УДК 004.4'2

СОЗДАНИЕ ЖИЗНЕСПОСОБНЫХ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ С УПРАВЛЯЕМЫМИ ДЕКЛАРАТИВНЫМИ КОМПОНЕНТАМИ

Грибова Валерия Викторовна
Д-р техн. наук, зам. дир. по научной работе, e-mail: gribova@iacp.dvo.ru,
Москаленко Филипп Михайлович
Кандидат технических наук, старший научный сотрудник
Тимченко Вадим Андреевич
Кандидат технических наук, старший научный сотрудник
Шалфеева Елена Арефьевна
Кандидат технических наук, старший научный сотрудник
Федеральное государственное бюджетное учреждение науки Институт автоматики и процессов управления Дальневосточного отделения Российской академии наук,
690041, г. Владивосток, ул. Радио 5

Аннотация. В работе обсуждается проблема обеспечения жизнеспособности одного из классов программных систем - систем с базами знаний. Рассмотрены программные средства, предназначенные для разработки систем данного класса, проведен анализ механизмов, направленных на достижение этого важного свойства систем с базами знаний. Предложены механизмы повышения их жизнеспособности, основанные на построении каждого компонента по его декларативной модели, созданной с помощью единого языка описания моделей. Предложена трехуровневая расширяемая архитектура инструментария, реализующая все предложенные механизмы.

Ключевые слова: система, основанная на знаниях; экспертные системы; сопровождение программных систем; жизнеспособность программных систем; инструментальные средства разработки.

Цитирование: Грибова В.В., Москаленко Ф.М., Тимченко В.А., Шалфеева Е.А. Создание жизнеспособных интеллектуальных систем с управляемыми декларативными компонентами // Информационные и математические технологии в науке и управлении. 2018. № 3 (11). С. 6–17. DOI:10.25729/2413-0133-2018-3-01

Введение. Обеспечение жизнеспособности программных систем (ПС) является одной из ключевых проблем в области инженерии программного обеспечения [19]. Под жизнеспособностью понимается устойчивость (сохранение работоспособности) ПС к изменениям среды функционирования и способность к развитию (эволюционируемость) в течение жизни [2, 5, 19]. Жизнеспособность напрямую связана с прозрачностью (transparency) ПС, которая характеризуется тремя основными свойствами: доступность, понятность и релевантность информации (под информацией в данном контексте понимаются компоненты ПС) для заинтересованных групп [20].

Среди множества ПС выделяется их особый класс - системы с базами знаний (СБЗ), которые в настоящее время активно используются для решения многих научных и практических задач, можно говорить о периоде зрелости данного класса систем. В

архитектуре СБЗ, помимо традиционных компонентов - баз данных, бизнес-логики (решателя) и пользовательского интерфейса, имеется дополнительный компонент - база знаний. Для СБЗ проблема обеспечения жизнеспособности стоит наиболее остро, поскольку команда разработчиков систем этого класса включает, помимо программистов и дизайнеров интерфейса, инженеров по знаниям и экспертов предметной области. Для этого класса программных систем характерна постоянная изменчивость знаний, методов решения, объяснительной компоненты и пользовательского интерфейса.

Несмотря на ряд успехов, достигнутых как российскими так и зарубежными коллективами, разрабатывающими инструменты для проектирования данного класса систем, проблема жизнеспособности СБЗ остается "острой": эксперты предметной области все еще не могут самостоятельно (без посредников в лице инженеров знаний и программистов) решать задачу формирования баз знаний и их сопровождения; в решатель задач "встроены" часть знаний предметной области, что значительно затрудняет модификацию, а его структура сложна для понимания и внесения изменений; пользовательский интерфейс не адаптируется к требованиям пользователей, платформы, предметной области, он, как правило, имеет "жесткую" структуру, встроенную в решатель задач. Указанные особенности накладывают необходимость использования дополнительных специализированных механизмов обеспечения жизнеспособности данного класса систем.

Целью работы являются описание новых моделей и методов, направленных на обеспечение жизнеспособности СБЗ.

Обзор литературы. Можно выделить три основных типа средств разработки СБЗ: языки программирования, оболочки, а также специализированные инструментальные системы. Языки программирования общего назначения (PYTHON, C#, Java и др.) либо специализированные (LISP, ПРОЛОГ, SMALLTALK, FRL, Interlisp и др.) являются универсальным инструментом разработки СБЗ, основное преимущество которых - возможность реализовать широкий спектр систем с заданной моделью представления знаний, решателем и интерфейсом. Вместе с тем это наиболее трудоемкий способ их создания. В [3] отмечается, что сложность разработки интеллектуальных систем с использованием языков программирования оказалась настолько велика, что стала практически недоступной.

