САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

*На правах рукописи*

**ЛЕЩЕВА Ирина Анатольевна**

# Метод автоматизированного наполнения баз знаний онтологического типа

Специальность: 2.3.5. Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

*Диссертация на соискание ученой степени кандидата технических наук*

Научный руководитель:
д-р техн. наук, профессор
Гаврилова Татьяна Альбертовна

Санкт-Петербург
2022

# ОГЛАВЛЕНИЕ

# ВВЕДЕНИЕ

**Актуальность темы.** Эффективность принятия управленческих решений зависит, в частности, от полноты и непротиворечивости имеющейся информации, а также от возможности гибкой ее обработки, в том числе на семантическом уровне. Применение корпоративных онтологий и баз знаний онтологического типа на предприятиях и в организациях создает потенциал для значительного повышения качества информационной поддержки и эффективности управления, обеспечивает сохранение интеллектуальных активов компании и поддержку важнейших бизнес-процессов.

При помощи корпоративных онтологий можно описывать не только бизнес-процессы, но и продукты, документы, компетенции, технологии, функции, стратегии, финансовые потоки и пр. Это инструмент, ориентированный на интероперабельность, под которой понимается возможность использования на различных уровнях иерархии, в разных подразделениях, на разных технических и программных платформах специалистами разной квалификации. Сумма корпоративных онтологий служит универсальным каркасом знаний организации, и одновременно мостом для понимания и трансфера знаний и технологий.

Но в настоящий момент слаба взаимосвязь между потребностями предприятий и организаций и существующими технологиями в области инженерии знаний и онтологического инжиниринга. Методы и инструменты работы с корпоративными знаниями пока недостаточно зрелы для решения практических задач управления знаниями и информационного менеджмента. В результате предприятия и организации применяют устаревшие и неэффективные технологии. Нарастающий интерес к вопросам инженерии знаний тормозится сложностью разработки практически-направленных онтологий и их интеграции с накопленными массивами данных различной структуры.

Значительный вклад в разработку и исследование моделей, методов и средств создания интеллектуальных систем (включая онтологии, базы знаний и программные средства для их разработки) внесли Грубер Т., Гуарино Н., Ленат Д., Осипов Г.С., Поспелов Д.А., Гаврилова Т.А., Голенков В.В., Грибова В.В., Загорулько Ю.А., Клещев А.С., Кудрявцев Д.В., Массель Л.В., Тузовский А. Ф., Хорошевский В.Ф., Штудер Р. и др. [Гаврилова, Кудрявцев, Муромцев, 2020; Грибова В. В. и др., 2018; Голенков и др., 2019; Грибова В. В. и др., 2018; Gruber, 1995; Giaretta, Guarino, 1995; Zagorulko, Borovikova, Zagorulko, 2021; Клещев, Москаленко, Черняховская, 2005; Гаврилова, Хорошевский, 2000; Lenat, Guha, 1991; Ерженин, Массель, 2020; Осипов, 2011; Тузовский, 2007; Studer R. et al., 2004].

Одной из частей разработки онтологии является наполнение онтологии индивидами и отношениями, которое тем или иным образом затрагивает все этапы жизненного цикла базы

знаний, но наибольшее влияние оказывает на этапы разработки и поддержки в процессе использования. При этом именно наполнение онтологий остается одной из нерешенных задач онтологического инжиниринга. Существующие методы наполнения баз знаний, основанных на онтологиях, обладают рядом недостатков, в частности, узкой направленностью на решение задач конкретного проекта, сложностью описания моделей источников данных и их трансформации, высокими требованиями к квалификации пользователей, обеспечением возможности интеграции данных из источников только одного типа, отсутствием модульности и расширяемости, а также удобного инструментария. Это определяет актуальность создания новых методов автоматизированного наполнения и обогащения онтологий или баз знаний онтологического типа путем консолидации, что позволит снизить трудоемкость наполнения и обогащения онтологий, используя накопленные массивы информации независимо от формы их хранения и представления.

**Целью диссертационной работы** является разработка автоматизированного метода и алгоритмов интеграции разнородных источников структурированных данных путем консолидации для наполнения баз знаний онтологического типа.

Для достижения поставленной цели в диссертационной работе поставлены и решены следующие **задачи**:

- Построить генеалогия языков представления знаний на основе онтологий.
- Провести исследование когнитивных аспектов структурирования знаний с использованием онтологической модели представления знаний.
- Разработать универсальный метод автоматизированной интеграции разнородных источников структурированных данных путем консолидации для наполнения баз знаний онтологического типа.
- Разработать правила отображения, используемые методом, только средствами онтологий без привлечения дополнительных языков со своим синтаксисом.
- Разработать шаблоны построения правил отображения исходной онтологии в базу знаний предметной области.
- Сформулировать критерии выбора программной платформы для реализации предложенного метода.
- Разработать архитектуру и реализовать процедуру консолидации данных, хранящихся в структурированных источниках данных, в базу знаний предметной области, апробировать на реальных БЗ.

Данные задачи являются **положениями, выносимыми на защиту.**

**Объектом исследования** являются базы знаний онтологического типа.

**Предметом исследования** являются модели, методы и алгоритмы наполнения и обогащения баз знаний, основанных на онтологиях.

**Методы исследования.** В работе использовались методы системного анализа и инженерии знаний, трансформации моделей, а также методы и средства искусственного интеллекта и онтологического моделирования.

**Научная новизна**.

1. Расширена и доработана генеалогия языков представления знаний, предложенная ранее в [Казекин, 2008].
2. При проведении исследования влияния когнитивных стилей на создание онтологических моделей впервые были применены количественные методы, что позволило показать статистическую значимость найденных зависимостей между когнитивным типом человека и характеристиками построенной им онтологии [Gavrilova, Leshcheva, Ontology…, 2015; Гаврилова, Лещева, 2016; Гаврилова и др., 2013; Кудрявцев и др., 2019; Gavrilova и др., 2013; Gavrilova, Leshcheva, Building…, 2015; Gavrilova, Leshcheva, The interplay…, 2015, Leshcheva, Gavrilova, 2015].
3. Предложен новый универсальный алгоритм наполнения онтологий, лишенный большинства недостатков существующих решений интеграции данных распределенной природы в онтологии (см.выше) в силу предложенного алгоритма. В частности, предлагаемое решение является расширяемым для обеспечения потенциальной возможности работы с новыми форматами данных в будущем.
4. Впервые реализован процесс трансформации структур данных в онтологическую модель без использования дополнительных языков со своим синтаксисом, а только средствами онтологического моделирования, что позволяет исключить необходимость в дополнительной инструментальной поддержке метода.

Научное и практическое значение диссертации заключается в разработке нового метода анализа обработки информации, а именно метода консолидации данных в онтологию для их дальнейшей обработки и использования при принятии управленческих решений.

**Апробация работы.** Основные результаты диссертационной работы, её отдельные положения, а также результаты конкретных прикладных исследований и разработок были представлены более чем на 20 международных, всероссийских, региональных научных и научно-практических конференциях (см. список в конце).

**Публикации и личный вклад автора.** Результаты диссертационного исследования опубликованы в 36 печатных работах (в том числе 5 статей, рекомендованных ВАК для опубликования результатов диссертаций, 8 статей в журналах, индексируемых в Web of Science

и Scopus, 3 статьи в профильных российских журналах и 20 публикации в трудах международных и всероссийских конференций).

**Структура и объем диссертации.** Диссертационная работа состоит из введения, трех глав, заключения, списка литературы, включающего 127 наименований, и 1 приложения. Объем составляет 103 страницы основного текста, включая 29 рисунков, 1 таблицу.

**Содержание работы.** Во введении обосновывается актуальность диссертационной работы, формулируется цель и задачи исследования, определяется научная новизна и практическая значимость результатов.

В первой главе дается анализ современных подходов к созданию и наполнению онтологий и БЗ онтологического типа.

Во второй главе описаны метод METEOP (МЕтодология и ТЕхнология наполнения Онтологии на основе интегРации с гетерогенными структурированными источниками данных) и алгоритмы интеграции разнородных источников структурированных данных путем консолидации для наполнения баз знаний онтологического типа.

Новизна и основная идея предложенного метода METEOP заключается в использовании вспомогательной Онтологии источников данных (ОИД), которая может быть расширена за счет добавления дополнительных типов источников данных. Использование ОИД дополнено разработкой требований к формированию правил отображения, позволяющих связать ОИД и конкретную наполняемую онтологию предметной области (ОПО) для наполнения ее экземплярами. Предложенный метод лишен большинства недостатков, характерных для существующих решений интеграций данных в онтологии.

Применение разработанного метода можно описать алгоритмом, состоящим из 5 шагов, два из которых выполняются автоматически:

1. Идентификация источников данных.
2. Спецификация источников данных с помощью вспомогательной ОИД.
3. Извлечение структуры данных из источников данных в ОИД.
4. Задание правил отображения в ОПО.
5. Наполнение ОПО.

Ключевым шагом алгоритма наполнения ОПО является задание правил отображения. Оно осуществляется с помощью встроенного в OWL механизма аннотирования. В работе введены новые аннотационные свойства: 4 основных и 2 вспомогательных. Такой подход позволил исключить необходимость в дополнительной инструментальной поддержке метода METEOP, так как аннотирование может быть выполнено с помощью любого редактора онтологий.

Также была разработана новая архитектура программного прототипа, реализующего предложенный метод METEOP. В основу архитектуры положена известная архитектура ETL-

систем. Особенностью предлагаемой архитектуры является то, что получателем данных является не хранилище или база данных, а онтология или база знаний.

Процесс наполнения онтологии с помощью метода METEOR проиллюстрирован в диссертации на условном примере

В третьей главе обсуждаются особенности применения разработанного метода на примере создания и наполнения онтологий из 4 предметных областей:

a) Онтология для интеграции данных эмпирических исследований EMPIRION;
b) Онтология орфанных заболеваний Ontomorf;
c) Онтология сборочных единиц для автоматизированного производства автомобилей;
d) Онтология для администрирования учебной деятельности ВУЗа.

В заключении сформулированы основные научные результаты диссертационной работы.

Результаты исследований были получены в рамках выполнения исследований по проектам РФФИ:

- РФФИ № 17-07-00228 «МЕтодология и ТЕхнология формирования Онтологий на основе интегРации с гетерогенными источниками данных (METEOR)»;
- РФФИ № 20-07-00854 «Формирование баз знаний на основе данных эмпирических исследований: онтологический подход (ЭМПИРИОН)».

Некоторые предварительные результаты были получены в ходе выполнения проектов РФФИ 11-07-00140 2011-2013 «Структурирование знаний и КОнтента МЕтодами группового дизайна онТологий (КОМЕТ)» и 14-07-00294 «Интеллектуальные Сервисы поддержки ПОРТалов знаний на основе онтологий (ИнС-ПОРТ)».

# ГЛАВА 1 СОВРЕМЕННОЕ СОСТОЯНИЕ ИССЛЕДОВАНИЙ В ОБЛАСТИ ОНТОЛОГИЧЕСКОГО ИНЖИНИРИНГА

## 1.1 Проблемы разработки баз знаний онтологического типа

### 1.1.1 К вопросу об определении онтологии

Цифровая экономика диктует новые законы работы с информацией, и базы знаний (БЗ) из редких применений в интеллектуальных системах, становятся стандартом de facto корпоративных информационных систем и платформ [Тузовский, 2007]. Несмотря на известные методологии проектирования разработки БЗ [Stadnicki, Pietroń, Burek, 2020; Yunianta et al., 2019], существуют проблемы их практического формирования. Особенно остро эта проблема стоит для баз знаний на онтологиях, которые вот уже более 20 лет являются наиболее популярной моделью представления знаний.

Термин «онтология» пришел из философии, где он обозначает учение о бытии как таковом; раздел философии, изучающий фундаментальные принципы бытия, наиболее общие сущности и категории сущего [Доброхотов, 2010].

В конце XX века термин «онтология» стал использоваться в искусственном интеллекте, в частности, в инженерии знаний [Studer et al., 2004]. Позже онтологии стали рассматриваться в качестве основы для построения Семантической Сети — нового этапа развития сети WWW (Word Wide Web). [Berners-Lee, Hendler, Lassila, 2001].

Существует множество подходов к определению понятия «онтология». Одно из самых известных определений онтологии дал Том Грубер: «Онтология — это спецификация концептуализации» [Gruber, 1993]. Никола Гуарино определяет онтологию следующим образом: «Онтология — это формальная теория, ограничивающая возможные концептуализации мира» [Giaretta, Guarino, 1995]. В обеих формулировках используется понятие «концептуализация», требующее, в свою очередь, определения. Под «концептуализацией» понимается строгое описание системы понятий, объектов и других сущностей и отношений, связывающих их друг с другом. [Genesereth, Nilsson, 1987] Можно сказать, что концептуализация — это упрощенная модель мира, созданная для определенных целей с использованием системного подхода.

Можно привести более развернутое практически-ориентированное определение: «Онтология — это спецификация предметной области или формальное ее представление, которое включает словарь указателей на термины предметной области и логические выражения, которые описывают, что эти термины означают, как соотносятся друг с другом, и как они могут или не могут быть связаны между собой» [Гаврилова, Муромцев, 2007].

В работе [Гаврилова, Хорошевский, 2000] под формальной моделью онтологии понимается упорядоченная тройка вида:

$$O=\{X, \Re, \Phi\},$$

где $X$ — конечное множество концептов (понятий, терминов) предметной области, которую представляет онтология $O$;

$\Re$ — конечное множество отношений между концептами заданной предметной области;

$\Phi$ — конечное множество функций интерпретации (аксиоматизация), заданных на концептах и/или отношениях онтологии $O$.

В работе [Давидовский, 2011] это определение уточняется следующим образом:

«Онтология — это кортеж

$$O = \{C, R, I, T, V, \leq, \perp, \in, =\},$$

где

$C$ — множество концептов (или классов);

$R$ — множество отношений (объектные свойства и свойства типов данных);

$I$ — множество индивидов (или экземпляров);

$T$ — множество типов данных;

$V$ — множество значений;

$\leq$ — рефлексивное, асимметрическое транзитивное отношение на $(C \times C) \cup (R \times R) \cup (T \times T)$, называемое специализацией, которое формирует частичные упорядочения на C и R, называемые соответственно иерархией концептов и иерархией отношений;

$\perp$ — иррефлексивное и симметрическое отношение на $(C \times C) \cup (R \times R) \cup (T \times T)$, называемое исключением;

$\in$ — отношение над $(I \times C) \cup (V \times R)$, называемое инстациацией (конкретизацией);

$=$ — отношение над $I \times P \times (I \cup V)$, называемое присваиванием;

(множества $C, R, I, T, V$ являются попарно непересекающимися).»

В общем виде структура онтологии представляет собой следующий набор элементов: понятия, отношения, аксиомы, отдельные экземпляры.

Понятия представляют собой концептуализацию класса всех представителей некой сущности или явления. Таким образом, каждый класс описывает совокупность индивидуальных сущностей, которые объединены на основании наличия общих свойств.

Между понятиями могут быть заданы отношения, которые служат как для связи классов, так и для их описания. В частности, понятия могут быть связаны функциональным отношением, обычно обозначающимся глаголом. Например, при описании реализации в программировании

могут использоваться отношения «реализует» (класс реализует интерфейс) и «расширяет» или «наследует» (один класс расширяет другой).

Самым распространенным типом отношений, использующимся во всех онтологиях, является таксономическое отношение, то есть отнесение к определенной категории. Этот тип отношений имеет и другие названия: отношение категоризации, родовидовое отношение, отношение «класс-подкласс», Is-a, AKO (a kind of) [Gavrilova, Leshcheva, Rumyantseva, 2011].

Другой тип часто используемых отношений — «часть–целое» (партономическое отношение). Отношение «часть-целое» является одним из основных примитивов структурирования Вселенной, определение этого отношения между объектами требуется во многих приложениях, например, в системах диагностики неисправностей, географических приложениях и т.п. Исследование отношений типа «часть-целое» является большой самостоятельной областью науки — "мереология" и "мереотопология".

Аксиомы задают условия, связывающие понятия и отношения. [http://www.sci-innov.ru/icatalog_new/entry_68352.htm] Они позволяют выразить ту информацию, которая не может быть отражена в онтологии только посредством построения иерархии понятий и установки различных отношений между понятиями. Аксиомы позволяют в дальнейшем осуществлять дополнительные выводы в рамках онтологии. Аксиомы могут также представлять собой ограничения, накладываемые на какие-либо отношения, делающие возможным выведение логических заключений. Если в онтологии задана аксиома, говорящая, что «Преподаватель» — это «Человек», который преподает по крайней мере одну «Дисциплину», а затем указано, что «Иванов Иван Иванович» (представитель класса «Человек») преподает «теорию вероятностей» (где «теория вероятностей» входит в класс «Дисциплина»), то будет выведено, что «Иванов Иван Иванович» является «Преподавателем».

Наряду с указанными элементами онтологии в нее также входят так называемые «экземпляры». Экземпляры — это отдельные представители класса сущностей или явлений, то есть конкретные элементы какой-либо категории («Иванов Иван Иванович» в предыдущем абзаце является примером экземпляра класса «Человек»). Онтологию, в которую были добавлены экземпляры, будем называть онтологической базой знаний.

Основные принципы, которым должны удовлетворять онтологии [Gruber, 1995]:

1. Доходчивость, ясность (Clarity).
2. Согласованность (Coherence)
3. Расширяемость (Extendibility).
4. Минимальное влияние кодирования (Minimal encoding bias).
5. Минимальные онтологические обязательства (Minimal ontological commitment).

Моделирование структуры базы знаний является краеугольным камнем онтологического инжинирнга, но для практического использования важно наполнить онтологию реальными данными, или индивидами (экземплярами информационных объектов) и отношениями между ними.

При этом существующие методы наполнения баз знаний, основанных на онтологиях, обладают рядом недостатков:

- узкой направленностью на решение задач конкретного проекта,
- сложностью описания моделей источников данных и их трансформации, высокими требованиями к квалификации пользователей,
- обеспечением возможности интеграции данных из источников только одного типа, отсутствием модульности и расширяемости, а также удобного инструментария.

Это определяет актуальность данного исследования по созданию новых методов автоматизированного наполнения и обогащения онтологий. При этом акцент ставится на так называемую консолидацию данных и знаний, что позволяет снизить трудоемкость наполнения и обогащения онтологий, используя накопленные массивы информации независимо от формы их хранения и представления.

### 1.1.2 Классификации онтологий

Существует множество классификаций онтологий [Ruiz, Hilera, 2006; Berdier, Roussey, 2007; Bullinger, 2009; Гаврилова, Хорошевский, 2000]. На рисунке 1 представлены общепринятые критерии и классификации.


***Рисунок 1 —*** *Классификация онтологий*

[Составлено по: Гаврилова, Хорошевский, 2000; Гаврилова, Кудрявцев, Муромцев, 2020]

По степени формальности различают:

- Неформальные онтологии — предполагают описание на любом естественном языке. В эту категорию входят: списки терминов (каталоги), глоссарии (список терминов с описанием их значений на естественном языке), тезаурусы (предоставляют дополнительную по сравнению с глоссариями информацию о семантических отношениях между терминами, например, о синонимичности).
- Слабо формализованные — предполагают наличие структуры (обычно, таксономии), но не формальной логики. Сюда относят: структурированные глоссарии, фолксономии (неформальные таксономии), формальные таксономии, XML DTDs, схемы баз данных и т.п.
- Сильно формализованные — предполагают описание на языке формальной логики (логики предикатов первого порядка, дескрипционной логики и т.п.).

Спектр различных видов онтологий по степени формальности представлен на рисунке 2.

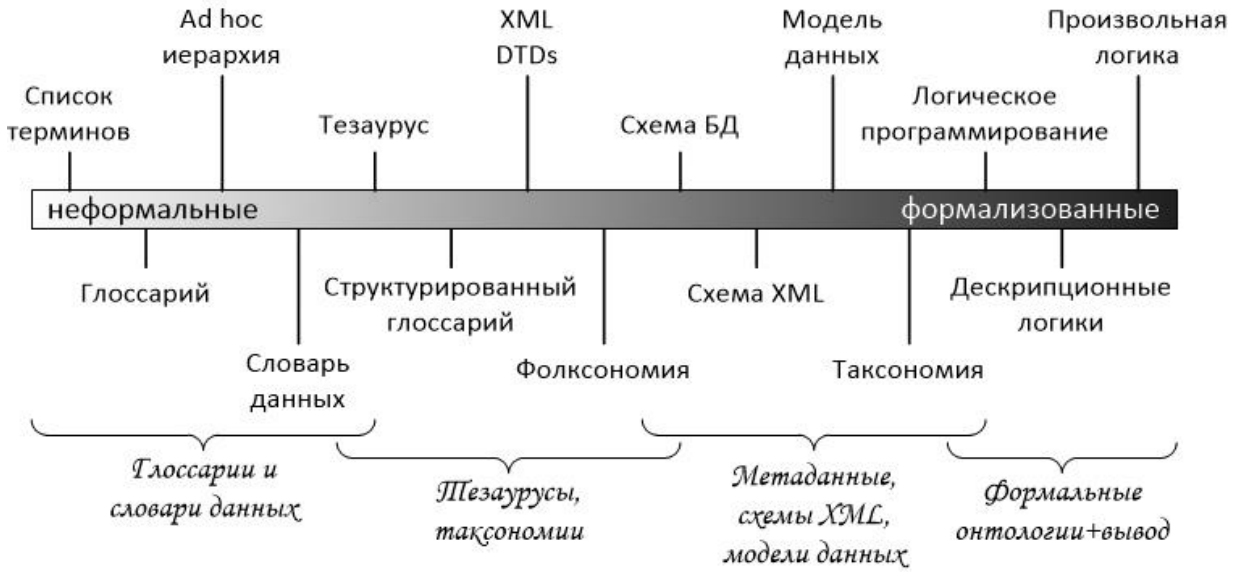


***Рисунок 2 —*** *Спектр онтологий*

[Источник: Uschold, Gruninger, 2004]

По уровню выразительности используемых средств описания (глубине описания предметной области) онтологии делятся на [Gómez-Pérez, Fernandez-Lopez, Corcho, 2004]:

- "весомые" онтологии (Heavyweighted), содержащие аксиомы и
- "легковесные" (Lightweighted), их не содержащие.

По степени зависимости от задачи или предметной области:

- Онтологии представления (мета-онтологии) — представляют собой концептуализацию формализмов представления знаний.
- Онтологии верхнего уровня — описывают общие знания о мире, не зависящие от предметной области.
- Онтологии предметной области — разрабатываются совместно с экспертами в предметной области для установления единой терминологии и совместного использования информации в своей области.
- Проблемно-ориентированные онтологии — разрабатываются для использования конкретной прикладной программой, предназначенной для решения определенных задач.
- Прикладные онтологии — обладают свойствами двух предыдущих видов онтологий: содержат знания о предметной области и предназначены для решения некоторого круга прикладных задач.

По типу отношений выделяют:

- Таксономии: ведущее отношении — родовидовое («kind-of», «is-a»)
- Партономии: ведущее отношение «имеет частью» («состоит», «has part»)

- Генеалогии: ведущее отношение «потомок-предшественник» («отец-сын»)
- Функциональные: отношения выражаются глаголами
- Ассоциативные: основное отношение «ассоциируется с»
- Атрибутивные: основное отношение «обладает свойством»
- Смешанные онтологии: онтологии с любыми типами отношений

### 1.1.3 Области применения и примеры баз знаний онтологического типа

Список предметных областей, для которых создаются онтологии, весьма обширен. Базы знаний уже широко используются для управления бизнес-процессами в различных отраслях. Например, в основе СППР для засушливых сельскохозяйственных районов Мексики лежит база знаний, с помощью которой фермеры могут получить совет, как им следует поступить в сложившейся ситуации для получения максимального урожая. [Sanchez-Cohen et al., 2015].

Возможно, наибольшее число онтологий и онтологических баз знаний создано для медицины, так как диагностирование было одной из первых прикладных предметных областей для инженерии знаний. Это и различные словари (Metathesaurus http://www.nlm.nih.gov/research/umls/, SNOMED http://www.snomed.org/, MeSH http://www.nlm.nih.gov/mesh/meshhome.html, GALEN http://www.opengalen.org), онтологии для систем диагностики, например, онтология MIAKT для диагностики рака груди у женщин [Patlak et al., 2001] и другие.

Информационные технологии и системы, основанные на знаниях, могут значительно улучшить управление сложными распределенными системами здравоохранения, где важна поддержка междисциплинарности и существенны коммуникации и синхронизация между различными специалистами. Например, мультиагентная платформа HeCaSe2, в основе которой лежит онтологическая база знаний, помогает врачам собирать информацию о пациентах, предоставляет клинические руководства по диагностике и лечению заболеваний, координирует всех участников процесса оказания медицинской помощи, контролирует выполнение всех назначений. [Isern, Sánchez, Moreno, 2012]. Онтология DO4MG создана для поддержки оказания помощи в экстренных ситуациях, особенно когда пострадали сразу много людей. Ее задача — обеспечение своевременного реагирования и лечения угрожающих жизни травм или болезней в случае массового поражения, эффективного взаимодействия и координации между учреждениями и чрезвычайными командами, предоставления доступа к информации в режиме реального времени. [Haghighi et al., 2010].

Но кроме медицины, онтологии используются практически во всех сферах человеческой жизнедеятельности: в биологии [Stevens et al., 2007], в сельском хозяйстве [Sanchez-Cohen et al., 2015], в образовании [Никитин, 2006; Гаврилова, Лещева, Кудрявцев, 2012; Гаврилова, Лещева,

Лещев, 2000; Гаврилова, Лещева, Страхович, 2011], в научных исследованиях [Загорулько, Боровикова, 2007; Лещева, Лещев, 2014; Gavrilova, Leshcheva, Strakhovich 2015], для городского планирования [Berdier, Roussey, 2007], для взаимодействия разнородных устройств [Christopoulou, Kameas, 2005], для автоматизированных производств [Гаврилова и др., 2019] и даже для представления знаний о футболе [Buitelaara et al., 2008].

## 1.2. Разработка баз знаний онтологического типа

Разработка больших систем (например, БЗ организации) связана с большой трудоемкостью создания единой онтологии [Тузовский, 2007], в связи с чем предлагаются две модели создания БЗ на основе онтологий:

- отказ от глубокой декомпозиции системы и включение в онтологию только наиболее значимых понятий из рассматриваемых предметных областей;
- выделение одного из направлений деятельности организации и создание для этого направления детальной, но узкоспециализированной онтологии.

При этом первый подход дает слишком грубую и обобщенную модель, а второй не позволяет использовать модель для взаимодействия между всеми подразделениями компании. Для решения этой проблемы предлагается создать иерархию областей знаний организации и создавать отдельно онтологии разных подобластей, которые могут иметь различную глубину в зависимости от потребностей в их детальном моделировании.

Необходимо отметить, что онтологии носят субъективный характер [Gavrilova, Leshcheva, Ontology…, 2015] и зависят от многих факторов, например от:

- задач, которые призвана решить создаваемая онтология;
- используемой методологии создания онтологий;
- языка представления онтологий;
- программного инструмента, который используется для работы над онтологией;
- когнитивных стилей как экспертов, так и аналитиков, работающих над созданием онтологии.

Основные результаты, полученные при исследовании связи когнитивных стилей экспертов с характеристиками построенных ими онтологий изложены в [Gavrilova, Leshcheva, Ontology…, 2015; Гаврилова, Лещева, 2016].

Основные последние методологии разработки онтологий рассмотрены в работе [Гаврилова, Кудрявцев, Муромцев, 2016; Simperl, Luczak-Rösch, 2014]: DOGMA-MESS [De Moor, De Leenheer, Meersman, 2006; De Leenheer, Debruyne, 2008], DILIGENT [Tempich et al, 2005, 2007], NeOn (www.neon-project.org) [Suárez-Figueroa, Gómez-Pérez, Fernández-López, 2012] и др.

Важнейшая современная рекомендация по разработке онтологий — использовать готовые доступные онтологии предметной области, которые можно найти в Swoogle (http://swoogle.umbc.edu/) или в LOV (Linked Open Vocabularies [http://lov.okfn.org/]), либо максимально включить их в процесс создания онтологии [Suárez-Figueroa, Gómez-Pérez, Fernández-López, 2012]. Также возможна интеграция нескольких онтологий для получения единого информационного пространства [Бова, 2015], например, на основе сравнения онтологических схем и интеграции концептов, с последующим формированием шаблона запросов для взаимного отображения онтологий друг в друга.

Таким образом, выделяют три основных подхода к созданию онтологий и основанных на них баз знаний [Petasis et al, 2011]:

1. Интеграция существующих онтологий. В процессе интеграции осуществляется попытка выделения общего в онтологиях, описывающих одинаковые или сходные предметные области, чтобы создать новую онтологию. Было предложено несколько методов, например:
    a. Слияние (merging) онтологий для создания единой согласованной онтологии;
    b. Выравнивание (alignment) онтологий путем установления связей между ними, позволяя им повторно использовать информацию друг друга;
    c. Отображение (mapping) онтологий путем нахождения соответствия между элементами онтологий.
2. Построение онтологии «с нуля» или расширение (наполнение и обогащение) существующей онтологии, как правило, на основе информации, извлеченной из предметно-ориентированного контента.
3. Специализация общей онтологии для того, чтобы адаптировать ее к определенной предметной области.

Под наполнением онтологии (ontology population) понимают добавление в онтологию экземпляров с их свойствами, а под обогащением (ontology enrichment) — добавление новых отношений и аксиом, использующих эти отношения.

Разработка базы знаний онтологического типа включает 4 шага [Pan et al, 2017]:

1. Выявление требований к создаваемой системе и её базе знаний.
2. Формирования онтологии или сети онтологий.
3. Наполнение онтологии или сети онтологий.
4. Публикация полученной базы для дальнейшего использования.

Далее приводится краткое описание языков представления онтологий и инструментов онтологического инжиниринга.

### 1.2.1 Генеалогия языков представления онтологий

Язык описания онтологий — формальный язык, используемый для кодирования онтологии. Наиболее известные среди них: OWL — Ontology Web Language, стандарт W3C (World Wide Web Consortium, https://www.w3.org/), язык для семантических утверждений, разработанный как расширение RDF (Resource Description Framework) и RDFS (RDF Schema); KIF (Knowledge Interchange Format — формат обмена знаниями) — основанный на S-выражениях синтаксис для логики; CycL — онтологический язык, использующийся в проекте Cyc, основан на исчислении предикатов с некоторыми расширениями более высокого порядка; DAML+OIL (DARPA Agent Markup Language, Ontology Inference Layer).

На рисунке 3 представлена генеалогия языков представления онтологий, привязанная к временной шкале.

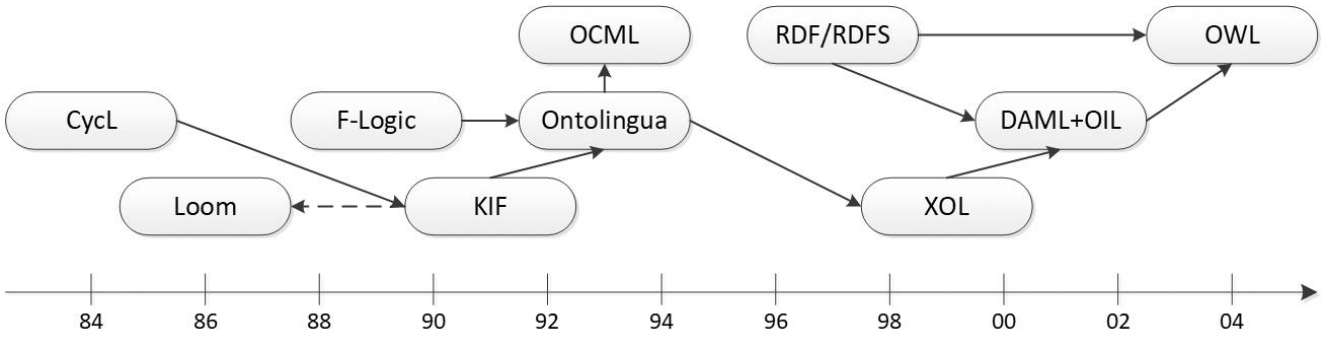


***Рисунок 3 —*** *Генеалогия языков представления онтологий*

[Составлено по: Казекин, 2008]

На временной шкале отмечен момент начала разработки языка (или первого релиза). В целом, привязка к временной шкале носит условный характер, так как языки развивались, причем с различной скоростью. Так, например, несмотря на то, что работы по созданию языка Loom начались раньше, в процессе развития он подвергся влиянию языка KIF.

***CycL***

CycL — это декларативный язык, используемый для представления общеизвестных знаний в коммерческой базе знаний Cyc [http://www.cyc.com/]. Проект по созданию этой базы знаний был начат в 1984 году Дугласом Ленатом (Douglas Lenat). CycL — это гибридный язык, в котором объединены свойства фреймов и логики предикатов первого порядка. Синтаксис языка CycL схож с синтаксисом языка Lisp. Несмотря на непривычный синтаксис, CycL обладает такими привлекательными средствами, как вложенность (в частности, вложенность кванторов), сколемизация, микротеории, система диагностики ошибок, а также пять уровней истинности утверждений. Микротеория — это контекст, содержащий тело — набор утверждений — и

предположения, истинные над всеми утверждениями. При этом истинность или ложность утверждения в CycL зависит от контекста, представленного как микротеория. Сколемизация — это избавление от кванторов существования и преобразование их в кванторы общности. CycL поддерживает выражения логики предикатов первого порядка и достаточно выразителен для того, чтобы на нем представлять онтологии [Lenat, Guha, 1991; Reed, Lenat, 2002].

***Loom***

Loom — это язык для построения интеллектуальных приложений. Был разработан в ISI (Information Sciences Institute) в Университете Южной Калифорнии при поддержке DARPA под руководством Роберта Макгрегора (Robert Macgregor) в 1986 г. Сердце Loom представляет собой систему представления знаний, которая используется, чтобы обеспечить дедуктивную поддержку декларативной части языка Loom. Дедуктивный механизм вывода, называемый классификатор, использует обратный вывод, семантическое объединение и объектно-ориентированные технологии, чтобы скомпилировать декларативные знания в сеть, предназначенную для эффективной поддержки обработки дедуктивных он-лайн запросов. Декларативные знания в Loom состоят из определений, правил, фактов и правил по умолчанию. Высокая степень интеграции между декларативной и процедурной составляющими языка Loom позволяет программистам использовать логическое программирование, продукции, и парадигму объектно-ориентированного программирования в одном приложении [http://www.isi.edu/isd/LOOM/LOOM-HOME.html] [MacGregor, 1999].

