УДК 004.89

Н. О. Дородных, аспирант, e-mail: tualatin32@mail.ru,
А. Ю. Юрин, канд. техн. наук, зав. лаб., e-mail: iskander@icc.ru,
Институт динамики систем и теории управления им. В. М. Матросова СО РАН (ИДСТУ СО РАН), г. Иркутск

Использование диаграмм классов UML для формирования продукционных баз знаний[1]

Дан анализ возможностей использования концептуальных моделей для автоматизированного формирования продукционных баз знаний. В качестве основных источников концептуальных моделей рассмотрены диаграммы классов UML как наиболее распространенный способ концептуализации знаний в процессе проектирования программного обеспечения в соответствии с объектным подходом. Описаны алгоритмы формирования элементов баз знаний CLIPS на основе анализа описания моделей, представленных в формате XML.

Ключевые слова: *база знаний, продукции, автоматизированное создание баз знаний, приобретение знаний, концептуальная модель, UML, CLIPS*

Введение

На настоящее время разработка новых методов и подходов к созданию прикладных программных систем искусственного интеллекта и, в частности, экспертных систем (ЭС) остается перспективной областью научных исследований. Ядром ЭС и систем, основанных на знаниях, является база знаний (БЗ), которая содержит как общие знания, так и информацию о частных случаях [1]. При этом сложность и трудоемкость процесса разработки ЭС обусловлены главным образом особенностью этапа разработки БЗ, который включает задачи идентификации (получения), концептуализации (структурирования) и формализации (представления) знаний, что считается традиционно "узким местом" проектирования ЭС [2].

Повышение эффективности получения (приобретения) знаний (*knowledge acquisition*) [2] из различных источников (баз данных, документов, концептуальных моделей и т. д.) является актуальной задачей. При этом особый интерес представляет повторное использование (в том числе и трансформация) моделей, построенных с использованием различных программных средств онтологического, когнитивного моделирования, CASE-средств (например, IBM Rational Rose и др.) [3]. Однако существующие в данной области решения имеют ряд недостатков, в частности, отсутствие возможности совместной распределенной и одновременной работы пользователей; отсутствие или ограниченность кодогенерации на языках представления знаний (ЯПЗ).

Целью работы, результаты которой представлены далее, являлось повышение эффективности процесса разработки продукционных БЗ путем автоматизированного анализа концептуальных моделей предметных областей, выполненных при помощи различных программных средств. Для достижения поставленной цели была осуществлена разработка алгоритмического и программного обеспечения [4], предназначенного для преобразования концептуальных моделей в онтологию предметной области (расширенную возможностью определения причинно-следственных отношений), а также для моделирования (модификации) продукций с использованием специальной нотации RVML (*Rule Visual Modeling Language*) [5] и их отображения (трансляции) в ЯПЗ CLIPS (*C Language Integrated Production System*) [6]. Использование онтологии с причинно-следственными отношениями позволяет унифицировать представление продукций, что обеспечивает возможность расширения набора доступных (поддерживаемых) в программном средстве ЯПЗ. Применение web-технологий в свою очередь обеспечивает возможность совместной и распределенной работы пользователей (экспертов, инженеров по знаниям и системных аналитиков) над проектами БЗ.

В качестве источников концептуальных моделей предлагается использовать диаграммы классов, построенные с использованием унифицированного языка моделирования UML (*Unified Modeling Language*) [7] и сохраненные в формате XML в соответствии со стандартом XMI (*XML Metadata Interchange*) [8].

[1] Работа выполнена при частичной поддержке гранта РФФИ 15-07-05641.

1. Постановка задачи

Анализ проблематики получения, структурирования и формализации знаний показал [1, 2], что в настоящее время наибольший интерес для исследователей представляют автоматизированные методы получения знаний. Это обстоятельство обусловлено главным образом наличием больших объемов накопленной информации, в том числе и в форме концептуальных моделей. При этом можно выделить основные группы программных средств, которые обеспечивают построение концептуальных моделей (рис. 1).

