Инженерный вестник Дона, №4 (2017)
ivdon.ru/ru/magazine/archive/n4y2017/4604

Аналитическое программное обеспечение прогнозирования потребления ресурсов в системе комплексного учета, регистрации и анализа потребления энергоресурсов и воды промышленными предприятиями и объектами ЖКХ

В.Н. Подсвиров[1], Е.С.Семенистая[1], В.Б. Подопригора[2]

[1]Южный федеральный университет, Таганрог
[2]ОАО «НПП КП «Квант», Ростов-на-Дону

Аннотация: в данной работе рассматриваются вопросы разработки аналитического программного обеспечения, являющегося частью аппаратно-программного комплекса по учету, регистрации и анализа потребления энергоресурсов. Основной функцией аналитического программного обеспечения является прогнозирование расхода энергоресурсов и аварийных состояний, а также нештатных ситуаций. Для создания аналитического программного обеспечения предлагается использование контекстно-доопределяемых машинных языков. Рассмотрены вопросы реализации таких языков для интеллектуальных агентов (ИОА), которые используются для решения задач прогнозирования потребления ресурсов в сфере ЖКХ. Особое внимание уделяется методам поддержки изменения реализующих функций. Этот же подход может быть использован для решения и других задач. Основная идея состоит в вычленении частей алгоритма компонента и организации соответствующей связи между такой частью и контекстом, который может изменять ее прямо или косвенно.
Ключевые слова: программное обеспечение, учет энергоресурсов, контекстно-доопределяемые языки, интеллектуальные агенты, машинные языки.

Введение

Автоматизированные системы учета энергоресурсов на сегодняшний день все более активно внедряются в систему ЖКХ. Применение АСКУЭ дает множество преимуществ, как для потребителей энергоресурсов, так и ресурсоснабжающих организаций.

Практически все современные системы представляют собой многоуровневые аппаратно-программные комплексы, с гибкой настраиваемой структурой [1].

Основной целью создания и применения автоматизированной информационно-измерительной системы коммерческого учета для поставщиков и потребителей энергоресурсов, является получение полной и достоверной информацией необходимой для коммерческих расчетов, об

объемах поставленных, переданных и потребленных энергоресурсов в технологическом процессе функционирования всей энергосистемы.

Программная часть аппаратно-программного комплекса учета расхода энергоресурсов и воды состоит из следующих классов программного обеспечения: клиентское ПО; сервисное ПО; аналитическое ПО.

В данной системе клиентское программное обеспечение (КПО) представляет собой совокупность следующих модулей: основной модуль, Веб-интерфейс, мобильный модуль. С помощью КПО осуществляется взаимодействие с внешними системами; разграничение и протоколирование доступа персонала; подготовка отчетов различного уровня для отображения оператору; регистрация информации о состоянии приборов учета и линии связи; хранение данных, полученных с приборов учета, в специализированной базе данных; обеспечение удаленного доступа и управления; получение необходимой информации из центра сбора информации (аналитическое ПО) для предоставления её пользователю.

Сервисное программное обеспечение (СПО) предназначено для решения следующих основных задач: настройка основных параметров функционирования приборов учета; аналитическое программное обеспечение предназначено для статистической обработки ранее собранных данных с целью прогнозирования потребления природного газа, электроэнергии и воды; упорядочивания выборок информации от приборов учета расхода ресурсов; формирования отчетной информации о потреблении природного газа, электроэнергии, горячей и холодной воды; выгрузки (передачи) информации о потреблении ресурсов в информационные системы организаций жилищно-коммунального хозяйства (ЖКХ) и ресурсоснабжающих организаций (РСО).

Аналитическое программное обеспечение (АПО) является частью сервисного программного обеспечения и состоит из следующих

функциональных компонентов: аналитическое ядро, набор функций и процедур в виде библиотек для аналитических и прогнозных задач [2].