***F-Logic***

F-Logic был разработан Михаэлем Кифером (Michael Kifer) и Георгом Лаусеном (Georg Lausen) и является формализмом для представления знаний. F-Logic находится в таком же отношении к объектно-ориентированному программированию, как классическое исчисление предикатов соотносится с программированием реляционных баз данных. Функциональность включает в себя, среди прочего, идентификаторы объектов, сложные объекты, наследование, полиморфизм, методы запросов, инкапсуляцию. Сильные стороны F-Logic — это расширяемость и способность непосредственно представлять фундаментальные понятия, которые приходят из языков объектно-ориентированного программирования и языков, основанных на фреймах. Главная слабость F-Logic связана с математическими и логическими понятиями, необходимыми для программирования на этом языке [Kifer, Lausen, 1989].

***KIF***

KIF — специальный язык, предназначенный для обмена знаниями между разными компьютерными системами (созданными разными программистами, в разное время, на разных языках, и так далее). Язык первоначально был создан Майклом Дженесерет (Michael Genesereth)

и другими участниками Проекта Обмена Знаниями DARPA. (Defense Advanced Research Projects Agency — агентство передовых оборонных исследовательских проектов — агентство Министерства обороны США, отвечающее за разработку новых технологий для использования в вооружённых силах). KIF был разработан, чтобы использоваться как синтаксис для логики предикатов первого порядка, которая удобна для компьютерной обработки. Как и CycL, KIF наследует синтаксис из Lisp. KIF был задуман как язык–посредник, он не предназначен в качестве основного языка для взаимодействия с пользователем, он также не предназначен для внутреннего представления знаний в компьютерных программах (хотя он может быть использован для этих целей). Обычно, когда программа читает базу знаний в KIF, она преобразует данные в свой внутренний формат. Все вычисления выполняются с использованием этого внутреннего формата. Когда программа должна общаться с другой программой, она отображает внутренние структуры данных в KIF. Неудобством этого языка является его слишком высокая вычислительная сложность [Genesereth, Fikes, 1992].

***Ontolingua***

Оригинальный язык Ontolingua был разработан в KSL (Knowledge System Laboratory) отделения информатики Стэндфордского Университета. Ontolingua использует принципы объектно-ориентированного подхода и является расширением компьютерно ориентированного языка KIF для обмена знаниями и взаимодействия между программами [Gruber, 1993]. Грубер расширил подъязык определений KIF, чтобы обеспечить дополнительные идиомы, которые часто бывают в онтологиях, и добавил фреймовые конструкции (в соответствии с протоколом OKBC (Open Knowledge Base Connectivity) — прикладной интерфейс программирования для доступа к базам знаний.) для того, чтобы стало возможным специфицировать онтологии в псевдо объектно-ориентированном стиле, используя знакомые отношения и функции, такие как Class, Subclass-Of, Slot, Slot-Value-Type, SlotCardinality и Facet. Недостатком языка является то, что сам по себе он не предоставляет возможности логического вывода.

***OCML***

OCML (Operational Conceptual Modeling Language) был разработан в контексте проекта VITAL [Shadbolt, Motta, Rouge, 1993]. Основной целью языка является поддержка моделирования на уровне знаний. OCML позволяет задавать спецификацию функций, отношений, классов, экземпляров и правил. Он также включает механизмы для описания онтологий и методов решения задач — основные технологии, разработанные в области представления знаний. Около десятка проектов в KMI (Knowledge Media Institute) использовали OCML для разработки моделей в таких областях, как управление знаниями, разработка онтологии, электронная торговля и системы обработки знаний [Motta, 1998].

***RDF и RDF Schema***

RDF (Resource Description Framework — среда описания ресурса) является стандартной моделью для обмена данными по сети [http://www.w3.org/RDF/]. RDF представляет утверждения о ресурсах в виде, пригодном для машинной обработки. Ресурсом в RDF может быть любая сущность — как информационная (например, веб-сайт или изображение), так и неинформационная (например, человек, река или некое абстрактное понятие). Структура, лежащая в основе любых выражений в RDF, это коллекция триплетов, каждый из которых состоит из субъекта, предиката и объекта. Набор таких триплетов называется RDF-графом. Узлы этого графа — объекты и субъекты. Связь всегда направлена от субъекта к объекту. Таким образом, RDF-граф образует семантическую сеть. Утверждение «голубой вертолет» в RDF-терминологии можно представить следующим образом: субъект — «вертолет», предикат — «имеет цвет», объект — «голубой». Для обозначения субъектов, отношений и объектов в RDF используются URI [http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/]. URI (Uniform Resource Identifier) — унифицированный (единообразный) идентификатор ресурса [https://tools.ietf.org/html/rfc3986]. Язык RDF менее выразителен, чем языки, основанные на исчислении логики предикатов, такие как CycL и KIF, но намного проще для реализации. Так как его основным назначением является аннотирование ресурсов всемирной паутины, то это не является недостатком или ограничением. RDF сам по себе не предоставляет механизмов для описания свойств или отношений, поэтому понадобилось дополнение, которым стала RDF Schema — язык описания словарей. По сути, RDF Schema добавляет типы данных к RDF. Первая версия рекомендаций W3C датирована мартом 1999 года [http://www.w3.org/TR/1999/PR-rdf-schema-19990303/].

***XOL***

XOL (XML Based Ontology Exchange Language) — как следует из названия, это язык, предназначенный для обмена описаниями онтологий, в основе синтаксиса которого лежит XML. Исследование языков представления онтологий показало, что ни один из существующих языков обмена онтологиями не удовлетворяет требованиям сообщества биоинформатики. Было заявлено о необходимости создания языка с семантикой объектно-ориентированных систем представления знаний, но с синтаксисом XML. Чтобы удовлетворить эти две цели был разработан язык XOL. Авторы языка — Питер Карп (Peter Karp), Виней Чаудри (Vinay Chaudhri) и Джером Томере (Jerome Thomere). Семантика XOL основана на OKBC-Lite, который представляет собой упрощенную форму модели знаний для OKBC [Karp, Chaudhri, Thomere, 1999].

***DAML+OIL***

Язык DAML+OIL [http://www.w3.org/Submission/2001/12/], предлагаемый W3C как стандарт представления онтологий в интернет, опирается на более ранние стандарты W3C, такие как RDF и RDF Schema, и расширяет эти языки более богатыми примитивами моделирования. OIL (Ontology Inference Layer) использует набор примитивов, заимствованный из фреймовых языков представления знаний (в основе языка лежит XOL) и механизм вывода в рамках дескрипционной логики. В OIL онтология — это структура, состоящая из нескольких компонентов, организованных в три слоя: уровень объектов (для определения экземпляров), первый мета-уровень (который содержит определение онтологии) и второй мета-уровень (который содержит информацию о свойствах онтологии, например, об ее авторах). На основе OIL был создан язык семантической разметки веб-ресурсов DAML+OIL. DAML расшифровывается как DARPA Agent Markup Language. Язык имеет ясную и хорошо определенную семантику.

***OWL***

Язык OWL разработан для использования приложениями, которые должны обрабатывать содержимое информации, а не только представлять эту информацию людям. OWL обеспечивает более полную машинную обработку Веб-контента, чем та, которую поддерживают XML, RDF, и RDF Schema, предоставляя наряду с формальной семантикой дополнительный терминологический словарь. Рекомендован и развивается пол эгидой консорциума Всемирной паутины (W3C [http://www.w3.org/]). Различают две версии OWL (OWL 1 и OWL 2), в OWL 1 — три диалекта (в порядке возрастания выразительности): OWL Lite, OWL DL и OWL Full, в OWL 2 — два диалекта: OWL 2 Full и OWL 2 DL, в последнем выделяют три подъязыка (профиля): OWL 2 EL, OWL 2 QL и OWL 2 RL. Каждый из профилей поступается различными аспектами выразительной силы OWL в обмен на другие вычислительные и/или реализационные преимущества.

OWL исходит из предположения об открытости мира, т.е. истинность утверждения не зависит от того, есть ли информация о его верности. Другими словами, нельзя утверждать, что утверждение ложно, пока это не задано явно. Это предположение противоположно предположению о замкнутости мира, из которого следует, что ложно любое утверждение, о котором не известно, что оно истинно.

Далее речь пойдет об OWL 2. Для сериализации OWL-онтологий используются несколько синтаксисов (RDF/XML, OWL/XML, Functional Syntax, Manchester Syntax, Turtle), из которых только первый является обязательным, т.е. любое программное обеспечение, работающее с OWL-онтологиями, обязано уметь открывать и сохранять онтологии с использованием этого синтаксиса.

Основными элементами языка являются классы, индивиды и свойства. Классы OWL интерпретируются как множества, элементами которых являются индивиды. Они описываются, используя формальные конструкции, которые декларируют требования для членства в классе. По умолчанию классы могут пересекаться, если явным образом не указано обратное.

Свойства в OWL представляют бинарные отношения. Свойства, как и классы, могут быть организованы иерархически. Существует два основных типа свойств: объектные свойства (Object Property) и свойства-значения (DataType Property). Объектное свойство является отношением между двумя индивидами. Свойства-значения связывают индивида со значением (например, с числовым значением, строкой, датой и т.п.). OWL также имеет третий тип свойства — свойства аннотации. Свойства аннотации могут использоваться для добавления информации (метаданные — данные о данных) для классов, отдельных индивидов и свойств.

Объектное свойство может быть обратным к другому свойству. Свойства $P$ и $Q$ являются обратными друг к другу, если для любых индивидов $x$ и $y$ $P(x, y)$ эквивалентно $Q(y, x)$, т.е. $\forall\, x,y \in I\ P(x,y) \Leftrightarrow Q(y,x)$. Можно указать, что свойства не пересекаются (disjoint), т.е. $P,Q \in R$ не пересекаются, если $\forall\, x,y \in I\ P(x,y) \Rightarrow \neg Q(x,y)\ \&\ Q(x,y) \Rightarrow \neg P(x,y)$. Также свойство объекта может обладать следующими характеристиками:

- Транзитивность ($P \in R$ транзитивно, если $\forall\, x,y,z \in I\ P(x,y) \& P(y,z) \Rightarrow P(x,z)$)
- Симметричность ($P \in R$ симметрично, если $\forall\, x,y \in I\ P(x,y) \Leftrightarrow P(y,x)$)
- Ассиметричность ($P \in R$ ассимметрично, если $\forall\, x,y \in I\ P(x,y) \Rightarrow \neg P(y,x)$)
- Функциональность ($P \in R$ функционально, если $\forall\, x,y,z \in I\ P(x,y) \& P(x,z) \Rightarrow y=z$)
- Обратная функциональность ($P \in R$ обратно функционально, если $\forall\, x,y,z \in I\ P(y,x) \& P(z,x) \Rightarrow y=z$)
- Рефлексивность ($P \in R$ рефлексивно, если $\forall\, x \in I\ P(x,x)$)
- Иррефлексивность ($P \in R$ иррефлексивно, если $\forall\, x \in I\ \neg P(x,x)$)

Возможно определение композиции (цепочки) свойств. Свойство $S \in R$ является композицией свойств $P$ и $Q$, если $\forall\, x,y,z \in I\ P(x,y) \& Q(y,z) \Rightarrow S(x,z)$.

Классы могут задаваться с помощью ограничений, накладываемых на свойства. Ограничения в OWL делятся на 3 категории:

- ограничения квантификатора;
- ограничения мощности;
- ограничения переменных.

Ограничения квантификатора, в свою очередь, подразделяются на экзистенциальные и универсальные. Экзистенциальные ограничения описывают классы индивидов, которые участвуют по крайней мере в одном отношении по указанному свойству с представителем

некоторого заданного класса. Например, класс «Преподаватель» может быть определен как совокупность индивидов, принадлежащих классу «Человек», и связанных отношением «преподает» хотя бы с одним индивидом из класса «Дисциплина». Универсальные ограничения описывают классы индивидов, которые связаны указанным отношением только с представителями некоторого заданного класса. Например, можно выделить класс преподавателей, которые читают лекции только по элективным дисциплинам, как набор индивидов, принадлежащих классу «Преподаватель», и связанных отношением «преподает» только с индивидами класса «Элективная дисциплина».

Ограничения мощности определяют минимальное, максимальное или точное число указанных отношений (свойств). С их помощью, например, можно описать классы преподавателей, которые ведут ровно одну дисциплину или не менее трех дисциплин.

Ограничение переменной позволяет описывать класс индивидов, которые связаны неким отношением с определенным индивидом или значением. Например, класс преподавателей, работающих на определенной кафедре, может быть описан как класс индивидов, которые связаны отношением «работает на кафедре» с индивидом, представляющим эту кафедру.

***Язык представления правил SWRL***

Одним из существенных недостатков OWL является отсутствие возможности естественным образом определять свойства у свойств. Это не позволяет моделировать свойства-значения у объектных отношений, n-арные отношения и свойства-значения у свойств-значений. Также отсутствует возможность производить какие-либо операции над свойствами-значениями (например, сравнивать, вычислять формулы и т.п.). Для устранения или обхода этих недостатков используются различные дополнительные средства. Одним из таких средств является язык SWRL (Semantic Web Rule Language), позволяющий добавлять в онтологию правила, которые невозможно представить на «чистом» OWL.

Язык SWRL является объединением технологий OWL и RuleML (Rule Markup Language, который, как указано в спецификации [RuleML Primer http://ruleml.org/papers/Primer], в будущем должен стать Rule Modeling Language или даже Rule MetaLogic). В свою очередь RuleML основывается на языке Datalog, который является синтаксическим подмножеством Prolog-а.

Правило в SWRL состоит из двух частей: антецедента и консеквента. Обе части представляют собой конъюнкцию (возможно, пустую) так называемых атомов. Неформально правило можно интерпретировать следующим образом: если все атомы, входящие в антецедент, выполняются (являются истинными утверждениями), то и все атомы консеквента должны выполняться. Атомы могут иметь следующую форму:

- *C(x)*: здесь в качестве *C* может выступать некоторое OWL-описание (чаще всего класс) или диапазон данных. В первом случае *x* — это индивид или объектная переменная, а во-втором — значение или переменная. Выполняется, если *x* принадлежит *C*.
- *P(x, y)*: *P* — это OWL-свойство, как объектное, так и свойство значение. *x* — это индивид или объектная переменная, *y* обозначает индивида или значение в зависимости от типа свойства *P*. Выполняется, если у *x* есть свойство *P*, значением которого является *y*.
- *sameAs(x, y)*: устанавливает эквивалентность *x* и *y*. Выполняется, если *x* и *y* являются одним и тем же индивидом.
- *differentFrom(x,y)*: утверждает, что *x* и *y* являются различными индивидами. Выполняется, если *x* и *y* представляют различных индивидов.

Кроме этих форм атомов, SWRL предлагает целый ряд встроенных формул (built-ins), позволяющих выполнять различные операции над объектами и данными. Наличие этих встроенных конструкций как раз и является сильной стороной языка SWRL. На момент релиза спецификации языка SWRL [http://www.w3.org/Submission/SWRL/] определены built-ins для выполнения сравнения, математических операций, работы со строками, датой, длительностями и списками.

Несколько примеров правил на языке SWRL (перед обозначением переменных ставится вопросительный знак):

- *Человек(?x) ∧ преподает(?x, ?y) ∧ Дисциплина(?y) ⇒ Преподаватель(?x)*

  Это правило записывает предыдущий пример, утверждающий, что любой индивид, принадлежащий классу Человек, который связан отношением «преподает» хотя бы с одной дисциплиной, также входит в класс Преподаватель.
- *Преподаватель(?x) ∧ преподает(?x, ?y) ∧ Дисциплина(?y) ∧ формирует (?y, ?z) ∧ Компетенция (?z) ⇒ формирует (?x, ?z)*

  Если преподаватель преподает некоторую дисциплину, которая формирует компетенцию, то можно сказать, что преподаватель формирует у студентов эту компетенцию.

Необходимо отметить, что оба приведенных правила не требуют использования SWRL, а могут быть записаны на OWL.

- *Прямоугольник (?x) ∧ длина (?x, ?a) ∧ ширина (?x, ?b) ∧ swrlb:equal (?a, ?b) ⇒ Квадрат (?x) ∧ длина_стороны (?x, ?a)*

  Если длина и ширина прямоугольника равны между собой, то это квадрат с длиной стороны, равной длине (или ширине).

### 1.2.2 Программные системы онтологического инжиниринга: анализ и сравнение

Согласно исследованиям сайта XML.com [http://www.xml.com/pub/a/2004/07/14/onto.html] в 2002 году список инструментов для работы с онтологиями состоял из 52х, в 2004 году — 93х продуктов. В 2008 году на сайте http://www.hozo.jp/OntoTools/ уже перечислены более 150-ти систем. В настоящий момент на сайте http://www.w3.org/2001/sw/wiki/Tools указано 340 различных инструментальных средств. Рост числа инструментов демонстрирует стремительно растущий интерес к разработке онтологий.

Ниже проведено сравнение нескольких бесплатных систем по следующим критериям:

- формализм, лежащий в основе представления онтологических знаний;
- язык представления онтологий;
- функциональность редактора онтологий, в частности:
    - возможность просматривать и редактировать онтологии, используя веб-интерфейс;
    - отображение и возможность редактирования онтологии в визуальной форме;
    - механизм логического вывода;
    - поддержка языка запросов;
    - расширяемость системы (т.е. возможность подключения дополнительных модулей, созданных сторонними разработчиками).

Дополнительно в состав инструментальных средств создания и поддержки онтологий могут включаться модули для выравнивания (alignment), отображения (mapping), объединения (merging) и оценки (evaluation) онтологий, но их наличие или отсутствие не представляет интереса в рамках данной работы.

***Ontolingua Server***

Ontolingua Server [http://www.ksl.stanford.edu/software/ontolingua/] был первым инструментом для совместного создания, просмотра, редактирования и использования онтологий в среде WWW [Farquhar, Fikes, Rice, 1997]. Он разработан в 1995 г. лабораторией KSL (Knowledge System Laboratory) отделения информатики Стенфордского университета.

Редактор онтологий реализован на основе форм HTML. Система включает графический браузер, позволяющий просмотреть иерархию концептов, включая экземпляры, но не позволяющий редактировать таксономию концептов, представленную в графической форме.

Одна из функций системы Ontolingua — это интеграция с другими системами представления знаний. Эта функция выполняется подсистемой OKBC на базе языка

межмашинного обмена знаниями KIF. В последнее время в качестве стандарта языка обмена онтологиями в Ontolingua используется и язык OWL.

Кроме редактора онтологий и подсистемы OKBC Ontolingua включает в себя сетевое приложение Webster (получение определений концептов) и Chimaera (анализ, объединение, интегрирование онтологий). Все приложения, кроме сервера OKBC, реализованы на основе форм HTML. Система представления знаний реализована на Lisp. Формализм, лежащий в основе представления знаний — фреймы и логика первого порядка.

Сервер поддерживает совместную разработку онтологии несколькими пользователями, для чего используются понятия пользователей и групп. Ontolingua Server также предоставляет архив онтологий, включающий большое количество онтологий различных предметных областей, что позволяет создавать онтологии из уже существующих.

К недостаткам можно отнести отсутствие проверки онтологии на непротворечивость. Корни этого кроятся в языке Ontolingua, который является основным языком представления знаний в системе.

***OntoSaurus***

OntoSaurus [http://www.isi.edu/isd/ontosaurus.html] был реализован примерно в то же время, что и Ontolingua. Он создавался как Web-редактор и браузер для Loom-онтологий. В OntoSaurus обеспечиваются автоматический контроль совместимости, дедуктивная поддержка рассуждений и некоторые другие функции [Swartout et al., 1996]. При этом он предоставляет ограниченные средства редактирования, поэтому для построения сложных онтологий необходимо осваивать язык Loom.

Серверная часть также написана на Lisp. В качестве модели представления знаний используется дескрипционная логика.

***WebOnto***

WebOnto [http://projects.kmi.open.ac.uk/webonto/] — редактор для совместного просмотра, создания и редактирования онтологий, использующих язык OCML [Domingue, 1998; Domingue, Motta, Garcia, 1999]. Он был разработан в 1997 году для Tadzebao — инструмента, который позволяет инженерам по знаниям вести синхронные и асинхронные дискуссии об онтологии. Tadzebao учитывает тот факт, что неотъемлемой частью совместной разработки онтологий является диалог, а так как разработка зачастую ведется удаленно, то необходимо средство для поддержки ведения такого диалога.

В отличие от двух предыдущих инструментов клиент WebOnto представляет собой Java-апплет. Для моделирования онтологий используется язык OCML.

Пользователь может создавать различные структуры, в том числе классы со множественным наследованием, причем редактирование можно осуществлять в графической среде. Инструмент предоставляет средства масштабирования для построения больших онтологий и имеет ряд полезных особенностей, например, раздельный просмотр отношений, классов и правил. Также WebOnto проверяет вновь вводимые данные с помощью контроля целостности кода OCML.

Формализм, лежащий в основе представления знаний — фреймы и логика первого порядка.

***OilEd***

OilEd — автономный графический редактор онтологий, разработанный в рамках проекта On-To-Knowledge [Bechhofer S. et al., 2001]. OilEd был разработан в 2001 году, как редактор OIL онтологий. С появлением DAML+OIL, OilEd был адаптирован под управление DAML+OIL онтологиями, позже его адаптировали под OWL.

OilEd сочетает в себе фреймовую структуру и выразительность дескрипционной логики с сервисами рассуждения, что позволило обеспечить понятный и интуитивный стиль интерфейса пользователя и преимущества поддержки рассуждения (обнаружение логически противоречивых классов и скрытых отношений подкласса).

OilEd написан на Java. Он свободно распространяется по общедоступной лицензии GPL, но больше не поддерживается.

OilEd так и не был закончен и не обеспечивает полноценную среду разработки онтологий. В нем отсутствует контроль версий, не поддерживается разработка онтологий большого масштаба и, вследствие использования механизма рассуждений FaCT, ограничена поддержка экземпляров классов.

***Protégé***

Protégé [http://protege.stanford.edu/] — локальный бесплатный редактор онтологий, разработанный группой медицинской информатики Стенфордского университета (первая версия была выпущена в 1987, сейчас выпущена 5-я версия) [Gennari et al., 2003; Knublauch et al., 2004]. В настоящее время является наиболее широко используемым инструментом, находящимся в свободном доступе.

Protégé рассчитан на создание прикладных онтологий с нуля, позволяет быстро и относительно просто сконструировать небольшую предметную онтологию и имеет расширяемую архитектуру, которая позволяет легко встраивать его в прикладные программы. Структура онтологии сделана аналогично иерархической структуре каталога. На основе сформированной онтологии Protege может генерировать формы получения знаний для введения

экземпляров классов и подклассов. Инструмент имеет графический интерфейс, удобный для использования неопытными пользователями, снабжен справками и примерами.

Protégé является Java-программой с открытым исходным кодом. Возможно расширение этого инструмента с помощью плагинов. Платформа Protégé поддерживает два пути моделирования онтологий — на основе фреймовой модели представления знания OKBC и с помощью OWL. Соответственно, могут быть реализованы основные выразительные возможности OWL — различение индивидов, классов и свойств; иерархия классов и свойств; указание области применения и области значений для объектных свойств и др. Также существует возможность указать, что пара или группа классов являются взаимоисключающими. Классы могут не только описываться, но и определяться через указание его необходимых и достаточных свойств. Индивид может принадлежать нескольким классам одновременно или не принадлежать ни одному классу. Онтологии могут экспортировать в различные форматы, включая RDF, RDFS, OWL и XML Schema.

Развивается веб-версия — WebProtege [https://webprotege.stanford.edu/].

***Fluent Edidor***

Fluent Editor [http://www.cognitum.eu/semantics/FluentEditor/], онтологический редактор, использующий контролируемый естественный язык (Controlled Natural Language), является универсальным инструментом для создания и обработки сложных онтологий. Fluent Editor обеспечивает более подходящую для пользователей альтернативу редактору OWL на основе XML. Главной особенностью является использование контролируемого английского языка как языка моделирования знаний. Поддерживается с помощью специального редактора Predictive Editor, запрещающего вводить любые предложения, которые являются грамматически или морфологически некорректными, и активно помогающего пользователю во время написания предложения.

Еще одной особенностью Fluent Editor является использование так называемых активных правил (Active Rules). Активные правила — это механизм Fluent Editor, обеспечивающий выполнение пользовательского императивного кода на C# при соблюдении определенных критериев, описанных как правила SWRL. С помощью активных правил можно вставлять знания в базу знаний, удалять их оттуда и печатать сообщения в окно вывода.

Функциональность Fluent Editor может быть расширена с помощью различных плагинов. В частности, установив специальный плагин, можно мгновенно синхронизировать представление онтологии между различными окнами Fluent Editor и Protégé в одном месте. Совместимость таких мощных редакторов онтологий, как Protégé и Fluent Editor открывает большие возможности, позволяя использовать сильные стороны каждого редактора для работы с одной онтологией.

***Сравнение инструментов онтологического инжиниринга***

В таблице 1 представлено сравнение инструментов онтологического инжиниринга по выбранным ранее критериям.

**Таблица 1 —** Сравнение редакторов онтологий

| **Критерии сравнения** | **Ontolingua Server** | **OntoSaurus** | **WebOnto** | **OilEd** | **Protégé** | **Fluent Editor** |
|---|---|---|---|---|---|---|
| Формализм | Фреймы<br>Логика первого порядка | Дескрипционная логика | Фреймы<br>Логика первого порядка | Дескрипционная логика | Фреймы<br>Логика первого порядка | Дескрипционная логика |
| Язык | Ontolingua | Loom | OCML | DAML+OIL | OKBC, OWL, SWRL | OWL, SWRL |
| Интерфейс пользователя | HTML | HTML | Java-апплеты | Локальная программа | Локальная и WebProtégé | Локальная программа |
| Редактирование в визуальной форме | – | – | + | – | + | – |
| Механизм логического вывода | JTP | Классификатор Loom | OCML | FaCT | HermiT, FaCT++, Pellet | HermiT |
| Поддержка языка запросов | – | – | – | – | + | + |
| Расширяемость | – | – | – | – | Плагины | Плагины |

Из таблицы следует, что наиболее развитым и удобным для решения поставленной задачи являются Protégé и Fluent Editor, так как они предоставляют возможности редактирования онтологий и в текстовой, и в визуальной форме, расширения функциональности с помощью плагинов и поддерживают языки описания правил (SWRL, Active Rules) и запросов SPARQL. При необходимости доступа к онтологии через веб-интерфейс можно перейти на WebProtégé. Кроме того, Protégé и Fluent Editor оказались одними из немногих свободно-распространяемых инструментов, которые продолжают активно развиваться. Но у инструмента Fluent Editor есть недостаток: он ориентирован на названия классов, отношений и т.п. исключительно на английском языке, что ограничивает среду его применений.

## 1.3. Постановка задачи исследования по наполнению онтологий и жизненный цикл онтологий

Существует множество теорий, методологий и инструментов для моделирования различных видов семантики с использованием онтологий. Однако после того, как организация завершила моделирование структуры онтологии, для практического использования эта онтология должна быть наполнена индивидами (экземплярами) и отношениями. Этот процесс может быть достаточно трудоемким, особенно если производить его «вручную», поэтому его необходимо автоматизировать. Причем требуются методы, которые могут быть применены для интеграции гетерогенных данных в онтологию предметной области, как на стадии ее проектирования, так и на стадии эксплуатации.

Наполнение онтологий на практике сопряжено с рядом трудностей, вызванных, в основном, гетерогенностью источников данных. Эта гетерогенность может иметь различную природу: форматы представления данных, модели данных, используемая терминология, различия в типах данных, единицах измерения или в множестве допустимых значений и т.п. И даже если все данные находятся в одном источнике, интерпретация того, каким образом трактовать семантику данных и осуществлять наполнение онтологии зависит не столько от модели источника данных, сколько от модели, заложенной в онтологию предметной области.

Для того, чтобы проанализировать существующие методы наполнения онтологий и выявить их ограничения или недостатки, рассмотрим жизненный цикл онтологий с точки зрения наполнения и связанным с ним проблем.

Жизненный цикл онтологий может быть рассмотрен двояко. В широком смысле он охватывает этапы от появления идеи до реализации онтологии и её распространения [Farquhar et al., 1995; Fernandez, Gómez-Pérez, Juristo, 1997; Gandon, 2002; Li, Raskin, Ramani, 2007]. В более узком смысле он рассматривается как непосредственная разработка онтологии [Uschold et al., 1998; Noy, McGuinness, 2001; Nanda et al., 2006]. Попытка согласовать эти видения была сделана авторами методологии NeOn [Suárez-Figueroa et al., 2012], которые жизненный цикл понимают в широком смысле.

В модели жизненного цикла в широком смысле выделяются несколько общих стадий, на каждой из которых могут возникнуть проблемы наполнения онтологии экземплярами. Ряд таких проблем описан в [Petasis et al, 2011; Celjuska and Vargas-Vera, 2004]. Классифицируем проблемы по этапам жизненного цикла онтологии:

1. Спецификация онтологии — определение цели построения и границ содержания. На этом же этапе должны быть решены вопросы включения в онтологию «чувствительных» (например, персональных) данных.
2. Разработка онтологии, в которую входят: сбор информации; концептуализация — формирование классов, свойств, их иерархии и отношений и др. На этом же этапе производится первичное наполнение онтологии данными. При этом можно выделить ряд проблем, связанных с наполнением. Среди них:
    a) распознавание объектов — определение частей данных, которые являются экземплярами;
    b) соотнесение объектов с классами;
    c) синонимия объектов — наличие разных значений, относящихся к одному объекту;
    d) омонимия объектов — когда похожие объекты относятся к разным классам.
3. Оценка — проверка онтологии перед использованием. В случае автоматического или полуавтоматического наполнения должна быть проверена адекватность экземпляров категориям, их уникальность, согласованность, полнота и отсутствие избыточности.
4. Поддержка в процессе использования. На этом же этапе может происходить эволюция — развитие онтологии, в том числе наполнение новыми терминами и обогащение. Развитие онтологии порождает проблемы обеспечения согласованности новых данных со старыми (от синтаксических, например, изменения формата ввода до семантических и структурных).
5. Документирование, в том числе контроль изменений и взаимного соответствия версий. В процессе документирования должна описываться процедура наполнения онтологии и использованные приёмы.

Как видно из приведенного выше списка, наполнение онтологии тем или иным образом затрагивает все этапы жизненного цикла базы знаний, но наибольшее влияние оказывает на этапы разработки и поддержки в процессе использования. Таким образом, задача исследования — это разработка метода и алгоритмов наполнения баз знаний онтологического типа для интеграции разнородных источников структурированных данных. Основными особенностями метода должны стать:

- Возможность его использования для наполнения базы знаний на различных этапах жизненного цикла.
- Возможность одновременного наполнения из нескольких гетерогенных источников.

- Интегрированность с мировыми стандартами в области языков описания онтологий для обеспечения возможности разработки и эксплуатации базы знаний с использованием существующих инструментов онтологического инжиниринга.

## Выводы по главе 1

Современные информационные системы активно используют не только базы данных, но и базы знаний. При этом ведущей моделью представления знаний является онтология. Обзор и анализ литературы, представленный в первой главе, демонстрирует, что среди множества проблем промышленного применения онтологий, важной и актуальной является задача автоматизированного наполнения баз знаний онтологического типа экземплярами, которой и посвящена данная диссертация.

Также в первой главе были даны определения основных понятий онтологического инжиниринга, рассмотрены основные классификации онтологий, приведены примеры областей применения и баз знаний.

Существующие методы наполнения баз знаний, основанных на онтологиях, обладают рядом недостатков, в частности, узкой направленностью на решение задач конкретного проекта, сложностью описания моделей источников данных и их трансформации, высокими требованиями к квалификации пользователей, обеспечением возможности интеграции данных из источников только одного типа, отсутствием модульности и расширяемости, а также удобного инструментария.

Основная сложность информационного моделирования состоит в гетерогенности, т.е. разнородности описываемых компонент, данных и знаний. Причем наибольшее влияние гетерогенность источников данных оказывает на процесс наполнения баз знаний. Проблемы, возникающие при наполнении, были классифицированы по этапам жизненного цикла базы знаний. Было выявлено, что основные трудности возникают на этапах разработки и эксплуатации.

Обзор исследований в области онтологического инжиниринга за последние 10 лет показал, что несмотря на наличие большого количества завершенных проектов и промышленных приложений онтологий имеются существенные недоработки в области теоретического осмысления программных механизмов формирования баз знаний онтологического типа. В частности, стоит проблема более глубокого изучения проблематики наполнения баз знаний с учетом семантики предметной области.

На основании обзора литературы [Simperl, Luczak-Rösch, 2014; Suárez-Figueroa, Gómez-Pérez, Fernández-López, 2012; Tempich C. et al., 2005] были выявлены основные подходы к

разработке баз знаний онтологического типа, а также проанализированы их особенности. Расширена и дополнена генеалогия языков представления онтологий. Выполнен краткий обзор языков представления правил с акцентом на язык SWRL. Также в главе представлен результат актуального анализа свободно-распространяемых инструментов создания и наполнения онтологий.

Проведенный анализ выявил ограничения существующих подходов и методов. В первой главе сформулирована задача исследования по созданию метода наполнения баз знаний онтологического типа и интеграции разнородных источников структурированных данных с учетом семантики предметной области. Основными особенностями описанного далее метода являются:

- Возможность его использования для наполнения базы знаний на различных этапах жизненного цикла.
- Возможность одновременного наполнения из нескольких гетерогенных источников.
- Интегрированность с мировыми стандартами в области языков описания онтологий для обеспечения возможности разработки и эксплуатации базы знаний с использованием существующих инструментов онтологического инжиниринга.

Глава 2 посвящена описанию разработанного нового метода наполнения онтологий и архитектуры программного прототипа.

# ГЛАВА 2 МЕТОД НАПОЛНЕНИЯ БАЗ ЗНАНИЙ ОНТОЛОГИЧЕСКОГО ТИПА

## 2.1 Наполнение онтологий

Классическая онтология ориентирована на моделирование предметных областей в понятиях классов и подклассов. По завершении моделирования классов, атрибутов, отношений и иерархии, необходимо наполнить онтологию экземплярами. Во многих крупных организациях данные или знания могут принимать различные форматы, такие как реляционные БД, веб-страницы, документация, и т.д. Чтобы сделать эту информацию доступной для общей онтологии, требуется преобразовать их из своего текущего представления в формат представления знаний.

Наполнение онтологий возможно в ручном и автоматизированном режимах. Варианты ручного наполнения БЗ рассмотрены в [Григорьев, Заблоцкий, Кудрявцев, 2012]. Для автоматизированного наполнения с точки зрения источников можно выделить:

- Использование структурированных данных (БД, таблицы, xml-файлы…) [Song, Zacharewicz, Chen, 2013];
- Использование неструктурированных данных (тексты). Применяется анализ текстов на естественном языке. Основные термины/подходы — семантическое аннотирование (semantic annotation), извлечение информации (information extraction) [Domingue, Fensel, Hendler, 2011].