Результаты анализа данных систем моделирования позволяют выделить следующие их недостатки:

- практически отсутствуют системы (web-сервисы), обеспечивающие совместную, распределенную работу пользователей в сети Интернет;
- обеспечивая создание концептуальных моделей, а также генерацию различных отчетных документов в разных форматах, исследуемые системы либо не предусматривают возможность преобразования построенных моделей в структуры ЯПЗ, либо эта возможность ограничена (неполное преобразование или единственный ЯПЗ), что в свою очередь затрудняет возможности практического использования построенных моделей при разработке экспертных систем.

Существуют примеры успешного преодоления данных недостатков по отдельности, в частности, для обеспечения совместной распределенной работы над проектами возможно использование программных средств контроля конфигурации и целостности проекта, однако инструментальных средств, сочетающих данные возможности с функциями генерации программного кода баз, авторами статьи не обнаружено.

В целях преодоления отмеченных недостатков предлагается разработать алгоритмическое и программное обеспечение, которое не только позволяет генерировать БЗ на определенном языке программирования с использованием концептуальных моделей и технологии визуального моделирования, но и поддерживает возможность совместной и распределенной работы пользователей. Для апробации результатов такой разработки в качестве языка программирования выбран CLIPS.

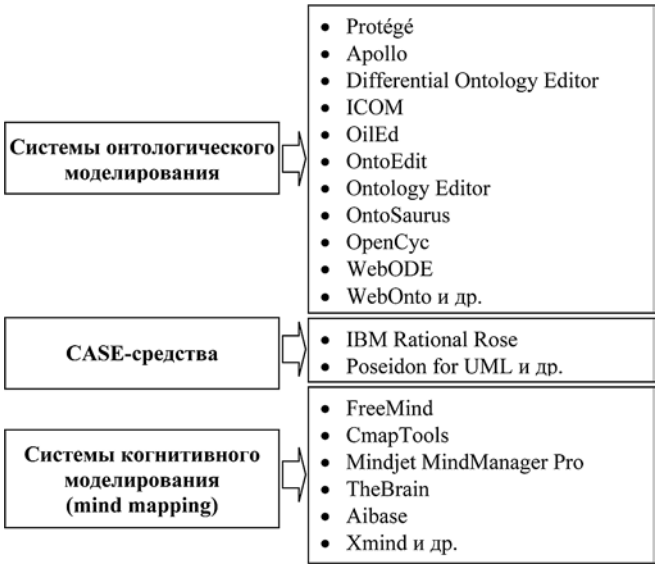

Рис. 1. Основные группы программных систем, обеспечивающих построение концептуальных моделей

Формализовать постановку задачи можно следующим образом. Необходимо определить оператор преобразования концептуальной модели T:

$$T: M \rightarrow Code^{CLIPS}, \quad (1)$$

где M — концептуальная модель; $Code^{CLIPS}$ — программный код на языке CLIPS.

2. Источники концептуальных моделей

Целью анализа программных систем, обеспечивающих построение концептуальных моделей, являлось в том числе выявление стандартов и языков, которые используются как при создании программного обеспечения (ПО), так и в процессе целенаправленного моделирования предметной области и построения ее онтологии (рис. 2).

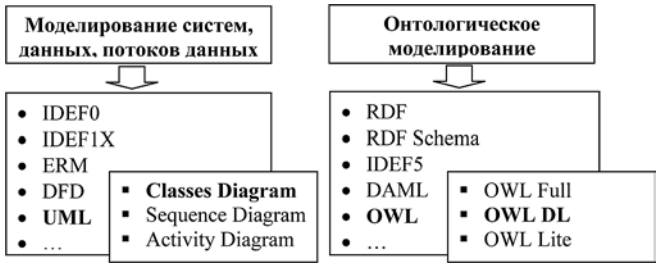

Рис. 2. Стандарты (нотации и языки) представления концептуальных моделей

На основе анализа был сделан вывод, что при моделировании программных систем (в процессе проектирования ПО) наиболее широко распространен язык UML [7]. Из языков описания онтологий особый интерес представляют языки, основанные на web-стандартах, в частности, OWL [9]. Поэтому в качестве основных источников концептуальных моделей могут быть рассмотрены диаграммы классов UML и онтологии OWL DL.