Проблемно-независимые и специализированные оболочки значительно упрощают создание СБЗ, однако ограничивают возможности их развития: имеют "жесткий" решатель и встроенный в него интерфейс, которые, в случае изменения требований, невозможно модифицировать. Также к недостаткам специализированных оболочек можно отнести ограничения области их использования, а проблемно-независимых - их "непрозрачность", прежде всего для экспертов предметной области, которые без инженеров знаний не могут самостоятельно формировать и сопровождать базу знаний, при этом часть знаний встроена в машину логического вывода, что также делает такие инструменты непрозрачными [3,8,9,16].

Специализированные инструментальные системы ориентированы на широкий класс СБЗ. Типичными представителями инструментальных систем являются: Level5 Object, G2, Clips, Loops, VITAL, KEATS, OSTIS, АТ-ТЕХНОЛОГИЯ и др. [3, 4, 6-8, 10, 11, 15]. Существенные отличия различных инструментальных систем связаны с набором предлагаемых разработчику компонентов, уровнем поддержки этапов жизненного цикла создания и сопровождения СБЗ, поддержке различных технологий разработки,

используемыми формализмами представления знаний и способе их формирования, отладки и связывания, используемым механизмам вывода, средствам формирования пользовательских интерфейсов.

Рассматривая данные инструменты с позиции жизнеспособности создаваемых с их использованием СБЗ, можно заметить, что эволюционное развитие инструментальных систем так или иначе ориентировано на достижение этой важной цели, что отражается, прежде всего, в средствах поддержки создания баз знаний (БЗ), которые являются одним из наиболее сложных этапов разработки таких систем, а также методам сопряжения БЗ с решателем задач.

Согласно [8], наиболее распространенной моделью представления знаний остается модель, основанная на правилах несмотря на то, что к настоящему времени число систем с данной моделью представления снизилось примерно 4 раза. Тренд на уменьшение систем с продукционной моделью можно объяснить расширением доступного инструментария, поддерживающего другие модели представления знаний, наиболее адекватные предметным областям и задачам, для которых продукционное представление не является принятым и удобным в предметной области. Делаются определенные попытки, например в G2, упростить понимание правил экспертами, представляя их на естественном языке или в графическом виде. Учитывая потребности пользователей в различных моделях представления знаний, многие инструментальные системы разработки предлагают смешанный механизм их представления. Например, LOOP, G2 использует два типа представления знаний: правила и объектно-ориентированное представление, ART - правила для процедурных, фреймоподобные и объектно-ориентированные модели для декларативных знаний, KEE-продукции, фреймы с наследованием, объектно-ориентированные модели. Однако предлагаемые типы представлений не ориентированы на самостоятельное (без инженеров знаний) формирование и сопровождение знаний (их модифицирование и развитие) экспертами предметной области.

Для формирования баз знаний можно рассматривать специализированные инструменты, основанные на онтологиях: IWE, Protégé, OntoEdit, GrOWL, Graphl, RDFGravity, WebVOWL, Ontolingua, OntoSaurus, OilEd, WebOnto, WebODE [18]. Однако они реализуют, как правило, объектно-ориентированную парадигму представления знаний, непонятную большинству экспертов предметной области. Также остается открытым вопрос их сопряжения с решателем задач и пользовательским интерфейсом.

В соответствии с моделью представления знаний предлагается и соответствующий механизм реализации решателя (рассуждений). Разработчики инструментария, поддерживающего данный механизм рассуждений, сами отмечают, что "длительное их использование показало, что этот подход громоздкий и негибкий". Для обеспечения некоторой степени гибкости и, как следствие, жизнеспособности, используются различные механизмы вывода, например, G2 поддерживает обратную цепочку рассуждений (дедуктивную, поиск цели), прямую (индуктивную, на основе событий), а также смешанные заключения, при этом одно и то же правило может использоваться как в прямой, так и в обратной цепочке рассуждений. Для объектно-ориентированного представления знаний часто используются такие языки программирования, как Java и C++ и C#. Некоторые системы для реализации решателя используют языки логического программирования, например, инструмент OLP (Ontological Logic Programming), который сочетает в себе

преимущества рассуждения на основе онтологий (ontological reasoning) и логическое программирование на языке Prolog. Если в инструментальной системе поддерживается несколько моделей представления знаний, соответственно, поддерживается несколько механизмов и языков реализации решателя, например, система SWORIER использует механизм рассуждений на основе онтологий и правил. Такие решения, с одной стороны, бесспорно, направлены на возможность выбора наиболее адекватной модели представления знаний и соответствующего ей решателя, с другой стороны, прозрачность таких систем остается достаточно низкой.

