УДК 004.4`2 + 004.89 DOI: 10.17587/it.25.738-746

В. В. Грибова, д-р техн. наук, ст. науч. сотр,
зам. директора по научной работе, e-mail: gribova@iacp.dvo.ru,
Е. А. Шалфеева, канд. техн. наук, доц., ст. науч. сотр, e-mail: shalf@dvo.ru,
Институт автоматики и процессов управления ДВО РАН, Владивосток

Обеспечение жизнеспособности систем, основанных на знаниях[1]

Рассматривается модель жизнеспособности систем, основанных на знаниях, конструируемых с помощью онтолого-ориентированной среды развития, где каждый компонент формируется в терминах явно представленной онтологии предметной области с использованием адаптируемой инструментальной поддержки.

Ключевые слова: *интеллектуальные системы, база знаний, основанные на знаниях системы, онтолого-ориентированный алгоритм, жизнеспособность, декларативный инструментарий, среда развития*

Введение

Перед разработчиком любой программной системы (ПС) стоит задача не только создать программный продукт, отвечающий всем текущим требованиям к нему, но и заложить в него механизмы, обеспечивающие его неизбежное последующее сопровождение, вызываемое изменениями требований пользователей к интерфейсу и функциональности, знаний предметной области, условий эксплуатации.

На процесс сопровождения приходится значительная доля трудозатрат в процессе жизненного цикла программной системы, по оценкам специалистов — от 50 до 80 %. Поэтому необходимо проектировать ПС, закладывая механизмы для упрощения этого процесса.

Известными механизмами, используемыми в программной инженерии для упрощения сопровождаемости ПС, являются [1—4]:

- архитектурные решения (разделение на слабо сцепленные компоненты с логически понятными функциями);
- декларативное представление компонентов ПС (например, с помощью структурных и графических редакторов) и использование "методов 4GL", смещающих усилия "от программирования к конструированию";
- средства автоматизации построения проектных моделей и автоматической генерации по ним фрагментов кода (генерации схем СУБД, генерации программного кода);
- средства связывания компонентов друг с другом (например, классов, использующих друг друга, или хранимых таблиц БД с визуализируемыми результатами запросов);
- использование типовых архитектурных решений, "навязываемых" так называемыми оболочками (от СУБД типа MSAccess до платформы типа 1С);
- введение в процесс жизненного цикла вида деятельности "управление требованиями" (с инструментальной поддержкой, как в Rational Rose для RUP);
- разделение компетенций (между дизайнером интерфейса, специалистом по сетевому взаимодействию, проектировщиком базы данных, наполнителями баз, специалистами по "большим данным", программистами и другими типами разработчиков).

Поскольку *системы, основанные на знаниях* (СОЗ), — особый класс систем, дополнительным архитектурным компонентом которых является база знаний, и команда разработчиков пополняется экспертами предметной области и когнитологами, проблема сопровождаемости СОЗ стоит особо остро [5—7]. Указанная специфика накладывает необходимость использования дополнительных специализированных

[1]Работа выполнена при частичной финансовой поддержке РФФИ (проекты № 18-07-01079 и 19-07-00244).

механизмов обеспечения их сопровождаемости и жизнеспособности.

Целью работы является представление обобщенной архитектурной модели сопровождаемой СОЗ и совокупности средств обеспечения ее жизнеспособности.

1. Понятия сопровождаемости и жизнеспособности программных систем

С появлением сложных программных систем, на разработку которых стали требоваться усилия, измеряемые человеко-годами, стала актуальна проблема их сопровождаемости.

Традиционно *сопровождаемость* ПС означает возможность ее исправления, переструктурирования (рефакторинга), адаптации (к аппаратуре и системному программному обеспечению, к новым видам человеко-машинных интерфейсов, к новому типу пользователей) и расширения (новыми функциями по желанию пользователей). Но следующие десятилетия обозначили новое полезное свойство систем — жизнеспособность как устойчивость (сохранение работоспособности и полезности) ПС к изменениям окружающей среды [8] и эволюционируемость как способность приспосабливаться к изменчивости требований в течение "жизни" с наименее возможной стоимостью, поддерживая архитектурную целостность [9].

Будем называть *жизнеспособностью* устойчивость (сохранение работоспособности) ПС к некоторым изменениям среды функционирования и способность к развитию (эволюционируемость) в течение жизни, которые обеспечиваются через адаптивность (автоматическое изменение) к изменениям в среде и адаптируемость (интерактивное изменение) ПС в ответ на новые требования.

Адаптируемость реализуется через:

- декларативное представление компонентов, обеспечивающее, с одной стороны, программно-обрабатываемое их представление, с другой, наглядное и понятное, соответствующее системе терминов (системе понятий, языку общения) разработчика;
- наличие структурных и графических редакторов для поддержки разработки всех компонентов ПС;
- автоматизацию разработки компонентов (генерацию схем СУБД, генерацию программного кода) с проверкой их архитектурной целостности.