Далее рассмотрим подробнее использование диаграмм классов UML при автоматизированном формировании баз знаний и расширим выражение (1), доопределив концептуальную модель:

$$M = \left\langle M^{UML} \right\rangle. \quad (2)$$

3. Алгоритмическое обеспечение

Основной задачей алгоритмического обеспечения является преобразование концептуальных моделей в конструкции (элементы) ЯПЗ CLIPS, что может

Рис. 3. Процесс формирования кода БЗ на основе концептуальных моделей

быть представлено в виде последовательности действий (рис. 3). На этапах 1 и 2 такого преобразования средствами внешних программ пользователь строит концептуальную модель, которая представляется в формате XML. Этот формат является универсальным и наиболее распространенным способом интеграции программных систем и обеспечения обмена информацией между приложениями.

При представлении UML-моделей в виде XML-структур используется стандарт XMI (*XML Metadata Interchange*) [8], предназначенный главным образом для хранения UML-структур, а также любых других данных, метамодель которых задана с помощью MOF (*Meta Object Facility*), и обмена ими между различными инструментальными средствами разработки.

На этапе 3 процесса анализа UML-модели выделяются понятия предметной области и их отношения, далее (этап 4) на их основе формируется онтология, как универсальное представление продукций, независящее от используемого ЯПЗ. При помощи специальной графической нотации RVML [5] предоставляется возможность модификации, визуализации и проверки полученных продукций (этап 5). На этапе 6 происходит генерация кода базы знаний в формате CLIPS на основе онтологии.

Таким образом, уточним оператор преобразования концептуальной модели (1):

$$T = \langle T_{CM-XML}, T_{XML-ONT}, T_{ONT-Code} \rangle,$$

$$T_{CM-XML} : M_{CM} \to M_{XML},\ T_{XML-ONT} : M_{XML} \to \to M_{ONT},\ T_{ONT-Code} : M_{ONT} \to Code_{CLIPS}, \quad (3)$$

где T_{CM-XML} — оператор преобразования концептуальной модели в XML-структуру; $T_{XML-ONT}$ — оператор преобразования XML-структуры в онтологию; $T_{ONT-Code}$ — оператор преобразования онтологии в код на ЯПЗ CLIPS; M_{XML} — представление концептуальной модели в XML-формате; M_{ONT} — представление концептуальной модели в онтологии.

Вопросы, связанные с исследованием свойств предложенного алгоритмического обеспечения, в данной статье не рассмотрены. Отметим только конечный характер алгоритмов, что определяется конечным числом строк анализируемых файлов и обрабатываемых (извлеченных и преобразованных) элементов.

3.1. Описание концептуальной модели

Используя формулы (2) и (3), доопределим M_{XML} из выражения (3):

$$M_{XML} = M_{XML}^{UML}, \quad (4)$$

где M_{XML}^{UML} — модель UML (диаграмма классов) в формате XML.

Используя формулу (4), формализуем описание модели M_{XML}^{UML}:

$$M_{XML}^{UML} = \langle C^{UML}, DT^{UML}, R_A^{UML} \rangle,$$

где C^{UML} — описание классов; DT^{UML} — типы данных; R_A^{UML} — отношения между классами типа "ассоциация".

Уточним описание классов и отношений следующим образом:

- $C^{UML} = \{c_1^{UML} \dots c_m^{UML}\}, c_i^{UML} = \langle name_i, A_i^{UML} \rangle, i = \overline{1,m},$

где $name_i$ — имя i-го класса; A_i^{UML} — атрибуты i-го класса; m — число классов; в свою очередь, $A^{UML} = \{a_1^{UML} \dots a_k^{UML}\}$, где $a_j^{UML} = \langle name_j, type_j, value_j \rangle$, $j \in \overline{1,k}$, где $name_j$ — имя j-го атрибута; $type_j$ — тип j-го атрибута; $value_j$ — возможное значение, при этом $type_j \in DT^{UML}$; k — число атрибутов;

