ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ

Гвоздевский И. Н., аспирант
Белгородский государственный технологический университет им. В.Г. Шухова

ИСПОЛЬЗОВАНИЕ ОНТОЛОГИЙ ДЛЯ УНИФИКАЦИИ ВЗАИМОДЕЙСТВИЯ КОМПОНЕНТОВ АВТОМАТИЗИРОВАННЫХ СИСТЕМ ДИСПЕТЧЕРСКОГО УПРАВЛЕНИЯ

igorek@intbel.ru

Многообразие программных и аппаратных элементов в современных автоматизированных системах диспетчерского управления усложняют процесс интеграции их в единую коммуникационную среду. Различное оборудование, протоколы, службы прикладных уровней современных систем диспетчеризации имеют возможность взаимодействия в рамках одной отдельно взятой подсистемы комплекса в формате вертикального межуровневого общения. Обмен между подсистемами происходит с помощью специализированного прикладного интерфейса или коннектора, что влияет на качество получаемых данных и в результате снижает показатели функционирования автоматизированной системы диспетчерского управления.

Совершенствование механизма обеспечения коммуникации между элементами подсистем используя онтологический подход, позволяет сформировать унифицированную среду. В которой возможно применение различных методов сбора информации, анализа, структурирования, а также методов принятия решений о функционировании и восстановлении работоспособности на основе интеллектуальных агентов, повышающих уровень работы автоматизированной системы.

Ключевые слова: *агентный онтологический поход, онтология, распределенные вычислительные системы, автоматизированная система диспетчерского управления.*

Введение. Применение различных технологий при построении автоматизированной системы диспетчерского управления БГТУ позволяет использовать многообразие инструментов для решения задачи программного контроля различных технологических систем [2]. Одним из недостатков данного подхода является использование центральной системы прикладного уровня для сбора, анализа и принятия управленческих решений о функционировании среды, которая имеет особенность увеличения времени реакции на возникающий запрос обслуживания. Применение онтологического подхода с использованием интеллектуальных агентов [3], позволяет существенно повысить уровень обслуживания компонентов автоматизированной системы диспетчерского управления и сформировать структурированное хранилище информации о элементах, режимах функционирования среды.

Методология. Применение в существующей АСДУ подхода по интеграции компонентов различных технологических подсистем без привязки к конкретному программному решению позволяет хранить информацию об объектах среды в унифицированном онтологическом виде. В данном случае распределенная агентная сеть, работающая в рамках всего комплекса может использовать данный вид структурированных знаний для решения задач мониторинга, анализа, интеллектуального принятия решений по оптимизации и восстановлению работоспособности элементов.

Основная часть. Внедрение агентной структуры в существующую автоматизированную систему диспетчерского управления создает предпосылки для хранения информации генерируемой системой или интеллектуальными агентами в структурированном виде. Так как на различных уровнях существующего объекта работают различные по своей специфике элементы, необходимо провести классификацию структур, обеспечивающих сбор, анализ, хранение и обработку информации на каждом базовом сегменте [1].

- агенты управления – объединяют нижний технологический слой функционирования автоматизированной системы;
- агенты обеспечения взаимодействия – являются интеграционным элементом между нижним технологическим уровнем, обеспечивают унификацию различных протоколов связи в единую среду;
- агенты УМПР – представляют надстройку над прикладным SCADA-уровнем, позволяют обеспечивать управление, мониторинг и приня-

тие решений над агентами нижестоящих уровней.

Вопрос использования онтологий для эффективной организации взаимодействия подсистем АСДУ [2] возможен только при условии создания единого информационного пространства. Исследования показывают, что спроектированная модель информационного пространства должна включать в себя следующие элементы:

- набор словарей (тезаурусов), по соответствующей предметной области;
- базы знаний и данных различных подсистем;
- комплекс специализированной документации по элементам АСДУ в электронном виде.

Будем рассматривать онтологию в виде формального описания концептов, их свойств и отношений между объектами. Фактически вопрос эффективного использования данного метода сводится к решению задачи более точного описания предметной области, разработки методов по структурированию получаемой от объектов комплекса информации, а также наиболее сложного вопроса расширения и модификации, текущих объектов онтологий[7]. Предполагается, что разрабатываемая надстройка над существующими модулями управления АСДУ в виде агентной платформы, может быть использована для:

- обеспечения информационного взаимодействия подсистем АСДУ;
- накопления и структуризации данных для повторного использования;
- анализа получаемой информации о функционировании всей единой среды АСДУ;
- обеспечения реализации моделей интеллектуального поведения агентов на основе онтологических представлений.