Адаптивность обеспечивается механизмами:

- машинного обучения, которые на основе наборов входных данных улучшают и совершенствуют алгоритмы обработки данных;
- обратной связи с пользователем, окружающей средой, обеспечивающих настройку интерфейса и сценария взаимодействия с ПС.

Таким образом, жизнеспособность расширяет традиционное понятие сопровождаемости, прежде всего, наличием механизмов адаптивности.

2. Потребность разных классов программных систем в развитии

Затраты на фазу сопровождения в жизненном цикле ПС занимают в среднем около 50 % [10, 11], имеются отчеты про долю до 80...90 % [12], хотя известно, что некоторым типам ПС (таким как встроенное программное обеспечение) сопровождение не требуется [1].

Потребуется ли модифицирование в связи с изменениями условий эксплуатации, требований пользователей, предметной области и др., зависит от назначения ПС, от уникальности (рыночная или заказная) и от типа обрабатываемой информации.

Если в качестве категорий ПС взять:

- прикладные ПС (в том числе СОЗ), создаваемые и приобретаемые для поддержки профессиональной деятельности специалистов или для улучшения обслуживания клиентов, а также для обучения,
- оболочки для создания прикладных ПС и
- инструментальные ПС,

то оценка потребности модифицирования приблизительно такова.

В процессе эксплуатации как прикладных ПС, так и инструментальных средств для их разработки появляется потребность в:

- исправлении ошибок;
- добавлении пользовательских функций (расширение);
- адаптации (новые приборы, устройства, платформы);
- изменении пользовательского интерфейса (ПИФ) специалистов;
- уточнении формата или состава информации (улучшение).

В случае с прикладными СОЗ, связанными с решением традиционных интеллектуальных

задач (диагностика, планирование, прогноз и т. д.), акценты смещаются. Здесь реже характерна потребность в добавлении новых пользовательских функций ("расширении"), скорее ожидается изменчивость знаний (появление новых методов диагностики, выявление новых факторов, влияющих на ситуации, и т. п.) и методов решения, объяснительной компоненты и пользовательского интерфейса. Причина в том, что постановки многих известных задач — диагностики, планирования и т. д. — достаточно устойчивы, а новые интеллектуальные задачи чаще всего являются конкретизацией/уточнением известных или их комбинацией. Развиваемость (модифицируемость) прикладных ПС также зависит и от инструментария для их разработки.

Разным категориям ПС свойственны и разные подходы к развитию (модернизации).

Для программных систем с заранее предопределенной архитектурой (обеспечиваемой фреймворками и оболочками) сопровождение и объединение разных компонентов заметно проще, поскольку проблема адаптируемости к устройствам, платформам, гибкость ПИФ, формата хранимой и другой информации "перекладывается" на отмеченные инструменты.

Сами фреймворки, оболочки, инструментальные среды подвергаются адаптации и расширению в рамках своих языковых сред (более низкого уровня), причем необходимые пользователю компоненты инструментария могут быть как "собственными", так и заимствованными или интегрированными из других инструментов (так, в Rational Rose-инструментарии "проектируются собственно объектная модель, модель данных, компонентная модель, т. е. архитектурно существенные элементы", а "пользовательский интерфейс проектируется в среде Delphi" с преобразованием в программный код "в модели Rational Rose").

Подход к сопровождению прикладных программных систем схож с сопровождением средств разработки, т. е. инструментальных систем.

3. Особенности программных систем, основанных на знаниях

Системами, основанными на знаниях, называют класс таких программных систем искусственного интеллекта, в которых предметные знания представлены в явном виде и отделены от прочих знаний системы. При разработке таких систем первична постановка задачи, знания, метод ее решения; типичное специальное требование предъявляется к представлению информации — все должно быть в форме, понятной пользователям без посредников: данные, знания, результат решения задач и ожидаемое объяснение результата.

Спецификация задачи на основе баз знаний есть предикат $P(x, k, y)$, где $x \in X$, $k \in K(X, Y)$, $y \in Y$. Особенностью спецификации задачи является то, что для нее не является справедливым утверждение о существовании решения [13]. Предполагается, что существует "правильная" база знаний $k^* \in K(X, Y)$, такая что $\forall x \in X \; \exists \, y \in Y \; P(x, k^*, y)$; однако предположение это не может быть доказано. В общем случае эта "правильная" база знаний неизвестна, а для любой другой базы знаний утверждение о существовании решения не обязано быть справедливым [13]. Но в предположении справедливости "слабого" утверждения (о существовании "правильной" базы знаний) можно разрабатывать алгоритм решения задачи (реализацию на алгоритмическом языке частично определенного функционального отображения A: $<X, K(X, Y)> \; —> \; Y$) [13].