Поддержка разработки интерфейса осуществляется несколькими способами. Разработчику предлагается набор средств, поддерживаемых инструментарием, например, [4, 7]. Он зависит от конкретной системы. Это может быть специализированный язык программирования, средства, похожие на Построители интерфейса, предлагаемые различными CASE-средствами: набор элементов WIMP-интерфейса, которые может определить пользователь, указать их свойства и связать их с командами (действиями пользователя и/или решателя) и/или данными (входными или выходными). Сценарий взаимодействия в этом случае встраивается в решатель. Разработка интерфейса может осуществляться средствами языка, на котором проектируется решатель, возможно взаимодействие с набором различных библиотек, поддерживаемых инструментарием.

Таким образом, наиболее гибким инструментом для реализации СБЗ являются специализированные инструментальные системы, они позволяют реализовать различные классы СБЗ. Однако проблема жизнеспособности данного класса систем все еще далека от окончательного решения. Поэтому поиск новых, улучшенных механизмов повышения жизнеспособности таких систем остается актуальной задачей.

Основные принципы обеспечения жизнеспособности СБЗ. Жизнеспособность ПС и СБЗ в частности во многом определяется их прозрачностью. Одним из основных атрибутов прозрачных программных систем является понятность для заинтересованных групп. В СБЗ такими группами являются эксперты предметной области, ответственные за разработку и сопровождение БЗ, программисты, обеспечивающие создание и сопровождение решателя, а также дизайнеры интерфейса, реализующие как интерфейс решателя СБЗ, так и интерфейс редакторов для создания и сопровождения БЗ. Для СБЗ принципиально важно использование актуальных знаний, которые должны формировать и сопровождать эксперты предметной области, и/или которые должны быть сформированы индуктивно, но их представление должно быть понятно экспертам. Это возможно только в случае, если язык представления знаний будет ориентирован на класс решаемых задач, а его терминология будет ясна и понятна экспертам. В терминах этого языка должны формироваться не только базы знаний и данных, но также описываться все контекстные условия и наборы ограничений, которые необходимо учитывать при формировании базы знаний, а также осуществляться запросы к базе знаний.

Для обеспечения прозрачности создаваемых СБЗ также необходим решатель, в который не включены предметные знания, а структура и модули, из которых он состоит, должны быть понятны сопровождающему. Это возможно в случае, если большая часть решателя представлена декларативно (что позволит управлять решателями с помощью редакторов). Немаловажным фактором является повторная используемость уже разработанных модулей.

Прозрачность пользовательского интерфейса может быть обеспечена, во-первых, за счет предоставления пользователям различных типов пользовательского интерфейса, наиболее адекватно подходящего модели представления информации (это касается также и редакторов компонентов СБЗ), во-вторых, за счет разделения данных, логики их обработки и способа представления с целью обеспечения возможности независимого изменения каждого из компонентов.

Для реализации данных требований предлагаются следующие основные решения:

- единые принципы создания компонентов СБЗ;
- двухуровневый подход к формированию компонентов: сначала формируется структурная декларативная модель (онтология компонента), затем по ней создается необходимый компонент СБЗ;
- единый язык и редактор для формирования моделей всех компонентов;
- единый унифицированный внутренний формат всех компонентов;
- автоматическая генерация редакторов для создания компонентов по их моделям;
- агентный подход к созданию решателей задач;
- реализация инструментария и СБЗ как облачных сервисов.

В основе всех предлагаемых решений лежит язык описания моделей, который обладает следующими основными свойствами: язык является декларативным, он позволяет описывать произвольные модели, ориентированные и адаптированные к терминологии и форме, принятой у разработчиков компонентов СБЗ; формирование необходимых компонентов выполняется "сверху-вниз", с переходом от общих понятий к деталям, которые уточняют общие понятия и сущности; язык поддерживает соответствие между моделью и компонентом СБЗ, формируемым на ее основе. Модели компонентов СБЗ, сформированные на языке описания моделей, представляются в форме связных размеченных корневых иерархических бинарных орграфов с возможными петлями и циклами. Разметку имеют как вершины орграфа, так и его дуги. Разметка определяет семантику правил формирования (создания и модификации) компонентов, накладывая ограничения на их структуру и содержание. Подробное описание языка приведено в работах [12,13].

