВНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО ПОБУДОВАНА НА ЗНАННЯХ ІЗ КОРПОРАТИВНОЇ ПАМ'ЯТІ

І.Є. Бібічков, В.В. Сокол, О.Ю. Шевченко

Проведено аналіз методів управління корпоративною пам'яттю з метою створення моделі самоадаптивного програмного забезпечення (ПО). Запропоновано концепцію розробки ПО, яке адаптується під індивідуальні потреби користувача для систем, в яких дані містяться в корпоративній пам'яті в онтологічному вигляді. Запропоновано спеціалізовану структуру, що побудована на онтології для розробки самоадаптивного ПО. Запропоновано механізм персоналізації для самоадаптивної моделі, який фільтрує дані і видає користувачеві актуальну інформацію. В результаті досліджень було розроблено модель самоадаптивного ПО, що побудована на корпоративній пам'яті, представленої у вигляді онтології. Розроблено модель, що адаптується, відповідно до змін, які були внесені в онтологію.

Ключові слова: *база знань, зовнішній вигляд, самоадаптація програмного забезпечення, онтологія, корпоративна пам'ять.*

MODEL OF SELF-ADAPTIVE SOFTWARE BASED ON KNOWLEDGE FROM CORPORATE MEMORY

I. Bibichkov, V. Sokol, O. Shevchenko

This paper contains analysis of methods for managing corporate memory with the purpose of creating self-adaptive software. There is proposed a software development concept which is based on using corporate memory as the main information storage. In this case the corporate memory is presented as ontology. There is proposed a specialized structure, which is based on ontology, for the development of self-adaptive software. There is proposed personalization implementation for self-adaptive model which filter user data and provide to user relevant information. As a result it was developed self-adaptation model, built on the corporate memory which is presented as an ontology. The created model is adapted when changes are done to the ontology, which contains corporate information.

The results of the research expand the application area of ontological knowledge bases. Particularly, it expands on the creation of a universal knowledge base, which determines the behavior and look of the software depending on the domain. The main advantage of this approach is the cost minimization of necessary changes to the source code of the application in combination with high development flexibility when the domain is changed.

Keywords: *knowledge base, appearance, self-adaptation of software, ontology, corporate memory.*