Такие задачи в современных СОЗ решаются "с применением эвристик, включая эмпирическую индукцию, аналогию и дедукцию" [14], часто с использованием более одного метода имитации интеллектуальной деятельности человека [15]. Современные СОЗ предлагают удобный ПИФ, они могут получать данные от современных средств измерения и часто интегрируются со статистическими и другими программными компонентами, не использующими формализованные знания (в том числе со стандартными пакетами прикладных программ) [16, 17].

Жизнеспособность для системы, основанной на знаниях, — это сохранение работоспособности и полезности в условиях изменчивости предметной области (чаще — знаний, реже — онтологии, среды функционирования и пользовательского интерфейса).

Поскольку жизнеспособность СОЗ проявляется, прежде всего, в условиях изменчивости знаний предметной области, а "способность к адаптации в соответствии с изменением множества фактов и знаний" считается одним из аспектов интеллекта [14], то для большинства предметных областей, связанных с решением интеллектуальных задач, подразумевается эво-

люционируемость знаний. Более того, в предметных областях, где важны влияние факторов и событий на состояние системы (объекта, ситуации), их изменение во времени, влияние индивидуальных характеристик системы и одних ее процессов на другие, развиваемость/эволюционируемость баз знаний — главный "вызов" современных "условий" по отношению к СОЗ.

Развиваемость баз знаний (БЗ) позволяет надеяться на получение "эталонной" базы знаний $k^* \in K(X, Y)$, от актуальности (соответствия современному представлению) и качества которой зависит успех применения СОЗ (для получения объяснения $y \in Y$ при любых $x \in X$).

4. Модель жизнеспособности систем, основанных на декларативных знаниях

Актуальность знаний (базы знаний) достигается тремя основными способами [3, 7, 18]: адаптируемостью (способностью к интерактивному изменению базы знаний), адаптивностью с использованием методов машинного обучения СОЗ (могут применяться средства индуктивного формирования знаний по отобранным прецедентам, средства выявления знаний из "больших данных"), либо их комбинацией.

"Успешность" адаптируемости (интерактивного изменения) зависит от онтологии предметной области, которая является системой понятий, удовлетворяющей всех участников разработки, и от ПИФ редактора БЗ, который должен соответствовать требованиям и ожиданиям экспертов предметной области. При этом онтология — структурная основа как инструментов для экспертов (средств редактирования), так и программных компонентов СОЗ (решателей интеллектуальных задач и ПИФ).

Таким образом, традиционная архитектура СОЗ — "база знаний + база фактов + решатель интеллектуальной задачи + ПИФ интеллектуальный" [14] — расширяется новым компонентом: "онтология + база знаний + база фактов + + онтолого-ориентированный решатель задачи + ПИФ интеллектуальный".

Архитектурные свойства систем, основанных на декларативных знаниях. В онтологии предметной области определены типы утверждений (позволяющих решать задачи в предметной области) и ограничения на интерпретацию смысла понятий (терминов). Например, для диагностики процессов одним из самых распространенных типов предложений (утверждений) является <класс отклонений-*k*, признак-*j*, диапазон значений-*kj* признака-*j*>; предложения типа "необходимое условие существования процесса", "вариант процесса изменения значений признака" и т. д.).

БЗ содержит предметные знания, которые представляются в явном виде. Они формируются вручную или индуктивно, в том числе по обучающим выборкам из архивов и баз данных (индуктивное обобщение в машинном обучении [14], байесовские классификаторы, алгоритмы кластеризации и обучение с подкреплением [19, 20]).

База фактов содержит набор фактов (утверждений), наблюдаемых или объективно измеренных в рассматриваемой ситуации, относительно которой решается задача.

Явное представление (и группирование) в онтологии всех структурных типов утверждений (и отделение онтологии от базы знаний) диктует необходимость "замены" решателей-рассуждателей, интерпретирующих продукционные правила, на специализированные алгоритмы на основе онтологий.

Онтолого-ориентированный алгоритм (ontological reasoner) выполняет для поиска либо опровержения гипотез обход (декларативной) базы знаний. Такой онтолого-ориентированный решатель задач "перебирает" утверждения БЗ каждого типа, относящегося к гипотезе. Он сопоставляет эти предложения входной информации (об объекте). Алгоритм обхода БЗ и сопоставления ее предложений элементам структуры объектов предметной области в общем случае несложен, так как число "правил сопоставления" определяется типами предложений БЗ, соответствующих решаемой задаче.