Аналитическое ПО является изолированным модулем, его работа достаточно автономна, что позволяет выбрать алгоритмы отличные от применяемых в других частях системы. Принципы прогнозирования выбраны с учетом особенностей структуры аналитического ПО, в частности наличия в нем двух блоков: подсистемы статистической обработки и подсистемы прогнозирования потребления. Данное разделение допускает применение метода интеллектуальных агентов, однако помимо этих двух подсистем данный метод может быть применен и в подсистеме прогнозирования аварий [3].

Интеллектуальный агент — программа, самостоятельно выполняющая задание, указанное пользователем компьютера, в течение длительных промежутков времени. Такие агенты, как и любые прочие, имеют сложный, зачастую реализуемый нейросетями алгоритм [4].

На рис.1 представлена структура обучающегося агента. Обучающийся агент может состоять из производительного элемента (ПЭ), определяющего действия, и обучающего элемента (ОЭ), который модифицирует производительный элемент в целях получения лучших решений. На структуру ОЭ оказывают влияние компоненты ПЭ, обратные связи обучения, способы представления компонентов [5].

Рассмотрим типовые компоненты ПЭ: 1. Средства прямого отображения условий в действия. 2. Средства логического вывода релевантных свойств мира из последовательности результатов восприятия. 3. Информация о том, как развивается внешний мир и какие результаты возможных действий могут быть получены агентом. 4. Информация о полезности, которая показывает, насколько желательными являются те или иные состояния мира. 5. Информация о ценности действий, показывающая

желательность действий. 6. Цели, описывающие классы состояний, достижение которых максимизирует полезность для агента.

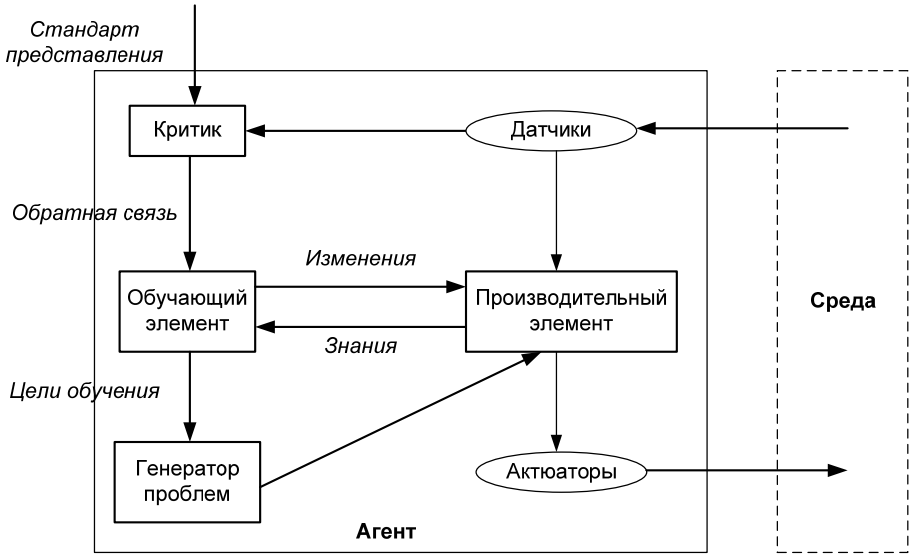

Рис. 1 – Общая структура обучающего агента

Создание систем, реализующих контекстно-доопределяемые языки (КДЯ), требует особенного подхода, так как не все традиционные средства компиляции и интерпретации подходят для решения этой задачи. Разумеется, элементы контекстного доопределения имеются в современных языках программирования и средства их реализации проработаны в той или иной степени. Только решаемая ими задача является более частной, а потому и более простой [6].

В условиях реализации системы учета энергоресурсов и анализа потребления одной из типичных задач реализации КДЯ является обнаружение и правильная интерпретация контекста. То есть некоторая совокупность синтаксических правил выступает в качестве предиката, вычисление которого дает нам значение «истина» или «ложь». Соответственно значение «истина» запускает следующий этап распознавания и генерации (интерпретации), а значение «ложь» - возвращает процесс обратно, к дальнейшему разбору предложений входного текста [7].