- $R_A^{UML} = \{r_{A_1}^{UML} \dots r_{A_n}^{UML}\}, r_l^{UML} = \langle name_l, r_{A-LHS_l}^{UML}, r_{A-RHS_l}^{UML} \rangle,$

$l \in \overline{1,n}$, где $name_l$ — имя отношения; $r_{A-LHS_l}^{UML} = \langle name_r, m_r, c^{UML} \rangle$ — левая часть связи; $name_r$ — наименование роли класса на определенной части связи, m_r — кратность (множественность) ассоциации; c^{UML} — ссылка на класс; $r_{A-RHS_l}^{UML} = \langle name_r, m_r, c^{UML} \rangle$ — правая часть связи, при этом $m_r \in M_r^{UML} = \{0; 0..1; 0..*; 1; 1..*\}$, n — число отношений.

3.2. Модель онтологии

Для унифицированного хранения и представления знаний разработана специальная модель онтологии. Разработанная модель позволяет абстрагироваться от особенностей описания продукций в различных языках программирования, используемых при реализации БЗ (например, CLIPS, JESS), и позволяет хранить знания в собственном независимом формате.

Формализуем описание модели онтологии, используя модель, предложенную в работе [10]:

$$M_{ONT} = \langle DT_{ONT}, CN_{ONT}, PN_{ONT}, Obj_{ONT}, R_{ONT}, E_{KRL} \rangle,$$

где DT_{ONT} — перечень базовых типов данных, при этом DT_{ONT} = {литерал, объект, коллекция}; CN_{ONT} — имена классов; PN_{ONT} — имена свойств классов; Obj_{ONT} — понятия (константы, объекты) предметной области; R_{ONT} — конечное множество отношений между концептами заданной предметной области; E_{KRL} — элементы (конструкции) языка программирования, соответствующие элементам модели онтологии, $E_{KRL} = \langle t_{KRL}, f_{KRL}, r_{KRL}, s_{KRL} \rangle$, где t_{KRL} — шаблон; f_{KRL} — факт; r_{KRL} — правило; s_{KRL} — слот (свойство), в данной работе $KRL = CLIPS$.

Уточним введенные понятия онтологии и расширим определение из работы [10] путем введения причинно-следственного отношения:

$R_{ONT} = \{R^{IS-A}_{ONT}, R^{P}_{ONT}, R^{C}_{ONT}\}$, где R^{IS-A}_{ONT} — отношение наследования между классами CN_{ONT}; R^{P}_{ONT} — отношение между классами и свойствами, pnR^{P}_{ONT} $\langle cn, cn\_bt \rangle$, где $pn \in PN_{ONT}$; $cn \in CN_{ONT}$; $cn\_bt \in CN_{ONT} \cup DT_{ONT}$, последнее означает, что свойство с именем pn_i характеризует класс cn_j, а значениями свойства могут быть элементы типа другого класса из CN_{ONT} или основного множества типов DT_{ONT}; R^{C}_{ONT} — причинно-следственные отношения между концептами. В свою очередь, $R^{C}_{ONT} = \{r^{C}_{ONT_1} ... r^{C}_{ONT_g}\}$, $r^{C}_{ONT_j} = \langle name_j, cn_{LHS_j}, cn_{RHS_j}, op_j \rangle$, $j \in \overline{1, g}$, где $name_j$ — имя отношения; cn_{LHS_j} — левый класс отношения (ссылка на класс); cn_{RHS_j} — правый класс отношения (ссылка на класс); op_j — оператор отношения, $op_j \in \{and, or, not\}$; g — число отношений.

3.3. Преобразование диаграмм классов UML

Далее кратко рассмотрим преобразование диаграммы классов UML в формате XML в онтологию продукций приложения:

$$T_{XML-ONT} : M^{UML}_{XML} \rightarrow M_{ONT}.$$

Следует отметить, что в данной работе в модели UML рассматриваются отношения только типа "ассоциация". По этой причине далее под отношением (связью) будем понимать именно ассоциацию.

Процесс преобразования представляет собой анализ структур XML-файла в целях выделения конструкций, описывающих элементы модели, а также поиска соответствия в таблице преобразований (табл. 1).

