DOI: 10.18137/RNU.V9187.22.04.P.149 УДК004.5:004.4’2

М.Э. Нагорных, А.Н. Быков, С.А. Чернышёв

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ФОРМИРОВАНИЯ ОНТОЛОГИИ ПРЕДМЕТНОЙ ОБЛАСТИ И ЕЕ КОДОГЕНЕРАЦИИ

Аннотация. Разработано программное обеспечение для построения онтологий и их кодогенерации. Представлены аналоги приложения, выявлены их сильные и слабые стороны. Подробно рассмотрены процессы описания онтологий и кодогенерации, а также стек технологий разработки, архитектура приложения. Продемонстрированы результаты работы программы на абстрактных примерах.

Ключевые слова: онтология, кодогенерация, Python, xmlschema.

M.E. Nagornykh, A.N. Bykov, S.A. Chernyshev

SOFTWARE FOR DOMAIN ONTOLOGY FORMATION AND CODE GENERATION

Abstract. Software for building ontologies and their code generation has been developed. During development, analogues of the application were considered, their strengths and weaknesses were identified. Also, the processes of ontology description and code generation were considered in detail. The article presents: a stack of development technologies, application architecture, and also demonstrates the results of the program on abstract examples.

Keywords: ontology, code generation, Python, xmlschema.

Введение

В современной ИТ-сфере применяется большое количество решений с использованием онтологий, например, искусственный интеллект, моделирование бизнес-процессов и семантическая паутина, категоризация и анализ знаний о предметной области. Они также используются для решения проблем интеграции данных путем предоставления общего согласованного словаря и ограничений между ними, которые затем используются для того, чтобы программное обеспечение понимало, что сущность «автомобиль» реляционной базы данных на самом деле означает то же самое, что и «транспортное средство» в некотором прикладном программном обеспечении.

Стоит отметить, что с ростом сложности разрабатываемых систем усложняется и процесс создания онтологий. Постоянное взаимодействие с базой знаний, содержащей онтологию предметной области или агента в мультиагентной системе, при наличии большого количества таких объектов уменьшает общую производительность системы. Этого можно избежать при отсутствии требований к модификации онтологии поведения агента в процессе работы мультиагентных систем путем использования автоматической кодогенерации и внедрения онтологии поведения непосредственно в код.

Поскольку приложений, позволяющих производить кодогенерацию по онтологиям, мало, более того, многие из них не производят ее на высоуровневых языках программирования, основная цель проделанной работы – создание и апробация программного обеспечения, которое позволило бы производить формирование онтологии и ее дальнейшую кодогенерацию.

Нагорных Максим Эдуардович
магистрант. Санкт-Петербургский государственный университет аэрокосмического приборостроения, Санкт-Петербург. Сфера научных интересов: имитационное моделирование; мультиагентные системы; системы поддержки принятия решений; разработка распределенных систем. Автор 7 опубликованных научных работ.
Электронный адрес: nagornykh_max@mail.ru

Быков Алексей Николаевич
магистрант. Санкт-Петербургский государственный университет аэрокосмического приборостроения, Санкт-Петербург. Сфера научных интересов: имитационное моделирование; мультиагентные системы; системы поддержки принятия решений; разработка распределенных систем. Автор 7 опубликованных научных работ.
Электронный адрес: alexey_bykovoff@mail.ru

Чернышёв Станислав Андреевич
кандидат технических наук, доцент кафедры информатики. Санкт-Петербургский государственный экономический университет; доцент кафедры прикладной информатики. Санкт-Петербургский государственный университет аэрокосмического приборостроения, Санкт-Петербург. Сфера научных интересов: имитационное моделирование; мультиагентные системы; системы поддержки принятия решений; разработка распределенных систем. Автор более 40 опубликованных научных работ.
Электронный адрес: chernyshev.s.a@bk.ru

Обзор существующих решений

Чтобы разобраться в принципах работы и особенностях существующих решений, позволяющих производить описание онтологий и их кодогерацию, было рассмотрено 3 существующих программы:

- Protege 5.0;
- NeOnToolkit;
- Software Ideas Modeler.

Protege. Protege является свободным программным средством с открытым исходным кодом для редактирования онтологий и систем управления знаниями. Программа имеет современный пользовательский интерфейс, в котором пользователь может создавать и описывать онтологии посредством классов и аннотаций к ним [16].