В зависимости от соглашений по контексту, можно искать начало контекстных изменений в начале строки или в произвольном месте строки. Контекст может занимать строку целиком или ее часть, может распространяться на некоторое количество строк и так далее. Рассмотрим работу агента на абстрактном примере. Представим входной текст в виде:

Строка 1

Строка 2

...

Строка i

...

Строка n

Положим для простейшего случая, как представлено выше, контекст занимает целое количество строк и действие распространяется на некоторое количество целых строк, тогда разбор текста, в частности, может иметь вид:

Строка 1

Контекст 1 (Строка 2)\{

    Строка 3

    Строка 4

\}

Контекст 2 (Строка 5, Строка 6)\{

    Строка 7

\}

Строка n

Скобочки "\{", "\}" определяют область действия контекста. В исходном тексте не присутствуют, условно расставляются системой распознавания. "Строка 2" распознается как "Контекст 1", а строки: "Строка 5"и "Строка 6" распознаются как "Контекст 2". "Контекст 1" и "Контекст 2"

являются условно независимыми, так как области действия их не пересекаются. Но может быть и так:

```
Строка 1
Контекст 1 (строка 2)\{
    Строка 3
    Строка 4
    Контекст 2 (Строка 5, Строка 6)\{
        Строка 7
    \}
    Строка 8
\}
Строка n
```

В этом случае "Контекст 2" можно рассматривать как дополняющий "Контекст 1" или заменяющий его действие в зависимости от алгоритма реализации контекста. По аналогии с обычными процедурными языками программирования, это могут быть циклы, вложенные друг в друга. В общем случае КДЯ позволяет оставить "Контекст 2" последействие для "Контекст 1": например, запретить выполнять область вложенного "Контекст 2" в цикле, образованном "Контекст 1" и предложить в рамках "Контекст 1" выполнить последействие "Контекст 2". Это принципиально отличается от традиционного подхода, обобщая его.

Методы, обеспечивающие последействие, могут быть переданы не только охватывающим контекстам, но и контекстам, следующим один за другим:

```
Строка 1
Контекст 1 (строка 2)\{
    Строка 3
    Строка 4
```

Контекст 2 (Строка 5, Строка 6)\{

Строка 7

\}

Строка 8

Контекст 3 (Строка 9, Строка 10)\{

Строка 11

\}

Строка 12

\}

...

Строка n

"Контекст 3" может получить доступ к методу последействия "Контекст 2". Для "Контекст 3" и "Контекст 1" это могут быть разные методы.

Система, отличающая контекст от обычного текста, может содержать противоречивые и взаимоисключающие правила распознавания. Кроме того, может изменяться множество этих правил и последовательность их применения. В результате чего текущее состояние системы распознавания определяется кортежем {А, П}, содержащим ссылку на соответствующий алгоритм последовательности выборки правил А и массив ссылок на правила распознавания П [8]. Правила распознавания в общем случае могут воздействовать на общий алгоритм выбора правил и на множество выбираемых правил, что может порождать зацикливание при некорректном использовании.

Объектом может считаться как один символ текста, так и некоторая их совокупность. В том числе и не образующую последовательность элементов текста, а состоящая из разрозненных символов или групп символов текста.

В общем случае удобно использовать стек тегов. Тогда значительно облегчается обработка блоков, присутствующих в исходном тексте. Также такой стек удобен при организации отладки самой реализующей системы контекстно-доопределяемого языка. В таком стеке должны быть определены операции: чтения, записи и ознакомления с содержимым стека. Операции чтения и записи представляют собой классические операции, а операция ознакомления аналогична операции чтения без удаления информации. Кроме того, операция ознакомления предполагает индексное чтение стека вплоть до достижения дна стека [10].