При этом в результате анализа формируются множества классов CN_{ONT}, свойств классов PN_{ONT}, объектов Obj_{ONT}, типов данных DT_{ONT} и отношений R_{ONT}

Таблица 1

Анализируемые элементы XMI UML

Элементы XMI UML	Описание
UML:Model	Модель диаграммы UML
UML:Class	Класс
UML:Attribute	Атрибут класса
UML:Expression	Значение атрибута
UML:DataType	Тип данных
UML:Association	Связь типа "ассоциация"
UML:AssociationEnd	Часть ассоциации (роль связи)
UML:Multiplicity	Мощность (кратность) связи
UML:MultiplicityRange	Интервал (диапазон) кратности связи

Таблица 2

Пример соответствия элементов онтологии и CLIPS

Элемент онтологии	Элемент CLIPS
Класс	(**deftemplate** <имя класса> "<описание класса>" <свойства>)
Свойство	(**slot**)
Значение свойства по умолчанию	(**default** "<значение>")
Причинно-следственное отношение	(**defrule** <имя отношения> "<описание>" <объекты> => <объекты>)
...	

онтологии продукций приложения. В частности, происходит преобразование ассоциаций в причинно-следственные отношения: $R_A^{UML} \to R_{ONT}^C$, при этом:

$$r_{A-LHS}^{UML} \to cn_{LHS}^{ONT},$$
$$r_{A-RHS}^{UML} \to cn_{RHS}^{ONT},$$
$$op = \begin{cases} and, \ if \ m_r \in \{1,1..n\} \\ or, \ if \ m_r \in \{0,0..n\} \end{cases}.$$

Таким образом, на формирование логического оператора влияет мощность (кратность) отношения правого элемента, если мощность не равна нулю, то назначается оператор *"and"*.

Необходимо отметить, что предложенная интерпретация отношения "ассоциация" не является общепринятой, поскольку отсутствует практика использования UML-моделей для формирования элементов БЗ.

На основе сформированной онтологии может быть получен программный код, например, на языке CLIPS:

$$T_{ONT-Code}: M_{ONT} \to Code_{CLIPS}.$$

Примеры соответствия элементов онтологии и CLIPS приведены в табл. 2.

4. Пример формирования продукций на основе концептуальных моделей

Рассмотрим пример автоматизированного формирования продукций на примере фрагмента диаграммы классов (рис. 4) [11].

В качестве источника UML-моделей, описывающих классы предметной области и их отношения, используется CASE-средство IBM Rational Rose. Анализируемый XML-файл с моделями получен с помощью модуля трансформации моделей XMI UML [8].

Графическому представлению модели (рис. 4) соответствует фрагмент XML-кода (рис. 5). При этом выделены ключевые конструкции (см. табл. 1), на основе которых осуществлялось извлечение необходимых элементов концептуальной модели.

Понятия онтологии могут быть представлены в нотации RVML [5] (рис. 6) для их дальнейшего уточнения и проверки.

На основе понятий онтологии генерируется код базы знаний в формате CLIPS (рис. 7).