Платформа Protégé поддерживает два основных способа моделирования онтологий – средствами редакторов Protégé-Frames и Protégé-OWL. Онтологии, построенные в Protégé, могут быть экспортированы во множество форматов, включая RDF (RDF Schema), OWL и XML Schema.

NeOnToolkit. Это многоплатформенный редактор онтологий с открытым исходным кодом, который поддерживает разработку онтологий в F-Logic, OWL и RDF. Редактор основан на платформе Eclipse и предоставляет набор подключаемых модулей (в настоящее время доступно 45 подключаемых модулей для последней версии v2.4), охватывающих ряд инженерных работ по онтологии, включая аннотации и документацию, модуляризацию, настройку, повторное использование, эволюцию онтологий и др. [5]. Главное преимущество NeOnToolkit перед конкурентами – наличие множества плагинов, разработанных сообществом программистов.

Software Ideas Modeler. Мощное средство для создания диаграмм, которое оснащено профессиональными и экономящими время функциями. Создание проекта не требует специальных навыков, так как рабочее пространство и опции интуитивно понятны, однако может потребоваться некоторое время, чтобы разобраться во всех возможностях инструмента [1]. Этот программный продукт не задумывался разработчиками для описания онтологии и ее кодогенерации, но, благодаря его обширным возможностям и множеству языков описания систем, его можно использовать для достиения высокого результата описания предметной области. На данный момент поддерживаются все 14 типов диаграмм, указанных в UML 2.5, а также ER-диаграммы, блок-схемы, объектно-ролевое моделирование, диаграммы потоков данных и др.

Используя предоставляемый функционал, можно описать предполагаемые объекты системы, их атрибуты, связи между ними, а также взаимосвязи между собой, что, в свою очередь, является главной задачей в построении онтологий. Также приложение позволяет производить кодогенерацию на основе построенных диаграмм, при этом поддерживается большинство современных высокоуровневых языков программирования, стандарты XML и SQL. Это позволяет хранить базу знаний предметной области в виде программного кода или в отдельных структурах.

Пример кодогенерации представлен на Рисунке 1.

```
class Admin :  public User
{
    private:
        string telephone;
        string nameOfAdmin;

    public:
        void createObj ();
        void delObj ();
        void addTeacherl ();
        void delTeacher ();
        void addPrice ();

};
```

Рисунок 1. Пример кодогенерации средствами Software Ideas Modeler

В результате рассмотренных аналогов можно выделить несколько их главных преимуществ: открытый исходный код, интерес к данным проектам сообщества разработчиков и постоянное развитие проектов. Однако почти все они не позволяют производить процесс кодогенерации по составленным онтологиям, так как требуют использования для этого дополнительных программных средств.

В связи с этим целью стала разработка программного обеспечения, которое позволит составить онтологию предметной области и также произвести ее кодогенерацию, что, в свою очередь, позволит сократить трудозатраты на интеграцию классов и ограничений в существующую информационную систему.

Стек технологий

Для реализации программного обеспечения был выбран язык программирования Python. На данный момент это один из самых популярных и востребованных языков программирования в мире за счет большого набора полезных переносимых функций, начиная от функционала для работы с текстом и заканчивая средствами для написания сетевых приложений [8; 9].

Хранить разрабатываемую средствами программного обеспечения онтологию, ее объекты и параметры было решено в XML-формате, для работы с которым выбрана библиотека Python xmlschema. Она может выполнять кодирование, декодирование файлов XML и позволяет производить проверку данных XML на валидность [10].

Для создания пользовательского интерфейса использовалась библиотека PySide2, которая обладает множеством готовых элементов взаимодействия пользователя с программой: кнопки, поля для ввода данных, заголовки с текстом, вкладки [3]. Для визуализации онтологии и представления ее в виде графов используется библиотека PyQtGraph [2].

Архитектура приложения

Разработанное приложение состоит из 6 модулей:

- MainWindow описывает логику поведения главного окна приложения;
- Manager Data отвечает за хранение данных об онтологии, ее классах, связях и ограничениях;
- Widgets хранит окна интерфейса приложения;
- Flow Chart используется для визуализации спроектированной онтологии в виде графов;
- Utils содержит в себе вспомогательные функции, такие как проверка данных на валидацию и реализацию паттернов проектирования;
- Code Generation отвечает за формирования кода на языке Python на основе составленной онтологии.