Реализация "обхода" декларативной БЗ в основном состоит в поиске значений элементов "оперативной" информации о ситуации в предметной области, относительно которой требуется принять решение, сравнения их с областями (возможных) значений соответствующих понятий (терминов) БЗ в целях выдвижения гипотез на основе декларативно представленных отношений между гипотезами и этими понятиями (терминами) в БЗ. (Пример: реализация "обхода" декларативной БЗ диагностики обычно состоит в поиске значений признаков у диагностируемого объекта и сравнения их с областями (ожидаемых) значений с целью отклонить или подтвердить соот-

ветствующие гипотезы о классах отклонений). Таким образом "структурная" сложность онтолого-ориентированного алгоритма, как правило, невелика. Его сложность определяется:

- числом и сложностью типов аксиом (предложений) онтологии;
- длиной цепочек причинно-следственных отношений (ПСО) между искомыми/проверяемыми гипотезами и наблюдениями — элементами описаний объектов предметной области (ПрОбл);
- числом наблюдений (элементов описаний) объектов и/или числом ограничений/целевых условий. Например, в задаче проектирования условия будут касаться размеров и разных свойств конструируемого объекта (моста, аппаратного комплекса...); в задаче лечения ограничение = "объект должен стать здоров"). На продолжительность обработки (вычисления) влияет объем БЗ.

Такие решатели способны стать повторно-используемыми, поскольку, во-первых, набор традиционно решаемых задач (диагностики, прогноза, управления, планирования....) известен [21, 22], во-вторых, традиционный набор типов связей в предметных областях с ПСО (для решения задач диагностики, прогноза, управления состоянием объекта) [21, 22] ограничен. Повторное использование состоит в выборе задачи в соответствии с классификацией задач и их уточнениями в соответствии с особенностями предметной области (дополнительными типами отношений в онтологии), поиске решателя задачи и, возможно, адаптации его к формату результата работы (объяснению).

Использование онтологии для формирования БЗ, данных (фактов), а также структуры объяснения дает возможность генерировать ПИФ в терминах (понятиях), понятных специалисту, а также, как правило, генерировать сценарий диалога, соответствующий структуре объяснения результатов.

СОЗ как разновидность ПС может также:

- включать в себя базы данных для хранения справочной или другой информации;
- принимать на вход файлы или базы данных с оперативной информацией (о ситуации, относительно которой требуется принять решение);
- создавать в качестве результата файлы или базы данных с результатами работы и объяснением принятого решения.

Таким образом, в составе жизнеспособной СОЗ должны быть:

- база знаний, выраженная в терминах онтологии;
- онтолого-базированный решатель (умеющий выдвигать и объяснять гипотезы);
- онтолого-зависимый адаптивный и адаптируемый ПИФ.

Наиболее характерные свойства СОЗ, относящиеся к жизнеспособности, таковы:

- регулярная обновляемость знаний;
- допустимость усовершенствования метода принятия решения;
- допустимость изменения или добавления функций (например, формирования дополнительных результатов);
- адаптивность ПИФ пользователя в связи с изменением входных данных и формируемых результатов функций;
- допустимость расширения онтологии (добавления понятий, отношений).

Изменения в онтологии, как правило, ведут к корректировке всех компонентов СОЗ; в этом случае целесообразно говорить о новой версии СОЗ.

Онтологический подход к средствам сопровождения и развития СОЗ. Онтология чаще соответствует классу решаемых задач или одному классу задач в одной предметной области. Для конструирования прикладных СОЗ может потребоваться не одна онтология, а их взаимосвязанная совокупность, в том числе для нескольких смежных классов задач (например, диагностика неполадок и исправление неполадок).

Инструментарий для конструирования прикладных систем, обрабатывающих декларативные знания, основан на онтологии предметной области, поскольку алгоритмы обрабатывают знания и другую информацию предметной области. Обрабатываемая информация — формализованные знания о законах предметной области, вводимые через ПИФ факты, хранимые в базах данных архивные данные и документы. Онтологические соглашения о правилах сопоставления фактов знаниям должны быть известны разработчикам алгоритмов решения задач. Знания и другая информация должны формироваться в единых понятиях предметной области (составляющих ее терминологический справочник) и в соответствии со структурой, определяемой онтологией ПрОбл, включающей онтологии знаний, входных данных и ожидаемых результатов.

Рассмотрим инструментарий, позволяющий создавать компоненты СОЗ на базе онтологии ПрОбл, специализированной среды развития СОЗ.

Специализированная среда развития СОЗ должна, как минимум, предоставлять:

- редактор БЗ;
- редактор БД;
- редактор структуры результатов решения и объяснения;
- библиотеку программных решателей, соответствующих классам решаемых задач (и их компонентов — программных единиц);
- средства поиска и выбора повторно используемых компонентов (ПИК) — БЗ и программных решателей;
- средства интеграции БЗ и решателей;
- специализированный редактор для ПИФ.