Разработка и сопровождение базы знаний (базы данных). В соответствии двухуровневым подходом к формированию компонентов СБЗ, на первом этапе формируется специализированная модель представления знаний (данных) - онтология знаний, которая учитывает специфику организации знаний и данных в конкретной предметной области. Чтобы упростить ее разработку, обеспечив интерактивное формирование онтологии, используется редактор моделей, ориентированный на широкий круг пользователей. Его использование освобождает разработчиков от изучения синтаксиса языка. Разработчик при таком редактировании выбирает допустимые в каждой конкретном контексте разработки операторы для формирования необходимой модели знаний (данных). Далее, по модели знаний (данных) генератор редакторов компонента формирует редактор базы знаний/ базы данных (см. рис.1).

Созданные на основе модели знаний базы знаний и сложно-структурированных данных имеют единое представление и внутренний формат хранения, что позволяет обеспечить единые средства их формирования и сопровождения, а также стандартизированный и расширяемый набор программных интерфейсов для унификации и упрощения доступа к ним. Эксперты предметной области получают возможность

формировать базы знаний и данных в терминах своих систем понятий, а не в терминах языка представления знаний и данных.

Разработка и сопровождение решателя задач. Решатель задач представляет собой множество агентов, взаимодействующих друг с другом посредством обмена сообщениями. В соответствие с двухуровневым подходом (сначала разрабатывается модель компонента, а затем компонент), разработчикам предлагаются единые для всех СБЗ модели (онтологии) агента, шаблона сообщений и решателя, освобождающие разработчиков соответствующих компонентов от их создания. Для организации запуска решателей с конкретными наборами входных и выходных данных, выделяется также модель сервиса. Агенты и шаблоны сообщений помимо декларативной части, имеют императивную компоненту, которая разрабатывается на языке Java. Для повышения прозрачности императивной части агента и шаблона сообщений после описания их декларативной части, выполняется генерация заготовки исходного кода разрабатываемого агента или шаблона сообщения. Разработанный императивный код связывается с соответствующей вершиной модели агента или шаблона сообщений.

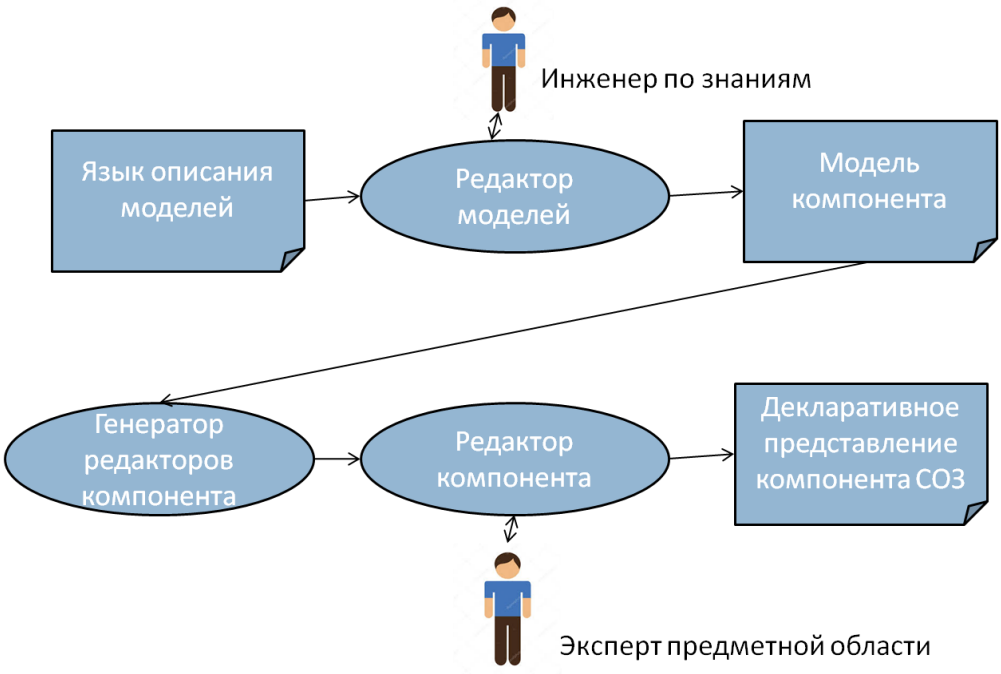

Рис. 1. Процесс формирования базы знаний

Разработка и сопровождение пользовательского интерфейса. Под разработкой интерфейса СБЗ подразумевается разработка web-интерфейса, поскольку СБЗ реализуются как облачные сервисы. В основу проектирования интерфейса положена концепция “MVC” (Модель-Представление-Контроллер). Ее основополагающий принцип состоит в разделении данных, логики их обработки и способа их представления с целью обеспечения возможности независимого изменения каждого из компонентов. Проекция данной концепции на модель интерфейса представлена следующим образом. Модель включает в себя: модель абстрактного пользовательского интерфейса, содержащего описание структуры стандартных интерфейсных элементов (простых и интерфейсных элементов-контейнеров) WIMP-интерфейсов и способа их рекурсивной организации в единую вложенную структуру, а также программный интерфейс для формирования абстрактных интерфейсов – набор