Подходы к решению проблемы наполнения и обогащения онтологии зависят от уровня интеграции, свойств источников данных и требуемых способов и методов интеграции. Классификация подходов представлена на рисунке 4.


***Рисунок 4 —*** *Подходы к наполнению онтологий*

В зависимости от решаемой задачи для наполнения онтологии могут быть использованы два подхода. Первый подход — отображение по требованию (mapping on-demand) — аналогичен отображению онтологий, но устанавливается связь не между двумя или несколькими

онтологиями, а между онтологией и данными, хранящимися в распределенных источниках данных.

Подход хорошо зарекомендовал себя в контексте очень больших массивов децентрализованных данных. Он гарантирует, что возвращаемые значения всегда актуальны, так как копирование данных в онтологию не производится. Кроме того, он обеспечивает соблюдение политик управления доступом, реализованных в системе управления базами данных. С другой стороны, в случае интеграции большого количества источников данных или использования машины логического вывода для обработки сложных запросов производительность запросов к базам данных может стать существенным ограничением [Sahoo et al., 2009]. Было высказано предположение [Erling, 2008], что выразительность запросов к базе знаний должна быть ограничена, так как, в частности, переменная в позиции предиката (resourceA ?r resourceB) или переменная в позиции объекта, представляющего класс (resourceA rdf:type ?c) может приводить к эффекту «взрыва бомбы»: экспоненциальному росту количества порождаемых запросов.

Второй подход — включение данных (консолидация) в базу знаний в качестве значений свойств объектов (data materialization) — используется, в частности при построении хранилищ данных (data warehouse). После завершения процесса консолидации все необходимые данные оказываются помещенными в онтологию, что позволяет в полной мере использовать преимущества баз знаний по сравнению с базами данных, а именно инструменты логического вывода и средства для визуализации связей между объектами. Но этот подход противоречит принципу разделения кода онтологии и данных, что затрудняет разработку, а главное — поддержку онтологии. При появлении нового индивида или при изменении свойств ранее описанного индивида необходимо создать/найти соответствующий объект в базе знаний и задать/изменить значения его свойств. Для решения устаревания данных приходится регулярно запускать процесс консолидации, что приводит к необходимости нахождения компромисса между затратами на обновление данных и чувствительностью приложения к устаревшим данным. Вторым ограничением может послужить количество данных, так как полученная база знаний может превысить объем доступных ресурсов.

В зависимости от того чья структура является первичной или более важной — структура онтологии или источников данных — различают прямую интеграцию (Direct Mapping) и интеграцию, учитывающую семантику предметной области (Domain Semantics-Driven Mappings) [Michel, Montagnat, Faron-Zucker, 2014]. Прямая интеграция реализуется автоматически, причем структура результирующей онтологии полностью «наследуется» от структуры источника данных. Но структура источника данных редко может служить хорошим описанием предметной области, что является результатом как процесса оптимизации производительности, так и длительной истории обслуживания, и, в любом случае, сама схема не будет представлять собой

полное описание предметной области, так как релевантные связи часто сохраняют в коде или в кодировке различных атрибутов [Dolbear, Goodwin, 2007]. Поэтому либо результат прямой интеграции используется в качестве отправной точки для создания онтологии, более точно описывающей предметную область, например, методом выравнивания, либо применяется интеграция, учитывающая семантику предметной области. При использовании второго подхода за основу берется онтология предметной области и с помощью некоторого языка описания отображения (mapping description language), обрабатываемого специализированным (пре)процессором, задаются правила, каким образом осуществляется наполнение онтологии.

Основными типами источников структурированных данных являются реляционные базы данных, XML-документы, электронные таблицы, а также текстовые файлы с проприетарной структурой. На данный момент методы интеграции данных из источников различного типа развиваются параллельно. Рассмотрим основные из них.

### 2.1.1 Трансляция реляционных баз данных в онтологию

Для хранения структурированных источников данных чаще всего используются реляционные базы данных. В случае прямой интеграции схема базы данных напрямую переводится в онтологию с помощью специальной программы-транслятора. Правила отображения устанавливаются разработчиками, поэтому при использовании различных программ из одной базы данных можно получить целый спектр схожих, но не идентичных онтологий.

При преобразовании схемы базы данных в онтологию с учетом семантики предметной области правила отображения задаются явным образом с помощью языков описания отображения. Эти языки можно разделить на две группы. Языки первой группы существенно полагаются на SQL запросы для описания отображения данных, что потенциально является их недостатком, так как сложные случаи отображения не могут быть описаны с их помощью. С другой стороны, популярность SQL облегчает принятие этого подхода, так как нет необходимости в изучении нового языка. Языки второй группы используют специализированные языки описания отображения, что позволяет их создавать/расширять таким образом, чтобы удовлетворить любые специфические потребности, например, поиск ключевых слов и регулярных выражений. Но на практике выразительные возможности языков второй группы пока очень ограничены [Michel, Montagnat, Faron-Zucker, 2014].

Основные возможности, которые должны обеспечивать языки отображения и инструменты, их реализующие:

1. Генерация уникальных идентификаторов, определенных пользователем (способность генерировать идентификаторы ресурсов URI помимо простого использования

значений первичного ключа, например, с помощью комбинирования значений столбцов, и т.п.)

2. Логические таблицы (способность читать кортежи не только из таблиц, но и из представлений (SQLview) или из результата запроса SQL.)
3. Выбор полей (возможность выбрать только подмножество столбцов таблицы для перевода.)
4. Переименование полей (возможность отобразить столбец в свойство RDF с другим именем.)
5. Условие выбора (возможность перевода только подмножества кортежей (упорядоченный набор элементов) таблицы с помощью задания условия WHERE в операторе SELECT.)
6. Использование существующей онтологии (возможность отображения реляционных объектов в экземпляры существующих онтологий.)
7. Отображение одной таблицы в n классов (возможность использовать значения столбца в качестве шаблона категоризации: кортежи таблицы будут переведены в экземпляры различных онтологических классов на основе значения этого поля.)
8. Преобразование отношения многие-ко-многим в простые тройки (способность переводить соединенную таблицу, представляющую отношение многие-ко-многим, в простые тройки, в отличие от прямого отображения, в котором соединенная таблиц будет переведена в отдельный класс.)
9. Создание анонимных узлов (возможность создавать анонимные узлы (blank nodes) и ссылаться на них в пределах графа, полученного в процессе перевода.)
10. Трансляция типов данных (возможность обработки реляционных типов данных в типы данных RDF.)
11. Обработка данных (возможность применения функции преобразования к значениям перед формированием троек RDF, например, выполнение сложного преобразования типа, вычисление значения с помощью нескольких столбцов, обработка строк, и т.д.)

Первым языком описания отображений был язык DR2 MAP [Bizer, 2003]. Он относится к первой группе. Синтаксис языка базируется на XML. С помощью языка DR2 MAP можно описать только простые правила отображения, поэтому потребовалось его развитие, результатом которого стал язык DR2Q [Cyganiak et al., 2012]. Описание отображения на DR2Q определяется с помощью RDFS-схемы, но по-прежнему во многом опирается на SQL-фрагменты, например, чтобы использовать агрегатные функции. Этот язык используется в DR2-сервере [http://d2rq.org/; Bizer, Seaborne, 2004], который поддерживает оба типа интеграции и предоставляет различные способы получения данных. Другим последователем языка D2R был R2O [Barrasa, Corcho,

Gómez-Pérez, 2004]. Его синтаксис также базируется на XML, но выразительность этого языка намного больше, в частности, поддерживается поиск ключевых слов и регулярных выражений, арифметические операции, обработка строк, задание ограничений на диапазон значений и т.д. Язык использовался в проектах ODEMapster [http://neon-toolkit.org/wiki/ODEMapster] и DB2OWL [Cullot, Ghawi, Yetongnon, 2007]. Ввиду очень сложного синтаксиса этого языка или по какой-то другой причине разработчики отказались от его поддержки и заменили на R2RML-совместимую реализацию. Можно упомянуть еще несколько проектов, использовавших свои языки отображения, например, METAmorphoses [Svihla, Jelinek, 2004], RDBToOnto [Cerbah, 2008], Relational.OWL [http://sourceforge.net/projects/relational-owl/; De Laborda, Conrad, 2006], но их поддержка давно не осуществляется.

В 2012 году рабочая группа W3C RDB2RDF выпустила две рекомендации: A Direct Mapping of Relational Data to RDF [https://www.w3.org/TR/rdb-direct-mapping/] (Прямое отображение реляционных данных в RDF) и R2RML: RDB to RDF Mapping Language (R2RML: Язык отображения RDB в RDF) [Das, Sundara, Cyganiak, 2012; https://www.w3.org/TR/r2rml/]. Первая рекомендация, как следует из названия, регламентирует прямое отображение реляционных баз данных в RDF. Второй документ определяет язык описания отображения из реляционной базы данных в RDF, но не выдвигает никаких рекомендаций по реализации R2RML-процессора, поэтому существует ряд R2RML-совместимых инструментов с оригинальными подходами к реализации, например, Ultrawrap (http://capsenta.com/ultrawrap), DB2Triples (https://github.com/antidot/db2triples). R2RML вобрал в себя все лучшее от своих предшественников: он возник в результате их изучения, основывается на их опыте и охватывает большую часть их выразительности. Поэтому кажется, что использование R2RML неизбежно, когда речь идет о преобразовании реляционных данных в RDF. Тем более, что инструменты, появившиеся до R2RML, теперь также его поддерживают. Например, Virtuoso (http://virtuoso.openlinksw.com/dataspace/ doc/dav/wiki/Main/) содержит простой адаптер, который транслирует R2RML в синтаксис его собственного языка. Тем не менее, авторы [Michel, Montagnat, Faron-Zucker, 2014] полагают, что нельзя считать R2RML исчерпывающим решением, поэтому будут появляться все новые предложения. Это связано с тем, что некоторые подходы опираются на сложные шаблоны, которые не могут быть выражены на языке R2RML. Например, RDBToOnto [Cerbah, 2008] анализирует избыточность данных, чтобы выявить шаблоны категоризации. А в работе [Hu, Qu, 2007] применяются методы интеллектуального анализа данных для автоматического обнаружения отображений между базой данных и существующей онтологией. И, наконец, R2RML не обеспечивает функций манипулирования данными, но полагается на возможности реляционной СУБД вместо внутреннего интерфейса. К недостаткам также можно отнести необходимость изучения синтаксиса языка или создания инструмента,

облегчающего задание правил отображения на этом языке. Кроме того, не определятся, каким образом устанавливается соединение с базой данных, как проходит аутентификация и, соответственно, как интегрировать данные из нескольких баз данных одновременно.

### 2.1.2 Интеграция данных из XML-документов

XML (Extensible Markup Language) [https://www.w3.org/XML/] — это расширяемый язык разметки, простой гибкий текстовый формат, разработанный для структурирования, хранения и передачи информации между приложениями, ориентированный на использование в сети Интернет. XML-документы подразделяются на две категории: ориентированные на данные (data-oriented) и на документы (content-oriented), но только первая категория рассматривается в контексте преобразования XML-схем в онтологию.

Первые методы преобразования XML-документов в онтологию [Ferdinand, Zirpins, Trastour, 2004; García, Celma, 2005] реализовывали прямую интеграцию, затем появились методы, позволяющие влиять на преобразование XML-схемы в онтологию, например [Anicic, Ivezic, Marjanovic, 2007; Cruz, Nicolle, 2008]. Один из наиболее развитых с точки зрения предоставляемых возможностей метод описан в [Rodrigues, Rosa, Cardoso, 2006], в частности, он позволяет отображать несколько XML-схем в одну существующую онтологию, включая создание индивидов. Для этого авторы разработали собственный язык отображения, на котором описывается, каким образом XML-узлы преобразуются в элементы онтологии.

Все существующие подходы можно разделить на две группы [Hacherouf, Bahloul, Cruz, 2015]. К первой группе относятся подходы, позволяющие работать с XML-документами без XML-схемы. [O'Connor, Das, 2011; Bohring, Auer, 2015]. Вторая группа включает подходы, основанные на обработке XML-схемы [Aussenac-Gilles, Kamel, 2009; Bakkas, Jakjoud, Bahaj, 2014; Quix, Kensche, Li, 2007]. Необходимо отметить, что ни один из предложенных методов не стал стандартом и в каждом проекте, где требуется реализовать преобразование XML-документов в онтологию, используется свой инструмент, разработанный специально для нужд этого проекта и учитывающий его особенности.

### 2.1.3 Преобразование таблиц в онтологию

Несмотря на то, что электронные таблицы являются превалирующим инструментом для представления и обработки структурированных данных, вопросу преобразования таблиц в онтологию посвящено относительно мало работ. Можно предположить, что это связано с тем, что таблицы могут быть представлены в форме реляционных баз данных, для которых уже разработан ряд методов преобразования в онтологию. Тем не менее, необходимость дополнительного преобразования для проектов, в которых используется множество

разрозненных таблиц различной структуры, может стать существенным ограничением, поэтому создание метода преобразования таблиц в онтологию является целесообразным.

В работе [Tijerino et al., 2005] предлагается подход TANGO (Table ANalysis for Generating Ontologies), суть которого состоит в следующем: на первом шаге таблицы «канонизирутся», т.е. осуществляется преобразование в стандартизованную форму, пригодную для дальнейшего анализа; на втором шаге создается набор «мини-онтологий», каждая из которых описывает структуру одной канонизированной таблицы; затем осуществляется слияние «мини-онтологий» в одну результирующую онтологию. Таким образом, подход TANGO реализует прямую интеграцию, и созданная в результате применения этого подхода онтология описывает структуру таблиц и в общем случае не может считаться онтологией предметной области.

Основная трудность при получении данных из таблиц для наполнения онтологии заключается в том, что структура таблицы может быть произвольной. Первые методы наполнения исходили из предположения, что таблица представлена в «стандартизированном» виде, т.е. в первой строке содержатся заголовки столбцов, а под ними располагаются данные (аналогично представлению таблиц в базах данных), но структура таблиц, использующихся на практике, редко соответствует этому предположению [O'Connor, Halaschek-Wiener, Musen, 2010]. Более поздние подходы пытались преодолеть это предположение, импортируя ячейки как отдельные объекты или блоки [Abraham, Erwig M, 2004; Ozturk, 2020] или добавляя семантическую структуру в таблицу [Nederstigt et al., 2014; O'Connor, Halaschek-Wiener, Musen, 2010; Han L. et al., 2008; Langegger, Wöß, 2009].

## 2.2 Метод автоматизированного наполнения онтологий из структурированных источников данных

В данной работе разработана методология и технология наполнения онтологии на основе интеграции с гетерогенными структурированными источниками данных (с учетом их семантики) из структурированных источников данных (метод METEOP).

Идея метода METEOP заключается в следующем: пользователь хочет импортировать данные из разных структурированных источников (таких как реляционные базы данных, XML-документы, электронные таблицы и т.п.) в единую онтологию предметной области (ОПО). Для этого ему необходимо выполнить два действия:

- Первое — описать структуру источников данных с помощью Онтологии источников данных (ОИД).
- Второе — сформулировать правила отображения для соответствующих сущностей из источников данных и ОПО, которую необходимо заполнить.

Оба действия выполняются с помощью системы, которая берет данные, извлекает их структуру и вставляет данные в ОПО. Архитектура этой системы описана далее в параграфе «2.2.1 Архитектура системы». Подробно этапы работы системы описаны в параграфе «2.2.2 Процесс наполнения онтологии предметной области из структурированных источников данных».

### 2.2.1 Архитектура системы

Предлагаемый подход к наполнению онтологий основан на технологии извлечения, преобразования и загрузки (ETL: extract-transform-load). Соответственно, архитектура системы, реализующей предлагаемый метод наполнения онтологий, сходна с архитектурой ETL-систем [Паклин, Орешков, 2013], так как они выполняют аналогичные задачи (см. Рисунок 5 —). Основное отличие заключается в том, что получателем данных является не хранилище или база данных, а онтология или база знаний, что обуславливает и остальные особенности. Этапы основаны на [Vassiliadis, Simitsis, 2009] и адаптированы к базам знаний следующим образом:

**Извлечение**: источники данных для вставки в онтологию идентифицируются и специфицируются, а данные извлекаются.

**Преобразование**: очистка данных и разрешение конфликтов выполняется вместе с мэппингом (отображением) структуры данных на онтологию предметной области (т.е. структуру базы знаний).

**Загрузка**: данные из источников вставляются в соответствующее место в базе знаний (т.е. осуществляется наполнение ОПО).

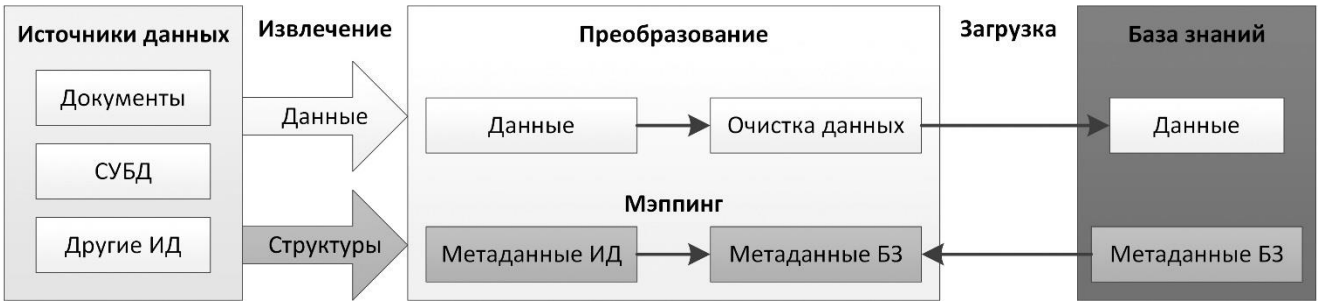


***Рисунок 5 —*** *Адаптация ETL-технологии для наполнения онтологических баз знаний*

**Обобщенная структура наполнения базы знаний**

С точки зрения процесса наполнения онтологии архитектуру системы можно представить в виде пяти компонентов (Рисунок 6 —), таких как:

- источники данных — содержат структурированные данные в виде совокупности таблиц и/или файлов, данные в которых, например, упорядочены в типизированные столбцы, отделенные друг от друга некоторыми символами-разделителями;
- промежуточная область — содержит вспомогательные таблицы, соответствующие источникам данных, и вспомогательный граф знаний — набор триплетов, базирующийся на мета-шаблонах — создаваемый исключительно для организации процесса наполнения;
- получатель данных — онтология, в которую должны быть помещены извлеченные данные;
- правила обработки — требования к организации потоков данных описываются аналитиком с помощью мета-шаблона;
- редактор онтологий — используется в качестве инструмента при создании онтологии и при описании правил обработки или отображения.


***Рисунок 6 —*** *Элементы архитектуры наполнения онтологической базы знаний*

**Поток данных при наполнении онтологии**

Для создания вспомогательных таблиц используется Power Query — технология подключения к данным, доступная в классических версиях MS Excel и Power BI.

Вспомогательный граф знаний создается с помощью онтологии источников данных. Он содержит структуру данных, которые будут обрабатываться в соответствии с правилами, установленными совместно инженером по знаниям и экспертом предметной области. База знаний создается после наполнения онтологии предметной области.

Особенностью подхода является то, что правила обработки задаются инженером по знаниям при взаимодействии с экспертом предметной области в виде онтологии, что не требует специальных инструментов, навыков программирования или знания специализированных языков описания отображений.

Подход реализован в виде четырех модулей, работа двух из которых требует ручного ввода от пользователя (Рисунок 7 —). Модули служат для выполнения следующих функций:

**Модуль спецификации данных**. Этот модуль создает спецификацию источников данных, используя информацию о соответствующих источниках данных и их местонахождении, предоставленную пользователем.

**Модуль извлечения структуры данных**. Модуль обращается к источникам данных, затем извлекает их структуру и загружает ее в ОИД вместе с информацией об источнике данных, из которого были извлечены данные. Результатом этого действия является заполненная ОИД, то есть вспомогательный граф знаний, который будет использоваться для интеграции данных из соответствующих источников в ОПО.

**Модуль отображения** устанавливает соответствие между объектами из вспомогательного графа и ОПО, которая в результате должна быть наполнена. Для этого используются правила отображения и информация о приоритетах источников данных, предоставленные пользователем.

**Модуль наполнения** извлекает данные из источников данных в соответствии с их структурой, хранящейся во вспомогательном графе знаний, выполняет проверки согласованности и избыточности и вставляет их в ОПО.

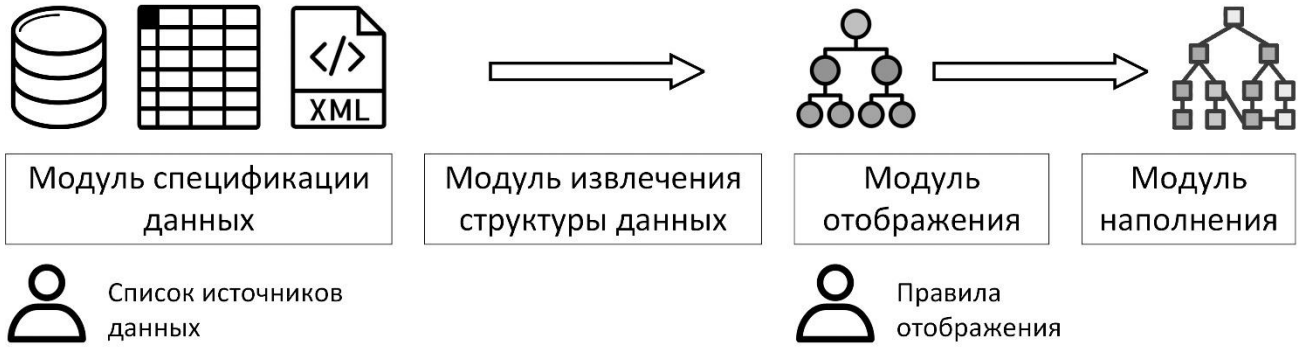


***Рисунок 7 —*** *Реализация метода с помощью 4 модулей*

Основными модулями являются модуль извлечения структуры данных и модуль наполнения. Они реализованы как рабочий прототип на Python и могут служить расширением для любого редактора онтологий (автор, в основном, использует Protégé [Musen, 2015; https://protege.stanford.edu/]). Такая архитектура была выбрана потому, что модуль спецификации данных и модуль отображения требуют ручного ввода от пользователя, который может быть осуществлен в любом онтологическом редакторе, с которым знаком пользователь. Таким образом обеспечивается возможность использования знакомого пользователю программного обеспечения.

### 2.2.2 Процесс наполнения онтологии предметной области из структурированных источников данных

**Компоненты, необходимые для применения метода:**

1. Онтология предметной области.
2. Онтология источников данных.
3. Модули, описанные в предыдущем параграфе.

**Онтология предметной области**, используемая организацией для решения определенных задач или получения ответов на заранее определенные вопросы. Именно эта онтология будет наполнена в результате использования метода. Для создания онтологии используются экспертные знания, существующие онтологические и неонтологические источники знаний, вопросы компетенции и другая информация в соответствии с выбранной методологией создания онтологий. Разработка онтологии предметной области выходит за рамки метода.

Базовая **Онтология источников данных** является мета-шаблоном для описания структуры и свойств источников данных следующих типов: электронные таблицы, реляционные базы данных и документы XML. На рисунке 8 отражены все ее существенные классы и отношения (для визуализации использовалась нотация из книги «Demystifying OWL for the Enterprise» [Uschold, 2018]). Эта онтология основана на спецификациях и на онтологиях, найденных в открытом доступе. Далее описаны основные классы такой онтологии.

Класс **DataSource** — это класс верхнего уровня, объединяющий разные типы источников данных. Любой источник данных имеет название и местоположение (путь к нему). Класс **DataSource** включает такие подклассы как **Workbook** (для табличных источников данных), **Database** (для реляционных баз данных) и **XML** (для xml-документов).

Экземпляры класса **Workbook** связаны отношением *hasPart* с экземплярами класса **WBSheet**, которые связаны таким же отношением с экземплярами класса **WBTable**, а они, в свою очередь, — с экземплярами класса **WBColumn**. Экземпляры всех этих классов характеризуются своим именем (свойство *hasName*). С помощью этого мета-шаблона можно описать любую рабочую книгу редактора электронных таблиц MS Excel.

Для описания структуры реляционных баз данных используются классы **Database**, **DBTable**, **DBColumn**, а также классы для описания ключей — **Key**, **PrimaryKey** и **ForeignKey**. Для описания структуры XML-документов используются классы **XML**, **XMLElement** и **Atribute**.

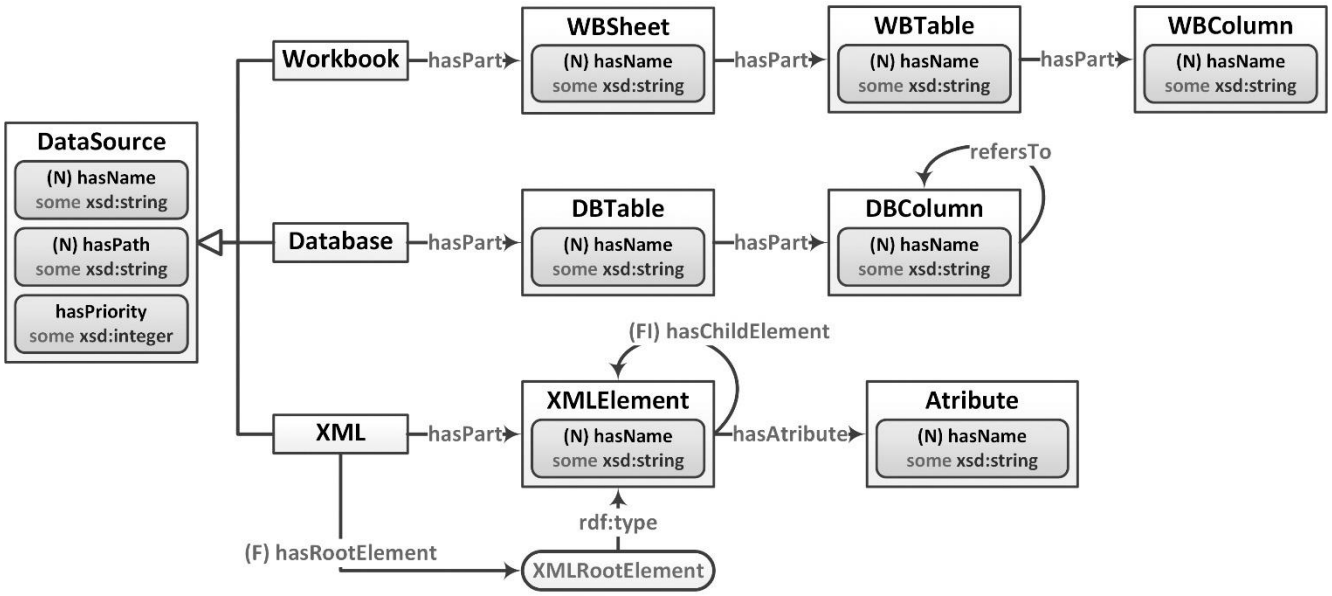


***Рисунок 8 —*** *Базовая онтология источников данных*

Онтология источников данных расширяемая — при необходимости в нее добавляются подклассы класса **DataSource** (по одному для каждого типа источников данных), а также другие классы и отношения, требующиеся для описания мета-структур других типов источников данных.

Также эта онтология задает необходимые аннотационные свойства, которые используются для описания отображения вспомогательного графа в ОПО на шаге 4.

**Процесс наполнения:**

Процесс наполнения онтологии предметной области включает пять шагов (рисунок 9). Для каждого шага кратко описан основной процесс, а также вход и выход.

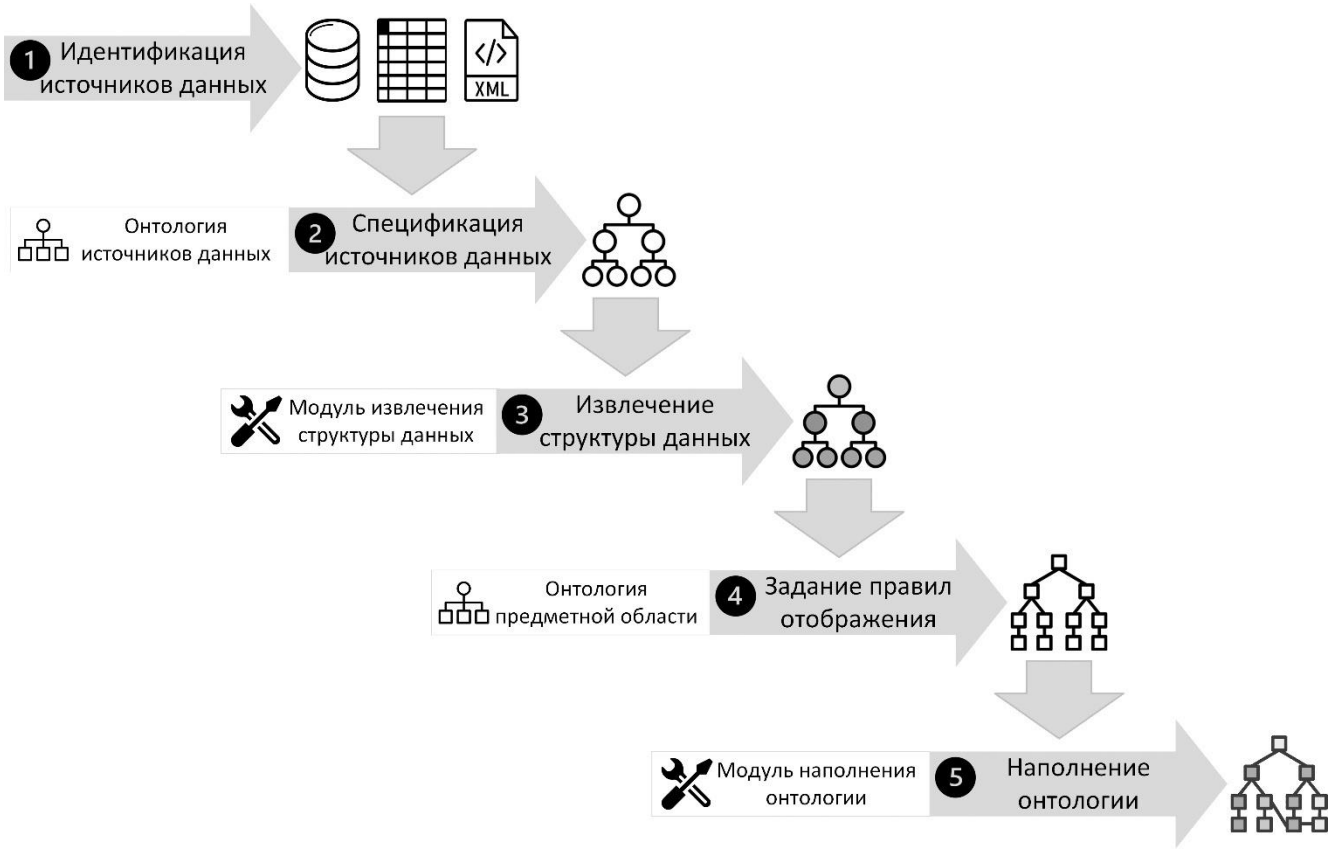


***Рисунок 9 —*** *Схема процесса наполнения онтологии предметной области*

**Шаг 1**: Идентификация источников данных, из которых будет наполняться онтология. На этом этапе должны быть определены все источники данных, данные в которых имеют отношение к предметной области наполняемой онтологии (т.е. источники данных, необходимые для решения задач и ответов на вопросы, для которых была создана онтология). Их идентификация также выходит за рамки метода — метод не ограничивает типы данных, поскольку может обрабатывать данные в различных структурированных форматах (например, документы XML, электронные таблицы и базы данных).

В случае необходимости интеграции данных из плохо структурированных электронных таблиц или для упрощения дальнейших операций, требующих ручного ввода от пользователя, создаются вспомогательные таблицы. Для создания вспомогательных таблиц используется Power Query — технология подключения к данным, доступная в последних версиях MS Excel и MS Power BI. С помощью этой технологии данные из различных источников могут быть объединены, и их структура трансформирована при необходимости.

***Вход***: экспертные знания, описание организационных процессов, хранилища данных.

***Выход***: список источников данных, данными из которых будет наполняться ОПО.

**Шаг 2**: Спецификация источников данных. Спецификация — это описание источников данных. Спецификация данных требует ручного ввода пользователя. Онтология источников данных (описанная ранее) используется в качестве мета-шаблона для описания структуры и

характеристик данных и дальнейшего импорта в онтологию предметной области — таким образом, нет необходимости в специализированном языке отображения. Для каждого источника данных необходимо создать экземпляр соответствующего подкласса класса **DataSource**, а также обязательно должны быть заполнены свойства *hasName* (имя источника данных) и *hasPath* (путь к источнику данных). На этом этапе также должна быть предоставлена информация о приоритетах источников данных, если их несколько — свойство *hasPriority* (для разрешения конфликтов при наполнении онтологии).

В случае необходимости интеграции данных из источников, тип которых отличается от наиболее широко используемых (электронные таблицы, реляционные базы данных и документы XML (описание которых уже задано в базовой ОИД)), выполняется расширение базовой Онтологии источников данных.

В случае выявления на шаге 1 источников данных «нестандартных» типов, для которых не могут быть построены вспомогательные таблицы, на этом шаге в ОИД должны быть добавлены подклассы класса **DataSource** (по одному для каждого «нестандартного» типа), экземпляры которых будут соответствовать источникам данных соответствующего типа, а также все классы и свойства, необходимые для описания всех «нестандартных» типов источников данных, выявленных на первом шаге. Также необходимо расширить классы модуля извлечения структуры данных и модуля наполнения, которые будут отвечать за обработку данных из «нестандартных» источников.

***Вход***: список источников данных, определенных на шаге 1; базовая ОИД.

***Выход***: единый файл с онтологическим описанием идентифицированных источников данных.

**Шаг 3**: Извлечение структуры данных. Структура данных из разных источников данных извлекается и объединяется в единую структуру (автоматически с помощью модуля извлечения структуры данных).