Существует возможность простого вычленения контекста из функций, варьирующих обучающие составляющие, что и позволяет производить гибкую вариацию алгоритмов таких функций. Использование такой возможности может быть не всегда оправданно, но для интеллектуальных агентов со сложным поведением могут оказаться эффективным решением возникающих проблем управления.

Как пример, рассмотрим применение данного подхода для программы прогнозирования потребления энергоресурсов и воды, разрабатываемой в рамках проекта «Разработка и создание высокотехнологичного производства инновационной системы комплексного учета, регистрации и анализа потребления энергоресурсов и воды промышленными предприятиями и объектами ЖКХ»

Общая диаграмма логической структуры программы представлена на рис.2.

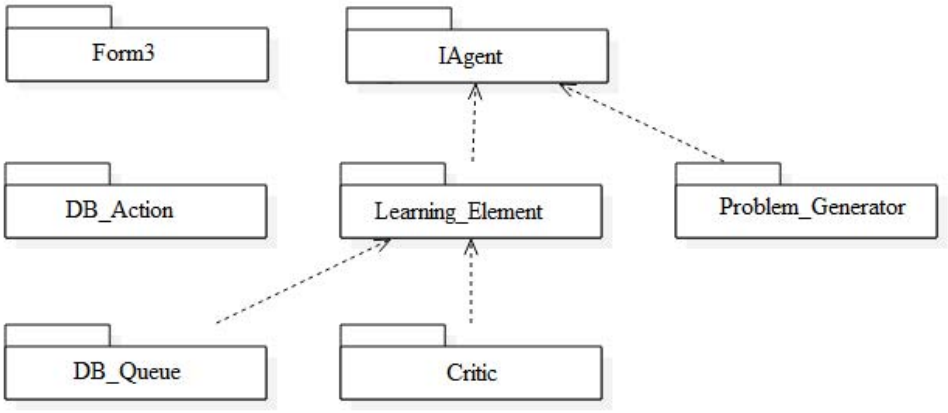

Рисунок 2 - Общая диаграмма логической структуры программы

Класс Form3 имеет вспомогательное значение (создание окон программы, запуск программы создания рабочей БД, Web - клиента, Клиента предсказаний).

Класс DB_Action предназначен для формирования входной БД на основе рабочей БД. В процессе формирования происходит статистическая обработка данных и последующее сохранение результатов в создаваемых таблицах.

Класс DB_Queue содержит средства работы с очередью, он обеспечивает следующие функции:

InitQueueTable – обеспечивает создание и инициализацию таблицы очереди;

AddNewLine – добавляет новую строку в таблицу очереди;

GetNextLine – читает очередную строку из очереди;

DeleteQueueTable – удаляет все содержимое таблицы очереди;

ClearQueueTable – очищает таблицу, если такая имеется, от уже отправленных ответов;

PutOutXML – вставляет в указанную строку (определяется по UID) таблицы результат обработанного запроса;

GetWebString – получает у Web - сервера очередной пакет запросов;

PutWebString – отправляет на Web - сервер очередную сформированную строку ответа на соответствующий запрос (идентифицируется их общим UID);

WaitQueue – позволяет приостановить обработку уже имеющихся в очереди запросов (ненулевое значение рассматривается как препятствующее обработке);

Класс IAgent имеет основную связь с обсуждаемым в статье вопросами, он обеспечивает формирование предсказаний. Этот класс, в свою очередь опирается на классы:

Learning_Element - обеспечивает "обучение" (динамическую адаптацию) используемых алгоритмов предсказаний.

Problem_Generator - отвечает за возможные проблемы функционирования алгоритмов предсказаний.

Critic - выявляет критические отклонения алгоритмов предсказаний.

Собственно модификации грамматики КДЯ в зависимости от контекста реализуются именно в этом классе.