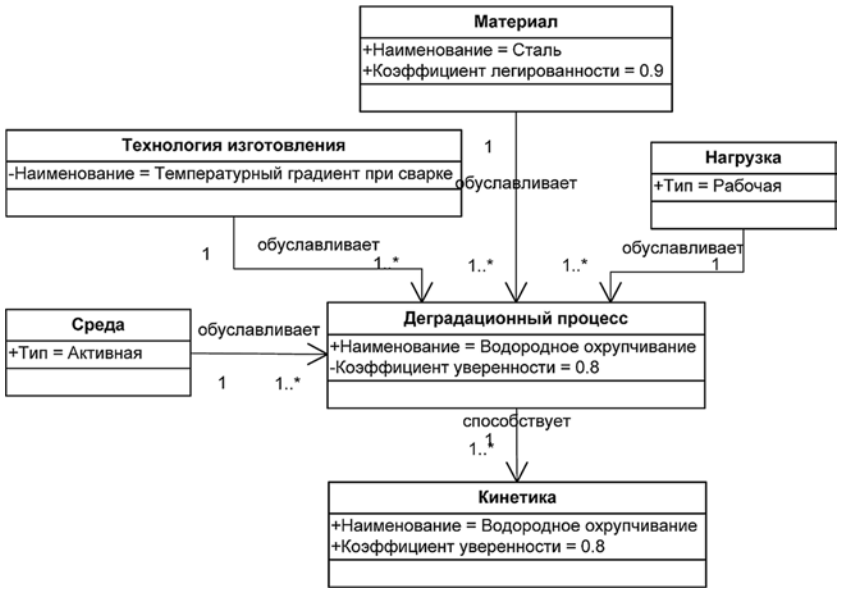

Рис. 4. Фрагмент анализируемой диаграммы классов UML

```
...
    <UML:Association xmi.id = 'G.4' name = 'обуславливает' ... >
        <UML:AssociationEnd xmi.id = 'G.5' type = 'S.350.1300.40.10' ... >
            <UML:Multiplicity>
                <UML:MultiplicityRange xmi.id = 'id.3510400.3' lower = '1' upper = '-1' />
            </UML:Multiplicity>
        </UML:AssociationEnd>
        <UML:AssociationEnd xmi.id = 'G.6' type = 'S.350.1300.40.1' ... >
            <UML:Multiplicity>
                <UML:MultiplicityRange xmi.id = 'id.3510400.4' lower = '1' upper = '1' />
            </UML:Multiplicity>
        </UML:AssociationEnd>
    </UML:Association>
...
    <UML:DataType xmi.id = 'G.16' name = 'String' ... />
    <UML:DataType xmi.id = 'G.17' name = 'Double' ... />
...
    <UML:Class xmi.id = 'S.350.1300.40.1' name = 'Материал' ... >
        <UML:Attribute xmi.id = 'S.350.1300.40.2' name = 'Наименование' type = 'G.16' ... >
            <UML:Expression language = '' >
              <UML:Expression.body>Сталь</UML:Expression.body>
            </UML:Expression>
        </UML:Attribute>
        <UML:Attribute xmi.id = 'S.350.1300.40.3' type = 'G.17' ... >
          <UML:ModelElement.name>Коэффициент легированности</UML:ModelElement.name>
            <UML:Expression language = '' body = '0.9' />
        </UML:Attribute>
    </UML:Class>
...
```

Рис. 5. Фрагмент анализируемой диаграммы классов UML в формате XML

Рис. 6. Пример полученной продукции в нотации RVML

```
;************************ Шаблоны ******************
(deftemplate material
   (slot naimenovanie (default "СТАЛЬ"))
   (slot koefficient_legirovannosti (default "0.9"))
   (slot cf (default "1"))
)
…
;************************ Правила ******************
(defrule pravilo_p_01 ""
   (declare (salience 1))
   (material
      (naimenovanie "СТАЛЬ")
      (koefficient_legirovannosti "0.9")
      (cf "1")
    )
…
=>
  (assert
       (degradacionnyy_process
            (naimenovanie "ВОДОРОДНОЕ ОХРУПЧИВАНИЕ")
            (cf "0.8")
        )
  )
)
…
```

Рис. 7. Фрагмент кода CLIPS, соответствующий RVML-модели

Заключение

Решение задач, связанных с получением (приобретением), структурированием и формализацией знаний, позволяет повысить эффективность процесса разработки БЗ интеллектуальных систем. В настоящей работе предложено автоматизировать анализ концептуальных моделей и автоматически формировать программный код на основе графических примитивов. Такой подход позволяет избежать ошибок программирования, которые, как правило, связаны с большими издержками на последующих этапах жизненного цикла программного продукта.

При решении задач исследования проведен анализ методов, подходов и систем, а также различных стандартов (языков) концептуального и онтологического моделирования. На основании этого анализа были выявлены основные источники концептуальных моделей: диаграммы классов UML и онтологии OWL DL. В качестве целевого ЯПЗ использован CLIPS. Подробно описан алгоритм преобразования концептуальных моделей (диаграмм классов UML) в программный код продукционной базы знаний.