На Рисунке 2 представлена структурная схема взаимодействия модулей.

Такая структура позволяет каждому модулю выполнять свои задачи независимо от других, что делает возможным их динамическую замену, а также добавление новой функциональности без изменения кода других модулей.

Процесс построения онтологий

В настоящее время существует множество стандартов для формирования онтологии предметной области, например: ER-модель, RDF, OWL, XML-схема и др.

В работе [4] подробно рассмотрены существующие способы формирования онтологии предметной области при моделировании. Отмечена целесообразность использования XML-схемы в задачах кодогенерации разрабатываемой онтологии, так как данный формат обладает рядом преимуществ, а именно: избежание лишних данных за счет древовидной структуры, ограничения на входные данные, отсутствие ограничений на типы данных.

Пользовательский интерфейс программного обеспечения позволяет производить описание сущностей онтологий, их характеристик, а также взаимодействий. Пользователю доступно создание множества объектов онтологии с подробным описанием характеристик каждого. Также он может создавать общие интерфейсы, включающие в себя группу абстрактных методов для назначения на общие объекты в онтологии. В качестве характеристик используются следующие параметры:

• название – позволяет задать название объекта онтологии;

• описание – позволяет пользователю задать описание объекта для его лучшего восприятия;

• состояния – возможные состояния объекта (например, при создании объекта «машина» она будет иметь состояния «едет» и «стоит»);

• характеристики – параметры объекта (например, цвет машины);

• ограничения – описание ограничений объекта (например, параметр скорости машины не может быть выше 110 км/ч);

• взаимодействия – позволяет описывать, с кем объект может взаимодействовать (например, человек взаимодействует с машиной);

• методы – позволяют задавать описание методов и их сигнатуры, которые должен реализовывать объект системы;

• реализуемые интерфейсы – позволяют задавать перечисление интерфейсов, которые объект должен реализовывать.

Интерфейс приложения и процесс описания актора системы представлен на Рисунках 3, 4.