Примечание. В ситуации, когда не исключается возможность внесения изменений в онтологию ПрОбл, после построения БЗ среда развития СОЗ должна предоставлять инструмент редактирования онтологии. Внесение дополнительных понятий или новых связей между понятиями в онтологию не нарушает работоспособности программных решателей. (Онтолого-ориентированный решатель обладает таким свойством, поскольку обрабатывает в БЗ только известные ему типы предложений.)

Для формирования программных решателей понадобятся:

- средства кодирования новых программных единиц (ПрЕд) для решателя или их новых версий;
- средства каталогизации ПрЕд как ПИК;
- средства интеграции ПИК и новых ПрЕд в новые решатели или их новые версии.

Примечание. Возможность создания программных единиц для решателя может оказаться излишней (в среде развития), если готовый решатель вырабатывает правильные гипотезы в соответствии с постановкой задачи, а "наращивание" дополнительной функциональности не предполагается.

Ввиду важности развиваемости и эволюционируемости БЗ следует всерьез рассматривать только такие СОЗ, которые интегрированы с СУБЗ, т. е. программные компоненты для развития баз знаний, например, средства проверки баз знаний, оценивания их качества на эталонных архивах решенных задач — часть "интегрированной архитектуры". Тогда среда развития может предоставлять:

- средства проверки баз знаний;
- средства оценивания качества баз знаний на эталонных архивах решенных задач;
- средства индуктивного формирования баз знаний или фрагментов БЗ, интегрируемых с ней.

Вышеперечисленным свойствам жизнеспособной СОЗ соответствуют характерные свойства специализированной среды ее развития:

- поддержка обновления знаний;
- поддержка изменения компоновки онтолого-ориентированного решателя (в связи с заменой компонента, реализующего другую стратегию принятия решения или метода получения результата или в связи с добавлением компонента, реализующего дополнительную функцию, например, формирования дополнительного результата);
- поддержка усовершенствования ПИФ пользователя в связи с изменением функций;
- поддержка изменения в онтологии;
- поддержка усовершенствования ПИФ эксперта (и ПИФ пользователя) в связи с обновлением онтологии;
- поддержка кодирования новых версий программных единиц онтолого-ориентированного решателя или поддержка изменения кода ПрЕд онтолого-ориентированного решателя (т. е. адаптации обхода декларативной БЗ в связи с обновлением онтологии знаний).

Пример реализации сред развития СОЗ (в рамках онтологического подхода). Онтология чаще соответствует классу решаемых задач или одному классу задач в одной предметной области. Для конструирования прикладных СОЗ может потребоваться не одна онтология, а их взаимосвязанная совокупность, в том числе для нескольких смежных классов задач (например, диагностика неполадок и исправление неполадок).

Примером инструментария для формирования сред развития (для производства СОЗ) является реализованный на платформе IACPaaS комплекс инструментов для обработки информационных ресурсов, формируемых по явно представляемой онтологии (на платформе используется термин "метаинформация") [23]. Онтология задает правила порождения информации, правила и ограничения ее интерпретации. Структуру порождаемой информации и ряд ограничений интерпретации определяет пользователь платформы IACPaaS для

формирования собственных информационных ресурсов (когнитолог — для сообщества экспертов, специалистов, пользователей).

СОЗ является частным случаем прикладных программных сервисов на платформе IACPaaS (сервисов IACPaaS).

Комплекс инструментов для формирования информационных компонентов СОЗ таков:

- IACPaaS-Редактор онтологии;
- IACPaaS-Редактор БЗ, генерирующийся по онтологии с самоадаптирующимся ПИФ (при изменении онтологии);
- IACPaaS-Редактор БД (с самоадаптирующимся ПИФ).

Этот комплекс инструментов также содержит средства для управления программными компонентами СОЗ:

- "Мастер" формирования декларативной части ПрЕд (IACPaaS-агентов);
- конструктор решателя из корневого IACPaaS-агента и новых агентов (их версий), представляемого только декларативной частью;
- генератор заготовок программного кода для новых IACPaaS-агентов (новых версий);
- загрузчик байт-кода в декларативную часть IACPaaS-агента;
- средства тестирования ПрЕд и подготовки их для библиотеки ПИК (агентов).

IACPaaS-агенты кодируются на языке Java, для написания исходных кодов и создания байт-кода агента рекомендована среда разработки (IDE) Java Development Kit (JDK) для версии Java SE 8.

Среда развития СОЗ подразумевает кодирование новых версий агентов (когда надо учесть использование новых терминов или новых связей терминов) или кодирование новых IACPaaS-агентов (когда надо внести изменение в процесс принятия решения). Она может быть локальной для программиста (или размещенной на платформе IACPaaS).