В результате исследований была спроектирована и реализована межуровневая концепция хранения знаний о системе в структурированном виде. Выделенные уровни функционирования системы представляют собой наиболее законченный набор однородных объектов, что позволяет обеспечивать модернизацию АСДУ на каждом уровне за счет расширения элементной базы. На каждом из трех уровней(рис. 1) была разработана модель онтологии, включающая в себя унифицированное описание элементов подсистем.

Рис. 1. Модель онтологии аппаратного и транзитного уровня применяемой в АСДУ

Применение распределенной агентной сети позволяет использовать разработанную модель для обеспечения интеллектуальных агентов комплексом эталонных знаний о внешней среде функционирования автоматизированной системы [2], а также оперативной модели знаний, обеспечивающей стабильную работу всех элементов и подсистем. Причем разработанная он-

тология позволяет использовать такие виды агентов, как делиберативные, коммуникативные и когнитивные [3] не изменяя структуры знаний в целом, реализация взаимодействия конкретного агента с базой знаний будет осуществлена за счет использования определённой программной надстройки.

Определение онтология, часто трактуют, как «спецификация разделяемой разными людьми концептуализации» или «логическая теория, основанная на концептуализации» [4]. Исходя из этих определений в структуру онтологии обычно входят тезаурусы, предметной области, определения, атрибуты, отношения и правила вывода. Для создания унифицированного объекта, позволяющего более точно описать предметную область решаемой задачи к каждой разработанной модели предъявлялись, требования обеспечивающие полноту данных онтологий. Была разработана схема (рис. 2), позволяющая реализовать более полнофункциональные модели онтологий.

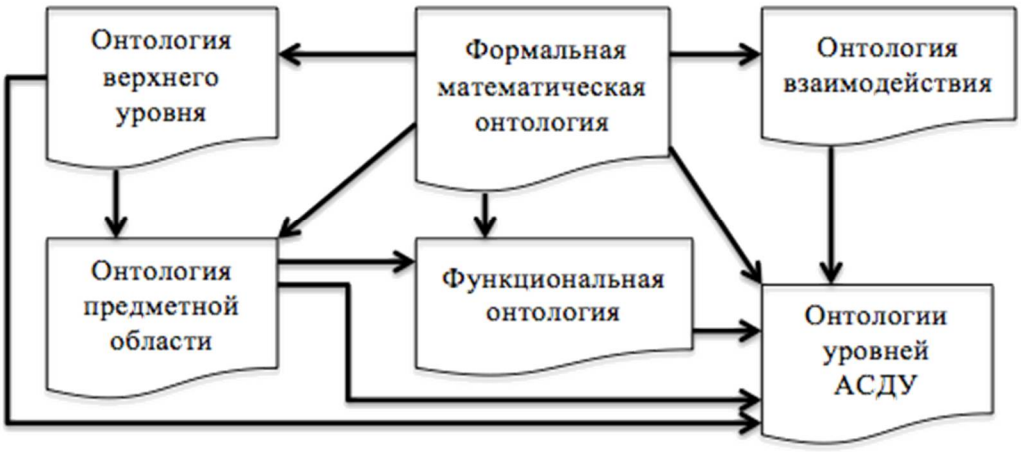

Рис. 2. Схема интеграции онтологий для прикладного взаимодействия

Данная схема позволила при проектировании структур использовать верхний описательный уровень в виде формальной математической спецификации онтологии, которая обеспечивает совместно с точным математическим аппаратом [5], создаваемых структур, также формальный анализ свойств объектов в них входящих.

При использовании разработанных моделей для обеспечения функционирования распределенной вычислительной сети, были рассмотрены основные способы хранения и взаимодействия с онтологиями [7], которые должны обеспечивать высокие показатели эффективности и производительности. Для хранения подобных структур необходимо кроме, объекта информации иметь возможность сопоставлять данные и набор метаданных, описывающих классы и их свойства, а также взаимосвязи между ними. В настоящее время существует несколько направлений по реализации онтологических хранилищ [8]:

- в виде плоской модели;
- отображение онтологии на SQL-подобную модель баз данных;
- прикладное реляционное хранилище rdf-триплетов, owl-репозиторий.