На основе разработанного алгоритмического обеспечения реализован исследовательский прототип web-сервиса для автоматизированного формирования продукционных баз знаний CLIPS [4]. Основной областью применения такого сервиса является обеспечение распределенной коллективной работы специалистов-предметников при создании БЗ для продукционных экспертных систем. Сервис использовался при выполнении работ по хоздоговору с ОАО "ИркутскНИИхиммаш" на создание проблемно-ориентированного редактора продукционных БЗ в области оценки технического состояния и остаточного ресурса нефтехимического оборудования [12].

Безусловно, данный подход не позволяет исключить ошибки, обусловленные неточностью или неполнотой анализируемых концептуальных моделей. Однако автоматическая генерация программного кода на основе моделей позволяет использовать принцип быстрого прототипирования при реализации БЗ с последующей их проверкой и тестированием в сторонних средствах. По результатам тестирования можно внести необходимые изменения в модели (диаграммы классов UML или RVML) и провести повторную генерацию программного кода.

В дальнейшем планируется расширить функции программного обеспечения за счет подсистемы тестирования сгенерированных БЗ, использовать в качестве источников концептуальных моделей OWL DL, а также осуществить интеграцию сервиса со средством создания продукционных экспертных систем на основе порождающего программирования (модельно-управляемого подхода) [5].

Список литературы

1. **Люгер Дж. Ф.** Искусственный интеллект: стратегии и методы решения сложных проблем, 4-е изд.: пер. с англ. М.: Вильямс, 2003. 864 с.

2. **Гаврилова Т. А., Хорошевский В. Ф.** Базы знаний интеллектуальных систем. СПб.: Питер, 2000. 384 с.

3. **Система** визуального моделирования для разработки приложений "IBM Rational Rose". 2014. URL: http://www-03.ibm.com/software/products/ru/enterprise (дата обращения: 10.12.2014).

4. **Дородных Н. О., Юрин А. Ю.** Web-сервис для автоматизированного формирования продукционных баз знаний на основе концептуальных моделей // Программные продукты и системы. 2014. № 4. С. 103—107.

5. **Грищенко М. А., Юрин А. Ю., Павлов А. И.** Разработка экспертных систем на основе трансформации информационных моделей предметной области // Программные продукты и системы. 2013. № 3. С. 143—147.

6. **Частиков А. П., Гаврилова Т. А., Белов Д. Л.** Разработка экспертных систем. Среда CLIPS. СПб.: БХВ-Петербург, 2003. 608 с.

7. **Документация** спецификации Unified Modeling Language (UML). 2014. URL: http://www.omg.org/spec/UML (дата обращения: 10.12.2014).

8. **Документация** спецификации XML Metadata Interchange (XMI). 2014. URL: http://www.omg.org/spec/XMI (дата обращения: 10.12.2014).

9. **Документация** стандарта Web Ontology Language (OWL). 2014. URL: http://www.w3.org/TR/owl-semantics (дата обращения: 10.12.2014).

10. **Николайчук О. А.** Методы, модели и инструментальное средство для исследования надежности и безопасности сложных технических систем: Автореферат дис. ... д-ра техн. наук. М., 2011. 37 с.

11. **Берман А. Ф.** Информатика катастроф // Проблемы безопасности и чрезвычайных ситуаций. 2012. № 3. С. 17—37.

12. **Юрин А. Ю., Грищенко М. А.** Редактор баз знаний в формате CLIPS // Программные продукты и системы. 2012. № 4. С. 83—87.

N. O. Dorodnykh, Postgraduate Student, e-mail: tualatin32@mail.ru,
A. Yu. Yurin, Head of Laboratoty, e-mail: iskander@icc.ru, Institute for System Dynamics and Control Theory Siberian Branch of the Russian Academy of Sciences, Irkutsk

Using UML Class Diagrams for Design of Knowledge Bases of Rule-Base Expert Systems

The paper describes the approach for computer-aided design of rule-bases on the basis of conceptual models. The proposed approach consists of six steps: design of the conceptual model, representation of the conceptual model in the form of a XML-document, analysis (parsing) of the XML-document, generation ontology of rules, visualization and modification of rules, model-based generation of a program code. UML class diagrams are considered as the main source of conceptual models, because the analysis shown that it is the most common way of conceptualizing knowledge in software development. The original notation — Rule Visual Modeling Language (RVML) is proposed for visualization and modification of rules. The C Language Production System (CLIPS) is selected as a targeted programming language.