Примечание. На платформе IACPaaS "Мастер" формирования декларативной части ПрЕд и Конструктор уникального решателя являются редакторами, генерируемыми по соответствующей метаинформации — шаблонам декларирования ПрЕд и решателя (из онтологии технологии IACPaaS). Самоадаптирующийся ПИФ для формирования данных (и знаний) и просмотра результата работы СОЗ предоставляет три способа визуализации, один из которых является графовым.

Таким образом, построение СОЗ с помощью онтолого-ориентированных сред развития принципиально увеличивает их жизнеспособность, существенно повышая роль и долю средств управления (для внесения изменений в декларативные компоненты) по отношению к средствам сопровождения (для внесения изменений в исходный код).

5. Обеспечение жизнеспособности СОЗ разными инструментами сред развития IACPaaS

Вышеописанные характерные свойства СОЗ, относящиеся к жизнеспособности, обеспечиваются в средах развития, реализуемых на платформе IACPaaS. Рассмотрим это на примере свойства *Обновляемость знаний и поддержка обновления знаний.*

Если требуется обновление базы знаний в связи с получением новых знаний (утверждений), естественно модифицировать их вручную (и оценить согласованность с имеющимися фактами). Для этого процесса (в среде развития) требуются редакторы для всех БЗ.

На платформе IACPaaS редактор ИнфоРесурсов (штатный) всегда является частью среды развития. В настоящее время разрабатывается средство проверки качества новой версии БЗ — проверки неухудшения правильности решения множества эталонных задач при замене версии БЗ (согласно важности монотонного усовершенствования баз знаний [24]). Способ проверки неухудшения правильности знаний: подача на вход решателю, интегрированному с новой версией БЗ, входных условий эталонной задачи и сравнение полученного объяснения с выходным результатом эталонной задачи.

Если требуется обновление БЗ в связи с получением "фактов" (прецедентов, решений), не согласованных со знаниями (когда прецедент содержит правильный результат решения задачи, который не соответствует результату, получаемому с помощью СОЗ), то эффективно автоматически (индуктивными методами) формировать новую версию БЗ (и оценить согласованность работы СОЗ по обновленной БЗ с имеющимися прецедентами). (Пример такого обновления знаний: из медучреждения через определенный промежуток времени поступает правильный результат диагностики или лечения, он сравнивается с результатом (объяснением гипотез) от СОЗ: не противоречат ли они друг другу.)

Для реализации процесса монотонного усовершенствования БЗ (на основе методов индуктивного формирования фрагментов БЗ [24]) требуются:

- средства индуктивного формирования знаний для каждой решаемой интеллектуальной задачи (с известной постановкой: диагностика, планирование, прогноз...);
- средство поддержки выбора прецедентов (решенных правильно задач в одной постановке);
- средство проверки правильности (качества) новой версии БЗ (тот же — вышеописанный).

Способ проверки наличия (у СОЗ) свойства "Обновляемость знаний" — проверка наличия и работоспособности вышеперечисленного инструментария.

На платформе IACPaaS штатный (самоадаптирующийся) редактор всегда есть, остальные инструменты добавляются в среду развития по мере готовности.

Заключение

Предложенная модель жизнеспособности СОЗ основана на построении такой системы с помощью онтолого-ориентированной среды развития, где каждый компонент СОЗ формируется в терминах (понятиях) единой явно представленной онтологии ПрОбл с использованием инструментальной поддержки.

Онтолого-ориентированная среда развития реализует механизмы адаптивности таких систем. Она развивается и эволюционирует, когда создаваемые в ней прикладные СОЗ нуждаются в адаптации к условиям функционирования и уточнению знаний ПрОбл, а также когда "базовый инструментарий" нуждается в адаптации к новой аппаратуре и системному ПО, к новым видам человеко-машинных интерфейсов. Эти механизмы адаптивности обеспечивают возможность усовершенствования каждого компонента силами узких специалистов, гарантируя целостность всей СОЗ.