В качестве примера конкретной семантики ответа приведем вид пакета запросов, имеющий следующий формат:

```
<?xml version="1.0" encoding="utf-8"?>
<req>
  <server></server><!--Идентификатор сервера КПО -->
  <rq_tp>ana</rq_tp><!--Тип запроса. -->
  <ac_tp>get</ac_tp><!—Метод. -->
  <data>
    <row>
      <id></id><!--глобальный идентификатор данного запроса-->
  <alhorytm_code></alhorytm_code><!--номер типа алгоритма -->
```

<period_ht_strat></period_ht_strat><!—начало периода исторических данных -->

<period_ht_end></period_ht_end> <!—окончание периода исторических данных -->

<prognoz_type></prognoz_type><!--вид прогнозирования -->

<horizont_start></horizont_start><!—начало горизонта прогнозирования >= -->

<horizont_end></horizont_end><!—окончание горизонта прогнозирования <= -->

<detail_code></detail_code><!--код вида детализации результата прогноза-->

<resource></resource><!--вид (глобальный идентификатор) ресурса -->

<zone></zone><!--код (глобальный идентификатор) зоны (поддерева)-->

<org>

<id></id>

<!--...-->

<id></id>

</org><!--код/коды (глобальный идентификатор) организации -->

<pomeschenie>

<id></id>

<!--...-->

<id></id>

</pomeschenie><!--код/коды (глобальный идентификатор) помещения -->

<client>

<id></id>

<!--...-->

```
            <id></id>
        </client><!--код/коды (глобальный идентификатор) плательщиков -->
        <lic_schet>
            <id></id>
            <!--...-->
            <id></id>
        </lic_schet><!--код/коды (глобальный идентификатор) лицевых счетов
-->
        <counter>
            <id></id>
            <!--...-->
            <id></id>
        </counter><!--код/коды (глобальный идентификатор) приборов учета -
->
        <balance_group>
            <id></id>
            <!--...-->
            <id></id>
        </balance_group><!--код/коды (глобальный идентификатор)
балансовых групп -->
          </row>
        </data>
      </req>
```

Поскольку это фрагмент запроса из работающего коммерческого варианта информационной системы, часть полей запроса предназначена для обработки функциями других классов системы.

Основное отличие применения КДЯ – самообучение системы, т.е. получение новых знаний при много кратных применениях редукционных правил.

Подводя итог, отметим, что предлагаемый подход подразумевает типовое внедрение в функцию обратной связи компонентов алгоритмов обучения. Технически это реализуется средствами КДЯ путем выделения контекста, определяющего вариации, как средство обучения, так и алгоритмов варьирующих эти средства. То есть речь идет об адаптации изменяющих (обучающих) функций к постоянно изменяющимся внешним условиям, в которых существует интеллектуальный агент. В принципе процесс внедрения изменяющихся составляющих в ранее написанные изменяющиеся составляющие может быть повторен неопределенное количество раз. Естественно очередное внедрение такого рода определяется целесообразностью и ограничивается сложностью получаемого общего алгоритма функционирования интеллектуального агента.

Результаты исследований, изложенные в данной статье, получены при финансовой поддержке Минобрнауки РФ в рамках реализации проекта «Разработка и создание высокотехнологичного производства инновационной системы комплексного учета, регистрации и анализа потребления энергоресурсов и воды промышленными предприятиями и объектами ЖКХ» по постановлению правительства №218 от 09.04.2010г. Исследования проводились в ФГАОУ ВО ЮФУ.

Литература

1. Корецкий А.А., Подопригора В.Б., Мирошниченко Е.П. Особенности разработки и внедрения системы учета энергоресурсов // Инженерный вестник Дона, №3. 2017. URL: ivdon.ru/ru/magazine/archive/N3y2017/4365.

2. Е.С.Семенистая, И.Г.Анацкий, Ю.А. Бойко Разработка программного обеспечения автоматизированной системы контроля и учета энергоресурсов

и воды // Инженерный вестник Дона, №4. 2016. URL: ivdon.ru/ru/magazine/archive/n4y2016/3897.