высокоуровневых функций для создания фрагментов абстрактных интерфейсов. Компонент Представление реализуется системным агентом Вид, основной функцией которого является формирование описания конкретного интерфейса на основе описания абстрактного интерфейса и правил отображения представлений интерфейсных элементов и их атрибутов в терминах модели абстрактного пользовательского интерфейса. Компонент Контроллер представлен агентами, играющими роль Интерфейсного контроллера в составе различных решателей задач. Эти агенты взаимодействуют с агентом Вид посредством обмена сообщениями по определенным шаблонам и реализуют логику обработки.

Концептуальная архитектура инструментария для разработки жизнеспособных СБЗ. Инструментарий для разработки ПС также является ПС, соответственно, его жизнеспособность также важна, как и жизнеспособность любого другого класса программного обеспечения. В работе [17] отмечается, что без непрерывного развития инструментария невозможно создание и развитие программного обеспечения на его основе, отвечающего постоянно изменяющимся требованиям. Соответственно, комплексное решение проблемы жизнеспособности СБЗ означает также и обеспечение жизнеспособности инструментария, с помощью которого СБЗ создаются и сопровождаются. Как правило, инструментарий развивается его разработчиками, но этого, недостаточно [17]. Жизнеспособный инструментарий должен развиваться также и его пользователями (в данном случае разработчиками СБЗ). Для успешной реализации этого требования необходимо, чтобы средства расширения инструментария и средства создания СБЗ были основаны на единых принципах и технологиях.

Учитывая вышесказанное, предлагается трехуровневая архитектура инструментария для разработки СБЗ, состоящая из *Инструментального ядра, Базового инструментария* и *Расширяемого инструментария*.

Инструментальное ядро определяет основной принцип построения компонентов как самого инструментария, так и СБЗ (созданных с помощью инструментария), и включает Язык описания моделей; Редактор моделей; Генератор редакторов компонентов. Декларативный Язык описания моделей используется для создания моделей компонентов, независимо от назначения, основные принципы его создания описаны выше. Для упрощения создания моделей на этом языке в состав ядра входит Редактор моделей, который позволяет разработчикам создавать их в наиболее простой и удобной форме. В Инструментальное ядро также входит Генератор редакторов компонентов, который предназначен для автоматической генерации редакторов декларативных компонентов по их моделям (онтологиям). Генератор редакторов компонентов отвечает за генерацию пользовательского интерфейса и сценария формирования компонента с проверкой заданных в модели контекстных условий, полноты его формирования. Инструментальное ядро достаточно для создания всех декларативных компонентов СБЗ по их моделям и управления ими.

Основная задача *Базового инструментария* заключается в следующем: предоставить пользователю - разработчику набор средств создания программных компонентов как СБЗ, так и других прикладных или инструментальных сервисов; обеспечить управление процессом создания таких программных компонентов, включая их сборку, связывание с информационными компонентами (например, базой знаний, данных, разработанных с использованием Инструментального ядра), запуск, а также организацию инфраструктуры на всех уровнях инструментария.

Поскольку все компоненты формируются по их структурным декларативным моделям, соответственно, данный уровень инструментария включает редакторы для создания компонентов. Такие компоненты формируются Генератором редакторов компонентов для каждой структурной модели. Кроме указанных выше элементов, этот уровень содержит внешние программные элементы. Эти элементы используются для создания пользовательского интерфейса СОЗ и императивной части тех компонентов, где декларативного представления недостаточно для задания необходимых им функциональных механизмов, а также систему управления. Сопровождение элементов базового инструментария осуществляется его разработчиками.

Расширяемый инструментарий, в первую очередь, предназначен для разработчиков СБЗ, которые могут включать в его состав новые, удобные инструментальные средства для сопровождения разработанных ими СБЗ, новые технологии разработки СБЗ; специализированные либо универсальные оболочки экспертных систем. Расширение может осуществляться с использованием Инструментального ядра, Базового инструментария, а также за счет средств и инструментальных механизмов Расширяемого инструментария. Таким образом, достигается рекурсивное использование разработанных компонентов Расширяемого инструментария. Данный уровень инструментария может расширяться как разработчиками инструментария, так и его пользователями (экспертами, инженерами знаний, программистами, аналитиками).