Список литературы

1. **Pressman R. S.** Software Engineering: Practitioner's Approach. 7th ed. McGraw-Hill, 2010. 930 p.
2. **Dehaghani S. M. H., Hajrahimi N.** Which factors affect software projects maintenance cost more? // Acta Informatica Medica. 2013. N. 21(1). P. 63—66.
3. **Додонов А. Г., Ландэ Д. В.** Живучесть информационных систем. Киев: Наукова Думка, 2011. 256 с.
4. **Черников Б. В.** Управление качеством программного обеспечения. М.: ИД "ФОРУМ": ИНФРА-М, 2012. 240 с.
5. **Musen M.** The Protégé Project: A Look Back and a Look Forward // AI Matters. 2015 Jun; N. 1(4). P. 4—12.
6. **Mortensen J., Minty E., Januszyk M., Sweeney T., Rector A., Noy N., Musen M. A.** Using the wisdom of the crowds to find critical errors in biomedical ontologies: A study of SNOMED CT // Journal of the American Medical Informatics Association. 2015. N. 22(3). P. 640—648.
7. **Кряжич О. А.** Обеспечение жизнеспособности информации во времени при ее обработке в СППР // Математичні машини і системи. 2015. № 2. С. 170—176.
8. **Izurieta C., Bieman J. M.** A multiple case study of design pattern decay, grime, and rot in evolving software systems // Software Quality Journal. 2013. Vol. 21, N. 2. P. 289—323.
9. **Breivold H. P., Crnkovic I., Eriksson P. J.** Analyzing software evolvability // Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International. IEEE. 2008. P. 327—330.
10. **Брауде Э.** Технология разработки программного обеспечения. СПб.: Питер, 2004. 655 с.
11. **Гласс Р., Нуазо Р.** Сопровождение программного обеспечения. М.: Мир, 1983. 158 с.
12. **Islam M., Katiyar V.** Development of a software maintenance cost estimation model: 4th GL perspective // International Journal of Technical Research and Applications. 2014. Vol. 2, Iss. 6. P. 65—68.
13. **Грибова В. В., Клещев А. С.** Какой должна быть парадигма решения задач на основе баз знаний? // Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2014): Матер. IV Междунар. науч.-техн. конф. (Минск, 20—22 февраля 2014 года) / редкол. В. В. Голенков (отв. ред.) [и др.]. Минск: БГУИР, 2014. С. 131—136.
14. **Финн В. К.** Об интеллектуальном анализе данных // Новости искусственного интеллекта. 2004. № 3. С. 3—18.
15. **Гаврилова Т. А., Кудрявцев Д. В., Муромцев Д. И.** Инженерия знаний. Модели и методы: учебник. СПб.: Лань, 2016. 324 с.
16. **Рыбина Г. В., Блохин Ю. М., Иващенко М. Г.** Интеллектуальная технология построения интегрированных экспертных систем // Искусственный интеллект и принятие решений. 2011. № 3. С. 48—57.
17. **Рыбина Г. В.** Обучающие интегрированные экспертные системы: некоторые итоги и перспективы // Искусственный интеллект и принятие решений. 2008. № 1. С. 22—46.
18. **Грибова В. В., Клещев А. С.** Парадигма разработки управляемых интеллектуальных систем // Системы управления и информационные технологии. 2016. № 3(65). С. 32—38.
19. **Николенко С. И., Тулупьев А. Л.** Самообучающиеся системы. М.: Изд-во МЦНМО, 2009. 288 с.
20. **Golenkov V. V., Gulyakina N. A., Grakova N. V., Nikulenka V. Y., Eremeev A. P., Tarasov V. B.** From training intelligent systems to training their development means // Proceedings of the International Conference OSTIS-2018. Minsk. 2018. P. 81—98.
21. **Clancey W. J.** Heuristic Classification // Artificial Intelligence. 1985. N.27. P. 289—350.
22. **Клещев А. С., Шалфеева Е. А.** Онтология задач интеллектуальной деятельности // Онтология проектирования. 2015. Т. 5, № 2(16). С. 179—205.
23. **Грибова В. В., Клещев А. С., Москаленко Ф. М., Тимченко В. А., Шалфеева Е. А.** Расширяемый инструментарий для создания жизнеспособных систем с базами знаний // Программная инженерия. 2018. Т. 9, № 8. С. 339—348.
24. **Клещев А. С., Черняховская М. Ю., Шалфеева Е. А.** Парадигма автоматизации интеллектуальной профессиональной деятельности. Часть 1. Особенности интеллектуальной профессиональной деятельности // Онтология проектирования. 2013. № 3(9). С. 53—69.

V. V. Gribova, Research Deputy Director, e-mail: gribova@iacp.dvo.ru,
E. A. Shalfeeva, Senior scientific employee,, e-mail: shalf@dvo.ru,
Institute for Automation & Control Processes, FEBRUS, Russian Federation

Ensuring of Viability of Systems Based on Knowledge

The model of viability of knowledge based systems designed by means of the ontology-oriented development environment is considered in the article. In the conditions of high labor costs on software maintenance the maintenance problem for knowledge-based systems is particularly acute. Their architecture includes the knowledge base created by domain experts and cognitologists. It imposes need of use of additional specialized mechanisms of providing maintenance and viability. The generalized architectural model of an accompanied knowledge-based system and set of means of ensuring of her viability is presented in the paper. It is shown that a viable system should include: a knowledge base formalized in terms of ontology, an ontology-based solver and ontology-dependent adaptable GUI. The structure of instrumental specialized environment of development for creation components of any knowledge-based system is defined. Such environment of development is ontology-oriented. It evolves together with enhancement of knowledge, expansion of ontology and in process of adaptation to operating conditions. These mechanisms of adaptability provide a possibility of improvement of each component by narrow experts, guaranteeing integrity of all system. It is shown how different aspects of viability are being provided in the environments of development implemented on the IACPaaS platform.