Можно отметить следующие разработки: Jena API [11] (Jena SDB использует реляционную базу данных для хранения и выполнения запросов на RDF хранилищах, Jena TDB обеспечивает масштабируемое нетранзакционное хранилище триплетов и слой SPARQL запросов, Jena In-memory предоставляет возможность размещения структуры хранилища в памяти), Ontotext GraphDB (OWLim) – коммерческое программное обеспечение реализующее хранилище RDF СУБД с полной поддержкой SPARQL запросов [14], Sesame свободная разработка в виде автономного хранилища предоставляющего полную поддержку запросов SPARQL, NoSQL Graph Support – версия IBM DB2 для хранения онтологий, Urika GD – продукт компании Cray, обеспечивающий поддержку онтологических структур огромных массивов данных.

Для взаимодействия с разработанной моделью был проведен эксперимент, в котором предварительно сформированная структура онтологии одного из уровней функционирования модели автоматизированной системы диспетчерского управления была загружена в соответствующие типы онтологических хранилищ. Для эксперимента были использованы: связка в виде Jena версии 3.0.1, Jena Fuseki версии 2.3.1 и версии MySQL 5.6, как основная СУБД (Jena TDB, Jena SDB, In-mem) [12], Sesame версии 4.0.1, GraphDB (OWLim) версии 6.5 и плоская модель хранилища в виде файлового объекта OWL. Тестовый стенд состоит из среды Intel Core i5 4x2,5GHz/20Gb 1333MHz DDR3/HDD 500Gb на базе Ubuntu server 15.10.

На первом этапе эксперимента был рассмотрен процесс загрузки определенного количества триплетов в хранилище, что позволило оценить временные показатели скорости инициализации данных в объекте для работы (табл. 1).

Таблица 1

Тестирование скорости загрузки триплетов

Количество загруженных триплетов в модель	Скорость выполнения, тип хранения. сек						
	Плоская	Jena TDB	Jena SDB	In-mem	Sesame	GraphDB	MySQL
10K	0,54	0,24	2,89	0,75	2,35	2,46	0,011
50K	3,26	1,17	7,24	1,58	6,34	7,21	0,246
100K	6,19	4,45	14,35	7,02	15,32	14,99	0,573
250K	13,24	12,67	35,17	13,31	40,12	37,15	1,341
500K	30,01	24,75	68,87	25,17	85,69	81,14	3,027
1M	61,83	49,11	129,33	51,25	179,36	154,22	6,931
5M	341,55	272,56	684,11	289,16	1283,77	789,11	37,21

На втором этапе тестирования были проведены исследования характеристик выполнения основных запросов SPARQL. Созданы запросы, обеспечивающие тестовую выборку триплетов из структуры хранилищ, особое внимание было уделено конструкциям:

SELECT - предполагает выборку результатов подобно аналогичному оператору выбора в SQL нотации [14]. Имеет возможность возвращать определенные данные из соответствующим условию триплетов, причем выборки могут объединяться и не указанные элементы, переменные могут удаляться. Для выполнения тестирования был сформирован следующий запрос:

SELECT ?basic_property ?value WHERE {:log_data_cases ?basic_property ?value} (1)

При выполнения запроса SELECT SPARQL, была осуществлена выборка данных из загруженной модели соответствующего RDF хранилища. В таблице (табл. 2). указано время осуществления выборки на разных объемах количества загруженных в структуру триплетов.

Таблица 2

Тестирование скорости выполнения запросов SPARQL SELECT

Количество загруженных триплетов в модель	Скорость выполнения запроса SELECT, тип хранения. сек					
	Плоская	Jena TDB	Jena SDB	In-mem	Sesame	GraphDB
500	0,053	0,028	0,073	0,065	0,007	0,013
1500	0,091	0,035	0,081	0,078	0,015	0,024
3000	0,161	0,043	0,091	0,951	0,023	0,037
10000	0,411	0,123	0,437	0,549	0,069	0,101
20000	0,763	0,354	0,752	0,799	0,165	0,235

Для реализации проверки наличия элементов в структуре хранилища RDF, причем результат возвращается в виде логического значения ложь или истина, был сформирован запрос с использованием оператора ASK:

ASK WHERE { UNSAID {:health_status :critical_error ?any} (2)

В результате выполнения запроса, была осуществлена выборка состояний системы health_status, принимающих значения critical_error и получены следующие результаты (табл. 3).