The main advantage of the approach is minimization of errors in program codes.

The approach does not eliminate errors caused by inaccuracy or incompleteness of analyzed conceptual models, however, the automatic model-based generation of program codes allows to apply the principle of rapid prototyping for the design of knowledge bases with the subsequent inspection and testing obtained codes with the aid of third-party tools.

The proposed algorithms are implemented in a prototype of a web-service. The main area of application of the web-service is providing distributed collaboration of experts and analytics in design of knowledge bases for rule-based expert systems.

Keywords: *knowledge base, rules, computer-aided, knowledge acquisition, conceptual model, ontology, UML, CLIPS*

References

1. **Luger G. F.** *Iskusstvennyj intellekt: strategii i metody reshenija slozhnyh problem*, 4-e izd. (Artificial Intelligence: Structures and Strategies for Complex Problem Solving, 4th ed). Translated from English. Moscow: Williams, 2003, 864 p. (in Russian).

2. **Gavrilova T.A., Khoroshevskii V.F.** *Bazy znanij intellektual'nyx sistem* (Knowledge bases of intelligent systems). Sankt-Petersburg: Piter, 2000, 384 p. (in Russian).

3. **Sistema** vizual'nogo modelirovanija dlja razrabotki prilozhenij "IBM Rational Rose" (Visual modeling software for application development "IBM Rational Rose"). 2014, available at: http://www-03.ibm.com/software/products/ru/enterprise (accessed: December 10, 2014). (in Russian).

4. **Dorodnykh N. O., Yurin A. Yu.** Web-servis dlya avtomatizirovannogo formirovaniya produktsionnykh baz znanii na osnove kontseptual'nykh modelei. *Programmnye produkty i sistemy*, 2014, no. 4, pp. 103—107 (in Russian).

5. **Grishchenko M. A., Yurin A. Yu., Pavlov A. I.** Razrabotka ekspertnykh sistem na osnove transformatsii informatsionnykh modelei predmetnoi oblasti. *Programmnye produkty i sistemy,* 2013, no. 3, pp. 143—147 (in Russian).

6. **Chastikov A. P., Gavrilova T. A., Belov D. L.** *Razrabotka ejspertnyx sistem. Sreda CLIPS* (Development of expert systems. CLIPS environment). Sankt-Petersburg: BHV-Petersburg, 2003, 608 p. (in Russian).

7. **Documentation** of specification of Unified Modeling Language (UML), available at: http://www.omg.org/spec/UML/ (accessed December 10, 2014).

8. **Documentation** of specification of XML Metadata Interchange (XMI), available at: http://www.omg.org/spec/XMI/ (accessed December 10, 2014).

9. **Documentation** of standard Web Ontology Language (OWL). 2014, available at: http://www.w3.org/TR/owl-semantics (accessed December 10, 2014).

10. **Nikolaychuk O. A.** *Metodi, modeli i instrumentalnoe sredstvo dlia issledovania nadezhnosti i bezopasnosti slozhnih tehnicheskih system:* avtoref. diss. d-ra tehn. nauk. (Methods, models and software for investigation reliability and safety of complex technical systems. Thesis abstract on scientific degree Dr. of Techn. Sciences). Moscow, 2011. 37 p. (in Russian).

11. **Berman A. F.** Informatika katastrof. *Problemyi bezopasnosti i chrezvyichaynyih situatsiy*, 2012, no. 3, pp. 17—37 (in Russian).

12. **Yurin A. Yu., Grishchenko M. A.** Redaktor baz znanii v formate CLIPS. *Programmnye produkty i sistemy,* 2012, no. 4, pp. 83—87 (in Russian).