Keywords: *intellectual systems, knowledge base, knowledge base system, ontology-oriented algorithm, viability, declarative tools, development environment*

DOI: 10.17587/it.25.738-746

References

1. **Pressman R. S.** Software Engineering: Practitioner's Approach, McGraw-Hill, 2010, 930 p.
2. **Dehaghani S. M. H., Hajrahimi N.** *Acta Informatica Medica*, 2013, no. 21(1), pp. 63—66.
3. **Dodonov A. G., Landje D. V.** Viability of information systems, Kiev, Naukova dumkab, 2011, 256 p. (in Russian).
4. **Chernikov B. V.** Software quality management, Moscow, ID "FORUM": INFRA-M, 2012, 240 p. (in Russian).
5. **Musen M.** *AI Matters*, 2015 Jun, no. 1(4), pp. 4—12.
6. **Mortensen J., Minty E., Januszyk M., Sweeney T., Rector A., Noy N., Musen M. A.** *Journal of the American Medical Informatics Association*, 2015, no. 22(3), pp. 640—648.
7. **Krjazhich O. A**. *Matematichni mashini i sistemi*, 2015, no. 2, pp. 170—176 (in Russian).
8. **Izurieta C., Bieman J. M.** *Software Quality Journal*, 2013, vol. 21, no. 2, pp. 289—323.
9. **Breivold H. P., Crnkovic I., Eriksson P. J.** *Computer Software and Applications*, 2008, COMPSAC'08, 32nd Annual IEEE International, pp. 327—330.
10. **Braude E. J., Bernstein M. E.** Software Engineering: Modern approaches (2nd edition), John Wiley & Sons, Inc., 2010, 800 p.
11. **Glass R. L. Noiseux R.** Software Maintenance Guidebook, Prentice Hall, 1981, 193 p. (in Russian).
12. **Islam M., Katiyar V.** *International Journal of Technical Research and Applications*, 2014, vol. 2, iss. 6, pp. 65—68.
13. **Gribova V. V., Kleshhev A. S.** *Open SemanticTechnologies for Intelligent Systems (OSTIS-2014): materialy IV mezhdunar. nauch.-tehn. konf.* (Minsk, 20—22 fevralja 2014 goda), Minsk, Publishing house of BGUIR, 2014, pp. 131—136 (in Russian).
14. **Finn V. K.** *Novosti Iskusstvennogo intellekta*, 2004, no. 3, pp. 3—18 (in Russian).
15. **Gavrilova T. A., Kudrjavcev D. V., Muromcev D. I.** Engineering of knowledge. Models and methods: Textbook, SPb., Lan', 2016, 324 p. (in Russian).
16. **Rybina G. V., Blohin Ju. M., Ivashhenko M. G.** *Iskusstvennyj Intellekt i Prinjatie Reshenij*, 2011, no. 3, pp. 48—57 (in Russian).
17. **Rybina G. V.** *Iskusstvennyj Intellekt i Prinjatie Reshenij*, 2008, no. 1, pp. 22—46 (in Russian).
18. **Gribova V. V., Kleshhev A. S.** *Sistemy Upravlenija i Informacionnye Tehnologii.* 2016, no. 3(65), pp. 32—38 (in Russian).
19. **Nikolenko S. I., Tulup'ev A. L.** Samoobuchajushhiesja sistemy, Moscow, Publishing house of MCNMO, 2009, 288 p. (in Russian).
20. **Golenkov V. V., Gulyakina N. A., Grakova N. V., Nikulenka V. Y., Eremeev A. P., Tarasov V. B.** *Proceedings of the International Conference OSTIS-2018*, Minsk, 2018, pp. 81—98.
21. **Clancey W. J.** *Artificial Intelligence*, 1985, no. 27, pp. 289—350.
22. **Kleshhev A. S., Shalfeeva E. A.** *Ontologija proektirovanija*, 2015, vol.5, no. 2(16), pp. 179—205 (in Russian).
23. **Gribova V., Kleschev A., Moskalenko Ph., Timchenko V., Shalfeeva E. A.** *Programmnaja Inzhenerija*, 2018, vol. 9, no. 8, pp. 339—348 (in Russian).
24. **Kleschev A. S., Chernyakhovskaya M. Y., Shalfeeva E. A.** *Ontologija proektirovanija,* 2013, no. 3(9), pp. 53—69 (in Russian).