Таблица 3

Тестирование скорости выполнения запросов SPARQL ASK

Количество загруженных триплетов в модель	Скорость выполнения запроса ASK, тип хранения. мсек					
	Плоская	Jena TDB	Jena SDB	In-mem	Sesame	GraphDB
500	0,039	0,021	0,043	0,040	0,005	0,006
1500	0,068	0,026	0,051	0,051	0,012	0,013
3000	0,125	0,035	0,067	0,602	0,018	0,023
10000	0,312	0,098	0,297	0,356	0,051	0,060
20000	0,569	0,298	0,532	0,504	0,124	0,141

Использование оператора CONSTRUCT, позволяет сформировать структуру из полученной выборки RDF согласно параметрам, встроенным в запрос конструкции шаблона. Для каждого результата оператор связывает переменные и добавляет состояния в модель результата. Для тестирования был создан запрос SPARQL:

```
CONSTRUCT { ?x :hasSystemType ?o }
WHERE {{ ?x :hasSerialNum ?o }
UNION { ?x :hasIPaddrees ?o }
UNION { ?o :hasDevID ?x }
UNION { ?o :hasMode ?x }}
```
(3)

В результате выполнения которого была получена конструкция, определяющие параметра работы группы устройств имеющих определенные признаки, время выполнения для запроса CONSTRUCT (табл. 4).

Таблица 4

Тестирование скорости выполнения запросов SPARQL CONSTRUCT

Количество загруженных триплетов в модель	Скорость выполнения запроса CONSTRUCT, тип хранения. сек					
	Плоская	Jena TDB	Jena SDB	In-mem	Sesame	GraphDB
500	11,34	6,93	8,23	7,53	2,26	2,75
1500	29,67	22,85	27,68	25,13	6,87	7,28
3000	62,35	42,47	51,24	48,91	12,85	12,91
10000	181,28	152,35	173,31	169,33	44,61	45,13
20000	395,44	277,2	346,62	324,19	89,22	94,72

Оператор DESCRIBE позволяет делать выборку значений, находя ассоциированные триплеты и добавлять ее к результирующей модели. Обеспечивает получение большего количества параметров при запросе, чем другие операторы[14].

```
PREFIX system: <http://127.0.0.1/transit_level#>
DESCRIBE ?dev_id WHERE { ?dev_id system:status "critical_error" }
```
(4)

Результаты выполнения запроса, осуществляющего получение информации о критическом состоянии устройств, разработанной модели представлены в таблице (табл. 5).

Таблица 5

Тестирование скорости выполнения запросов SPARQL DESCRIBE

Количество загруженных триплетов в модель	Скорость выполнения запроса DESCRIBE, тип хранения. мсек					
	Плоская	Jena TDB	Jena SDB	In-mem	Sesame	OWLim
500	4,660	2,811	3,255	2,979	0,991	1,074
1500	12,192	9,270	10,949	9,940	3,013	2,844
3000	25,620	17,230	20,268	19,347	5,636	5,043
10000	74,490	61,808	68,553	66,979	19,565	17,629
20000	162,49	112,45	137,11	128,23	39,13	37,00

Выводы. Предложенная в работе модель хранения данных в виде онтологий показывает высокую эффективность при применении на реальной автоматизированной системе диспетчерского управления. Мультиагентная распределенная вычислительная сеть, используя данные наборы структурированной информации, позволяет без вмешательства оператора или вышестоящей прикладной SCADA-системы решать задачи поддержания высокого уровня коммуникации между элементами различных подсистем АСДУ [2]. Применение онтологических хранилищ и формирование моделей данных в предложенном виде показывает хороший уровень показателей быстродействия, кроме этого качественные показатели обработки знаний через интерфейсы SPARQL превосходят стандартные механизмы реляционных баз данных. Формирование эталонной модели функционирования, а также возможность агентов осуществлять свое развитие за счет анализа внешней среды или подключаемых источников знаний, увеличивает показатели отказоустойчивости и скорости реакции на возникающие инциденты. Использование онтологических моделей совместно с инструментами логического вывода позволяет находить новые знания в масштабах модели, а также применять методы интеллектуального анализа.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Поляков В.М., Буханов Д.Г., Синюк В.Г. Базовые структурные модели распределенных вычислительных систем в многоагентной диагностической среде. // Журнал «Информационные системы и технологии». 2012. №4(72). С. 52–56

2. Белоусов А.В., Глаголев С.Н., Быстров А.Б., Кошлич Ю.А. Демонстрационная зона по

энергосбережению БГТУ им. В.Г. Шухова – база для развития энергоэффективных проектов в регионе // Энергосбережение. Энергетика. Энергоаудит. 2013. № 10 (116). С. 10–17.