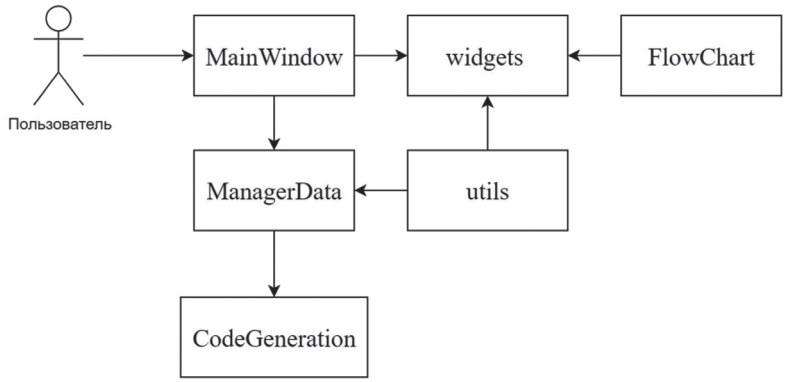

Рисунок 2. Структурная схема взаимодействия модулей

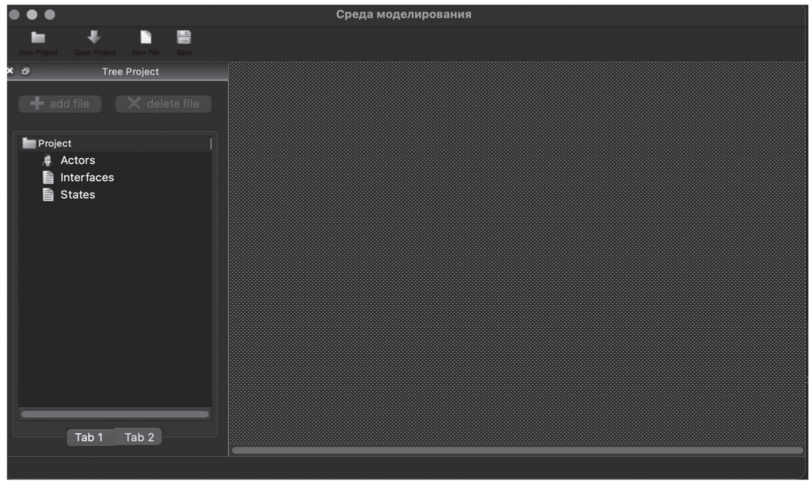

Рисунок 3. Общий вид интерфейса

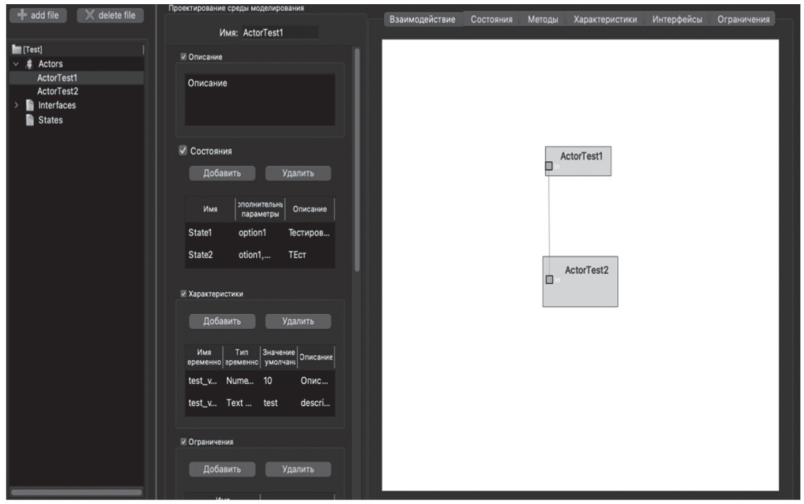

Рисунок 4. Создание актора системы

Разработанное программное обеспечение позволяет в полном объеме произвести описание онтологии с последующим формированием на ее основе XML-схемы.

На Рисунке 5 приведена XML-схема, в которой происходит описание транспортного средства.

```
<Actor name="ladaGranta">
<Description>The object of the ontology machine</Description>
<States>
<State name="stand" extra_options="test" description="car stands"
/>
<State name="drive" extra_options="test" description="the car
drive" />
</States>
<Characteristics>
<Characteristic name="Color" type="Text Sequence Type"
default="Red" description="Color car" />
</Characteristics>
<Interactions />
<Constrains>
<Constrain name="Color" description="cannot be yellow" />
</Constrains>
<Methods>
<Method name="drive" options="speed, diraction" description="car
drive" />
</Methods>
<Interfaces />
</Actor>
```

Рисунок 5. XML-схема с описанием транспортного средства

Процесс кодогенерации

После получения XML-схемы пользователь может произвести процесс кодогенерации на язык программирования Python, который заключается в поочередном анализе метаданных каждого объекта. На каждой итерации кодогенерирования формируются части сгенерированного кода для каждого из его параметров. После того, как они будут сформированы, в зависимости от указанных взаимодействий объектов в системе и их реализуемых интерфейсов запускается сборка генерации итого кода.

На Рисунке 6 представлен результат кодогенерации XML-схемы, рассмотренный в предыдущем пункте.

```
from abc import ABC

class ladaGranta:
  """The object of the ontology machine"""

  def __init__(self):
self.Color = 'Red'  # Color car

  @property
  def Color(self):
     return self.Color

  @Color.setter
  def Color(self, value):
     """cannot be yellow"""
     pass

  def drive(self, speed, diraction, *args, **kwarg):
     """car drive"""
     pass

class State(ABC):
  @property
  def stand(self):
     """car stands"""
     return self._stand

  @stand.setter
  def stand(self, test):
     pass

  @property
  def drive(self):
     """the car drive"""
     return self._drive

  @drive.setter
  def drive(self, test):
pass
```

Рисунок 6. Результат кодогенерации XML-схемы

Полученный код сохраняет в себе все параметры и ограничения, указанные пользователем при графическом моделировании онтологии и может служить основой будущей программной системы.

Заключение

Разработанное программное обеспечение позволяет описывать онтологии предметной области и производить ее кодогенерацию в высокоуровневый код. Это дает возможность на этапе проектирования системы заложить будущую архитектуру приложения, которое содержит все необходимые знания о предметной области.