Трехуровневая архитектура положена в основу облачной платформы IACPaaS (https://iacpaas.dvo.ru) [1, 14], которая в настоящее время доступна для использования всем разработчикам СБЗ и их компонентов. К настоящему времени на платформе созданы порталы знаний по медицине, математике, автономным необитаемым подводным аппаратам, экспериментальной диагностике сельскохозяйственных культур, защите информации, педагогической психологии, технологии программирования.

Заключение. В работе рассмотрены механизмы, направленные на обеспечение жизнеспособности одного из классов программных систем - систем с базами знаний. Их основным отличием от других классов программных систем является наличие базы знаний, подверженной постоянным изменениям в течение жизненного цикла, которую должны создавать и сопровождать эксперты предметной области. В основе предлагаемых решений лежит разработанный авторами язык описания моделей, который предоставляет средства спецификации моделей в форме связных размеченных корневых иерархических бинарных орграфов с возможными петлями и циклами. Созданные на основе модели компоненты СБЗ имеют единое представление и внутренний формат хранения, а также стандартизированный и расширяемый набор программных интерфейсов для унификации и упрощения доступа к ним. Эксперты предметной области получают возможность формировать базы знаний и данных в терминах своих систем понятий, а не в терминах языка представления знаний и данных. Архитектура решателя задач включает декларативно представляемые программные единицы, которые могут составлять динамическую конфигурацию, взаимодействовать посредством передачи сообщений, и структура которых также описывается по декларативной модели. Универсальный язык и унифицированное внутреннее представление как моделей, так и отдельных их компонентов, позволяет применить единые принципы генерации редакторов по типу компонентов СБЗ. Все предлагаемые авторами идеи реализованы на облачной платформе IACPaaS. При этом инструментальные сервисы,

обеспечивающие поддержку технологии разработки, создаются на тех же принципах, что и прикладные сервисы.

Вместе с тем, опыт использования данной платформы и наличие обратной связи с её пользователями поставили перед авторами ряд новых научных задач, среди которых создание языково-ориентированных запросов к базам знаний, методов создания адаптивных пользовательских интерфейсов разных типов для решателей СБЗ и редакторов баз знаний. Их решение откроет дополнительные новые возможности по повышению жизнеспособности данного класса систем.

Работа выполнена при частичной финансовой поддержке РФФИ (проекты 18-07-01079, 16-07-00340).

СПИСОК ЛИТЕРАТУРЫ

1. Грибова В.В., Клещев А.С., Москаленко Ф.М., Тимченко В.А., Федорищев Л.А., Шалфеева Е.А. Облачная платформа IACPaaS: текущее состояние и перспективы развития // Информационные и математические технологии в науке и управлении. 2016. № 2. С. 94–102.
2. Кряжич О.А. Обеспечение жизнеспособности информации во времени при ее обработке в СППР // Математические машины и системы. 2015. №. 2. С. 170−176.
3. Рыбина Г.В. Интеллектуальные системы: от А до Я. Серия монографий в трех книгах. Кн. 3. Проблемно-специализированные интеллектуальные системы. Инструментальные средства построения интеллектуальных систем. М.: Научтехлитиздат. 2015. 180 с.
4. Рыбина Г.В. Интеллектуальная технология построения обучающих интегрированных экспертных систем: новые возможности // Открытое образование. 2017. Т. 21. № 4. С. 43–57.
5. Черников Б.В. Управление качеством программного обеспечения. М.: ИД «ФОРУМ»:Инфра-М. 2012. 240 с.
6. C.F. Tan, L.S. Wahidin, S.N. Khalil, N. Tamaldin, J. Hu, G.W.M. Rauterberg. The application of expert system: A review of research and applications // ARPN Journal of Engineering and Applied Sciences. 2016. Vol. 11. Pp. 2448–2453.
7. Davydenko I. Semantic models, method and tools of knowledge bases coordinated development based on reusable components // Open Semantic Technologies for Intelligent Systems. 2018. Pp. 99–118.
8. Emmanuel C. Ogu, Adekunle Y.A. Basic Concepts of Expert System Shells and an Efficient Model for Knowledge Acquisition // Intern. J. of Science and Research Intern. Journal of Science and Research (IJSR). India Online ISSN: 2319-7064. 2013. Vol. 2. Issue 4. Pp. 554–559.
9. Gabriel Jakobson, Robert Weihmayer, Mark Weissman. A Domain-Oriented Expert System Shell for Telecommunication Network Alarm Correlation. Network Management and Control. Springer, Boston, MA. 1994. Pp. 365–380.
10. Gensym G2. The World's Leading Software Platform for Real-Time Expert System Application. Available at: URL: http://www.gensym.com/wp-content/uploads/Gensym-l-G2.pdf (accessed 25.05.2018)