3. Михайлов Н.В., Поляков В.М. Расширение функциональных возможностей банковских информационных систем с применением технологий интеллектуальных агентов. // Вестник Белгородского государственного технологического университета им. В.Г. Шухова. 2011. № 3. С. 159–161.

4. Семерханов И.А., Муромцев Д.И. Интеграция информационных систем на основе связанных данных // Научно-технический вестник информационных технологий, механики и оптики. 2013. № 5 (87). С. 123–127.

5. Корзун Д.Ж., Ломов А.А., Ванаг П.И. Автоматизированная модельно-ориентированная разработка программных агентов для интеллектуальных пространств на платформе SMART // Программная инженерия. 2012. № 5. С. 6–14.

6. Слободюк А.А., Маторин С.И., Четвериков С.Н. О подходе к созданию онтологий на основе системно-объектных моделей предметной области // Научные ведомости Белгородского государственного университета. Серия: Экономика. Информатика. 2013. Т. 28. № 22-1 (165). С. 186–194.

7. Муромцев Д.И., Варгин Г.В., Семерханов И.А. Применение онтологий в системе управления интеллектуальными ресурсами // Научно-технический вестник информационных технологий, механики и оптики. 2011. № 2 (72). С. 170.

8. Wylot M., Cudré-Mauroux P., Groth P. TripleProv: Efficient processing of lineage queries a native RDF store // В сборнике: WWW 2014 - Proceedings of the 23rd International Conference on World Wide Web 23. 2014. С. 455–465.

9. Amann B., Fundulaki I., Scholl M. Integrating ontologies and thesauri for RDF schema creation and metadata querying // International Journal on Digital Libraries. 2000. Т. 3. № 3. С. 221–236.

10.Филатов В.А. Мультиагентные технологии интеграции гетерогенных информационных систем и распределенных баз данных : Дис. д-ра техн. наук: 05.13.06 // ХНУР. Х., 2004. 341с. с. 313–336.

11.Использование онтологий в системах на базе Jena [Электронный ресурс]. Систем. требования: Веб-браузер. URL: http://jena.apache.org/documentation/ontology/ (дата обращения: 11.02.2016).

12.Спецификация языка RDF. W3C: Resource Description Framework (RDF). [Электронный ресурс]. Систем. требования: Веб-браузер. URL: http://www.w3.org/RDF/ (дата обращения: 23.01.2016).

13.Использование языка структурированных запросов SPARQL в системах на базе Jena [Электронный ресурс]. Систем. требования: Веб-браузер. URL: http://jena.apache.org/tutorials/sparql.html (дата обращения: 11.02.2016).

14.Рекомендации по работе с языком запросов SPARQL ver 1.1 [Электронный ресурс]. Систем. требования: Веб-браузер. URL: https://www.w3.org/TR/2013/REC-sparql11-query-20130321/ (дата обращения: 11.02.2016).

Gvozdevskiy I.N

USING ONTOLOGIES FOR THE UNIFICATION OF INTERACTION OF COMPONENTS AUTOMATED SUPERVISORY SYSTEMS

A variety of hardware and software elements in a modern automated systems of dispatching management complicate the process of their integration into a unified communications environment. Miscellaneous equipment, protocols, application service levels modern dispatching systems have the ability to interact within a single subsystem of the complex in the form of vertical cross-layer communication. Exchange between subsystems is done by specialized application interface connector and that affects the quality of the data and as a result reduces the performance of the automated dispatch control system. Improving the mechanism for communication between the elements of the sub-systems using an ontological approach allows you to create a unified environment. In which it is possible to use different methods of data collection, analysis, structuring, as well as methods of decision-making on the operation and restoration of health, based on intelligent agents that increase the level of the automated system.

Key words: *agent-based campaign ontological, ontology, distributed computing systems, automated dispatch control system*

Гвоздевский Игорь Николаевич, аспирант кафедры программного обеспечения и автоматизированных систем.

Белгородский государственный технологический университет им. В.Г. Шухова.

Адрес: Россия, 308012, Белгород, ул. Костюкова, д. 46.

E-mail: igorek@intbel.ru