***Вход***: файл с описанием источников данных из шага 2; источники данных; модуль извлечения структуры данных.

***Выход***: вспомогательный граф знаний, созданный на основе структур источников данных и мета-шаблона для их описания.

**Шаг 4**: Задание правил отображения. Вспомогательный граф знаний, созданный на предыдущем шаге, отличается от онтологии предметной области по крайней мере в двух аспектах. Во-первых, имена экземпляров и значения свойств во вспомогательном графе знаний берутся из данных и отличаются от имен в онтологии предметной области. Во-вторых, структуры вспомогательного графа (и источников данных) отличаются от структуры онтологии предметной области. Чтобы правильно импортировать данные, необходимо выполнить сопоставление между

ОПО и вспомогательным графом знаний, построенным на основе ОИД. Это сопоставление выполняется инженером по знаниям вместе с экспертом предметной области посредством создания правил отображения с использованием набора аннотационных свойств из ОИД. Подробное описание этих аннотационных свойств дано в следующем параграфе «Аннотационные свойства для описания отображения ОИД в ОПО».

***Вход***: онтология предметной области; вспомогательный граф знаний.

***Выход***: правила отображения между вспомогательным графом знаний и ОПО (заданные в ОПО).

**Шаг 5**: Наполнение онтологии. На предыдущем шаге классы в ОПО были связаны с соответствующими сущностями во вспомогательном графе знаний с помощью аннотационных свойств и их значений. Экземпляры во вспомогательном графе знаний отражают структуру импортируемых данных, но сами данные хранятся в описанных источниках данных. Чтобы вставить значения в ОПО, модуль наполнения обрабатывает аннотационные свойства (которые задают отображение структур источников данных, хранящихся во вспомогательном графе знаний, в концепты ОПО) и импортирует данные в ОПО.

***Вход***: онтология предметной области, содержащая правила отображения, вспомогательный граф знаний, источники данных.

***Выход***: база знаний; лог-файл с информацией о данных, выбранных для вставки в случае конфликтов, и другой вспомогательной информацией.

При этом в процессе наполнения создается иерархия онтологий (см. рисунок 10). Сплошная рамка вокруг прямоугольника, обозначающего онтологию, означает, что включение в онтологию более высокого уровня выполняется с помощью импортирования, а пунктирная линия показывает, что осуществляется изменение в самой онтологии (например, добавление аннотационных свойств или экземпляров). Так как онтология структуры источников данных импортируется в ОПО, то впоследствии ее легко можно исключить из ОПО, удалив команду импорта.


***Рисунок 10 —*** *Иерархия онтологий в процессе наполнения*

### 2.2.3 Аннотационные свойства для описания отображения ОИД в ОПО

Для описания правил наполнения используются аннотационные свойства, которые назначаются классам ОПО.

С точки зрения результирующей онтологии, ее наполнение — это:

- Создание экземпляров класса
- Задание значений объектных свойств (ObjectPropertyAssertion)
- Задание значений слотов (или атрибутов) (DataPropertyAssertion)
- Задание аннотаций (AnnotationPropertyAssertion)

### Создание экземпляров класса

Два аннотационных свойства дают указания модулю наполнения онтологии создать экземпляры класса: *getInstancesFrom* и *combineInstancesFrom*. К созданным экземплярам добавляются аннотации, в которых указывается, какое значение вызвало создание этого экземпляра, а также из какого источника это значение было взято.

***getInstancesFrom***

Значением свойства может быть любой экземпляр из ОИД, принадлежащий классу, описывающему объекты структуры источников данных, непосредственно содержащие однородные данные (например, WBColumn или Atribute). Каждое уникальное значение порождает новый экземпляр в аннотируемом классе, и его значение становится значением свойства rdfs:label порожденного экземпляра. Пререквизиты отсутствуют.

***combineInstancesFrom***

Обычно имеет 2 или более значений. Значением свойства могут быть любые классы, экземпляры которых должны быть скомбинированы для того, чтобы был создан объект описываемого класса. В результате обработки этого свойства создаются экземпляры, используя декартово произведение экземпляров соответствующих классов. Созданные экземпляры связываются с исходными экземплярами свойствами, заданными с помощью вложенного аннотационного свойства *useObjectProperty*. Пререквизиты: экземпляры комбинируемых классов должны быть созданы заранее.

### Задание объектных свойств

Для описания связывания экземпляров отношением (с помощью объектного свойства) служит свойство *makeReferenceTo*. Значением свойства может быть любой класс, содержащий экземпляры, созданные ранее с помощью свойств *getInstancesFrom* или *combineInstancesFrom*, которые должны стать значением создаваемых объектных свойств. Эти объектные свойства аннотируются с помощью свойства rdfs:label, значение которого указывает на источник данных, содержащий информацию о порожденном отношении. Ссылка на объектное свойство задается с помощью вложенного аннотационного свойства *useObjectProperty*. Пререквизиты: экземпляры связываемых классов должны быть созданы заранее.

### Задание значений слотов

Для задания атрибутов экземпляров класса используется свойство *getValuesFrom*. Значением свойства может быть любой экземпляр из ОИД, принадлежащий классу, описывающему объекты структуры источников данных, непосредственно содержащие

однородные данные (например, WBColumn или Atribute). Свойства аннотируются с помощью свойства rdfs:label, значение которого указывает на источник данных, содержащий информацию о порожденном отношении. Ссылка на свойство задается с помощью вложенного аннотационного свойства *useDataProperty*. Пререквизиты: экземпляры аннотируемого класса должны быть созданы заранее.

### 2.2.4 Иллюстрация процесса наполнения онтологии на примере

Пререквизиты:

- Онтология предметной области. Создание ОПО выходит за рамки метода. Пример ОПО, которую необходимо наполнить, показан на рисунке 11.

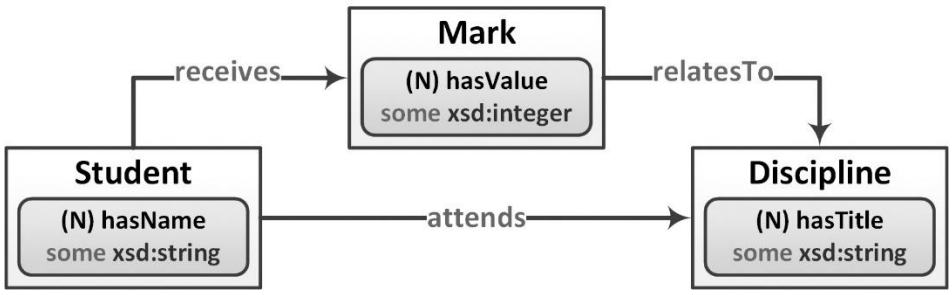


***Рисунок 11 —*** *Фрагмент онтологии предметной области*

- Базовая ОИД показана на рисунке 8.

**Шаг 1**: Идентификация источников данных. Этот шаг выходит за рамки метода и должен выполняться администратором в соответствии с его или ее знаниями о задаче. В примере будут использованы два разных источника данных об успеваемости студентов: таблица, представленная на рисунке 12, и фрагмент базы данных, схема и содержимое которого показаны на рисунке 13.

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **Образовательная программа** | | | Менеджмент | | | | | | | |
| 2 | **Год поступления** | | | 2017 | | | | | | | |
| 3 | | | | | | | | | | | |
| 4 | | | | | | **Семестр 1** | | | **Семестр 2** | | |
| 5 | **№** | **Фамилия** | **Имя** | **Отчество** | **Группа** | Математика | История бизнеса | Макроэкономика | Статистика | Маркетинг | Микроэкономика |
| 6 | st061234 | Аристов | Иван | Иванович | 17.Б01 | 5 | 3 | 4 | 5 | 4 | 5 |
| 7 | st061345 | Коваль | Елена | Олеговна | 17.Б04 | 4 | 5 | 5 | 4 | 5 | 4 |

***Рисунок 12 —*** *Пример таблицы успеваемости студентов*

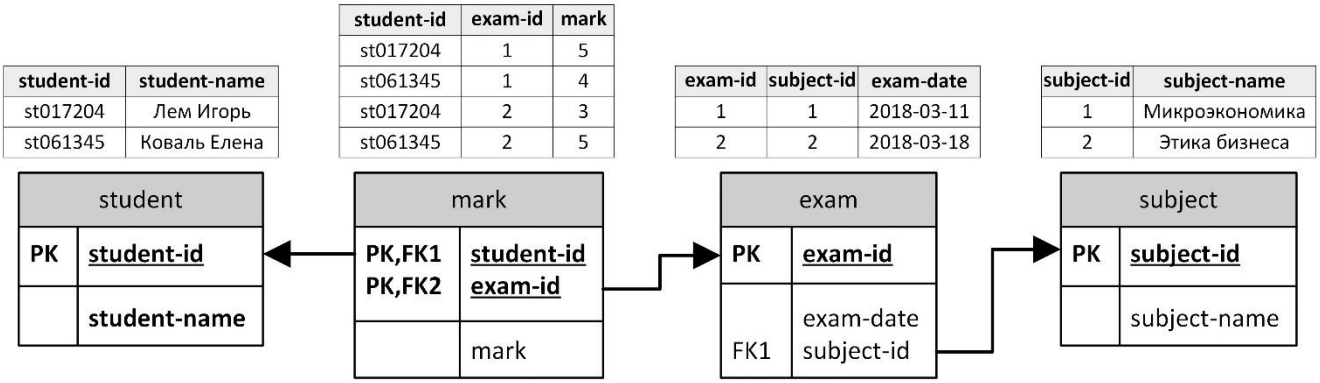


***Рисунок 13 —*** *Фрагмент базы данных успеваемости студентов*

С помощью технологии PowerQuery таблица, показанная на рисунке 12, была преобразована к «прямому» виду, показанному на рисунке 14. Столбцы **Образовательная программа** и **Год поступления** не были добавлены, так как не несут существенной информации для приводимого примера, но если бы стояла задача сравнения успеваемости студентов различных программ или потоков, то добавление этих столбцов легко осуществимо. Данные из базы данных также можно преобразовать в подобную таблицу при желании для наглядности, но это не обязательно.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | st | ФИО | Предмет | Оценка |
| 2 | st061234 | Аристов Иван Иванович | Математика | 5 |
| 3 | st061234 | Аристов Иван Иванович | История бизнеса | 3 |
| 4 | st061234 | Аристов Иван Иванович | Макроэкономика | 4 |
| 5 | st061234 | Аристов Иван Иванович | Статистика | 5 |
| 6 | st061234 | Аристов Иван Иванович | Маркетинг | 4 |
| 7 | st061234 | Аристов Иван Иванович | Микроэкономика | 5 |
| 8 | st061345 | Коваль Елена Олеговна | Математика | 4 |
| 9 | st061345 | Коваль Елена Олеговна | История бизнеса | 5 |
| 10 | st061345 | Коваль Елена Олеговна | Макроэкономика | 5 |
| 11 | st061345 | Коваль Елена Олеговна | Статистика | 4 |
| 12 | st061345 | Коваль Елена Олеговна | Маркетинг | 5 |
| 13 | st061345 | Коваль Елена Олеговна | Микроэкономика | 4 |

***Рисунок 14 —*** *Данные из таблицы успеваемости после преобразования*

**Шаг 2**: Спецификация источников данных. На этом шаге в ОИД добавляются экземпляры, описывающие источники данных, выявленные на первом шаге. Применительно к описываемому примеру, добавляются экземпляр класса **Workbook** и экземпляр класса **Database**, у которых указываются значения атрибутов *hasName*, *hasPath* и *hasPriority*. Для экземпляра класса **Database** также можно задать значение свойства *queryText*, которое будет использовано при извлечении

данных из базы данных, если на первом шаге было принято решение не создавать для нее вспомогательную таблицу. Значение свойства *queryText* для описываемого примера:

SELECT * FROM mark LEFT JOIN student USING(`student-id`) LEFT JOIN exam USING(`exam-id`) LEFT JOIN subject USING(`subject-id`);

**Шаг 3**: Извлечение структуры данных. Выполняется автоматически. На этом этапе структура (например, имена столбцов) отмеченных источников данных вставляются в ОИД как отдельные экземпляры соответствующих классов. В результате создаются все экземпляры и отношения между ними, описывающие структуру источников данных, специфицированных на предыдущем шаге. Важно отметить, что сами данные находятся в исходных источниках данных и не импортируются в ОИД. Однако связи между структурой (например, именем столбца) и данными (значениями в столбце) сохраняются и будут использоваться на этапе наполнения онтологии. Результат работы модуля извлечения структуры данных — структура, показанная на рисунке 15. Значения на рисунке — это структура источников данных (имена таблиц и столбцов), а не сами данные.

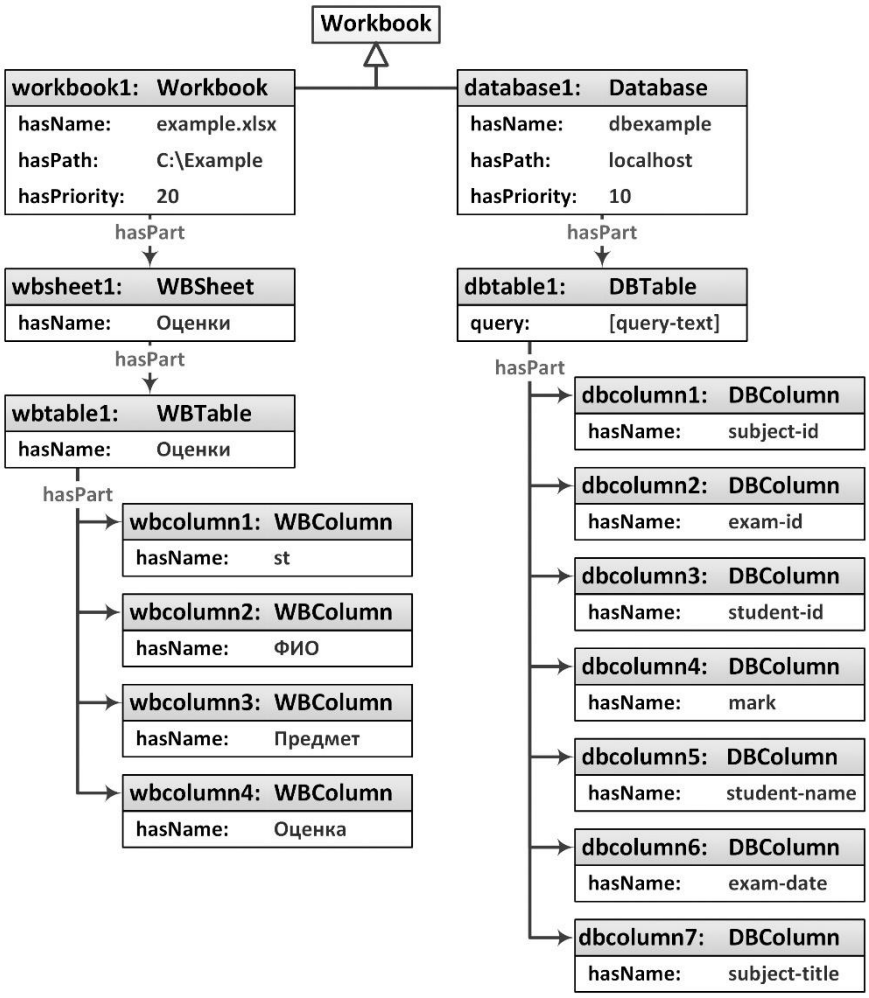


***Рисунок 15 —*** *Результат шага 3: описание структуры данных*

**Шаг 4**: Задание правил отображения. Зададим следующие аннотационные свойства:

Класс **Student**:

getInstancesFrom: wbcolumn1

getValuesFrom: wbcolumn2

useDataProperty: hasName

getInstancesFrom: dbcolumn3

getValuesFrom: dbcolumn5

useDataProperty: hasName

makeReferenceTo: Discipline

useObjectProperty: attends

makeReferenceTo: Mark

useObjectProperty: receives

Класс **Discipline**:

getInstancesFrom: wbcolumn3

getValuesFrom: wbcolumn3

useDataProperty: hasTitle

getInstancesFrom: dbcolumn7

getValuesFrom: dbcolumn7

useDataProperty: hasTitle

Класс **Mark**:

combineInstancesFrom: Student

combineInstancesFrom: Discipline

getValuesFrom: wbcolumn4

useDataProperty: hasValue

getValuesFrom: dbcolumn4

useDataProperty: hasValue

Эти свойства полностью описывают трансформацию структуры источников данных в структуру онтологии. Значения свойств соответствуют экземплярам из ОИД (то есть именам элементов структуры данных, которые должны быть загружены в онтологию предметной области).

**Шаг 5**: Наполнение онтологии.

Процесс наполнения, специфицированный на предыдущем шаге с помощью аннотационных свойств, выполняется по очереди в последовательности, показанной на рисунке 16. Результат заполнения онтологии показан на рисунке 17.

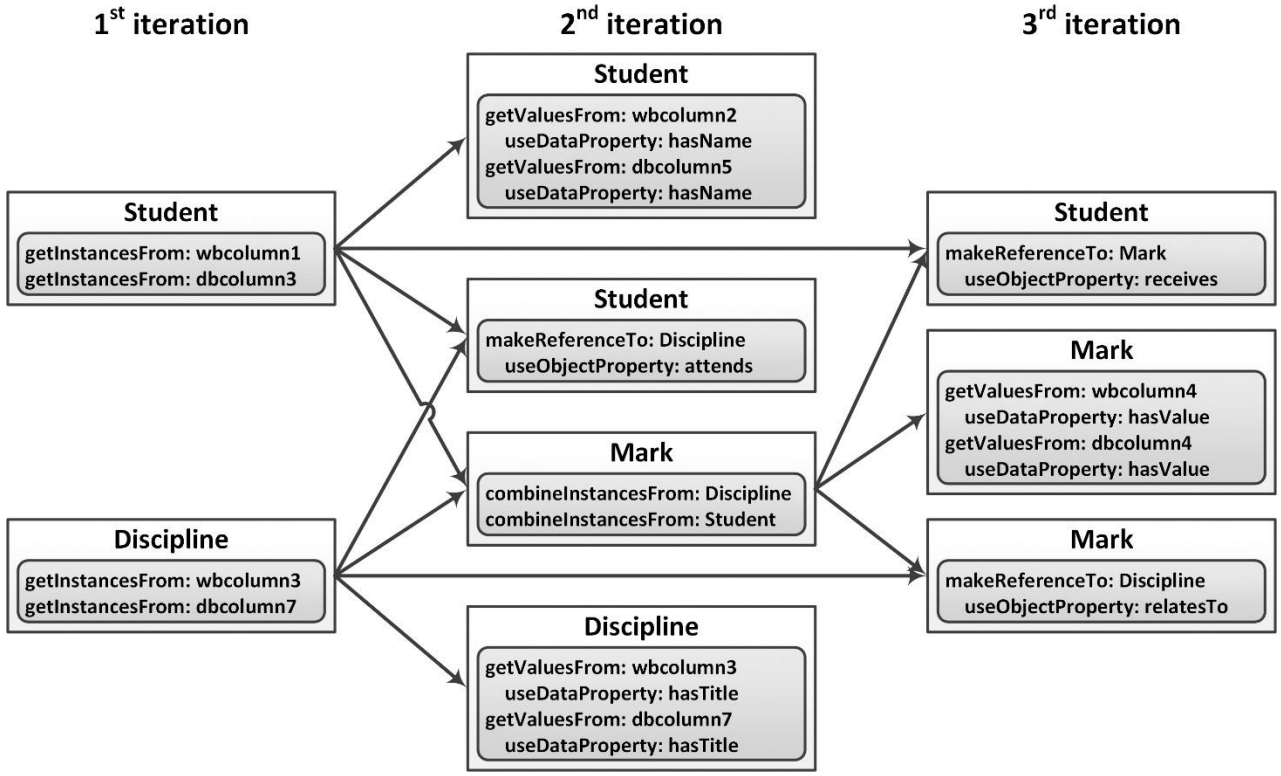


***Рисунок 16 —*** *Порядок наполнения онтологии для описываемого примера*

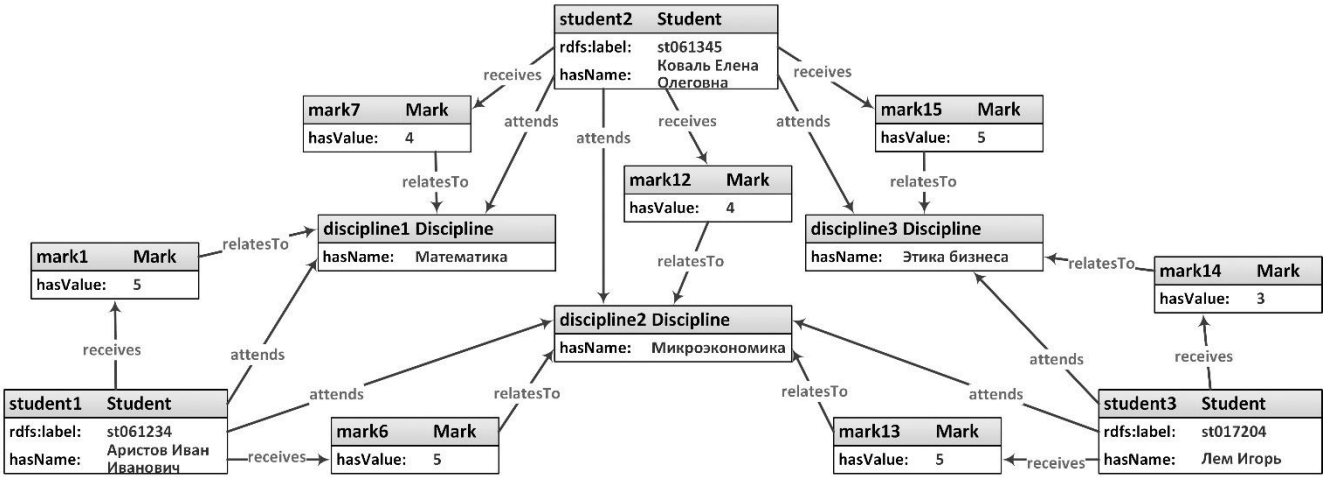


***Рисунок 17 —*** *Результат заполнения онтологии*

## Выводы по главе 2

Глава 2 посвящена разработке нового метода наполнения онтологий и архитектуре программного прототипа. Несмотря на то, что современные исследования предлагают большое количество подходов к интеграции данных в онтологии [Baghernezhad-Tabasi S. et al., 2021; Chasseray Y. et al., 2021; Lubani, Noah, Mahmud, 2019; Clarkson K. et al., 2018; Nederstigt et al., 2014; Petasis et al, 2011; Valarakos et al., 2004], большинство из них обладают рядом ограничений. Так, основные недостатки существующих решений заключаются в:

1. Необходимости использования и, соответственно, изучения синтаксиса нескольких специфических языков для описания отображений. Также, для эффективной работы понадобится создание удобного инструмента для каждого из используемых языков.
2. Узости методов: предложено большое количество различных методов для интеграции данных в онтологию, но каждый метод работает только для одного типа источника данных.
3. Отсутствии рекомендаций по интеграции данных из нескольких источников одновременно. Если есть необходимость интегрировать в одну онтологию данные, например, из базы данных и из XML-документа, то, используя существующие методы, придется действовать последовательно. Например, наполнить исходную онтологию данными из БД, затем полученную онтологию — данными из XML-документа. Так как данные в источниках подвергаются изменениям, то эту процедуру необходимо регулярно повторять, что ведет к неоправданным издержкам.
4. Наличии противоречий на этапе интеграции данных из различных источников. Так как авторы вышеприведенных методов фокусируются на интеграции данных из одного источника, то вопрос разрешения подобных противоречий не рассматривается.
5. Неясности, каким образом устанавливается соединение с источником данных, как проходит аутентификация и т.п.
6. Отсутствии рекомендаций ПО в случае, если компании могут использовать проприетарные (частные) форматы хранения данных, и в будущем могут появиться новые.

Анализ этих проблем позволил сделать вывод о необходимости создания нового метода, позволяющего интегрировать данные из источников различного типа (например, баз данных, электронных таблиц и XML-файлов). Создание такого метода снижает трудоемкость наполнения и обогащения онтологий. Метод использует накопленные массивы информации независимо от формы их хранения и представления.

Предлагаемая методика направлена на осуществление однотипных операций или на решение повторяющихся задач, требующих знаний правил и процедур, а также сопоставления информации из различных источников. Такого рода задачи возникают на предприятиях, в учебных заведениях, а также в организациях различной ведомственной принадлежности в рамках цифрового документооборота. Предложенная методика может быть положена в основу систем поддержки принятия решений, основанных на знаниях.

Например, в системах электронной коммерции широкое распространение получили рекомендующие системы. Используются различные подходы к составлению списка рекомендаций: коллаборативные рекомендации, рекомендации на основе содержимого,

рекомендации на основе знаний. Последний подход может быть реализован, в частности, с помощью предложенного метода. В онтологию закладываются знания о критериях выбора, совместимости объектов и т.п. Характеристики имеющихся объектов могут храниться в базе данных или отдельных файлах. Информация о предпочтениях пользователя может содержаться в запросе. По запросу данные об объектах «втягиваются» в онтологию, анализируются в соответствии с заложенными правилами и на основе проведенного анализа формируется рекомендация для конкретного пользователя.

Новизна предложенного метода заключается в использовании вспомогательной Онтологии источников данных (ОИД), которая может быть расширена за счет добавления дополнительных типов источников данных. Использование ОИД дополнено разработкой требований к формированию правил отображения, позволяющих связать ОИД и конкретную наполняемую онтологию предметной области (ОПО) для наполнения ее экземплярами. Предложенный метод лишен всех указанных в начале параграфа недостатков (первые четыре отсутствуют в силу алгоритма, для двух последних проблем предложен механизм их решения).

Применение разработанного метода METEOR (МЕтодология и ТЕхнология наполнения Онтологии на основе интегРации с гетерогенными структурированными источниками данных) можно описать алгоритмом, состоящим из 5 шагов, два из которых выполняются автоматически:

1. Идентификация источников данных.
2. Спецификация источников данных с помощью вспомогательной ОИД.
3. Извлечение структуры данных из источников данных в ОИД.
4. Задание правил отображения в ОПО.
5. Наполнение ОПО.

Ключевым шагом алгоритма наполнения ОПО является задание правил отображения. Оно осуществляется с помощью встроенного в OWL механизма аннотирования. В работе введены новые аннотационные свойства: 4 основных и 2 вспомогательных (подробнее см. параграф 2.2.3). Такой подход позволил исключить необходимость в дополнительной инструментальной поддержке метода METEOR, так как аннотирование может быть выполнено с помощью любого редактора онтологий.

Также была разработана новая архитектура программного прототипа, реализующего предложенный метод METEOR. В основу архитектуры положена известная архитектура ETL-систем. Особенностью предлагаемой архитектуры является то, что получателем данных является не хранилище или база данных, а онтология или база знаний.

Разработанный прототип реализует созданный метод METEOR и позволяет наполнять базы знаний онтологического типа из гетерогенных источников. Программный прототип включает 4 модуля, поддерживающих процессы спецификации источников данных, извлечения

структуры данных, задания правил отображения и собственно наполнения ОПО. Весь процесс наполнения онтологии с помощью метода METEOR проиллюстрирован на условном примере. Следующая глава посвящена практическому применению метода METEOR для наполнения 4 онтологий из различных предметных областей (медицинская диагностика орфанных заболеваний, сборочное производство, поиск данных эмпирических исследований и администрирование учебных процессов).

# ГЛАВА 3 ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ МЕТОДА НАПОЛНЕНИЯ ОНТОЛОГИЙ

## 3.1. Наполнение онтологии по обработке эмпирических исследований EMPIRION и его особенности

Предлагаемый в диссертации метод наполнения онтологии был апробирован в рамках проекта «Формирование баз знаний на основе данных эмпирических исследований: онтологический подход (ЭМПИРИОН)» (РФФИ № 20-07-00854). Была разработана онтология для описания данных эмпирических исследований (далее, empirion) [Leshcheva, Begler, 2020].

Эта онтология позволила обеспечить соблюдение основных традиционных принципов организации хранения и доступа к данным [Jacobsen и др., 2020a; Jacobsen и др., 2020b; Wilkinson, 2016]:

- Findability: обеспечить находимость данных и метаданных;
- Accessibility: обеспечить возможность доступа к данным;
- Interoperability: обеспечить возможность интеграции данных с другими данными используя общепринятые схемы метаданных;
- Reusability: обеспечить возможность повторного использования данных.

Онтология empirion обеспечивает хранение как общих свойств массива данных, включая дату создания, название, авторов, описание и т.п., так и его встроенные метаданные, например, заголовки столбцов с переменными, диапазоны их значений и способы кодирования, единицы измерения и т.п.

Основой массивов данных эмпирических исследований являются переменные, обычно представленные в виде столбцов данных. С точки зрения структуры онтологии все переменные можно разделить на 3 типа:

a. переменные, значения которых задаются некоторым списком, например, {correct, incorrect};
b. переменные, у которых числовые значения в некоторых единицах измерения, причем сами единицы измерения указаны в файле с метаданными;
c. переменные, значениями которых являются любые (нефиксированные) строки или безразмерные числа, представляющие собой скорее метки, а не собственно числовые значения (например, номер респондента).

В онтологии empirion используется трехуровневое представление переменных:

1. Экземпляры класса **Variable** представляют собой переменные из описываемых массивов данных.

2. Экземпляры класса **Representation** описывают варианты представления описываемых переменных. Для переменных типа (a) для каждого набора значений переменной создается экземпляр класса **Representation**, связанный с каждым из возможных значений. Сами значения являются экземплярами класса **RepresentationValue**. Для переменных типа (b) создается экземпляр класса **Representation**, связанный с единицами измерения, которые являются экземплярами класса **Metadata** -> **Variable Sensemaker** -> **Measurement Unit**. Для переменных типа (c) экземпляр класса **Representation** носит формальный характер и создается для единообразия запросов к создаваемой базе знаний.
3. Экземпляры класса **RepresentedVariable** представляют собой некоторую идеализированную измеряемую переменную, например, пол или возраст (независимую от ее представления в конкретных массивах данных). Для переменных типа (a) выбирается некоторый «эталонный» набор значений и создаются связи с этими значениями, представляющими собой экземпляры класса **RepresentedVariableValue**. Это необходимо для того, чтобы впоследствии устанавливать соответствие между значениями одной переменной, закодированной по-разному в различных массивах данных.

На рисунке 18 представлена структура основных классов онтологии empirion в виде концептуальной карты.

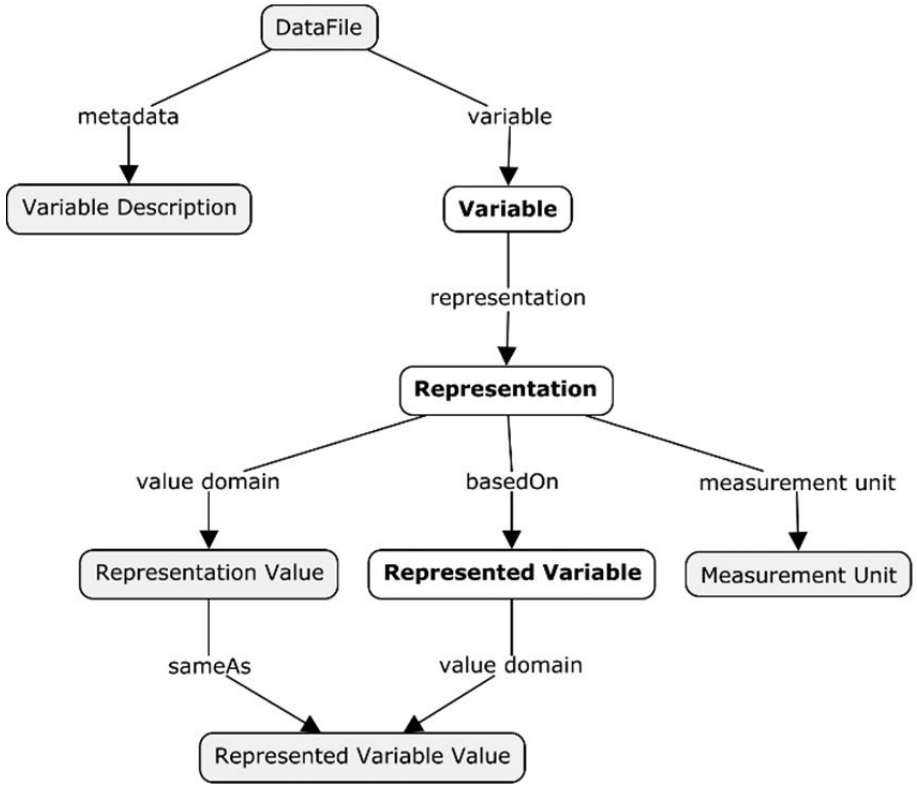


***Рисунок 18 —*** *Концептуальная карта основных классов онтологии empirion*

Для наполнения онтологии empirion использовался новый метод METEOP, описанный в главе 2. Пререквизитами метода является создание двух онтологий - наполняемой ОПО (empirion) и ОИД (базовая ОИД также носит имя meteor). Был использована метод процесса наполнения METEOP (см. главу 2), состоящий из 5 шагов.

**Шаг 1**: Идентификация источников данных. Считаем, что массив данных содержит следующие файлы:

1. собственно файл с таблицей с результатами эксперимента, где каждый столбец представляет собой значения некоторой переменной;
2. файл с метаданными — описанием эксперимента и переменных.

На основании этих файлов для каждого массива данных был создан файл, содержащий информацию о том, к каким классам онтологии empirion относятся описываемые переменные. Все файлы, созданные на шаге 1, были помещены в одну папку и, с помощью Power Query, преобразованы к табличному виду. Полученная таблица имеет следующую структуру:

1. Столбец DataFile — содержит название файла с результатами эксперимента. Уникальные значения из этого столбца отразятся в экземпляры класса **Data** -> **Data File**.
2. Столбец VariableDescription — содержит название файла с метаданными. Уникальные значения из этого столбца отразятся в экземпляры класса **Data** -> **Metadata File** -> **Variable Description**.
3. Столбец Variable — содержит названия переменных.
4. Столбец Representation — содержит все возможные значения для переменных вида (a), варианты значений которых ограничены некоторым списком.
5. Столбец RepresentedValue — содержит «эталонные» значения для переменных типа (a).
6. Столбец MeasureUnits — содержит единицы измерений для переменных типа (b). Уникальные значения из этого столбца отразятся в экземпляры класса **Metadata** -> **Variable Sensemaker** -> **Measurement Unit**.
7. Блок из столбцов, соответствующих подклассам класса **Variable**, экземплярами которых станут переменные из массива данных.
8. Блок из столбцов, соответствующих подклассам класса **Representation**, экземплярами которых станут различные представления переменных из массива данных.
9. Блок из столбцов, соответствующих подклассам класса **RepresentedVariable**, экземплярами которых являются переменные, не зависящие от конкретных массивов данных и представлений переменных в них.

**Шаг 2**: Спецификация источников данных. На этом шаге была создана вспомогательная онтология empirion_struct, импортирующая ОИД meteor. Далее был создан экземпляр класса **Workbook**, у которого заданы атрибуты hasPath (путь к файлу из шага 1) и hasName (имя файла из шага 1).