11. Golenkov V., Gulyakina N., Grakova N., Davydenko I., Nikulenko V., Eremeev A., Tarasov V. From training intelligent systems to training their development tools // Open Semantic Technologies for Intelligent Systems. 2018. Pp. 88–98.
12. Gribova V.V., Kleshchev A.S., Moskalenko F.M., Timchenko V.A. A Two-level Model of Information Units with Complex Structure that Correspond to the Questioning Metaphor // Automatic Documentation and Mathematical Linguistics. 2015. vol. 49. no. 5. Pp. 172–181.
13. Gribova V.V., Kleshchev A.S., Moskalenko F.M., Timchenko V.A. A Model for Generation of Directed Graphs of Information by the Directed Graph of Metainformation for a Two_Level Model of Information Units with a Complex Structure // Automatic Documentation and Mathematical Linguistics. 2015. vol. 49. no. 6. Pp. 221–231. ISSN 0005-1055.
14. Gribova V., Kleschev A., Moskalenko P., Timchenko V., Fedorischev L., Shalfeeva E. The IACPaaS cloud platform: Features and perspectives // Computer Technology and Applications (RPC), 2017 Second Russia and Pacific Conference on. IEEE. 2017. Pp. 80–84.
15. Jabbar H.K. and Khan R.Z. Development of Expert Systems // Intern. J. of Information Technology & Management Information System (IJITMIS). 2015. vol. 6. no. 2. Pp. 49-59.
16. Kumar S., Prasad R. Importance of Expert System Shell in Development of Expert System // Intern. J. of Innovative Research & Development. 2015. Vol. 4. Issue 3. Pp. 128–133.
17. Musen M. The protégé project: a look back and a look forward. Newsletter AI Matters. 2015. vol. 1. iss. 4. Pp. 4–12. DOI: 10.1145/2757001.2757003
18. Ontology Tools [Electronic resource], Open Semantic Framework mode of access: http://wiki.opensemanticframework.org/index.php/ Ontology_Tools.
19. Pressman R.S. Software engineering: a practitioner's approach. McGraw-Hill. 2010. 930 p. ISBN: 0073375977
20. Yu-Cheng Tu. Transparency in Software Engineering // A thesis submitted in fulfillment of the requirements of Doctor of Philosophy in Electrical and Electronic Engineering. The University of Auckland. New Zealand. 2014. 337 p.

UDK 004.4'2

VIABLE INTELLIGENT SYSTEMS DEVELOPMENT WITH CONTROLLED DECLARATIVE COMPONENTS

Valeria V. Gribova
Dr., Research Deputy Director, e-mail: gribova@iacp.dvo.ru,
Filipp M. Moskalenko
PhD, senior scientific employee,
Vadim A. Timchenko
PhD, senior scientific employee,
Elena A. Shalfeeva
PhD, senior scientific employee,
Institute for Automation and Control Processes, Far Eastern Branch,
Russian Academy of Sciences, 5, Radio Str., 690041, Vladivostok, Russia

Abstract. The paper discusses the problem of ensuring the viability of one of the classes of software systems - systems with knowledge bases. The software tools designed for the development of systems of this class are considered, the analysis of mechanisms of achieving this important property of systems with knowledge bases is carried out. The mechanisms of increasing their viability based on the each component development according to its declarative model created on a model description language are proposed. A three-level extensible architecture of tools that implements all the proposed mechanisms is proposed.

Keywords: knowledge-based system; expert systems; maintenance of software systems; viability of software systems; development tools

References

1. Gribova V.V., Kleshchev A.S., Moskalenko F.M., Timchenko V.A., Fedorishchev L.A., Shalfeeva E.A. Oblachnaya platforma IACPaaS: tekushchee sostoyanie i perspektivy razvitiya [IACPaaS cloud platform: Current state and evolution trends] // Informatsionnyye i matematicheskiye tekhnologii v nauke i upravlenii = Information and mathematical technologies in science and management. 2016. no. 2. Pp. 94–102 (in Russian).
2. Kryazhich O.A. Obespechenie zhiznesposobnosti informacii vo vremeni pri ee obrabotke v SPPR [Ensuring the viability of the information during its processing in decision support systems] // Matematicheskie mashiny i sistemy = Mathematical Machines and Systems. 2015. no. 2. Pp. 170–176 (in Russian).
3. Rybina G.V. Intellektual'nye sistemy: ot A do YA. Seriya monografij v trekh knigah. Kn. 3. Problemno-specializirovannye intellektual'nye sistemy. Instrumental'nye sredstva postroeniya intellektual'nyh system [Intelligent systems: A to Z. A series of monographs in three books. Book 3. Problem-specialized intelligent systems. Tools for building intelligent systems]. Moscow. Nauchtekhlitizdat = NAUCHTEKHLITIZDAT Publishing House. 2015. 180 p. (in Russian).
4. Rybina G.V. Intellektual'naya tekhnologiya postroeniya obuchayushchih integrirovannyh ehkspertnyh sistem: novye vozmozhnosti [Intelligent technology for construction of tutoring integrated expert systems: new aspects] // Otkrytoye obrazovaniye = Open Education. 2017. vol. 21. no 4. Pp. 43–57 (in Russian).