Благодаря такому подходу ускоряется процесс создания информационных систем, а графический интерфейс дает возможность в удобном для человека виде производить описание и формирование онтологии предметной области. Полученный код после генерации обладает одним архитектурным стандартом, что позволяет поддерживать читаемость и понятность кода с ростом переносимых в код объектов предметной области.

На разработанное программное обеспечение получено свидетельство о государственной регистрации программы для ЭВМ [7].

Литература

1. *Бедердинова О.И., Кремлева Л.В., Протасова С.В.* Моделирование информационных систем на платформе SOFTWARE IDEAS MODELER. М.: Инфра-М, 2019.

2. Библиотека PyQtGraph. Официальный сайт [Электронный ресурс]. URL: https://www.pyqtgraph.org/ дата обращения: 25 марта 2022 г.).

3. Библиотека PySide2. Материалы с сайта [Электронный ресурс]. URL: https://pypi.org/project/PySide2/ (дата обращения: 25 марта 2022 г.).

4. *Быков А.Н., Антонов А.А., Чернышев С.А.* Обзор существующих способ формирования онтологии предметной области при моделировании // Международный журнал информационных технологий и энергоэффективности. 2021. Т. 6, № 4 (22). С. 12–17.

5. Программа NeOn Toolkit. Официальный сайт [Электронный ресурс]. URL: http://neon-toolkit.org/ (дата обращения: 21 марта 2022 г.).

6. Программа Protege. Официальный сайт [Электронный ресурс]. URL: https://protege.stanford.edu/ (дата обращения: 17 марта 2022 г.).

7. Свидетельство о государственной регистрации программы для ЭВМ № 2021660091 от 22.06.2021 г. Программное обеспечение формирования онтологии предметной области. Основное программное обеспечение / А.Н. Быков, М.Э. Нагорных, С.А. Чернышев.

8. *Lamy Jean-Baptiste*. Ontologies with Python. Apress, 2021.

9. Python Documentation. Материалы с сайта [Электронный ресурс]. URL: https://www.python.org/doc/ (дата обращения: 25 марта 2022 г.).

10. XML Documentation. Материалы с сайта [Электронный ресурс]. URL: https://www.w3.org/TR/xmlschema-0/ (дата обращения: 25 марта 2022 г.).

References

1. Bederdinova O.I., Kremleva L.V., Protasova S.V. (2019) Modelirovanie informacionnyh system na platforme SOFTWARE IDEAS MODELER [Modeling of information systems on the SOFTWARE IDEAS MODELER platform]. Moscow, INFRA-M Publishing, 2019 (in Russian).

2. Library PyQtGraph. Available at: https://www.pyqtgraph.org/ (accessed: 25 March 2022) (in Russian).

3. Library PySide2. Available at: https://pypi.org/project/PySide2/ (accessed: 25 March 2022) (in Russian).

4. Bykov A.N., Antonov A.A., Chernyshev S.A. (2021) [Review of the existing method of forming the ontology of the subject area in modeling]. *Mezhdunarodnyj zhurnal informacionnyh tekhnologij i energoeffektivnosti*, vol. 6, No. 4 (22), pp. 12–17 (in Russian).
5. Programma NeOn Toolkit. Available at: http://neon-toolkit.org/ (accessed: 21 March 2022) (in Russian).
6. Programma Protege. Available at: https://protege.stanford.edu/ (accessed: 17 March 2022) (in Russian).
7. Bykov A.N., Nagornyh M.E., Chernyshev S.A. (2021) *Svidetel'stvo o gosudarstvennojregistraciiprogr ammydlya EVM № 2021660091. Programmnoe obespechenie formirovaniya ontologii predmetnoj oblasti. Osnovnoe programmnoe obespechenie* [Certificate of state registration of the computer program No. 2021660091 dated June 22, 2021. Software for the formation of the ontology of the subject area. Main software] (in Russian).
8. Lamy Jean-Baptiste. (2021) Ontologies with Python. Apress.
9. Python Documentation. Available at: https://www.python.org/doc/ (accessed: 25 March 2022) (in Russian).
10. XML Documentation. Available at: https://www.w3.org/TR/xmlschema-0/ (accessed: 25 March 2022) (in Russian).