**Шаг 3**: Извлечение структуры данных. С помощью модуля извлечения структуры данных онтология empirion_struct была наполнена экземплярами, описывающими структуру файла из шага 1.

**Шаг 4**: Задание правил отображения. На этом этапе онтология empirion_struct была импортирована в онтологию empirion, а затем заданы следующие правила отображения:

- Для класса **Data -> Data File**:
    - getInstancesFrom DataFile, где DataFile — экземпляр класса **WBColumn**, представляющий соответствующий столбец из таблицы, созданной на первом шаге.
    - makeReferenceTo **Variable Description** useObjectProperty metadata, где **Variable Description** — имя класса, а metadata — название отношения, которое должно быть использовано при создании связей.
    - Несколько правил вида makeReferenceTo <**Variable SubClass Name**> useObjectProperty variable, где вместо <**Variable SubClass Name**> указывается имя конкретного подкласса класса **Variable**, а variable – название отношения, которое должно быть использовано при создании связей.
- Для класса **Data -> Metadata File -> Variable Description**:
    - getInstancesFrom VariableDescription, где VariableDescription — экземпляр класса **WBColumn**, представляющий соответствующий столбец из таблицы, созданной на первом шаге.
- Для подклассов класса **Variable**:
    - getInstancesFrom <WBColumn Variable Instance Name>, где <WBColumn Variable Instance Name> – экземпляр класса **WBColumn**, представляющий соответствующий столбец из таблицы, созданной на первом шаге.
    - makeReferenceTo <**Representation SubClass Name**> useObjectProperty representation, где вместо <**Representation SubClass Name**> указывается имя конкретного подкласса класса **Representation**, а representation – название отношения, которое должно быть использовано при создании связей.
- Для подклассов класса **Representation**:

- getInstancesFrom <WBColumn Representation Instance Name>, где <WBColumn Representation Instance Name> — экземпляр класса **WBColumn**, представляющий соответствующий столбец из таблицы, созданной на первом шаге.
- makeReferenceTo <**Represented Variable SubClass Name**> useObjectProperty based on, где вместо <**Represented Variable SubClass Name**> указывается имя конкретного подкласса класса **Represented Variable**, a based on — название отношения, которое должно быть использовано при создании связей.
- makeReferenceTo **RepresentationValue** useObjectProperty has value domain.
- makeReferenceTo **Measurement Unit** useObjectProperty has measurement value.

- Для подклассов класса **Represented Variable**:
  - getInstancesFrom <WBColumn Represented Variable Instance Name>, где <WBColumn Represented Variable Instance Name> — экземпляр класса **WBColumn**, представляющий соответствующий столбец из таблицы, созданной на первом шаге.
  - makeReferenceTo **RepresentedVariableValue** useObjectProperty has value domain.
- Для класса **RepresentationValue**:
  - getInstancesFrom Representation, где Representation — экземпляр класса **WBColumn**, представляющий соответствующий столбец из таблицы, созданной на первом шаге.
  - makeReferenceTo **RepresentedVariableValue** useObjectProperty sameAs, где **RepresentedVariableValue** — имя класса, a sameAs — название отношения, которое должно быть использовано при создании связей.
- Для класса **RepresentedVariableValue**:
  - getInstancesFrom RepresentedValue, где RepresentedValue — экземпляр класса **WBColumn**, представляющий соответствующий столбец из таблицы, созданной на первом шаге.
- Для класса **Metadata** -> **Variable Sensemaker** -> **Measurement Unit**:
  - getInstancesFrom MeasureUnits, где MeasureUnits — экземпляр класса **WBColumn**, представляющий соответствующий столбец из таблицы, созданной на первом шаге.

Для задания правил отображения использовались аннотационные свойства getInstancesFrom, makeReferenceTo и useObjectProperty, подробно описанные в главе 2.

**Шаг 5**: Наполнение онтологии. Выполняется автоматически с помощью модуля наполнения. На рисунках 19-21 показаны фрагменты созданных структур для переменных различного типа.

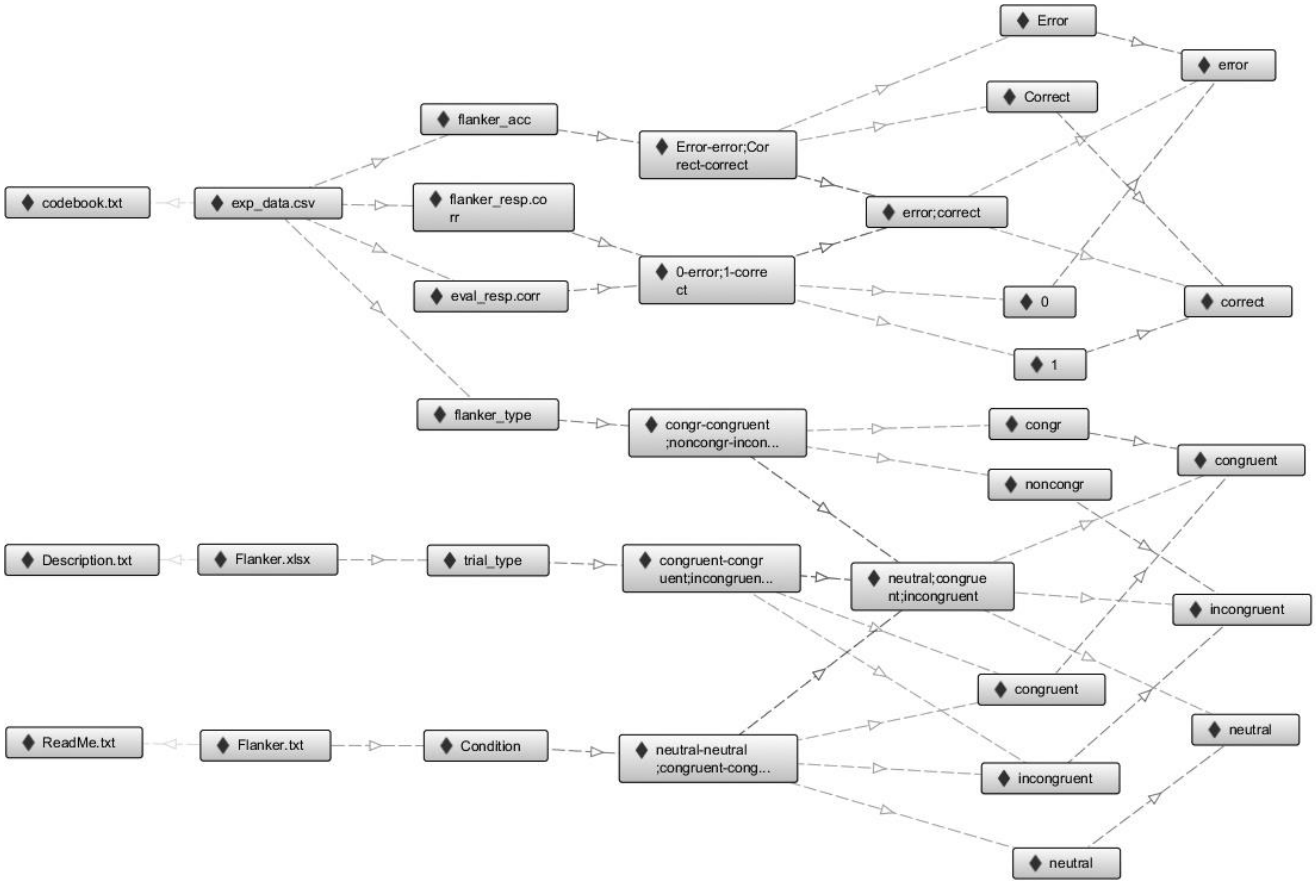


***Рисунок 19 —*** *Пример структуры, созданной для переменных типа (a)*

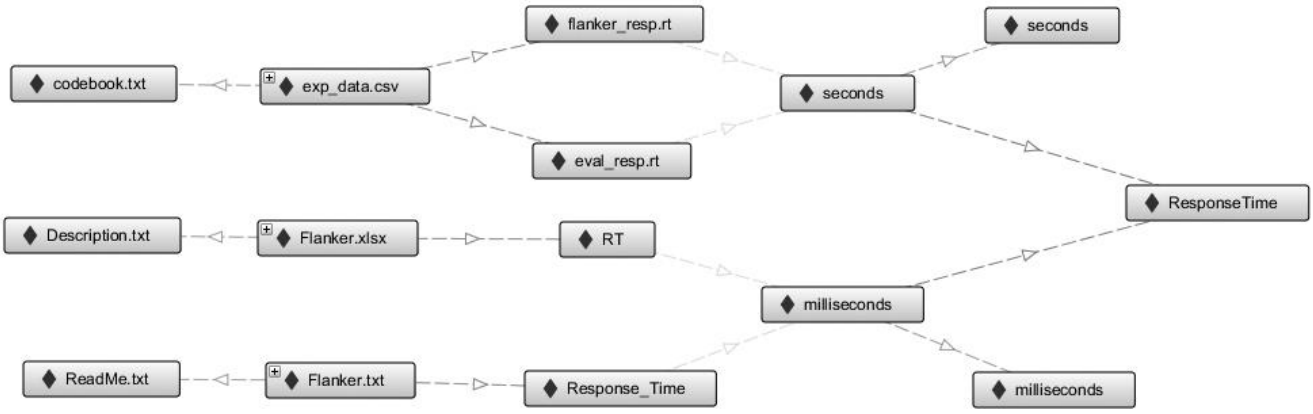


***Рисунок 20 —*** *Пример структуры, созданной для переменных типа (b)*


***Рисунок 21 —*** *Пример структуры, созданной для переменных типа (c)*

После наполнения онтологии возможно выполнение SPARQL-запросов для поиска датасетов с заданными характеристиками. Например, запрос ниже ищет все датасеты, которые измеряют переменные ResponseTime и StimulusCongruence.

```
PREFIX emp: <http://www.tempuri.org/empirion#>
PREFIX ddi: <http://rdf-vocabulary.ddialliance.org/discovery#>
SELECT DISTINCT ?subject
    WHERE {
        ?subject ddi:variable ?somevar1.
        ?somevar1 ddi:representation ?somerepresentation1.
        ?somerepresentation1 ddi:basedOn emp: responsetimerepresentedvariable1.
        ?subject ddi:variable ?somevar2.
        ?somevar2 ddi:representation ?somerepresentation2.
        ?somerepresentation2 ddi:basedOn emp: stimuluscongruencerepresentedvariable1
 }
```

### 3.2. Наполнение онтологии орфанных заболеваний Ontomorf

Онтологический подход широко используется в различных медицинских интеллектуальных системах и приложениях [Грибова В. В. и др., 2018; Клещев, Москаленко, Черняховская, 2005; Ivanović, Budimac, 2014]. Для диагностики орфанных (редких) заболеваний предложено использовать медицинские онтологии [Кобринский, 2018; Кобринский и др., 2019]. Анализ предметной области и работа с экспертами позволили предложить алгоритмы, в которых диагностирование осуществляется на основе шкал модальности признаков с их коэффициентами и формул для комплексной оценки совокупности факторов уверенности (выраженности и манифестации).

По результатам анализа были выделены 4 возрастных группы пациентов:

(1) до 1 года;

(2) 1-3 года;

(3) 4-6 лет;

(4) старше 6 лет.

Далее для каждой возрастной группы были определены коэффициенты модальности, манифестации и выраженности каждого признака (симптома).

Модальность ($M_{ik}$, где $i$ — номер признака, а $k$ — номер возрастного периода) характеризует релевантность признаков заболевания в каждом возрастном периоде. В соответствии с мнением экспертов для каждого заболевания признаки были разделены на главные, необходимые и второстепенные, и для каждой группы признаков были определены коэффициенты. При невозможности наличия признака в данном возрастном диапазоне модальность считалась нулевой ($M_{ik}$=0).

Манифестация ($n_{ij}$, где $i$ — номер признака, а $j$ — номер возрастного периода) характеризует меру доверия (уверенность) экспертов в том, что данный признак определённого заболевания манифестирует (обнаруживается) именно в данном возрасте (возрастной группе). Т.е. для каждого возрастного периода экспертами было определено свое значение манифестации, причем сумма этих значений для одного заболевания по всем возрастным группам не превышает 1. Суммарный фактор уверенности в манифестации ($m_{ik}$, где $i$ — номер признака, а $k$ — номер возрастного периода) рассчитывался по формуле:

$$m_{ik} = \sum_{j=1}^{k} n_{ij},$$

где $i$ — номер признака, а $k$ — номер возрастного периода.

Выраженность ($s_{ik}$, где $i$ — номер признака, а $k$ — номер возрастного периода) характеризует уверенность экспертов в том, что данный признак встречается в конкретной возрастной группе с определенной степенью выраженности. Изменение выраженности по возрастным периодам косвенно указывает на скорость развития заболевания (симптомов).

Для получения комплексной количественной оценки признака в определенном возрастном периоде использовалась следующая формула:

$$P_{ik} = M_{ik} * m_{ik} * s_{ik},$$

где $i$ — номер признака,

$k$ — номер возрастного периода,

$P_{ik}$ — количественная оценка признака заболевания (симптома) для указанного возрастного периода,

$M_{ik}$ — модальность признака для этого возрастного периода,

$m_{ik}$ —суммарный фактор уверенности манифестации для возрастного периода $k$,

$s_{ik}$ — фактор уверенности выраженности для этого же возрастного периода.

Для получения интегрированной количественной оценки суммы признаков для каждого диагностируемого случая использовалась формула:

$$I = \sum_{i=1}^{n} P_{ik},$$

где $n$ — количество признаков,

$k$ — номер возрастного периода, в который попадает возраст данного пациента,

$I$ — интегрированная оценка признаков,

$P_{ik}$ — количественная оценка признака заболевания $i$ для возрастного периода $k$.

Для реализации предложенного подхода была создана онтология ontomorf, основные элементы которой представлены на рисунке 22.

В левой части — классы, в которых содержится «постоянная» информация базы знаний. Предполагается, что в процессе работы базы знаний эта информация не меняется или меняется очень редко.

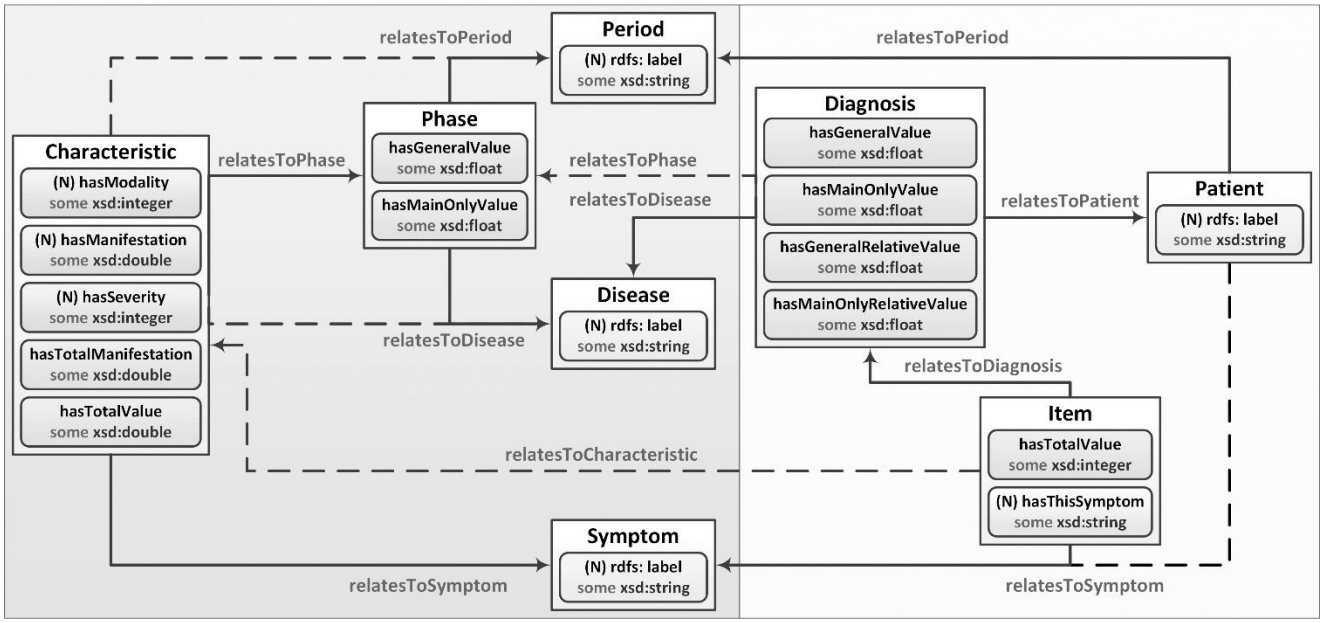


***Рисунок 22 —*** *Пример структуры, созданной для переменных типа (b)*

- Класс **Disease** характеризует заболевания, для диагностики которых предназначена сформированная база знаний.
- Класс **Symptom** служит для описания признаков заболеваний диагностируемой группы.
- Класс **Period** определяет возрастные периоды: до 1 года, 1-3 года, 4-6 лет, старше 6 лет.
- Класс **Characteristic**: каждый элемент класса соответствует одному признаку (симптому) определенного заболевания в определенный период и содержит значения модальности ($M_{ik}$), манифестации ($n_{ik}$) и выраженности ($s_{ik}$) этого признака для этого заболевания в этого период. Также элементы этого класса используются для расчетов суммарного фактора уверенности манифестации ($m_{ik}$),

комплексной количественной оценки признака ($P_{ik}$) и суммарных значений элементов класса **Phase**.

- Класс **Phase**: содержит суммарные (референсные) значения для каждого заболевания в определенный период.

Правая часть содержит «переменную» информацию о пациентах.

- Класс **Patient** содержит идентификационные данные пациента, возраст, симптомы.
- Класс **Diagnosis**: в этом классе для каждого пациента создаются экземпляры по одному на каждое заболевание. Содержит суммарные значения по каждому заболеванию для определенного пациента. Эти значения сравниваются с референсными значениями из соответствующего экземпляра класса **Phase** для формирования гипотез о диагнозе.
- Класс **Item**: вспомогательный класс для вычисления значений экземпляров класса **Diagnosis**.

Элементы, подсвеченные розовым, и отношения, показанные пунктирными стрелками, определяются на основе другой имеющейся информации. Для этого заданы 24 правила вывода.

Разработанный автором метод METEOP для наполнения онтологии ontomorf использовался несколько раз:

1) на этапе создания «базовой» (постоянной) части базы знаний (однократно);
2) на этапе эксплуатации для внесения информации о пациентах.

Далее подробно описан процесс наполнения «базовой» части базы знаний.

**Шаг 1**: Идентификация источников данных. Данные для наполнения «базовой» части базы знаний содержались в двух Excel-файлах symptoms.xlsx и modalities.xlsx. Структура файла symptoms.xlsx показана на рисунке 23.

| | Гурлер | | | | | | | | Г-Шейе | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | до 1 года | | 1-3 года | | 4-6 лет | | старше 6 ле | | до 1 года | | 1-3 года | | 4-6 лет | | старше 6 ле | |
| **Задержка роста** | 0,7 | 3 | 0,1 | 5 | 0,1 | 7 | 0,1 | 10 | 0 | 0 | 0,7 | 7 | 0,1 | 7 | 0,1 | 8 |
| **Короткая шея** | 0,1 | 2 | 0,1 | 3 | 0,3 | 7 | 0,4 | 9 | 0,1 | 3 | 0,1 | 3 | 0,2 | 6 | 0,2 | 6 |
| **Макроцефалия** | 0,7 | 2 | 0,1 | 4 | 0,1 | 5 | 0,1 | 8 | 0,4 | 2 | 0,1 | 3 | 0,1 | 4 | 0,1 | 4 |

***Рисунок 23 —*** *Пример структуры файла с коэффициентами манифестации и выраженности*

В первом столбце содержатся все признаки диагностируемых заболеваний, а далее в блоках для каждого заболевания и для каждой возрастной группы указаны значения коэффициентов факторов уверенности манифестации и выраженности.

Модальности заданы иерархическим списком следующего вида (см. рисунок 24).

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Мукополисахаридоз 1H (Гурлер) | | | | | | | | | | |
| | до 1 года | | | | | | | | | |
| | | (https://rarediseases.info.nih.gov/diseases/12559/mucopolysaccharidosis-type-ih) | | | | | | | | |
| | | | **Главные признаки (80%-99% больных имеют эти симптомы):** | | | | | | | |
| | | | | Короткая шея | | | | | | |
| | | | | Грубые черты лица | | | | | | |
| | | | | Гепатомегалия | | | | | | |
| | | | | Спленомегалия | | | | | | |
| | | | | Грыжи | | | | | | |
| | | | **Необходимые признаки (30%-79% больных имеют эти симптомы):** | | | | | | | |
| | | | | Задержка роста | | | | | | |
| | | | | Макроглоссия | | | | | | |
| | | | | Снижение слуха | | | | | | |
| | | | | Помутнение роговицы | | | | | | |
| | | | **Второстепенные признаки (менее 30% больных имеют эти симптомы):** | | | | | | | |
| | | | | Макроцефалия | | | | | | |
| | | | | Скафоцефалия | | | | | | |
| | | | | Гипертрихоз | | | | | | |
| | | | | Утолщенная кожа | | | | | | |
| | | | | Воронкообразная грудная клетка | | | | | | |
| | 1-3 года | | | | | | | | | |
| | | (https://rarediseases.info.nih.gov/diseases/12559/mucopolysaccharidosis-type-ih) | | | | | | | | |
| | | | **Главные признаки (80%-99% больных имеют эти симптомы):** | | | | | | | |
| | | | | Короткая шея | | | | | | |
| | | | | Грубые черты лица | | | | | | |
| | | | | Тугоподвижность крупных суставов | | | | | | |

***Рисунок 24 —*** *Пример структуры файла с модальностями признаков*

На первом шаге с помощью инструмента PowerQuery данные были сведены в единую нормализованную таблицу (см. рисунок 25). При этом понадобилась дополнительная таблица, в которой текстовым названиям модальностей сопоставлялись числовые значения (Главные — 5, Необходимые —4, Второстепенные —2).

| Disease | Period | Symptom | Manifestation | Severity | Modality |
|---|---|---|---|---|---|
| Гурлер | до 1 года | Короткая шея | 0,1 | 2 | 5 |
| Гурлер | до 1 года | Грубые черты лица | 0,7 | 5 | 5 |
| Гурлер | до 1 года | Гепатомегалия | 0,5 | 7 | 5 |
| Гурлер | до 1 года | Спленомегалия | 0,2 | 3 | 5 |
| Гурлер | до 1 года | Грыжи | 0,6 | 7 | 5 |
| Гурлер | до 1 года | Задержка роста | 0,7 | 3 | 4 |
| Гурлер | до 1 года | Макроглоссия | 0,3 | 5 | 4 |
| Гурлер | до 1 года | Снижение слуха | 0,3 | 4 | 4 |
| Гурлер | до 1 года | Помутнение роговицы | 0,2 | 4 | 4 |
| Гурлер | до 1 года | Макроцефалия | 0,7 | 2 | 2 |
| Гурлер | до 1 года | Скафоцефалия | 0,4 | 3 | 2 |
| Гурлер | до 1 года | Гипертрихоз | 0,4 | 5 | 2 |
| Гурлер | до 1 года | Утолщенная кожа | 0,3 | 2 | 2 |
| Гурлер | до 1 года | Воронкообразная грудная клетка | 0,6 | 2 | 2 |
| Гурлер | 1-3 года | Короткая шея | 0,1 | 3 | 5 |
| Гурлер | 1-3 года | Грубые черты лица | 0,1 | 7 | 5 |

***Рисунок 25 —*** *Структура таблицы после объединения и преобразования*

Таблица содержит 6 столбцов:

- Disease: названия диагностируемых заболеваний.
- Period: принадлежность к возрастной группе.
- Symptom: названия признаков заболеваний.
- Manifestation: значения манифестации ($n_{ik}$).
- Severity: значени выраженности ($s_{ik}$).
- Modality: значения модальности ($M_{ik}$).

**Шаг 2**: Спецификация источников данных. На этом шаге была создана вспомогательная онтология ontomorf_struct, импортирующая онтологию meteor. Далее был создан экземпляр класса **Workbook**, у которого заданы атрибуты hasPath (путь к файлу из шага 1) и hasName (имя файла из шага 1).

**Шаг 3**: Извлечение структуры данных. С помощью модуля извлечения структуры данных онтология ontomorf_struct была наполнена экземплярами, описывающими структуру файла из шага 1 (см. рисунок 26).


***Рисунок 26 —*** *Онтология, описывающая структуру источника данных*

**Шаг 4**: Задание правил отображения. На этом этапе онтология ontomorf_struct была импортирована в онтологию ontomorf, а затем заданы следующие правила отображения:

- Для класса **Period**:
  - getInstancesFrom Period, где Period — экземпляр класса **WBColumn**, представляющий соответствующий столбец из таблицы, созданной на первом шаге.
  - makeList (указывает на то, что экземпляры должны быть организованы в список с помощью свойства hasNext).
- Для класса **Disease**:
  - getInstancesFrom Disease, где Disease — экземпляр класса **WBColumn**, представляющий соответствующий столбец из таблицы, созданной на первом шаге.

- Для класса **Phase**:
    - combineInstancesFrom **Disease** useObjectProperty relatesToDisease;
    - combineInstancesFrom **Period** useObjectProperty relatesToPeriod (**Disease** и **Period** — имена классов, а relatesToDisease и relatesToPeriod — названия отношений, которые должны быть использованы при создании связей между экземплярами, представляющими комбинацию заболевания и периода, и исходными экземплярами классов **Disease** и **Period**).
- Для класса **Symptom**:
    - getInstancesFrom Symptom, где Symptom — экземпляр класса **WBColumn**, представляющий соответствующий столбец из таблицы, созданной на первом шаге.
    - makeList (указывает на то, что экземпляры должны быть организованы в список с помощью свойства hasNext).
- Для класса **Characteristic**:
    - combineInstancesFrom **Disease** useObjectProperty relatesToDisease;
    - combineInstancesFrom **Period** useObjectProperty relatesToPeriod;
    - combineInstancesFrom **Symptom** useObjectProperty relatesToSymptom (**Disease**, **Period** и **Symptom** — имена классов, а relatesToDisease, relatesToPeriod и relatesToSymptom — названия отношений, которые должны быть использованы при создании связей между экземплярами, представляющими комбинацию заболевания, периода и признака, и исходными экземплярами классов **Disease**, **Period** и **Symptom**).
    - getValuesFrom Manifestation, где Manifestation — экземпляр класса **WBColumn**, представляющий соответствующий столбец из таблицы, созданной на первом шаге.
    - getValuesFrom Modality, где Modality — экземпляр класса **WBColumn**, представляющий соответствующий столбец из таблицы, созданной на первом шаге.
    - getValuesFrom Severity, где Severity — экземпляр класса **WBColumn**, представляющий соответствующий столбец из таблицы, созданной на первом шаге.
- **Шаг 5**: Наполнение онтологии. Выполняется автоматически с помощью модуля наполнения. На рисунке 26 показаны фрагменты созданных структур для переменных различного типа.

После наполнения онтологии была запущена машина логического вывода (правила приведены в приложении), с помощью который были построены недостающие отношения и вычислены необходимые для процесса диагностирования значения. Все полученные результаты вывода были экспортированы в онтологию, так как процесс вывода занял значительное время, а, в данном случае, наполнялась «базовая» часть онтологии, т.е. та часть, регулярное изменение которой не предполагается.

Далее тот же алгоритм был применен для загрузки в полученную базу знаний информации о 15 тестовых пациентах, а затем запущена машина логического вывода для постановки диагноза. Полученные результаты приведены в приложении.

Таким образом использование предложенного в диссертации метода METEOP позволило создать прототип медицинской экспертной системы совместно с сотрудниками ФИЦ РАН.

## 3.3. Специфика наполнения онтологии сборочных единиц автомобилей

Задача создания и наполнения онтологической базы знаний сборочных единиц автомобилей возникла в связи с производственной необходимостью систематизации, упорядочивания и переиспользования знаний о сборочных единицах различных автомобилей внутри компании-автопроизводителя[1]. Элементами базы знаний данного предприятия являются сборочные единицы (СЕ) — детали различных автомобилей (в основном, грузовиков), производимых в компании. База знаний является основой для создаваемой информационно-справочной системы, в которой пользователи (инженер, менеджер по закупкам, экономист и др.) смогут найти необходимую им СЕ по характеристикам, а также сравнить её с аналогами. Актуальность данной системы объясняется тем, что компания занимается производством электромобилей с использованием компонент, уже представленных на рынке, и поиск информации об изделиях поставщиков занимает серьезную долю времени сотрудников. В системе предполагается иметь как информацию о тех СЕ, которые уже используются в компании, так и о тех, которые представлены на рынке в каталогах поставщиков. В перспективе базу знаний планируется использовать в роботизированном производстве (сборке), которое также развивается в компании. Заказчиком базы знаний было IT-подразделение, осуществляющее цифровизацию всех бизнес-процессов предприятия. Для пилотного проекта было взято 50 СЕ.

Для создания базы знаний СЕ использовалась методика ВИТОН (Визуальной коллекТивной разработки ОНтологического графа знаний) [Гаврилова и др., 2019], которая

1 Название компании не указывается в соответствии с подписанным соглашением о неразглашении конфиденциальной информации

ориентирована на решение задачи описания сборочных единиц (СЕ) для сложного высокотехнологичного производства и связана с формирование базы знаний большой экстенсивности (тысячи СЕ, экземпляров онтологии). Особенность создаваемой базы знаний заключается в многообразии и сложной структуре свойств СЕ, при этом количество классов онтологии незначительное.

В процессе создания базы знаний СЕ можно выделить 4 этапа: концептуализации, формализации, реализации и наполнения.

**Концептуализация** осуществлялась с помощью интеллект-карт. Сначала было применено нисходящее моделирование и создан макет онтологии в форме интеллект-карты. Затем было применено восходящее моделирование: на основе макета онтологии были созданы интеллект-карты для набора пилотных СЕ, предоставленных заказчиком. Эти интеллект-карты были оценены экспертами, а затем были внесены изменения в макет, в частности, были добавлены новые свойства, единицы измерения и диапазоны значений. Эти два способа моделирования реализовывались итеративно: интеллект-карты для каждой СЕ основывались на макете и сам макет непрерывно изменялся в соответствии с новыми свойствами, идентифицированными во время анализа СЕ.

На этапе **формализации** использовались электронные таблицы. После того, как были созданы интеллект-карты для примерно 50 СЕ и зафиксирован макет онтологии (т.е. дополнительный анализ СЕ немногое привносил в макет), была выполнена формализация свойств с использованием таблиц. Таблица со свойствами служила промежуточным этапом между интеллект-картами и онтологией. Главной целью создания таблицы было формирование иерархии свойств и словаря. Интеллект-карты для разных СЕ содержали характеристики, которые являются дочерними свойствами того же свойства высокого уровня. Например, воздушный компрессор серии ERC имеет выходную характеристику Free air delivery («свободное нагнетание воздуха»), которая связана со свойством Output («выход»), так же как свойство Air flow («поток воздуха») другого экземпляра. Кроме того, одни и те же характеристики СЕ могут быть названы по-разному в разных каталогах поставщиков, поэтому внесение в таблицу помогало выявлять синонимы. Похожее использование таблиц для создания онтологии было упомянуто в подходах Rubin’a [Rubin, 2008] и Ontorat [Xiang et al., 2015]. Преобразованные интеллект-карты составили основу таблицы (на рисунке 27 представлен фрагмент таблицы для СЕ на примере воздушного компрессора серии ERC).

| Свойства | Ярлык | Описание | ERC 507L Mattei ERC Series |
|---|---|---|---|
| hasFunctionalCharacteristics | Функциональные характеристики | «Атрибут или характеристика в использовании и результативности продукта» (Источник: ISO 14050:2009(en), 8.3.3.2) | |
| hasFunctionCharacteristics | Функция | Предположительное поведение артефакта | |
| hasType | Тип артефакта | Неформальное описание типа (мотор, батарея, и т.д) | компрессор |
| hasFunctionType | Тип функции | Тип функции в соответствии с IEC 81346 | GQ |
| hasOutputCharacteristics | Выход | Весь выход, который может быть предоставлен артефактом в случае правильного функционирования | |
| hasOutputAirCharacteristics | Характеристики выхода воздуха | Группа характеристик для описания выхода воздуха | |
| hasOutputFreeAirDelivery | Свободная подача воздуха | | 43 scfm |
| hasRatedPressure | Разрешенное давление | | 125 psig |

***Рисунок 27 —*** *Фрагмент таблицы для CE воздушного компрессора серии ERC*

Техника преобразования, схожая с описанными техниками для трансформации интеллект-карт [Sure et al., 2002; Křemen et al., 2014], была применена с некоторыми дополнениями:

- CE сформировали столбцы в таблице (субъекты).
- Понятия из макетов онтологий были преобразованы в свойства высокого уровня, а элементы низкого уровня из интеллект-карты CE — в дочерние свойства. В таблице они сформировали строки (предикаты). Во время преобразования имена свойств были унифицированы в соответствии с выбранной формой именования, были добавлены человекочитаемые имена в качестве ярлыков, а также описание.
- Самый нижний уровень интеллект-карты был помещен в ячейки таблицы (объекты). Для каждой CE все связанные свойства добавлялись в таблицу вместе с их размерностями (при наличии).

Создание таблицы состояло из трех этапов:

1. Первоначальный ввод свойств в соответствии с интеллект-картой макета онтологии.
2. Уточнение общей структуры свойств с учетом их семантической близости и уровня обобщения.
3. Ввод всех CE в созданную структуру свойств (с указанием имеющихся свойств и, при необходимости, их значений), утверждение и финализация структуры, добавление ярлыков и описаний свойств.

На этапе **реализации** иерархия свойств, созданная на предыдущем этапе, была преобразована в описание на языке OWL с помощью редактора онтологий (WebProtégé [https://webprotege.stanford.edu/]). Переход состоял из трех шагов. На первом шаге свойства были импортированы в редактор онтологий. Во время их переноса общая иерархия свойств была разделена на две отдельные иерархии для свойств-объектов (object property) и свойств-значений

(datatype property). При этом свойства верхнего уровня составляли верхний уровень классификации для обоих типов свойств (например, свойство с именем hasFunctionalCharacteristics существовало и в иерархии свойств-объектов, и в иерархии свойств-значений). Затем была создана иерархия классов с четырьмя классами верхнего уровня:

- Artifact («артефакт») — для экземпляров сборочных единиц и их частей;
- Organization («организация») — для экземпляров организаций поставщиков и производителей;
- QuantitativeValue («количественное значение») — для экземпляров количественных значений свойств;
- Reference («словари») — для экземпляров значения свойств импортированных из внешних источников (например, в этот класс входил список единиц измерения).