5. Chernikov B.V. Upravlenie kachestvom programmnogo obespecheniya [Software quality management]. Moscow: Publishing House «FORUM»:Infra-M. 2012. 240 p. (in Russian)
6. C.F. Tan, L.S. Wahidin, S.N. Khalil, N. Tamaldin, J. Hu, G.W.M. Rauterberg. The application of expert system: A review of research and applications // ARPN Journal of Engineering and Applied Sciences. 2016. Vol. 11. Pp. 2448–2453.
7. Davydenko I. Semantic models, method and tools of knowledge bases coordinated development based on reusable components // Open Semantic Technologies for Intelligent Systems. 2018. Pp. 99–118.
8. Emmanuel C. Ogu, Adekunle Y.A. Basic Concepts of Expert System Shells and an Efficient Model for Knowledge Acquisition // Intern. J. of Science and Research Intern. Journal of Science and Research (IJSR). India Online ISSN: 2319-7064. 2013. Vol. 2. Issue 4. Pp. 554–559.
9. Gabriel Jakobson, Robert Weihmayer, Mark Weissman. A Domain-Oriented Expert System Shell for Telecommunication Network Alarm Correlation. Network Management and Control. Springer, Boston, MA. 1994. Pp. 365–380.
10. Gensym G2. The World's Leading Software Platform for Real-Time Expert System Application. Available at: URL: http://www.gensym.com/wp-content/uploads/Gensym-l-G2.pdf (accessed 25.05.2018)
11. Golenkov V., Gulyakina N., Grakova N., Davydenko I., Nikulenko V., Eremeev A., Tarasov V. From training intelligent systems to training their development tools // Open Semantic Technologies for Intelligent Systems. 2018. Pp. 88–98.
12. Gribova V.V., Kleshchev A.S., Moskalenko F.M., Timchenko V.A. A Two-level Model of Information Units with Complex Structure that Correspond to the Questioning Metaphor // Automatic Documentation and Mathematical Linguistics. 2015. vol. 49. no. 5. Pp. 172–181.
13. Gribova V.V., Kleshchev A.S., Moskalenko F.M., Timchenko V.A. A Model for Generation of Directed Graphs of Information by the Directed Graph of Metainformation for a Two_Level Model of Information Units with a Complex Structure // Automatic Documentation and Mathematical Linguistics. 2015. vol. 49. no. 6. Pp. 221–231. ISSN 0005-1055.
14. Gribova V., Kleschev A., Moskalenko P., Timchenko V., Fedorischev L., Shalfeeva E. The IACPaaS cloud platform: Features and perspectives // Computer Technology and Applications (RPC), 2017 Second Russia and Pacific Conference on. IEEE. 2017. Pp. 80–84.
15. Jabbar H.K. and Khan R.Z. Development of Expert Systems // Intern. J. of Information Technology & Management Information System (IJITMIS). 2015. vol. 6. no. 2. Pp. 49-59.
16. Kumar S., Prasad R. Importance of Expert System Shell in Development of Expert System // Intern. J. of Innovative Research & Development. 2015. Vol. 4. Issue 3. Pp. 128–133.
17. Musen M. The protégé project: a look back and a look forward. Newsletter AI Matters. 2015. vol. 1. iss. 4. Pp. 4–12. DOI: 10.1145/2757001.2757003
18. Ontology Tools [Electronic resource], Open Semantic Framework mode of access: http://wiki.opensemanticframework.org/index.php/Ontology_Tools.
19. Pressman R.S. Software engineering: a practitioner's approach. McGraw-Hill. 2010. 930 p. ISBN: 0073375977
20. Yu-Cheng Tu. Transparency in Software Engineering // A thesis submitted in fulfillment of the requirements of Doctor of Philosophy in Electrical and Electronic Engineering. The University of Auckland. New Zealand. 2014. 337 p.