3. Podsvirov V.N. Context-Redefined Language Application for the Tasks of Intelligent Learning Agents (Resource Consumption Behavior Prediction Tasks). International Journal of Applied Engineering Research ISSN 0973-4562 Volume 11, Number 15 (2016) pp. 8471-8484.

4. Agent-Oriented Methodologies. Ed.by B.Henderson-Sellers and P.Giorgini. Idea Group Publishing, 2005. 413 p.

5. Подсвиров В.Н. Контекстно-доопределяемые языки и интеллектуальные агенты. Актуальные проблемы современной науки: Международная научно-практическая конференция. Алушта, 2015.С.192-194.

6. Рассел, Стюарт, Норвиг, Питер. Искусственный интеллект: современный подход, 2-е изд..: Пер. с англ. — М.: Издательский дом "Вильяме", 2006. — 1408 с.: ил. — Парал. тит. англ.

7. Franciso, M. Integrating Multi-Agent Systems: A Case Study. / Raymond, S. Staron, F. // IFIP International Federation for information Processing, Springer Verlag. 2005. vol. 159. pp. 99-108.

8. Brian Henderson-Sellers. Object-Oriented Metrics. Measures of Complexity. Prentice-Hall, Inc. Upper Saddle River, NJ, USA 1996. 234p.

9. Bauer B. UML Class Diagrams: Revisited in the Context of Agent-Based Systems: Agent-Oriented Software Engineering (AOSE) 2001, Montreal pp.l-8.

10. Barbosa, Fernanda, Cunha, Jos. A coordination language for collective agent based systems: GroupLog//ACM Symposium on Applied computing SAC'2000 Villa Olmo, Como, Italy, 2000. pp. 189-195.

References

1. Koretskiy A.A., Podoprigora V.B., Miroshnichenko Ye.P. Inženernyj vestnik Dona (Rus), 2017, №3. URL: ivdon.ru/ru/magazine/archive/N3y2017/4365.

2. Ye.S.Semenistaya, I.G.Anatskiy, YU.A. Boyko. Inženernyj vestnik Dona (Rus), 2016, №4. URL: ivdon.ru/ru/magazine/archive/n4y2016/3897.

3. Podsvirov V.N. Context-Redefined Language Application for the Tasks of Intelligent Learning Agents (Resource Consumption Behavior Prediction Tasks). International Journal of Applied Engineering Research ISSN 0973-4562 Volume 11, Number 15 (2016) pp. 8471-8484.

4. Agent-Oriented Methodologies. Ed.by B.Henderson-Sellers and P.Giorgini. Idea Group Publishing, 2005. 413 p.

5. Podsvirov V.N. Kontekstno-doopredelyayemyye yazyki i intellektual'nyye agenty [Context-pre-defined languages and intelligent agents]. Aktual'nyye problemy sovremennoy nauki: Mezhdunarodnaya nauchno-prakticheskaya konferentsiya. Alushta, 2015. pp.192-194.

6. Russell, Stuart, Norvig, Peter. Iskusstvennyj intellekt: sovremennyj podhod [Artificial intelligence: a modern approach], 2nd ed. Trans. with English. M.: Publishing house "William", 2006. 1408 p.

7. Franciso, M. Integrating Multi. Agent Systems: A Case Study. Raymond, S. Staron, F. IFIP International Federation for information Processing, Springer Verlag. 2005. vol. 159. pp. 99-108.

8. Brian Henderson-Sellers. Object-Oriented Metrics. Measures of Complexity. Prentice-Hall, Inc. Upper Saddle River, NJ, USA 1996. 234p.

9. Bauer B. UML Class Diagrams: Revisited in the Context of Agent-Based Systems: Agent-Oriented Software Engineering (AOSE) 2001, Montreal pp.l-8.

10. Barbosa, Fernanda, Cunha, Jos. A coordination language for collective agent based systems: GroupLog. ACM Symposium on Applied computing SAC'2000 Villa Olmo, Como, Italy, 2000. pp. 189-195.