На последнем этапе онтология была наполнена с помощью предложенного в работе метода.

1. Сборочные единицы были импортированы как экземпляры класса Artifact со свойствами rdfs:label (человекочитаемое название) и rdfs:description (описание, если доступно).
2. Организации импортировались как экземпляры класса Organization со свойствами rdfs:label (человекочитаемое название) и hasURL (ссылка на сайт организации).
3. Значения свойств из других словарей импортировались как экземпляры класса Reference.
4. Количественные значения импортировались как экземпляры класса QuantitativeValue. Каждый экземпляр описывал значение или диапазон значений. Например, компрессор серии ERC имеет свойство hasFreeAirDelivery («свободное нагнетание воздуха») с диапазоном значений от 5.1 до 460 scfm. Это импортируется как экземпляр «5.1-460scfm» класса Flow («поток», подкласс QuantitativeValue) со свойствами-значениями hasMaxValue (максимальное значение: 460), hasMinValue (минимальное значение 5.1; hasUnitCode, код единицы измерения: scfm).

Созданная таким образом база знаний продемонстрировала как жизнеспособность метода ВИТОН, так и возможности предлагаемого метода наполнения онтологий. Необходимо отметить, что этот пример не является целевым для разработанного метода наполнения онтологий, так как данные заносились в базу знаний из таблицы, структура которой могла быть изменена при необходимости и специально создавалась для удобства в том числе наполнения онтологии.

## 3.4. Наполнение онтологии для администрирования учебной деятельности

Система образования в России претерпевает значительные изменения. Эти изменения обусловлены несколькими факторами. Во-первых, это вступление Российской Федерации в болонский процесс 19 сентября 2003 года. Полноценное вхождение в Болонский процесс требует реформирования системы обучения в целом и высшего профессионального образования в частности. Реформа предусматривает, прежде всего, разработку образовательных программ, совместимых с европейскими, а для их реализации — соответствующую трансформацию вузовских структур, нормативной базы и, наконец, практики преподавания. Акцент переносится с содержания образования на результаты обучения, что обеспечивается за счет компетентностно-ориентированного подхода.

Во-вторых, 30 апреля 2014 года распоряжением Правительства Российской Федерации утвержден очередной План мероприятий («дорожная карта») «Изменения в отраслях социальной сферы, направленные на повышение эффективности образования и науки». В этом документе, в частности, указано, что основные направления изменений в сфере высшего образования включают в себя:

- Совершенствование структуры и сети государственных образовательных организаций высшего образования:
    - проведение ежегодного мониторинга эффективности образовательных организаций высшего образования;
    - реализация программы совершенствования сети государственных образовательных организаций высшего образования, в том числе путем реорганизации и присоединения организаций и их филиалов.
- Совершенствование структуры образовательных программ:
    - введение прикладного бакалавриата в высшем образовании;
    - мониторинг перехода на федеральные государственные образовательные стандарты подготовки кадров высшей квалификации и их актуализацию.
- Повышение результативности деятельности образовательных организаций высшего образования с учетом их специализации:
    - обновление программ развития федеральных университетов;
    - поддержку программ развития сети национальных исследовательских университетов;
    - реализацию программ развития ведущих университетов, получающих государственную поддержку в целях повышения их конкурентоспособности среди ведущих мировых научно-образовательных центров, и их мониторинг в соответствии с утвержденным планом мероприятий;

- реализацию программ стратегического развития образовательных организаций высшего образования.

- Развитие кадрового потенциала высшего образования:
  - разработку и внедрение механизмов эффективного контракта с научно-педагогическими работниками образовательных организаций высшего образования;
  - проведение мероприятий по аттестации научно-педагогических работников образовательных организаций высшего образования;
  - разработку и внедрение механизмов эффективного контракта с руководителями образовательных организаций высшего образования в части установления взаимосвязи между показателями качества предоставляемых государственных (муниципальных) услуг организацией и эффективностью деятельности руководителя образовательной организации системы высшего образования;
  - информационное и мониторинговое сопровождение введения эффективного контракта.

В-третьих, 21 декабря 2012 года Государственной Думой принят (а 26 декабря 2012 года Советом Федерации одобрен) Федеральный закон «Об образовании в Российской Федерации». Федеральные государственные образовательные стандарты высшего профессионального образования по направлениям подготовки, утвержденные ранее, приводятся в соответствие к требованиям этого закона.

Кроме того, сейчас широко обсуждается необходимость так называемого непрерывного образования. Непрерывное образование — это образование на протяжении всей жизни, которое обеспечивается единством и целостностью системы образования, созданием условий для самообразования и всестороннего развития личности, совокупностью преемственных, согласованных, дифференцированных образовательных программ различных ступеней и уровней, гарантирующих гражданам реализацию права на образование и предоставляющих возможность получать общеобразовательную и профессиональную подготовку, переподготовку, повышать квалификацию на протяжении всей жизни [Словарь согласованных терминов и определений в области образования государств-участников Содружества Независимых Государств, 2004].

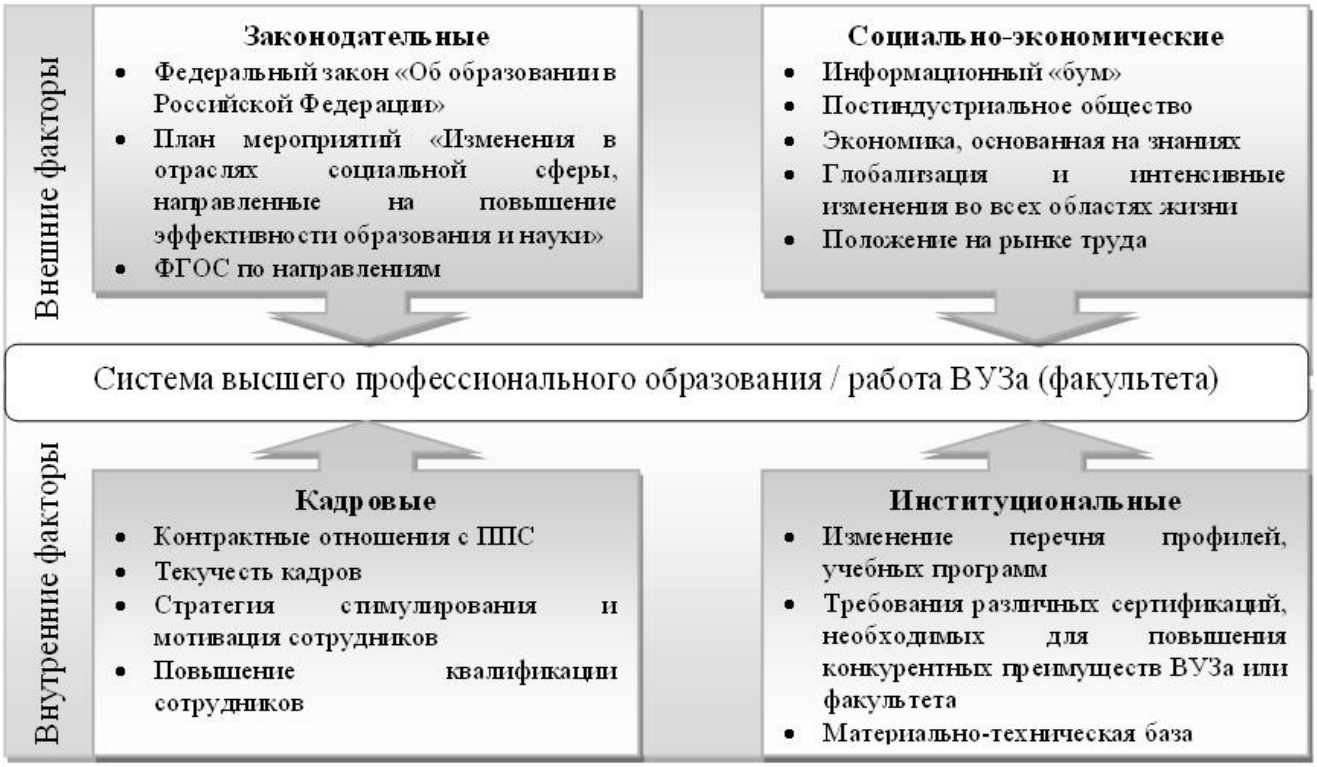


***Рисунок 28 —*** *Факторы, оказывающие влияние на систему высшего профессионального образования в целом и на работу ВУЗа в частности*

Важность непрерывного образования подтверждают следующие основные факторы:

- внедрение новой техники, технологии, производство современных товаров, рост коммуникационных возможностей создают условия для ликвидации или изменения некоторых видов работ. В связи с этим необходимая квалификация не может быть гарантирована базовым образованием.
- мир превращается в рынок без границ с высоким уровнем конкуренции между странами. Страны, имеющие программы непрерывного образования, лидируют в условиях этой конкуренции. Они имеют возможность в кратчайшие сроки ответить на любой "вызов" повышением производительности труда.
- изменения во всех областях жизни — главный элемент современности. Непрерывные и быстрые изменения в технологии и других сферах требуют непрерывного обучения;
- для бизнеса более эффективно и экономично повышение отдачи от уже работающих сотрудников на основе их непрерывного обучения, чем привлечение новых работников.

Все это ведет к тому, что образовательные стандарты, учебные планы и программы учебных дисциплин требуют постоянного обновления, чтобы соответствовать реалиям нашего времени (см. Рисунок 28).

При этом, как отмечается в работе [Благов, Лещева, Щербан, 2018], наблюдается разрыв между компетенциями, на которых базируются компетентностно-ориентированные учебные планы (КОУП) ВУЗов, и компетенциями, которые ожидают видеть у выпускников — кандидатов на должность — потенциальные работодатели. Для анализа существующих КОУПов и их выравнивания с точки зрения профессиональных стандартов была предложена онтологическая модель, содержащая помимо классов Преподаватели, Дисциплины, Компетенции и т.п., классы Трудовые функции и Профессиональные стандарты. Для наполнения этой онтологии использовались такие документы, как КОУП основной образовательной программы (ООП) бакалавриата по направлению «Менеджмент», три профессиональных стандарта «менеджер продуктов в сфере информационных технологий», «менеджер по информационным технологиям» и «специалист по управлению знаниями и информационными объектами» области информационных технологий, разработанные ассоциацией АПКиТ (Ассоциация предприятий компьютерных и информационных технологий) [Профессиональные стандарты в области ИТ [http://www.apkit.ru/committees/education/meetings/standarts.php], а также разработанный на их основе список дополнительных компетенций и их соответствие дисциплинам рассматриваемого КОУПа.

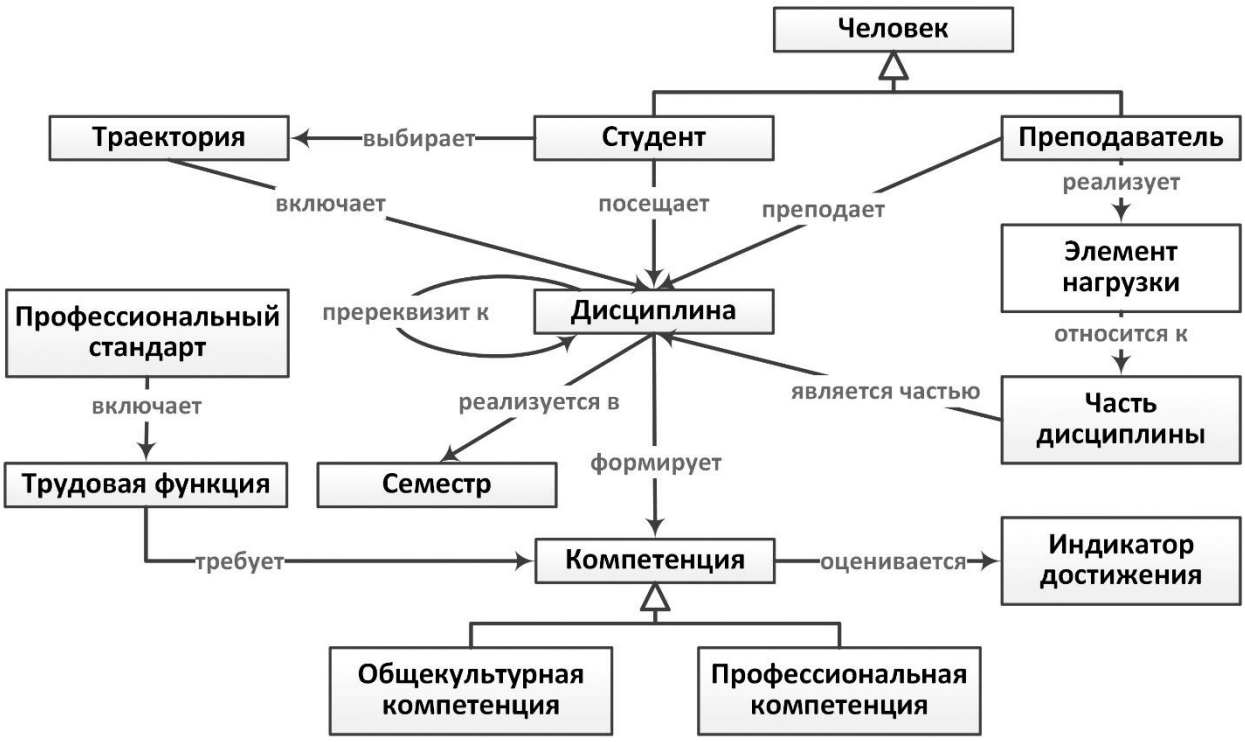


***Рисунок 29 —*** *Классы онтологии для анализа образовательных программ*

Процесс наполнения этой онтологии (см. рисунок 29) подробно не описывается, так как он аналогичен процессам наполнения онтологий, описанных в предыдущих параграфах. С

помощью полученной после наполнения базы знаний можно получить ответы на ряд вопросов, например:

1. Какой набор компетенций получил студент, прослушавший некоторый заданный ряд дисциплин?
2. Какой «минимальный» набор дисциплин обеспечивает освоение всех компетенций из некоторого набора?
3. Какие компетенции и дисциплины будут затронуты при удалении некоторой дисциплины из учебного плана?
4. Какие дисциплины по выбору можно исключить из учебного плана без ущерба для овладения заданным набором компетенций?
5. Какими компетенциями должен овладеть студент (и, соответственно, какие дисциплины прослушать), претендующий на занятие некоторой профессиональной позиции?

Таким образом, созданная база знаний может использоваться руководством образовательных программ и профильными кафедрами в качестве аналитического инструмента для анализа соответствия существующих КОУП требованиям работодателей и коррекции КОУП в соответствии с развитием профильных областей и профессиональных стандартов, а также входящих в них трудовых функций.

Но для построения образовательных траекторий студентами этой информации недостаточно, так как дисциплины по выбору объединены в блоки, и у студентов нет возможности освоить все дисциплины из одного блока. Поэтому онтология была дополнена классами, описывающими структуру учебного плана, включая блоки дисциплин по выбору, на каком семестре реализуется та или иная дисциплина, возможные траектории обучения с учетом пререквизитов к дисциплинам и т.п., а также была расширена информация о дисциплинах и преподавателях. Для наполнения расширенной онтологии дополнительно потребовались документы, содержащие информацию об аудиторной нагрузке преподавателей.

Расширенная база знаний компетентностно-ориентированного учебного процесса позволяет классифицировать преподавателей по нагрузке и компетенциям, а также отвечать на более сложные вопросы, например:

1. Какая траектория обучения обеспечит освоение некоторого заданного набора компетенций (с учетом ограничений, связанных с тем, что студент может прослушать только одну/несколько дисциплин из каждого блока по выбору, а также с тем, что для того, чтобы записаться на некоторые курсы, необходимо предварительно освоить ряд других дисциплин)?

2. В каком семестре рекомендуется выезжать на включенное обучение, чтобы влияние на выбранный набор компетенций было минимальным? На какие компетенции делать при этом упор при выборе дисциплин на включенном обучении?

Таким образом, созданная база знаний может быть использована не только руководством программ или кафедр, но и студентами профиля с целью построения траекторий индивидуального профессионального развития путем выбора траекторий, т.е. элективных дисциплин как профессионального, так и профильного характера.

## Выводы по главе 3

В третьей главе обсуждаются примеры создания онтологий различных предметных областей и их наполнение с помощью предложенного в работе и описанного во второй главе метода. Согласно этому алгоритму канонического наполнения онтологии из 5 шагов были реализованы и наполнены:

e) Онтология по обработке эмпирических исследований EMPIRION;
f) Онтология орфанных заболеваний Ontomorf;
g) Онтология сборочных единиц для автоматизированного производства автомобилей;
h) Онтология для администрирования учебной деятельности ВУЗа.

Наиболее показательным является пример наполнения онтологии Ontomorf, так как он предполагает не только единоразовое наполнение онтологии до начала эксплуатации, но и регулярное наполнение в процессе ее использования (внесение информации о пациентах для диагностики).

Предлагаемый в диссертации метод наполнения онтологии был апробирован в рамках проекта «Формирование баз знаний на основе данных эмпирических исследований: онтологический подход (ЭМПИРИОН)» (РФФИ № 20-07-00854). Эта онтология позволила обеспечить соблюдение основных традиционных принципов организации хранения и доступа к данным. Именно на примере этой онтологии детально продемонстрировано применение метода.

Задача создания и наполнения онтологической базы знаний сборочных единиц автомобилей возникла в связи с производственной необходимостью систематизации, упорядочивания и переиспользования знаний о сборочных единицах различных автомобилей внутри компании-автопроизводителя. Актуальность данной системы объясняется тем, что компания занимается производством электромобилей с использованием компонент, уже представленных на рынке, и поиск информации об изделиях поставщиков занимает серьезную долю времени сотрудников. Созданная таким образом база знаний продемонстрировала как

жизнеспособность метода ВИТОН, так и возможности предлагаемого метода наполнения онтологий.

Созданная и наполненная с помощью предложенного метода расширенная база знаний компетентностно-ориентированного учебного процесса позволяет классифицировать преподавателей по нагрузке и компетенциям, и может быть использована не только руководством программ или кафедр, но и студентами профиля с целью построения траекторий индивидуального профессионального развития путем выбора траекторий, т.е. элективных дисциплин как профессионального, так и профильного характера.

Все четыре созданные и описанные онтологии относятся к различным предметным областям. Структура источников данных также различается, продемонстрировано, что это не является препятствием для использования разработанного метода. Можно заметить, что для наполнения, в основном, использовались данные из одной или нескольких электронных таблиц. Это обусловлено тем, что именно электронные таблицы широко применяются в повседневной практике в различных областях, а все описанные проекты имеют практическую направленность и выполнялись для конкретных заказчиков, как внутренних, так и внешних.

Таким образом было продемонстрировано, что разработанный метод позволяет наполнять онтологии, относящиеся к различным предметным областям, независимо от их структуры, а также от структуры источников данных, что является отличительной особенностью предложенного метода.

Результаты исследований были получены в рамках выполнения исследований по проектам РФФИ:

- РФФИ № 17-07-00228 МЕтодология и ТЕхнология формирования Онтологий на основе интегРации с гетерогенными источниками данных (МЕТЕОР)
- РФФИ № 20-07-00854 Формирование баз знаний на основе данных эмпирических исследований: онтологический подход (ЭМПИРИОН)

# СПИСОК СОКРАЩЕНИЙ

АПКиТ — Ассоциация предприятий компьютерных и информационных технологий

БД —база данных

БЗ —база знаний

ВУЗ — высшее учебное заведение

ИТ — информационные технологии

КОУП — компетентностно-ориентированные учебные планы

ОИД — онтология источников данных

ООП — основная образовательная программа

ОПО — онтологию предметной области

ПО — программное обеспечение

СЕ — сборочная единица

СППР — система поддержки принятия решений

СУБД — система управления базами данных

АКО — a kind of, тип отношений

DARPA — Defense Advanced Research Projects Agency, агентство передовых оборонных исследовательских проектов

DTD — Document Type Definition, определение типов документа

ETL — extract-transform-load, технологии извлечения, преобразования и загрузки

HTML — HyperText Markup Language, «язык гипертекстовой разметки

IT — информационные технологии

JTP — Java Theorem Prover, объектно- ориентированная система логического вывода, www.ksl.stanford.edu/software/JTP

OCML — Operational Conceptual Modeling Language, язык операционного концептуального моделирования

OKBC — Open Knowledge Base Connectivity, прикладной интерфейс программирования для доступа к базам знаний

OWL — Ontology Web Language, язык описания онтологий для семантической паутины, стандарт W3C

RDF — Resource Description Framework, среда описания ресурса

SQL — structured query language, язык структурированных запросов

TANGO — Table ANalysis for Generating Ontologies, анализ таблиц для генерации онтологий

URI — Uniform Resource Identifier, унифицированный (единообразный) идентификатор ресурса

W3C — World Wide Web Consortium, консорциум WWW, https://www.w3.org/

WWW — Word Wide Web, всемирная паутина

XML — eXtensible Markup Language, расширяемый язык разметки.

XOL — язык обмена онтологиями на основе XML

# ЗАКЛЮЧЕНИЕ

Современные информационные системы все более активно используют базы знаний в своей работе. При этом, как правило, основной моделью представления знаний является онтология. Обзор и анализ литературы, представленный в первой главе, демонстрирует, что среди множества проблем применения онтологий можно выделить задачу автоматизированного наполнения баз знаний онтологического типа экземплярами, решению которой и посвящена данная диссертация.

Существующие методы наполнения баз знаний, основанных на онтологиях, обладают рядом недостатков, в частности, узкой направленностью на решение задач конкретного проекта, сложностью описания моделей источников данных и их трансформации, высокими требованиями к квалификации пользователей, обеспечением возможности интеграции данных из источников только одного типа, отсутствием модульности и расширяемости, а также удобного инструментария.

Основная сложность информационного моделирования состоит в гетерогенности, т.е. разнородности описываемых компонент, данных и знаний. Причем наибольшее влияние гетерогенность источников данных оказывает на процесс наполнения баз знаний. Проблемы, возникающие при наполнении, были классифицированы по этапам жизненного цикла базы знаний. Было выявлено, что основные трудности возникают на этапах разработки и эксплуатации.

Обзор исследований в области онтологического инжиниринга за последние 10 лет показал, что несмотря на наличие большого количества завершенных проектов и промышленных приложений онтологий, имеются существенные недоработки в области теоретического осмысления программных механизмов формирования баз знаний онтологического типа.

На основании обзора литературы была расширена и дополнена генеалогия языков представления онтологий. Проведенный анализ выявил ограничения существующих подходов и методов. В первой главе сформулирована задача исследования по созданию метода наполнения баз знаний онтологического типа и интеграции разнородных источников структурированных данных с учетом семантики предметной области. Основными особенностями описанного далее метода являются:

- Возможность его использования для наполнения базы знаний на различных этапах жизненного цикла.
- Возможность одновременного наполнения из нескольких гетерогенных источников.

- Интегрированность с мировыми стандартами в области языков описания онтологий для обеспечения возможности разработки и эксплуатации базы знаний с использованием существующих инструментов онтологического инжиниринга.

Сам разработанный метод наполнения онтологий описан во второй главе. Там же описана архитектура программного прототипа. Предложенный метод устраняет основные недостатки существующих решений:

1. Необходимости использования и, соответственно, изучения синтаксиса нескольких специфических языков для описания отображений. Также, для эффективной работы требуется создание удобного инструмента для каждого из используемых языков.
2. Узости методов: предложено большое количество различных методов для интеграции данных в онтологию, но каждый метод работает только для одного типа источника данных.
3. Отсутствии рекомендаций по интеграции данных из нескольких источников одновременно. Если есть необходимость интегрировать в одну онтологию данные, например, из базы данных и из XML-документа, то, используя существующие методы, придется действовать последовательно. Например, наполнить исходную онтологию данными из БД, затем полученную онтологию — данными из XML-документа. Так как данные в источниках подвергаются изменениям, то эту процедуру необходимо регулярно повторять, что ведет к неоправданным издержкам.
4. Наличии противоречий на этапе интеграции данных из различных источников. Так как авторы приведенных во 2 главе методов фокусируются на интеграции данных из одного источника, то вопрос разрешения подобных противоречий не рассматривается.
5. Неясности, каким образом устанавливается соединение с источником данных, как проходит аутентификация и т.п.
6. Отсутствии рекомендаций по наполнению онтологии в случае, если компании используют собственные форматы хранения данных, а также для новых форматов данных, которые могут появиться в будущем.

Предложенный метод METEOP (МЕтодология и ТЕхнология наполнения Онтологии на основе интегРации с гетерогенными структурированными источниками данных) использует накопленные массивы информации независимо от формы их хранения и представления (например, баз данных, электронных таблиц и XML-файлов). Применение метода снижает трудоемкость наполнения и обогащения онтологий.

Предлагаемая методика направлена на осуществление однотипных операций или на решение повторяющихся задач, требующих знаний правил и процедур, а также сопоставления информации из различных источников. Такого рода задачи возникают на предприятиях, в

учебных заведениях, а также в организациях различной ведомственной принадлежности в рамках цифрового документооборота. Предложенная методика может быть положена в основу систем поддержки принятия решений, основанных на знаниях.

Новизна предложенного метода заключается в использовании вспомогательной Онтологии источников данных (ОИД), которая может быть расширена за счет добавления дополнительных типов источников данных. Использование ОИД дополнено разработкой требований к формированию правил отображения, позволяющих связать ОИД и конкретную наполняемую онтологию предметной области (ОПО) для наполнения ее экземплярами.

Применение разработанного метода METEOP можно описать алгоритмом, состоящим из 5 шагов, два из которых выполняются автоматически:

1. Идентификация источников данных.
2. Спецификация источников данных с помощью вспомогательной ОИД.
3. Извлечение структуры данных из источников данных в ОИД.
4. Задание правил отображения в ОПО.
5. Наполнение ОПО.

Ключевым шагом алгоритма наполнения ОПО является задание правил отображения. Оно осуществляется с помощью встроенного в OWL механизма аннотирования. В работе введены новые аннотационные свойства: 4 основных и 2 вспомогательных (подробнее см. параграф 2.2.3). Такой подход позволил исключить необходимость в дополнительной инструментальной поддержке метода METEOP, так как аннотирование может быть выполнено с помощью любого редактора онтологий.

Также была разработана новая архитектура программного прототипа, реализующего предложенный метод METEOP. В основу архитектуры положена известная архитектура ETL-систем. Особенностью предлагаемой архитектуры является то, что получателем данных является не хранилище или база данных, а онтология или база знаний.

Разработанный прототип реализует созданный метод METEOP и позволяет наполнять базы знаний онтологического типа из гетерогенных источников. Программный прототип включает 4 модуля, поддерживающих процессы спецификации источников данных, извлечения структуры данных, задания правил отображения и собственно наполнения ОПО. Весь процесс наполнения онтологии с помощью метода METEOP проиллюстрирован на условном примере.

Работа является продолжением и развитием многочисленных исследований и разработок в области онтологического инжиниринга. В диссертации получены следующие результаты:

1. Предложена новая генеалогия языков представления знаний, которая существенно расширяет и дополняет предложенную ранее в [Казекин, 2008].

2. Исследовано влияние когнитивных стилей на создание онтологических моделей, где впервые были применены не только качественные, но и количественные методы исследования, что позволило получить новые статистически значимые зависимости между когнитивным типом человека и характеристиками построенной им онтологии, опубликованные в [Gavrilova, Leshcheva, Ontology…, 2015; Гаврилова, Лещева, 2016].
3. Предложен новый универсальный метод наполнения онтологий, который отличается комплексным подходом и впервые позволяет одновременно интегрировать данные из источников различного типа, учитывает их распределенную природу, а также предлагает решение проблемы разрешения противоречий в интегрируемых данных, что позволит снизить трудоемкость наполнения и обогащения онтологий, используя накопленные массивы информации независимо от формы их хранения и представления. Также, предлагаемое решение является расширяемым для обеспечения потенциальной возможности работы с новыми форматами данных в будущем. Разработанный метод реализации процесса трансформации структур данных в онтологическую модель не использует дополнительных языков для этого, а обходится только средствами онтологического моделирования.
4. Разработаны некоторые шаблоны построения правил маппирования исходной онтологии в базу знаний предметной области.
5. Сформулированы критерии выбора программной платформы для реализации предложенного метода.
6. Разработана архитектура и реализован программный прототип, реализующий процедуру консолидации данных, хранящихся в структурированных источниках данных, в базу знаний предметной области.

Полученные результаты были апробированы на 4-х предметных областях:

a) Онтология по обработке эмпирических исследований EMPIRION;
b) Онтология орфанных заболеваний Ontomorf;
c) Онтология сборочных единиц для автоматизированного производства автомобилей;
d) Онтология для администрирования учебной деятельности ВУЗа.

Результаты исследований были получены в рамках выполнения исследований по проектам РФФИ:

- РФФИ № 17-07-00228 МЕтодология и ТЕхнология формирования Онтологий на основе интегРации с гетерогенными источниками данных (МЕТЕОР)
- РФФИ № 20-07-00854 Формирование баз знаний на основе данных эмпирических исследований: онтологический подход (ЭМПИРИОН)

Некоторые предварительные результаты были получены в ходе выполнения проектов РФФИ 11-07-00140 2011-2013 «Структурирование знаний и КОнтента МЕтодами группового дизайна онТологий (КОМЕТ)» и 14-07-00294 «Интеллектуальные Сервисы поддержки ПОРТалов знаний на основе онтологий (ИнС-ПОРТ)».

Результаты работы опубликованы в трудах международных и всероссийских конференций:

1. XIX Национальная конференция по искусственному интеллекту с международным участием (КИИ-2021) (Таганрог, 2021)
2. GSOM Emerging Markets Conference (Санкт-Петербург, 2018)
3. Инжиниринг предприятий и управление знаниями (ИП&УЗ-2018) (Москва, 2018)
4. IV International Scientific Conference “The Convergence of Digital and Physical Worlds: Technological, Economic and Social Challenges” (Санкт-Петербург, 2018)
5. BIR 2018 Short Papers, Workshops and Doctoral Consortium, (Стокгольм, Швеция, 2018)
6. XXII Международная научно-практическая конференция «Системный анализ в проектировании и управлении» (Санкт-Петербург, 2018)
7. XVI Национальная конференция по искусственному интеллекту с международным участием КИИ-2018 (Москва, 2018)
8. GSOM Emerging Markets Conference 2018 (Санкт-Петербург, 2018)
9. Инжиниринг предприятий и управление знаниями (ИП&УЗ-2017) (Москва, 2017)
10. Системный анализ и информационные технологии (САИТ-2017) (Светлогорск, 2017)
11. IEEE 30th Neumann Colloquium (Будапешт, Венгрия, 2017)
12. XV Национальная конференция по искусственному интеллекту с международным участием КИИ-2016 (Смоленск, 2016)
13. GSOM Emerging Markets Conference-2016: Business and Government Perspectives (Санкт-Петербург, 2016)
14. Знания-Онтологии-Теории (ЗОНТ-15) (Новосибирск, 2015)
15. International Conference on Knowledge Engineering and Ontology Development (KEOD 2014) (Rome, Italy, 2014)
16. The 8th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS 2014) (UK, Birmingham, 2014)
17. Federated Conference on Computer Science and Information Systems (Kraków, Poland, 2013)
18. Знания-Онтологии-Теории (ЗОНТ-13) (Новосибирск, 2013)
19. Инжиниринг предприятий и управление знаниями (ИП&УЗ-2013) (Москва, 2013)

20. XI Всероссийская конференция «Преподавание информационных технологий в Российской Федерации» (Воронеж, 2013)

**Список работ автора по теме диссертации**:


1. Leshcheva I., Begler A. A method of semi-automated ontology population from multiple semi-structured data sources //Journal of Information Science. – 2020. DOI: 10.1177/0165551520950243.
2. Кудрявцев Д. В. и др. Метод коллективной визуальной разработки онтологического графа знаний/ Кудрявцев Д.В., Беглер А.М., Гаврилова Т.А., Лещева И.А., Кубельский М.В., Тушканова О.Н. //Искусственный интеллект и принятие решений. – 2019. – №. 1. – С. 27-38.
3. Благов Е. Ю., Лещева И. А., Щербан С. А. Онтологический подход в практике образовательной деятельности: формирование траекторий индивидуального профессионального развития студентов //Открытое образование. – 2018. – Т. 22. – №. 5. – С. 26-39.
4. Гаврилова Т. А., Лещева И. А. Понятийные структуры знаний и когнитивный стиль //Психология. Журнал Высшей школы экономики. – 2016. – Т. 13. – №. 1. с. 154-176.
5. Gavrilova T. A., Leshcheva I. A. Ontology design and individual cognitive peculiarities: A pilot study //Expert systems with Applications. – 2015. – Т. 42. – №. 8. – С. 3883-3892.
6. Gavrilova T. A., Leshcheva I. A. Building Collaborative Ontologies: A Human Factors Approach //Collaborative Knowledge in Scientific Research Networks. – IGI Global, 2015. – С. 305-324.
7. Gavrilova T., Leshcheva I., Strakhovich E. Gestalt principles of creating learning business ontologies for knowledge codification //Knowledge Management Research & Practice. – 2015. – Т. 13. – №. 4. – С. 418-428.
8. Gavrilova T., Leshcheva I. The interplay of knowledge engineering and cognitive psychology: learning ontologies creating //International Journal of Knowledge and Learning. – 2015. – Т. 10. – №. 2. – С. 182-197.
9. Leshcheva I., Gavrilova T. How the cognitive features testing can assist in evaluating collective ontology engineering //International Journal of High Performance Computing and Networking. – 2015. – Т. 8. – №. 3. – С. 275-284.
10. Лещева, И. А., Лещев, Д. В. Анализ динамики изменений научной области методами онтологического инжиниринга //Открытые семантические технологии проектирования интеллектуальных систем. – 2014. – № 4. – С. 483-486.
11. Гаврилова, Т. А., Кудрявцев, Д. В., Лещева, И. А., Павлов, Я. Ю. Об одном методе классификации визуальных моделей //Бизнес-информатика. – 2013. – № 4(26). – С. 21-34.

12. Gavrilova T., Bolotnikova E., Leshcheva I., Blagov E., Yanson A. Measuring psychological impact on group ontology design and development: an empirical approach //Communications in Computer and Information Science. – 2013. – Vol. 394. – P. 29-43. – DOI 10.1007/978-3-642-41360-5_3.
13. Гаврилова Т. А., Лещева И. А., Кудрявцев Д. В. Использование моделей инженерии знаний для подготовки специалистов в области информационных технологий //Системное программирование. – 2012. – Т. 7. – №. 1. – С. 90-105.
14. Gavrilova T. A., Leshcheva I. A., Rumyantseva M. N. Knowledge elicitation methods taxonomy: Russian view // Lecture Notes in Computer Science. – 2011. – Vol. 6881 LNAI. – No Part 1. – P. 337-346. – DOI 10.1007/978-3-642-23851-2_35.
15. Гаврилова Т. А., Лещева И. А., Страхович Э. В. Об использовании визуальных концептуальных моделей в преподавании //Вестник Санкт-Петербургского университета. Менеджмент. – 2011. – № 4. – С. 124-150.
16. Гаврилова Т. А., Лещева И. А., Лещев Д. В. Использование онтологии в качестве дидактического средства //Искусственный интеллект. – 2000. – № 3. – С. 34-39.

# СПИСОК ЛИТЕРАТУРЫ


1. Беглер А. М., Костоусов, С. А. Критерии выбора инструмента по работе с корпоративными базами знаний //Сборник студенческих работ V Международной научно-практической конференции “Управленческие науки в современном мире” 2018 / г. Москва, (2018). – С. 18–23.
2. Благов Е. Ю., Лещева И. А., Щербан С. А. Онтологический подход в практике образовательной деятельности: формирование траекторий индивидуального профессионального развития студентов //Открытое образование. – 2018. – Т. 22. – №. 5. – С. 26-39.
3. Бова В. В. Онтологическая модель интеграции данных и знаний в интеллектуальных информационных системах //Известия Южного федерального университета. Технические науки. – 2015. – №. 4 (165). – С. 225-237.
4. Гаврилова Т. А. и др. Метод визуальной коллективной разработки онтологического графа знаний //Искусственный интеллект и принятие решений. – 2019. – №. 1. – С. 27-38.
5. Гаврилова Т. А., Кудрявцев Д. В., Муромцев Д. И. Инженерия знаний. Модели и методы. – Санкт-Петербург : Лань, 2020. – 324 С.
6. Гаврилова, Т. А., Кудрявцев, Д. В., Лещева, И. А., Павлов, Я. Ю. Об одном методе классификации визуальных моделей //Бизнес-информатика. – 2013. – № 4(26). – С. 21-34.
7. Гаврилова Т. А., Лещева И. А. Понятийные структуры знаний и когнитивный стиль //Психология. Журнал Высшей школы экономики. – 2016. – Т. 13. – №. 1.
8. Гаврилова Т. А., Лещева И. А., Кудрявцев Д. В. Использование моделей инженерии знаний для подготовки специалистов в области информационных технологий //Системное программирование. – 2012. – Т. 7. – №. 1. – С. 90-105.
9. Гаврилова Т. А., Лещева И. А., Лещев Д. В. Использование онтологии в качестве дидактического средства //Искусственный интеллект. – 2000. – № 3. – С. 34-39.
10. Гаврилова Т. А., Лещева И. А., Страхович Э. В. Об использовании визуальных концептуальных моделей в преподавании //Вестник Санкт-Петербургского университета. Менеджмент. – 2011. – № 4. – С. 124-150.
11. Гаврилова Т. А., Муромцев Д. И. Интеллектуальные технологии в менеджменте: инструменты и системы. – СПб. : Высшая школа менеджмента. – 2007. – 488 С.
12. Гаврилова Т. А., Хорошевский В. Ф. Базы знаний интеллектуальных систем. – СПб. : Питер – 2000. – 384 С.

13. Голенков В. В. и др. Онтологическое проектирование гибридных семантически совместимых интеллектуальных систем на основе смыслового представления знаний //Онтология проектирования. – 2019. – Т. 9. – №. 1 (31). – С. 132-151.
14. Грибова В. В. и др. Онтология медицинской диагностики для интеллектуальных систем поддержки принятия решений //Онтология проектирования. – 2018. – Т. 8. – №. 1 (27). – С. 58-73.
15. Григорьев Л. Ю., Заблоцкий А. А., Кудрявцев Д. В. Технология наполнения баз знаний онтологического типа //Научно-технические ведомости Санкт-Петербургского государственного политехнического университета. Информатика. Телекоммуникации. Управление. – 2012. – №. 3 (150). – С. 27-36.
16. Давидовский М. В. Формализация задачи согласования онтологий в децентрализованных системах // Вестник запорожского национального университета. – 2011. – №. 1. – С. 25-29.
17. Доброхотов А. Л. Онтология//Новая философская энциклопедия //М.: Мысль. – 2010. – С. 149-152.
18. Ерженин Р. В., Массель Л. В. Онтологический подход к представлению знаний о методологии моделирования сложной системы управления //Онтология проектирования. – 2020. – Т. 10. – №. 4 (38). – С. 463-476.
19. Загорулько Ю. А., Боровикова О. И. Технология построения онтологий для порталов научных знаний //Вестник Новосибирского государственного университета. Серия: Информационные технологии. – 2007. – Т. 5. – №. 2. – С. 42-52.
20. Казекин М.М. История языков представления онтологий // Компьютерные инструменты в образовании. – 2008. – №4 –С. 3-11.
21. Клещев А. С., Москаленко Ф. М., Черняховская М. Ю. Онтология и модель онтологии предметной области «Медицинская диагностика» //Владивосток: Изд-во ИАПУ ДВО РАН. – 2005.
22. Кобринский Б. А. Триединство факторов уверенности в задачах медицинской диагностики //Искусственный интеллект и принятие решений. – 2018. – №. 2. – С. 62-72.
23. Кобринский Б.А., Благосклонов Н.А., Демикова Н.С., Грибова В.В., Шалфеева Е.А., Петряева М.В. Возможности применения онтологического подхода к диагностике орфанных заболеваний// Семнадцатая Национальная конференция по искусственному интеллекту с международным участием. КИИ-2019 (21–25 октября 2019 г., г. Ульяновск, Россия). Сборник научных трудов. В 2 т. – Ульяновск: УлГТУ, 2019. – Т.2. – С. 227. ISBN 978-5-9795-1940-1.
24. Кудрявцев Д. В. и др. Метод коллективной визуальной разработки онтологического графа знаний/ Кудрявцев Д.В., Беглер А.М., Гаврилова Т.А., Лещева И.А., Кубельский М.В., Тушканова О.Н. //Искусственный интеллект и принятие решений. – 2019. – №. 1. – С. 27-38.

25. Лещева, И. А., Лещев, Д. В. Анализ динамики изменений научной области методами онтологического инжиниринга //Открытые семантические технологии проектирования интеллектуальных систем. – 2014. – № 4. – С. 483-486.
26. Никитин В.В. Информационно-методическое обеспечение перечня формирования направлений и специальностей в области информационно-коммуникационных технологий // Москва: МАКС Пресс, 2006. – 272 С.
27. Осипов Г.С. Методы искусственного интеллекта. – М.: Физматлит, 2011.
28. Словарь согласованных терминов и определений в области образования государств-участников Содружества Независимых Государств : ок. 100 терминов / М-во образования Рос. Федерации, Департамент содерж. высш. проф. образования, Упр. междунар. образования и сотрудничества, Исслед. центр проблем качества подгот. специалистов Моск. гос. ин-та стали и сплавов (техн. ун-та); [авт.-сост.: О. Л. Ворожейкина и др.] ; под науч. ред. Н.А. Селезневой. – Москва : Исслед. центр проблем качества подгот. специалистов, 2004. – 167 с.
29. Тузовский А. Ф. Разработка систем управления знаниями на основе единой онтологической базы знаний //Известия Томского политехнического университета. 2007. Т. 310, № 2. С. 182-185.
30. Abraham R., Erwig M. Header and unit inference for spreadsheets through spatial analyses //2004 IEEE Symposium on Visual Languages-Human Centric Computing. – IEEE, 2004. – С. 165-172.
31. Anicic, N., Ivezic, N., Marjanovic, Z. Mapping XML Schema to OWL, Enterprise Interoperability, Springer London, 2007.
32. Aussenac-Gilles N., Kamel M. Ontology Learning by Analyzing XML Document Structure and Content //KEOD. – 2009. – Т. 9. – С. 159-165.
33. Baghernezhad-Tabasi S. et al. IOPE: Interactive ontology population and enrichment guided by ontological constraints //International Conference on Web Information Systems Engineering. – Springer, Cham, 2021. – С. 321-336.
34. Bakkas J., Jakjoud W., Bahaj M. Semantic mapping at the schema level of XML documents to ontologies //2014 International Conference on Next Generation Networks and Services (NGNS). – IEEE, 2014. – С. 165-169.
35. Barrasa J., Corcho Ó., Gómez-pérez A. R2O, an Extensible and Semantically based Database-to-Ontology Mapping Language //In Proceedings of the 2nd Workshop on Semantic Web and Databases (SWDB2004), Toronto, Canada. – 2004.
36. Bechhofer S. et al. OilEd: a reason-able ontology editor for the semantic web //KI 2001: Advances in Artificial Intelligence. – Springer Berlin Heidelberg, 2001. – С. 396-408.

37. Berdier C., Roussey C. Urban ontologies: The towntology prototype towards case studies //Ontologies for Urban Development. – Springer Berlin Heidelberg, 2007. – С. 143-155.
38. Berners-Lee T., Hendler J., Lassila O. The semantic web //Scientific american. – 2001. – Т. 284. – №. 5. – С. 28-37.
39. Bizer C. D2R MAP - a database to RDF mapping language. In Proceedings of the 12th International World Wide Web Conference (WWW 2003), Budapest, Hungary, 2003.
40. Bizer C., Seaborne A. D2RQ-treating non-RDF databases as virtual RDF graphs //Proceedings of the 3rd international semantic web conference (ISWC2004). – Hiroshima : Citeseer, 2004. – Т. 2004.
41. Bohring H., Auer S. Mapping XML to OWL ontologies //Marktplatz Internet: Von e-Learning bis e-Payment, 13. Leipziger Informatik-Tage (LIT 2005). – 2015 – pp. 147-156.
42. Buitelaara P., Cimianob P., Frankc A., Hartungc M., Racioppa S. Ontology-based information extraction and integration from heterogeneous data sources // Int. J. Human-Computer Studies – 2008. – Т. 66. – №. 11.– С. 759–788.
43. Bullinger A. C. Classification—The OntoCube //Innovation and Ontologies: Structuring the Early Stages of Innovation Management. – 2009. – С. 172-195.
44. Cerbah F. Learning highly structured semantic repositories from relational databases. – Springer Berlin Heidelberg, 2008. – С. 777-781.
45. Chasseray Y. et al. A generic metamodel for data extraction and generic ontology population //Journal of Information Science. – 2021. DOI: 10.1177/0165551521989641.
46. ChristopoulouE., Kameas A. GAS Ontology: An ontology for collaboration among ubiquitous computing devices // Int. J. Human-Computer Studies – 2005. – Т. 62. – №. 5 – С. 664–685.
47. Clarkson K. et al. User-centric ontology population //European Semantic Web Conference. – Springer, Cham, 2018. – С. 112-127.
48. Cruz C., Nicolle C. Ontology Enrichment and Automatic Population From XML Data //ODBIS. – 2008. – Т. 2008. – С. 17-20.
49. Cullot N., Ghawi R., Yetongnon K. DB2OWL : A tool for automatic database-to-ontology mapping. In Proceedings of the 15th Italian Symposium on Advanced Database Systems, pages 491–494, Torre Canne, Fasano, BR, Italy, 2007.
50. Cyganiak R., Bizer C., Maresch O., Becker C. The D2RQ mapping language v0.8, 2012. URL http://d2rq.org/d2rq-language
51. Das, S, Sundara, S, Cyganiak, R. R2RML: RDB to RDF mapping language, 2012. https://www.w3.org/TR/r2rml/
52. De Laborda C. P., Conrad S. Database to semantic web mapping using RDF query languages //Conceptual Modeling-ER 2006. – Springer Berlin Heidelberg, 2006. – С. 241-254.

53. De Leenheer P., Debruyne C. DOGMA-MESS: A tool for fact-oriented collaborative ontology evolution //OTM Confederated International Conferences "On the Move to Meaningful Internet Systems". – Springer, Berlin, Heidelberg, 2008. – pp. 797-806.
54. De Moor A., De Leenheer P., Meersman R. DOGMA-MESS: A meaning evolution support system for interorganizational ontology engineering //International Conference on Conceptual Structures. – Springer, Berlin, Heidelberg, 2006. – pp. 189-202.
55. Dolbear C., Goodwin J. Position paper on expressing relational data as RDF //W3C Workshop on RDF Access to Relational Databases. – 2007. – С. 25-26.
56. Domingue J. Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web. Proceedings of the 11th Banff Knowledge Acquisition Workshop, Banff, Alberta, Canada, April 18-23, 1998.
57. Domingue J., Fensel D., Hendler J. A. (ed.). Handbook of semantic web technologies. – Berlin, Heidelberg : Springer-Verlag Berlin Heidelberg, 2011. 1035 с.
58. Domingue J., Motta E., Garcia O. C. Knowledge Modelling in WebOnto and OCML: A User Guide //Knowledge Media Institute, Milton Keynes, UK. – 1999.
59. Erling O. Requirements for relational to RDF mapping, 2008. URL http://www.w3.org/wiki/Rdb2RdfXG/ReqForMappingByOErling.
60. Farquhar A., Fikes R., Rice J. The ontolingua server: A tool for collaborative ontology construction //International journal of human-computer studies. – 1997. – Т. 46. – №. 6. – С. 707-727.
61. Ferdinand M., Zirpins C., Trastour D. Lifting XML Schema to OWL, in: Koch, Nora and Fraternali, Piero and Wirsing, Martin (Hrsg.): Web Engineering, ICWE 2004, Munich, Germany, July 26-30, 2004.
62. García, R., Celma, O. Semantic Integration and Retrieval of Multimedia Metadata, ISWC, Galway, Ireland, 2005.
63. Gavrilova T., Bolotnikova E., Leshcheva I., Blagov E., Yanson A. Measuring psychological impact on group ontology design and development: an empirical approach //Communications in Computer and Information Science. – 2013. – Vol. 394. – P. 29-43.
64. Gavrilova T. A., Leshcheva I. A. Ontology design and individual cognitive peculiarities: A pilot study //Expert systems with Applications. – 2015. – Т. 42. – №. 8. – С. 3883-3892.
65. Gavrilova T. A., Leshcheva I. A. Building Collaborative Ontologies: A Human Factors Approach //Collaborative Knowledge in Scientific Research Networks. – IGI Global, 2015. – С. 305-324.
66. Gavrilova T., Leshcheva I. The interplay of knowledge engineering and cognitive psychology: learning ontologies creating //International Journal of Knowledge and Learning. – 2015. – Т. 10. – №. 2. – С. 182-197.

67. Gavrilova T. A., Leshcheva I. A., Rumyantseva M. N. Knowledge elicitation methods taxonomy: Russian view // Lecture Notes in Computer Science. – 2011. – Vol. 6881 LNAI. – No Part 1. – P. 337-346.
68. Gavrilova T., Leshcheva I., Strakhovich E. Gestalt principles of creating learning business ontologies for knowledge codification //Knowledge Management Research & Practice. – 2015. – Т. 13. – №. 4. – С. 418-428.
69. Genesereth M. R., Fikes R. Knowledge interchange format reference manual (version 3.0) //Computer Science Department, Stanford University. – 1992.
70. Genesereth M. R., Nilsson N. J. Logical Foundations of Artificial Intelligence // Los Altos, California : Morgan Kaufmann. – 1987. – 405 С.
71. Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., ... & Tu, S. W. The evolution of Protégé: an environment for knowledge-based systems development //International Journal of Human-computer studies. – 2003. – Т. 58. – №. 1. – С. 89-123.
72. Giaretta P., Guarino N. Ontologies and knowledge bases towards a terminological clarification //Towards very large knowledge bases: knowledge building & knowledge sharing. – 1995. – С. 25-32.
73. Gómez-Pérez A., Fernandez-Lopez M., Corcho O. Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. – Springer-Verlag London, 2004. – 404 С.
74. Gruber T. R. A translation approach to portable ontology specifications //Knowledge acquisition. – 1993. – Т. 5. – №. 2. – С. 199-220.
75. Gruber T. R. Toward principles for the design of ontologies used for knowledge sharing // International Journal of Human-Computer Studies. – 1995. – Т. 43. – №. 5-6. – С. 907-928.
76. Hacherouf M., Bahloul S. N., Cruz C. Transforming XML documents to OWL ontologies: A survey //Journal of Information Science. – 2015. – Т. 41. – №. 2. – С. 242-259.
77. Haghighi P. D. et al. Ontology-based service-oriented architecture for emergency management in mass gatherings //Service-Oriented Computing and Applications (SOCA), 2010 IEEE International Conference on. – IEEE, 2010. – С. 1-7.
78. Han L. et al. RDF123: from Spreadsheets to RDF //International Semantic Web Conference. – Springer, Berlin, Heidelberg, 2008. – С. 451-466.
79. Ivanović M., Budimac Z. An overview of ontologies and data resources in medical domains //Expert Systems with Applications. – 2014. – Т. 41. – №. 11. – С. 5158-5166.
80. Langegger A., Wöß W. XLWrap–querying and integrating arbitrary spreadsheets with SPARQL //International Semantic Web Conference. – Springer, Berlin, Heidelberg, 2009. – С. 359-374.

81. Hu W., Qu Y. Discovering simple mappings between relational database schemas and ontologies. – Springer Berlin Heidelberg, 2007. – С. 225-238.
82. Isern D., Sánchez D., Moreno A. Ontology-driven execution of clinical guidelines //Computer methods and programs in biomedicine. – 2012. – Т. 107. – №. 2. – С. 122-139.
83. Karp P. D., Chaudhri V. K., Thomere J. XOL: An XML-based ontology exchange language. – 1999.
84. Kifer M., Lausen G. F-logic: a higher-order language for reasoning about objects, inheritance, and scheme //ACM SIGMOD Record. – ACM, 1989. – Т. 18. – №. 2. – С. 134-146.
85. Knublauch, H., Fergerson, R. W., Noy, N. F., & Musen, M. A. The Protégé OWL plugin: An open development environment for semantic web applications //The Semantic Web–ISWC 2004. – Springer Berlin Heidelberg, 2004. – С. 229-243.
86. Křemen P. et al. Ontology-driven mindmapping //Proceedings of the 8th International Conference on Semantic Systems. – 2012. – С. 125-132.
87. Lenat D. B., Guha R. V. The evolution of CycL, the Cyc representation language //ACM SIGART Bulletin. – 1991. – Т. 2. – №. 3. – С. 84-87.
88. Leshcheva I., Begler A. A method of semi-automated ontology population from multiple semi-structured data sources //Journal of Information Science. – 2020. DOI: 10.1177/0165551520950243.
89. Leshcheva I., Gavrilova T. How the cognitive features testing can assist in evaluating collective ontology engineering //International Journal of High Performance Computing and Networking. – 2015. – Т. 8. – №. 3. – С. 275-284.
90. Lubani M., Noah S. A. M., Mahmud R. Ontology population: Approaches and design aspects //Journal of Information Science. – 2019. – Т. 45. – №. 4. – С. 502-515.
91. MacGregor R. Retrospective on LOOM //Information Sciences Institute, University of Southern California, Tech. Rep. – 1999.
92. Michel F., Montagnat J., Faron-Zucker C. A survey of RDB to RDF translation approaches and tools: Research Report. 2014. URL: http://hal.archives-ouvertes.fr/hal-00903568
93. Motta E. An overview of the OCML modelling language //the 8th Workshop on Methods and Languages. – 1998.
94. Musen M. A. The protégé project: a look back and a look forward //AI matters. – 2015. – Т. 1. – №. 4. – С. 4-12.
95. Nederstigt L. J. et al. FLOPPIES: a framework for large-scale ontology population of product information from tabular data in e-commerce stores //Decision Support Systems. – 2014. – Т. 59. – С. 296-311.
96. O'Connor M. J., Das A. Acquiring OWL ontologies from XML documents //Proceedings of the sixth international conference on Knowledge capture. – 2011. – С. 17-24.

97. O'Connor M. J., Halaschek-Wiener C., Musen M. A. Mapping master: a flexible approach for mapping spreadsheets to OWL //International Semantic Web Conference. – Springer, Berlin, Heidelberg, 2010. – С. 194-208.
98. Ozturk O. OPPCAT: Ontology population from tabular data //Journal of Information Science. – 2020. – Т. 46. – №. 2. – С. 161-175.
99. Pan J. Z. et al. (ed.). Exploiting Linked Data and Knowledge Graphs in Large Organisations. – Cham : Springer, 2017.
100. Patlak, M., Nass, S., Henderson, I., Lashof, J. (Eds.), 2001. Mammography and Beyond: Developing Technologies for the Early Detection of Breast Cancer. National Academy Press, Washington, DC.
101. Petasis G. et al. Ontology population and enrichment: State of the art // Knowledge-driven multimedia information extraction and ontology evolution. – Springer-Verlag, 2011. – pp. 134-166.
102. Quix C., Kensche D., Li X. Matching of ontologies with xml schemas using a generic metamodel //OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". – Springer, Berlin, Heidelberg, 2007. – С. 1081-1098.
103. Reed S.L., Lenat D.B. Mapping ontologies into Cyc //AAAI 2002 Conference Workshop on Ontologies For The Semantic Web. – 2002. – С. 1-6.
104. Rodrigues, T., Rosa, P., Cardoso, J. Mapping XML to Exiting OWL ontologies, International Conference WWW/Internet 2006.
105. Rubin D.L. Creating and curating a terminology for radiology: Ontology modeling and analysis // J. Digit. Imaging. 2008. Vol. 21, № 4. P. 355–362.
106. Ruiz F., Hilera J. R. Using ontologies in software engineering and technology //Ontologies for software engineering and software technology. – Springer Berlin Heidelberg, 2006. – С. 49-102.
107. Sahoo S. S. et al. A survey of current approaches for mapping of relational databases to RDF //W3C RDB2RDF Incubator Group Report. – 2009. URL http://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SurveyReport_01082009.pdf. Sahoo, S. S., Halb, W., Hellmann, S., Idehen, K., Thibodeau Jr, T., Auer, S., ... & Ezzat, A.
108. Sanchez-Cohen I. et al. A decision support system for rainfed agricultural areas of Mexico //Computers and Electronics in Agriculture. – 2015. – Т. 114. – С. 178-188.
109. Shadbolt N., Motta E., Rouge A. Constructing knowledge-based systems //Software, IEEE. – 1993. – Т. 10. – №. 6. – С. 34-38.
110. Simperl E., Luczak-Rösch M. Collaborative ontology engineering: a survey //The Knowledge Engineering Review. – 2014. – Т. 29. – №. 1. – С. 101-131.
111. Song F., Zacharewicz G., Chen D. An ontology-driven framework towards building enterprise semantic information layer //Advanced Engineering Informatics. – 2013. – Vol. 27. – №. 1. – С. 38-50.

112. Stadnicki A., Pietroń F. F., Burek P. Towards a Modern Ontology Development Environment //Procedia Computer Science. – 2020. – Т. 176. – С. 753-762.
113. Stevens R., Aranguren M.E., Wolstencroft K., Sattler U., Drummond N., Horridge M., Rector A. Using OWL to model biological knowledge // International Journal of Human-Computer Studies. Volume 65, Issue 7, July 2007, С. 583–594.
114. Studer R. et al. Situation and Perspective of Knowledge Engineering //Knowledge Engineering and Agent Technology. – 2004. – С. 237-252.
115. Suárez-Figueroa M. C., Gómez-Pérez A., Fernández-López M. The NeOn methodology for ontology engineering //Ontology engineering in a networked world. – Springer Berlin Heidelberg, 2012. – С. 9-34.
116. Sure Y. et al. OntoEdit: Collaborative Ontology Development for the Semantic Web // Semant. Web — ISWC 2002. 2002. P. 221–235.
117. Svihla M., Jelinek I. Two layer mapping from database to RDF //Proceedings of Electronic Computers and Informatics (ECI). – 2004.
118. Swartout B. et al. Ontosaurus: a tool for browsing and editing ontologies //9th Banff Knowledge Aquisition for KNowledge-based systems Workshop. – 1996.
119. Tempich C. et al. An argumentation ontology for distributed, loosely-controlled and evolving engineering processes of ontologies (DILIGENT) //European Semantic Web Conference. – Springer, Berlin, Heidelberg, 2005. – pp. 241-256.
120. Tijerino Y. A. et al. Towards ontology generation from tables //World Wide Web. – 2005. – Т. 8. – №. 3. – С. 261-285.
121. Uschold M. Demystifying Owl for the Enterprise (Synthesis Lectures on Semantic Web: Theory and Technology) //Morgan & Claypool Publishers, 2018. – С. 237.
122. Uschold M., Gruninger M. Ontologies and semantics for seamless connectivity //ACM SIGMod Record. – 2004. – Т. 33. – №. 4. – С. 58-64.
123. Valarakos A. G. et al. Enhancing ontological knowledge through ontology population and enrichment //International Conference on Knowledge Engineering and Knowledge Management. – Springer, Berlin, Heidelberg, 2004. – С. 144-156.
124. Vassiliadis P., Simitsis A. Extraction, Transformation, and Loading // In: Liu L and Tamer M (eds) Encyclopedia of Database Systems. New York: Springer – 2009. – Т. 10. – pp. 1095–1101.
125. Xiang Z. et al. Ontorat: Automatic generation of new ontology terms, annotations, and axioms based on ontology design patterns // J. Biomed. Semantics. 2015. Vol. 6, № 1. P. 1–10.
126. Yunianta A. et al. OntoDI: The methodology for ontology development on data integration //Int. J. Adv. Comput. Sci. Appl. – 2019. – Т. 10. – №. 1. – С. 160-168.

127. Zagorulko Y. A., Borovikova O. I., Zagorulko G. B. The development of a system for automated ontology building based on heterogeneous ontology design patterns //Journal of Physics: Conference Series. – IOP Publishing, 2021. – Т. 1715. – №. 1. – С. 012014.

# ПРИЛОЖЕНИЕ

Список правил для установления связей, показанных пунктирной линией в левой части рисунка 22.

```
#Связь Characteristic relatesToPeriod Period
Characteristic(?it), Phase(?ph), relatesToPhase(?it, ?ph), Period(?p), relatesToPeriod(?ph, ?p) -> relatesToPeriod(?it, ?p)

#Связь Characteristic relatesToDisease Disease
Characteristic(?it), Phase(?ph), Disease(?d), relatesToPhase(?it, ?ph), relatesToDisease(?ph, ?d) -> relatesToDisease(?it, ?d)

#Значение hasTotalManifestation для первого элемента, если Manifestation не равно -1
Pointer(pointer), Period(?per), hasFirst(pointer, ?per), Characteristic(?h), relatesToPeriod(?h, ?per), hasManifestation(?h, ?number), notEqual(?number, -1) -> hasTotalManifestation(?h, ?number)

#Значение hasTotalManifestation для первого элемента, если Manifestation равно -1
Pointer(pointer), Period(?per), hasFirst(pointer, ?per), Characteristic(?h), relatesToPeriod(?h, ?per), hasManifestation(?h, ?number), equal(?number, -1) -> hasTotalManifestation(?h, 0)

#Значение hasTotalManifestation для следующего элемента, если Manifestation не равно -1
Period(?p1), Period(?p2), hasNext(?p1, ?p2), Characteristic(?h1), relatesToPeriod(?h1, ?p1), Characteristic(?h2), Symptom(?sy), relatesToPeriod(?h2, ?p2), relatesToSymptom(?h1, ?sy), relatesToSymptom(?h2, ?sy), Disease(?di), relatesToDisease(?h1, ?di), relatesToDisease(?h2, ?di), hasTotalManifestation(?h1, ?tm), hasManifestation(?h2, ?m), add(?sum, ?tm, ?m), notEqual(?m, -1) -> hasTotalManifestation(?h2, ?sum)
#Значение hasTotalManifestation для следующего элемента, если Manifestation равно -1
```

```
Period(?p1), Period(?p2), hasNext(?p1, ?p2), Characteristic(?h1), relatesToPeriod(?h1, ?p1), Characteristic(?h2), relatesToPeriod(?h2, ?p2), Symptom(?sy), relatesToSymptom(?h1, ?sy), relatesToSymptom(?h2, ?sy), Disease(?di), relatesToDisease(?h1, ?di), relatesToDisease(?h2, ?di), hasTotalManifestation(?h1, ?tm), hasManifestation(?h2, ?m), add(?sum, ?tm, ?m), equal(?m, -1) -> hasTotalManifestation(?h2, ?tm)

#Значение hasTotalValue для следующего элемента, если Manifestation равно -1
Characteristic(?h), hasTotalManifestation(?h, ?tm), hasSeverity(?h, ?s), hasModality(?h, ?m), multiply(?res1, ?s, ?m), multiply(?res2, ?res1, ?tm) -> hasTotalValue(?h, ?res2)

#Устанавливаем значение GeneralPreliminary первого элемента типа Characteristic:
relatesToSymptom(?it, ?sy), hasTotalValue(?it, ?v), Symptom(?sy), hasFirst(pointer, ?sy), Characteristic(?it), Pointer(pointer) -> hasGeneralPreliminaryValue(?it, ?v)

#Вычисляем General-сумму последовательных элементов типа Characteristic
Characteristic(?it2), Symptom(?sy1), Symptom(?sy2), Characteristic(?it1), relatesToPhase(?it1, ?h), relatesToPhase(?it2, ?h), Phase(?h), relatesToSymptom(?it1, ?sy1), relatesToSymptom(?it2, ?sy2), hasGeneralPreliminaryValue(?it1, ?pv), hasNext(?sy1, ?sy2), add(?res, ?pv, ?tv), hasTotalValue(?it2, ?tv) -> hasGeneralPreliminaryValue(?it2, ?res)

#Заносим General-значение в элемент типа Phase
hasGeneralPreliminaryValue(?it, ?tv), relatesToSymptom(?it, ?sy), Symptom(?sy), hasLast(pointer, ?sy), relatesToPhase(?it, ?h), Characteristic(?it), Pointer(pointer), Phase(?h) -> hasGeneralTotalValue(?h, ?tv)

#Устанавливаем значение MainOnlyPreliminary первого элемента типа Characteristic в 0, если его модальность меньше 3:
```

```
relatesToSymptom(?it, ?sy), Symptom(?sy), hasFirst(pointer, ?sy), Characteristic(?it), Pointer(pointer), hasModality(?it, ?m), lessThan(?m, 3) -> hasMainOnlyPreliminaryValue(?it, 0)

#Устанавливаем значение MainOnlyPreliminary первого элемента типа Characteristic, если его модальность больше 3:
relatesToSymptom(?it, ?sy), hasTotalValue(?it, ?v), Symptom(?sy), hasFirst(pointer, ?sy), Characteristic(?it), Pointer (pointer), hasModality(?it, ?m), greaterThan(?m, 3) -> hasMainOnlyPreliminaryValue(?it, ?v)

#Вычисляем MainOnly-сумму последовательных элементов типа Characteristic, если значение модальности 2-го элемента больше 3:
Characteristic(?it2), Symptom(?sy1), Symptom(?sy2), Characteristic(?it1), Phase(?h), relatesToPhase(?it1, ?h), relatesToPhase(?it2, ?h), relatesToSymptom(?it1, ?sy1), relatesToSymptom(?it2, ?sy2), hasMainOnlyPreliminaryValue(?it1, ?pv), hasNext(?sy1, ?sy2), add(?res, ?pv, ?tv), hasTotalValue(?it2, ?tv), hasModality(?it2, ?m), greaterThan(?m, 3) -> hasMainOnlyPreliminaryValue(?it2, ?res)

#Вычисляем MainOnly-сумму последовательных элементов типа Characteristic, если значение модальности 2-го элемента меньше 3:
Characteristic(?it2), Symptom(?sy1), Symptom(?sy2), Characteristic(?it1), Phase(?h), relatesToPhase(?it1, ?h), relatesToPhase(?it2, ?h), relatesToSymptom(?it1, ?sy1), relatesToSymptom(?it2, ?sy2), hasMainOnlyPreliminaryValue(?it1, ?pv), hasNext(?sy1, ?sy2), hasModality(?it2, ?m), lessThan(?m, 3) -> hasMainOnlyPreliminaryValue(?it2, ?pv)

#Заносим MainOnly-значение в элемент типа Phase
hasMainOnlyPreliminaryValue(?it, ?tv), relatesToSymptom(?it, ?sy), Symptom(?sy), hasLast(pointer, ?sy), relatesToPhase(?it, ?h), Characteristic(?it), Pointer (pointer), Phase(?h) -> hasMainOnlyTotalValue(?h, ?tv)
```

Список правил для установления связей, показанных пунктирной линией в правой части рисунка 22.

```
#Связь Diagnosis relatesToPhase Phase
Diagnosis(?dg), Disease(?di), relatesToDisease(?dg, ?di), Phase(?ph), relatesToDisease(?ph, ?di), Period(?pe), relatesToPeriod(?ph, ?pe), Patient(?pa), relatesToPatient(?dg, ?pa), relatesToPeriod(?pa, ?pe) -> relatesToPhase(?dg, ?ph)

#Связь Item relatesToCharacteristic Characteristic
Characteristic(?ha), Phase(?ph), relatesToPhase(?ha, ?ph), Diagnosis(?dg), relatesToPhase(?dg, ?ph), Item(?it), relatesToDiagnosis(?it, ?dg), Symptom(?sy), relatesToSymptom(?it, ?sy), relatesToSymptom(?ha, ?sy) -> relatesToCharacteristic(?it, ?ha)

#Связь Patient relatesToSymptom Symptom
Item(?it), Diagnosis(?dg), relatesToDiagnosis(?it, ?dg), Patient(?pa), relatesToPatient(?dg, ?pa), Symptom(?sy), relatesToSymptom(?it, ?sy), hasThisSymptom(?it, "да") -> relatesToSymptom(?pa, ?sy)

#Значение hasTotalValue, если hasThisSymptom «нет»
Item(?it), hasThisSymptom(?it, "нет") -> hasTotalValue(?it, 0)

#Значение hasTotalValue, если hasThisSymptom «да»
Item(?it), Characteristic(?ha), relatesToCharacteristic(?it, ?ha), hasThisSymptom(?it, "да"), hasTotalValue(?ha, ?tv) -> hasTotalValue(?it, ?tv)

#Устанавливаем значение GeneralPreliminary первого элемента типа Item:
Symptom(?sy), Pointer (pointer), hasFirst(pointer, ?sy), Item(?it), relatesToSymptom(?it, ?sy), hasTotalValue(?it, ?tv) -> hasGeneralPreliminaryValue(?it, ?tv)

#Вычисляем General-сумму последовательных элементов типа Item
Symptom(?sy1), Symptom(?sy2), hasNext(?sy1, ?sy2), Item(?it1), Item(?it2), relatesToSymptom(?it1, ?sy1), relatesToSymptom(?it2,
```

```
?sy2), Diagnosis(?dg), relatesToDiagnosis(?it1, ?dg), relatesToDiagnosis(?it2, ?dg), hasGeneralPreliminaryValue(?it1, ?pv), hasTotalValue(?it2, ?tv), add(?res, ?pv, ?tv) -> hasGeneralPreliminaryValue(?it2, ?res)

#Заносим General-значение в элемент типа Diagnosis
hasGeneralPreliminaryValue(?it, ?tv), relatesToSymptom(?it, ?sy), Symptom(?sy), hasLast(pointer, ?sy), relatesToDiagnosis(?it, ?h), Item(?it), Pointer (pointer), Diagnosis(?h) -> hasGeneralValue(?h, ?tv)

#Устанавливаем значение MainOnlyPreliminary первого элемента типа Item в 0, если его модальность меньше 3:
relatesToSymptom(?it, ?sy), Symptom(?sy), hasFirst(pointer, ?sy), Item(?it), Pointer (pointer), hasModality(?ha, ?m), lessThan(?m, 3), Characteristic(?ha), relatesToCharacteristic(?it, ?ha) -> hasMainOnlyPreliminaryValue(?it, 0)

#Устанавливаем значение MainOnlyPreliminary первого элемента типа Item, если его модальность больше 3:
relatesToSymptom(?it, ?sy), hasTotalValue(?it, ?v), Symptom(?sy), hasFirst(pointer, ?sy), Item(?it), Pointer(pointer), hasModality(?ha, ?m), greaterThan(?m, 3), Characteristic(?ha), relatesToCharacteristic(?it, ?ha) -> hasMainOnlyPreliminaryValue(?it, ?v)

#Вычисляем MainOnly-сумму последовательных элементов типа Item, если значение модальности второго элемента больше 3:
Item(?it2), Symptom(?sy1), Symptom(?sy2), Item(?it1), Diagnosis(?dg), relatesToDiagnosis(?it1, ?dg), relatesToDiagnosis(?it2, ?dg), relatesToSymptom(?it1, ?sy1), relatesToSymptom(?it2, ?sy2), hasMainOnlyPreliminaryValue(?it1, ?pv), hasNext(?sy1, ?sy2), add(?res, ?pv, ?tv), hasTotalValue(?it2, ?tv), Characteristic(?ha), relatesToCharacteristic(?it2, ?ha), hasModality(?ha, ?m), greaterThan(?m, 3) -> hasMainOnlyPreliminaryValue(?it2, ?res)
```

```
#Вычисляем MainOnly-сумму последовательных элементов типа Item, если значение модальности второго элемента меньше 3:
Item(?it2), Symptom(?sy1), Symptom(?sy2), Item(?it1), Diagnosis(?dg), relatesToDiagnosis(?it1, ?dg), relatesToDiagnosis(?it2, ?dg), relatesToSymptom(?it1, ?sy1), relatesToSymptom(?it2, ?sy2), hasMainOnlyPreliminaryValue(?it1, ?pv), hasNext(?sy1,стран ?sy2), Characteristic(?ha), relatesToCharacteristic(?it2, ?ha), hasModality(?ha, ?m), lessThan(?m, 3) -> hasMainOnlyPreliminaryValue(?it2, ?pv)

#Заносим MainOnly-значение в элемент типа Diagnosis
hasMainOnlyPreliminaryValue(?it, ?tv), relatesToSymptom(?it, ?sy), Symptom(?sy), hasLast(pointer, ?sy), relatesToDiagnosis(?it, ?h), Item(?it), Pointer (pointer), Diagnosis(?h) -> hasMainOnlyValue(?h, ?tv)

#Значение hasGeneralRelativeValue инстанса типа Diagnosis:
Diagnosis(?dg), Phase(?ph), relatesToPhase(?dg, ?ph), hasGeneralValue(?dg, ?dtv), hasGeneralTotalValue(?ph, ?ptv), divide(?res, ?dtv, ?ptv), greaterThan(?ptv, 0) -> hasGeneralRelativeValue(?dg, ?res)

#Значение hasMainOnlyRelativeValue инстанса типа Diagnosis:
Diagnosis(?dg), Phase(?ph), relatesToPhase(?dg, ?ph), hasMainOnlyValue(?dg, ?dtv), hasMainOnlyTotalValue(?ph, ?ptv), divide(?res, ?dtv, ?ptv), greaterThan(?ptv, 0) -> hasMainOnlyRelativeValue(?dg, ?res)
```

**Таблица 2 —** Результаты работы машины логического вывода по постановке диагноза

| ***Пациент*** | ***Синдром*** | ***a*** | ***b*** | ***c*** | ***d*** |
|---|---|---|---|---|---|
| *Пример 1.*<br>*Пол: мужской*<br>*Возраст: 2 года 10 мес*<br>*Признаки:*<br>*• грубые черты лица,*<br>*• макроцефалия*<br>*• скафоцефалия*<br>*• деформация суставов кисти*<br>*• тугоподвижность крупных суставов*<br>*• гепатомегалия*<br>*• спленомегалия*<br>*• умственная отсталость*<br>*• кардиопатия*<br>*• кифосколиоз*<br>***Диагноз: Мукополисахаридоз II типа (болезнь Хантера)*** | Гурлер | 125,6 | 47% | 111,0 | 52% |
| | Гурлер-Шейе | 43,7 | 35% | 35,5 | 41% |
| | М-Лами быстро прогр | 6,6 | 15% | 1,6 | 4% |
| | М-Лами медл прогр | 0,0 | 0% | 0,0 | 0% |
| | Морк-А (быстро-прогр) | 1,8 | 10% | 0,8 | 5% |
| | Морк-А (медленно-прог) | 0,0 | 0% | 0,0 | 0% |
| | Морк-В | 0,0 | 0% | 0,0 | 0% |
| | Слая | 30,0 | 61% | 24,6 | 85% |
| | С-Ф D | 0,0 | 0% | 0,0 | 0% |
| | С-Ф А | 6,4 | 50% | 5,0 | 86% |
| | С-Ф В | 3,1 | 36% | 2,5 | 100% |
| | С-Ф С | 0,0 | 0% | 0,0 | 0% |
| | Хантер-Т | 34,2 | 41% | 29,4 | 39% |
| | Хантер-Л | 11,5 | 33% | 8,9 | 28% |
| | Шейе | 19,6 | 44% | 14,8 | 46% |
| *Пример 10*<br>*Пол: мужской*<br>*Возраст: 10 лет*<br>*Признаки:*<br>*• макроцефалия*<br>*• короткая шея*<br>*• грубые черты лица*<br>*• гепатомегалия*<br>*• спленомегалия*<br>*• деформацияя суставов кисти*<br>*• умственная отсталость*<br>***Диагноз: Мукополисахаридоз II типа (болезнь Хантера)*** | Гурлер | 247,7 | 41% | 220,5 | 47% |
| | Гурлер-Шейе | 126,8 | 38% | 98,0 | 44% |
| | М-Лами быстро прогр | 110,9 | 40% | 78,9 | 47% |
| | М-Лами медл прогр | 13,4 | 20% | 6,8 | 16% |
| | Морк-А (быстро-прогр) | 41,0 | 26% | 33,6 | 27% |
| | Морк-А (медленно-прог) | 12,0 | 22% | 10,8 | 23% |
| | Морк-В | 24,6 | 25% | 21,0 | 34% |
| | Слая | 85,0 | 47% | 53,4 | 47% |
| | С-Ф D | 14,2 | 56% | 8,0 | 55% |
| | С-Ф А | 53,0 | 41% | 35,0 | 41% |
| | С-Ф В | 55,3 | 48% | 37,5 | 51% |
| | С-Ф С | 14,2 | 48% | 8,0 | 55% |
| | Хантер-Т | 165,6 | 49% | 140,0 | 51% |
| | Хантер-Л | 53,6 | 53% | 36,0 | 45% |
| | Шейе | 67,2 | 46% | 52,4 | 65% |
| *Пример 11*<br>*Пол: женский*<br>*Возраст: 9 лет*<br>*Признаки:*<br>*• Задержка роста*<br>*• Утолщенная кожа*<br>*• Гипертрихоз (ВП)*<br>*• Скафоцефалия*<br>*• Короткая шея (НП)*<br>*• Кифосколиоз*<br>*• Тугоподвижность крупных суставов (ГП)*<br>*• Килевидная грудная клетка*<br>*• Грыжи*<br>*• Гепатомегалия (ВП)*<br>*• Спленомегалия (НП)*<br>*• Помутнение роговицы*<br>***Диагноз: Мукополисахаридоз VI типа (Марото-Лами)*** | Гурлер | 346,1 | 57% | 277,5 | 60% |
| | Гурлер-Шейе | 223,8 | 67% | 168,0 | 76% |
| | М-Лами быстро прогр | 158,6 | 57% | 101,2 | 61% |
| | М-Лами медл прогр | 43,2 | 66% | 27,6 | 63% |
| | Морк-А (быстро-прогр) | 104,8 | 67% | 91,2 | 72% |
| | Морк-А (медленно-прог) | 37,6 | 69% | 34,0 | 73% |
| | Морк-В | 73,6 | 74% | 56,0 | 90% |
| | Слая | 116,5 | 64% | 73,9 | 65% |
| | С-Ф D | 10,7 | 42% | 2,9 | 20% |
| | С-Ф А | 58,1 | 45% | 24,1 | 28% |
| | С-Ф В | 51,6 | 45% | 19,6 | 27% |
| | С-Ф С | 14,1 | 48% | 2,9 | 20% |
| | Хантер-Т | 140,6 | 41% | 103,0 | 37% |
| | Хантер-Л | 78,1 | 77% | 58,9 | 73% |
| | Шейе | 87,4 | 60% | 58,0 | 72% |

| *Пациент* | *Синдром* | *a* | *b* | *c* | *d* |
|---|---|---|---|---|---|
| *Пример 12*<br>*Пол: мужской*<br>*Возраст: 8 лет*<br>*Признаки:*<br>*• Задержка роста*<br>*• Грубые черты лица*<br>*• Короткая шея (НП)*<br>*• Кифосколиоз (НП)*<br>*• Тугоподвижность крупных суставов (ГП)*<br>*• Гепатомегалия (ВП)*<br>*• Спленомегалия (НП)*<br>*• Умственная отсталость(ВП)*<br>***Диагноз: Мукополисахаридоз VI типа (Марото-Лами)*** | Гурлер | 316,5 | 52% | 300,5 | 65% |
| | Гурлер-Шейе | 187,2 | 56% | 164,0 | 74% |
| | М-Лами быстро прогр | 147,3 | 53% | 119,3 | 71% |
| | М-Лами медл прогр | 30,0 | 46% | 21,6 | 50% |
| | Морк-А (быстро-прогр) | 70,2 | 45% | 58,4 | 46% |
| | Морк-А (медленно-прог) | 24,4 | 45% | 20,8 | 44% |
| | Морк-В | 54,0 | 54% | 48,0 | 77% |
| | Слая | 119,4 | 66% | 86,4 | 77% |
| | С-Ф D | 17,1 | 68% | 10,9 | 75% |
| | С-Ф А | 77,1 | 60% | 59,1 | 70% |
| | С-Ф В | 76,7 | 67% | 57,1 | 78% |
| | С-Ф С | 17,1 | 58% | 10,9 | 75% |
| | Хантер-Т | 169,0 | 50% | 141,0 | 51% |
| | Хантер-Л | 67,8 | 67% | 51,0 | 63% |
| | Шейе | 75,4 | 52% | 52,4 | 65% |
| *Пример 13*<br>*Пол: мужской*<br>*Возраст: 1 год 1 месяц*<br>*Признаки:*<br>*• Макроцефалия (ВП)*<br>*• Грубые черты лица (ГП)*<br>*• Помутнение роговицы*<br>*• Короткая шея(НП)*<br>*• Задержка роста*<br>*• Кифосколиоз(НП)*<br>*• Гиперлордоз(ВП)*<br>*• Воронкообразная грудная клетка*<br>*• Тугоподвижность крупных суставов (ГП)*<br>*• Грыжи(НП)*<br>*• Кардиопатия(ВП)*<br>***Диагноз: Мукополисахаридоз VI типа (Марото-Лами)*** | Гурлер | 124,4 | 46% | 106,8 | 50% |
| | Гурлер-Шейе | 66,9 | 53% | 47,5 | 55% |
| | М-Лами быстро прогр | 37,4 | 87% | 33,6 | 93% |
| | М-Лами медл прогр | 3,2 | 100% | 3,2 | 100% |
| | Морк-А (быстро-прогр) | 13,5 | 77% | 12,9 | 84% |
| | Морк-А (медленно-прог) | 2,2 | 92% | 2,2 | 100% |
| | Морк-В | 15,6 | 90% | 15,6 | 100% |
| | Слая | 33,2 | 68% | 21,2 | 73% |
| | С-Ф D | 0,0 | 0% | 0,0 | 0% |
| | С-Ф А | 6,4 | 50% | 2,0 | 34% |
| | С-Ф В | 4,9 | 56% | 0,5 | 20% |
| | С-Ф С | 1,2 | 100% | 0,0 | 0% |
| | Хантер-Т | 61,8 | 74% | 60,2 | 81% |
| | Хантер-Л | 30,7 | 87% | 30,5 | 95% |
| | Шейе | 32,9 | 74% | 23,9 | 74% |
| *Пример 14*<br>*Пол: женский*<br>*Возраст: 6 месяцев*<br>*Признаки:*<br>*• Грубые черты лица*<br>*• Помутнение роговицы(НП)*<br>*• Гипертрихоз(ВП)*<br>*• Кардиопатия (ВП)*<br>*• Гепатомегалия(ВП)*<br>*• Спленомегалия(НП)*<br>***Диагноз: Мукополисахаридоз VII типа (синдром Слая)*** | Гурлер | 45,2 | 47% | 41,2 | 50% |
| | Гурлер-Шейе | 4,0 | 20% | 4,0 | 45% |
| | М-Лами быстро прогр | 0,0 | 0% | 0,0 | 0% |
| | М-Лами медл прогр | 0,0 | 0% | 0,0 | 0% |
| | Морк-А (быстро-прогр) | 0,0 | 0% | 0,0 | 0% |
| | Морк-А (медленно-прог) | 0,0 | 0% | 0,0 | 0% |
| | Морк-В | 0,0 | 0% | 0,0 | 0% |
| | Слая | 3,0 | 32% | 3,0 | 86% |
| | С-Ф D | 0,0 | 0% | 0,0 | 0% |
| | С-Ф А | 0,4 | 25% | 0,0 | 0% |
| | С-Ф В | 0,4 | 25% | 0,0 | 0% |
| | С-Ф С | 0,0 | 0% | 0,0 | 0% |
| | Хантер-Т | 2,2 | 16% | 0,0 | 0% |
| | Хантер-Л | 0,0 | 0% | 0,0 | 0% |
| | Шейе | 0,0 | 0% | 0,0 | 0% |

| *Пациент* | *Синдром* | *a* | *b* | *c* | *d* |
|---|---|---|---|---|---|
| *Пример 15*<br>*Пол: женский*<br>*Возраст: 5 лет*<br>*Признаки:*<br>*• Грубые черты лица (ГП)*<br>*• Кардиопатия (ВП)*<br>*• Гепатомегалия (ВП)*<br>*• Спленомегалия (НП)*<br>*• Умственная отсталость (ГП)*<br>*• Снижение слуха (ВП)*<br>*• Кифосколиоз (ГП)*<br>***Диагноз: Мукополисахаридоз VII типа (синдром Слая)*** | Гурлер | 196,6 | 44% | 185,4 | 53% |
| | Гурлер-Шейе | 112,7 | 45% | 100,7 | 60% |
| | М-Лами быстро прогр | 37,9 | 31% | 30,3 | 36% |
| | М-Лами медл прогр | 10,0 | 45% | 7,6 | 49% |
| | Морк-А (быстро-прогр) | 25,4 | 36% | 20,4 | 36% |
| | Морк-А (медленно-прог) | 5,2 | 28% | 4,8 | 32% |
| | Морк-В | 12,8 | 23% | 8,8 | 23% |
| | Слая | 47,9 | 47% | 43,5 | 70% |
| | С-Ф D | 3,6 | 95% | 2,4 | 100% |
| | С-Ф А | 29,1 | 58% | 24,9 | 77% |
| | С-Ф В | 25,1 | 57% | 21,3 | 77% |
| | С-Ф С | 4,2 | 68% | 2,4 | 100% |
| | Хантер-Т | 54,9 | 31% | 47,1 | 32% |
| | Хантер-Л | 18,1 | 30% | 11,1 | 22% |
| | Шейе | 51,8 | 51% | 35,6 | 63% |
| *Пример 16*<br>*Пол: женский*<br>*Возраст: 6 лет*<br>*Признаки:*<br>*• Умственная отсталость*<br>*• Гипертрихоз (ВП)*<br>*• Грубые черты лица*<br>*• Макроцефалия (ВП)*<br>*• Задержка роста*<br>*• Воронкообразная грудная клетка (ВП)*<br>*• Кифосколиоз (ГП)*<br>***Диагноз: Мукополисахаридоз VII типа (синдром Слая)*** | Гурлер | 134,5 | 30% | 92,7 | 27% |
| | Гурлер-Шейе | 64,8 | 26% | 43,0 | 26% |
| | М-Лами быстро прогр | 32,1 | 27% | 19,5 | 23% |
| | М-Лами медл прогр | 2,4 | 11% | 1,6 | 10% |
| | Морк-А (быстро-прогр) | 16,9 | 24% | 15,3 | 27% |
| | Морк-А (медленно-прог) | 3,6 | 19% | 3,4 | 23% |
| | Морк-В | 20,6 | 37% | 18,8 | 48% |
| | Слая | 48,9 | 48% | 35,5 | 57% |
| | С-Ф D | 0,0 | 0% | 0,0 | 0% |
| | С-Ф А | 21,9 | 44% | 19,5 | 60% |
| | С-Ф В | 19,3 | 44% | 16,5 | 60% |
| | С-Ф С | 0,0 | 0% | 0,0 | 0% |
| | Хантер-Т | 94,9 | 54% | 88,3 | 60% |
| | Хантер-Л | 32,4 | 53% | 30,8 | 60% |
| | Шейе | 28,8 | 29% | 14,0 | 25% |
| *Пример 2*<br>*Пол: женский*<br>*Возраст: 5 лет*<br>*Признаки:*<br>*• грубые черты лица*<br>*• утолщенная кожа*<br>*• тугоподвижность крупных суставов*<br>*• помутнение роговицы*<br>*• гепатомегалия*<br>*• спленомегалия*<br>*• кардиопатия*<br>***Диагноз: Мукополисахаридоз I типа (синдром Гурлер)*** | Гурлер | 186,7 | 42% | 178,3 | 51% |
| | Гурлер-Шейе | 119,7 | 48% | 113,5 | 68% |
| | М-Лами быстро прогр | 32,1 | 27% | 18,3 | 22% |
| | М-Лами медл прогр | 6,0 | 27% | 2,0 | 13% |
| | Морк-А (быстро-прогр) | 21,8 | 31% | 12,0 | 21% |
| | Морк-А (медленно-прог) | 4,8 | 26% | 3,2 | 21% |
| | Морк-В | 7,6 | 14% | 2,4 | 6% |
| | Слая | 44,1 | 43% | 39,9 | 64% |
| | С-Ф D | 1,2 | 32% | 0,0 | 0% |
| | С-Ф А | 15,1 | 30% | 10,9 | 34% |
| | С-Ф В | 14,7 | 33% | 10,9 | 39% |
| | С-Ф С | 1,8 | 29% | 0,0 | 0% |
| | Хантер-Т | 39,1 | 22% | 29,1 | 20% |
| | Хантер-Л | 18,4 | 30% | 10,6 | 21% |
| | Шейе | 64,8 | 64% | 56,6 | 100% |

| *Пациент* | *Синдром* | *a* | *b* | *c* | *d* |
|---|---|---|---|---|---|
| *Пример 3*<br>*Пол: женский*<br>*Возраст: 3 месяца*<br>*Признаки:*<br>*• кардиопатия*<br>*• тугоподвижность крупных суставов*<br>*• гепатомегалия*<br>*• спленомегалия*<br>*• грубые черты лица*<br>*• кифосколиоз*<br>***Диагноз: Мукополисахаридоз I типа (синдром Гурлер)*** | Гурлер | 38,0 | 40% | 38,0 | 46% |
| | Гурлер-Шейе | 2,5 | 13% | 2,5 | 28% |
| | М-Лами быстро прогр | 0,0 | 0% | 0,0 | 0% |
| | М-Лами медл прогр | 0,0 | 0% | 0,0 | 0% |
| | Морк-А (быстро-прогр) | 0,0 | 0% | 0,0 | 0% |
| | Морк-А (медленно-прог) | 0,0 | 0% | 0,0 | 0% |
| | Морк-В | 0,0 | 0% | 0,0 | 0% |
| | Слая | 3,0 | 32% | 3,0 | 86% |
| | С-Ф D | 0,0 | 0% | 0,0 | 0% |
| | С-Ф А | 0,0 | 0% | 0,0 | 0% |
| | С-Ф В | 0,0 | 0% | 0,0 | 0% |
| | С-Ф С | 0,0 | 0% | 0,0 | 0% |
| | Хантер-Т | 1,8 | 13% | 0,0 | 0% |
| | Хантер-Л | 0,0 | 0% | 0,0 | 0% |
| | Шейе | 0,0 | 0% | 0,0 | 0% |
| *Пример 4*<br>*Пол: женский*<br>*Возраст: 2 года*<br>*Признаки:*<br>*• гепатомегалия*<br>*• умственная отсталость*<br>*• грубые черты лица*<br>*• деформация суставов кисти*<br>***Диагноз: Мукополисахаридоз III типа С (синдром Санфилиппо С)*** | Гурлер | 85,2 | 32% | 85,0 | 40% |
| | Гурлер-Шейе | 27,5 | 22% | 27,5 | 32% |
| | М-Лами быстро прогр | 2,4 | 6% | 0,0 | 0% |
| | М-Лами медл прогр | 0,0 | 0% | 0,0 | 0% |
| | Морк-А (быстро-прогр) | 1,2 | 7% | 0,8 | 5% |
| | Морк-А (медленно-прог) | 0,0 | 0% | 0,0 | 0% |
| | Морк-В | 0,0 | 0% | 0,0 | 0% |
| | Слая | 14,4 | 29% | 12,0 | 41% |
| | С-Ф D | 0,0 | 0% | 0,0 | 0% |
| | С-Ф А | 6,2 | 48% | 5,0 | 86% |
| | С-Ф В | 3,1 | 36% | 2,5 | 100% |
| | С-Ф С | 0,0 | 0% | 0,0 | 0% |
| | Хантер-Т | 13,7 | 16% | 10,5 | 14% |
| | Хантер-Л | 4,4 | 13% | 2,0 | 6% |
| | Шейе | 13,6 | 31% | 13,6 | 42% |
| *Пример 5*<br>*Пол: мужской*<br>*Возраст: 4 года*<br>*Признаки:*<br>*• грубые черты лица*<br>*• умственная отсталость*<br>*• килевидная грудная клетка*<br>***Диагноз: Мукополисахаридоз III типа В (синдром Санфилиппо В)*** | Гурлер | 77,3 | 17% | 67,5 | 19% |
| | Гурлер-Шейе | 26,0 | 10% | 15,0 | 9% |
| | М-Лами быстро прогр | 10,9 | 9% | 7,5 | 9% |
| | М-Лами медл прогр | 1,8 | 8% | 0,0 | 0% |
| | Морк-А (быстро-прогр) | 8,2 | 12% | 7,4 | 13% |
| | Морк-А (медленно-прог) | 2,3 | 12% | 2,3 | 15% |
| | Морк-В | 6,4 | 12% | 2,4 | 6% |
| | Слая | 31,5 | 31% | 25,5 | 41% |
| | С-Ф D | 0,0 | 0% | 0,0 | 0% |
| | С-Ф А | 18,5 | 37% | 18,5 | 57% |
| | С-Ф В | 16,5 | 38% | 16,5 | 60% |
| | С-Ф С | 0,0 | 0% | 0,0 | 0% |
| | Хантер-Т | 27,5 | 16% | 27,5 | 19% |
| | Хантер-Л | 3,0 | 5% | 3,0 | 6% |
| | Шейе | 16,4 | 16% | 14,0 | 25% |

| *Пациент* | *Синдром* | *a* | *b* | *c* | *d* |
|---|---|---|---|---|---|
| *Пример 7*<br>*Пол:*<br>*Возраст: 7 лет*<br>*Признаки:*<br>*• тугоподвижность крупных суставов*<br>*• помутнение роговицы*<br>*• грубые черты лица*<br>*• деформация суставов кисти*<br>*• задержка роста*<br>*• кардиопатия*<br>***Диагноз: Мукополисахаридоз VI типа (Марото-Лами)*** | Гурлер | 185,2 | 30% | 174,0 | 37% |
| | Гурлер-Шейе | 122,8 | 37% | 112,0 | 51% |
| | М-Лами быстро прогр | 98,1 | 35% | 58,5 | 35% |
| | М-Лами медл прогр | 22,8 | 35% | 12,0 | 28% |
| | Морк-А (быстро-прогр) | 51,8 | 33% | 40,8 | 32% |
| | Морк-А (медленно-прог) | 21,8 | 40% | 18,8 | 40% |
| | Морк-В | 32,2 | 32% | 21,0 | 34% |
| | Слая | 63,8 | 35% | 50,0 | 44% |
| | С-Ф D | 4,7 | 19% | 2,9 | 20% |
| | С-Ф А | 37,3 | 29% | 34,1 | 40% |
| | С-Ф В | 39,6 | 34% | 34,6 | 47% |
| | С-Ф С | 5,3 | 18% | 2,9 | 20% |
| | Хантер-Т | 108,6 | 32% | 79,8 | 29% |
| | Хантер-Л | 31,2 | 31% | 28,8 | 36% |
| | Шейе | 76,4 | 53% | 50,4 | 63% |
| *Пример 8*<br>*Пол:*<br>*Возраст: 20 лет*<br>*Признаки:*<br>*• деформация суставов кисти*<br>*• грыжи*<br>*• грубые черты лица*<br>*• помутнение роговицы*<br>*• тугоподвижность крупных суставов*<br>*• кардиопатия*<br>***Диагноз: Мукополисахаридоз VI типа (Марото-Лами)*** | Гурлер | 185,2 | 30% | 174,0 | 37% |
| | Гурлер-Шейе | 104,8 | 31% | 76,0 | 34% |
| | М-Лами быстро прогр | 104,5 | 38% | 80,9 | 48% |
| | М-Лами медл прогр | 27,2 | 42% | 20,0 | 46% |
| | Морк-А (быстро-прогр) | 51,0 | 33% | 40,0 | 32% |
| | Морк-А (медленно-прог) | 21,8 | 40% | 18,8 | 40% |
| | Морк-В | 31,2 | 31% | 20,0 | 32% |
| | Слая | 66,5 | 37% | 62,5 | 55% |
| | С-Ф D | 5,8 | 23% | 2,4 | 17% |
| | С-Ф А | 44,8 | 35% | 29,6 | 35% |
| | С-Ф В | 46,2 | 40% | 34,6 | 47% |
| | С-Ф С | 9,8 | 33% | 2,4 | 17% |
| | Хантер-Т | 100,6 | 30% | 71,8 | 26% |
| | Хантер-Л | 16,6 | 16% | 14,2 | 18% |
| | Шейе | 75,8 | 52% | 50,4 | 63% |
| *Пример 9*<br>*Пол: мужской*<br>*Возраст: 2 года*<br>*Признаки:*<br>*• грыжи*<br>*• умственная отсталость*<br>*• грубые черты лица*<br>*• тугоподвижность крупных суставов*<br>*• деформация суставов кисти*<br>*• гепатомегалия*<br>*• спленомегалия*<br>***Диагноз: Мукополисахаридоз II типа (болезнь Хантера)*** | Гурлер | 138,2 | 52% | 138,0 | 65% |
| | Гурлер-Шейе | 46,7 | 37% | 35,5 | 41% |
| | М-Лами быстро прогр | 8,8 | 20% | 6,4 | 18% |
| | М-Лами медл прогр | 1,6 | 50% | 1,6 | 50% |
| | Морк-А (быстро-прогр) | 4,4 | 25% | 4,0 | 26% |
| | Морк-А (медленно-прог) | 1,2 | 50% | 1,2 | 55% |
| | Морк-В | 3,6 | 21% | 3,6 | 23% |
| | Слая | 26,0 | 53% | 23,6 | 81% |
| | С-Ф D | 0,0 | 0% | 0,0 | 0% |
| | С-Ф А | 9,6 | 75% | 5,0 | 86% |
| | С-Ф В | 6,3 | 72% | 2,5 | 100% |
| | С-Ф С | 1,2 | 100% | 0,0 | 0% |
| | Хантер-Т | 22,4 | 27% | 18,4 | 25% |
| | Хантер-Л | 6,7 | 19% | 4,1 | 13% |
| | Шейе | 18,2 | 41% | 14,8 | 46% |

SAINT-PETERSBURG STATE UNIVERSITY

*A manuscript*

**LESHCHEVA Irina Anatolevna**

# Method of automated population of ontology knowledge bases

Specialization: 2.3.5. Mathematical and software support for computing systems, complexes and computer networks

*Dissertation is submitted for the degree of a candidate of technical science*

Translation from Russian

Science advisor:
Doctor of science (Technical science)
Gavrilova Tatiana Albertovna

Saint-Petersburg
2022

# TABLE OF CONTENTS

# INTRODUCTION

**Research motivation.** Effective management decision-making depends on the completeness and consistency of the information available, as well as on the flexibility of its processing, including at the semantic level. The application of corporate ontologies and ontological knowledge bases in enterprises and organizations has the potential to significantly improve the quality of information support and management, ensures the preservation of the company's intellectual assets and the support of the most important business processes.

Corporate ontologies can describe not only business processes, but also products, documents, competencies, technologies, functions, strategies, financial flows, etc. It is an interoperability-oriented tool, understood as the possibility of using different levels of hierarchy, in different units, on different technical and software platforms with different skills. The sum of corporate ontologies serves as a universal framework for the organization's knowledge, as well as a bridge for understanding and transferring knowledge and technology.

But at the moment there is little correlation between the needs of enterprises and organizations and existing technologies in the field of knowledge engineering and ontological engineering. Corporate knowledge practices and tools are not yet mature enough to address the practical challenges of knowledge management and information management. As a result, enterprises and organizations are using obsolete and inefficient technologies. The growing interest in knowledge engineering is hindered by the difficulty of developing practical-oriented ontologies and integrating them with the accumulated datasets of different structures.

Contributed significantly to the development and research of models, methods and tools of intelligent systems (including ontologies, knowledge bases and software tools for their development): Gruber T., Guarino N., Lenat D., Osipov G.S., Grilov D.A., Gavrilova T.A., Golenkov V.V., Gribova V.V., Zagorolko Y.A., Kleshchev A.S., Kudrivtsev D.V., Massel L.V., Tuzovsky A. F., Dobrovsky V.F., Studer R. and others. [Gavrilova, Kudryavtsev, Muromtsev, 2020; Gribova V. V. V. et al. 2018; Golenkov et al. 2019; Gribova V. V. V. etc. 2018; Gruber, 1995; Giaretta, Guarino, 1995; Zagorulko, Borovikova, Zagorulko, 2021; Kleshchev, Moskalenko, Chernyakhovskaya, 2005; Gavrilova, Dobrovsky, 2000; Lenat, Guha, 1991; Yezhenin, Massel, 2020; Osipov, 2011; Tuzovsky, 2007; Studer R. et al., 2004].

One part of ontology development is the population of ontology with individuals and relationships, which in one way or another affects all stages of the life cycle of the knowledge base, but most influences the stages of development and support in the process of use. At the same time, the population of ontologies remains one of the unsolved tasks of ontological engineering. Existing methods for populating ontology-based knowledge bases have a number of

disadvantages, such as the narrow focus on project-specific tasks, the complexity of describing models of data sources and their transformation, High user qualification requirements, possibility of integration of data from only one type of sources, lack of modularity and extensibility, as well as convenient tools. This determines the relevance of creating new methods of automated laboriousness and enrichment of ontologies or ontological-type knowledge bases by consolidation, which will reduce the laboriousness of population and enriching ontologies, using the accumulated data sets regardless of their form of storage and presentation.

**The objective of the thesis** is to develop an automated method and algorithms for integrating heterogeneous sources of structured data by consolidating to populate ontological knowledgebases.

In order to achieve this objective, the thesis work has set and achieved the following **objectives:**

- Construct genealogy of languages of representation of knowledge on the basis of ontologies.
- Study the cognitive aspects of knowledge structuring using the ontological model of knowledge representation.
- Develop a universal method of automated integration of heterogeneous sources of structured data by consolidation to populate ontological-type knowledge bases.
- Develop display rules used by the method only by ontologies without attracting additional languages with its syntax.
- Develop templates for constructing rules of representation of the initial ontology into the knowledge base of the subject area.
- Formulate criteria for selecting a programme platform to implement the proposed method.
- Develop an architecture and implement a procedure to consolidate data stored in structured data sources into a subject domain knowledge base, test on real knowledge bases.

These tasks are **protected provisions**.

**The research focuses** on ontological-type knowledge bases.

**The research focuses** on models, methods and algorithms for population and enriching ontology-based knowledge bases.

**Research Methods**. Methods of systems analysis and knowledge engineering, model transformation, and methods and tools of artificial intelligence and ontological modelling were used.

**Scientific novelty**.

1. The genealogy of knowledge presentation languages proposed earlier in [Kazekin, 2008] has been expanded and further developed.
2. In a study of the influence of cognitive styles on the creation of ontological models, quantitative methods were applied for the first time, showing the statistical significance of the relationships found between the cognitive type of a person and the characteristics of the ontology he constructed [Gavrilova, Leshcheva, Ontology…, 2015; Gavrilova and Leshcheva, 2016; Gavrilova et al., 2013; Kudryavtsev et al., 2019; Gavrilova et al, 2013; Gavrilova, Leshcheva, Building…, 2015; Gavrilova, Leshcheva, The interplay…, 2015; Leshcheva, Gavrilova, 2015].
3. A new universal ontology population algorithm is proposed, lacking most of the disadvantages of existing distributed nature data integration solutions in ontology (see above) due to the proposed algorithm. In particular, the proposed solution is scalable to provide a potential opportunity to work with new data formats in the future.
4. For the first time, the process of transforming data structures into an ontological model was implemented without the use of additional languages with their syntax, but only with ontological modelling tools, thus eliminating the need for additional instrumental support of the method.

The scientific and practical importance of the thesis lies in the development of a new method of information processing analysis, namely a method for consolidating data into ontology for further processing and use in management decisions

**Work validation.** The main results of the thesis work, its individual provisions, as well as the results of specific applied research and development were presented at more than 20 international, Russian, regional scientific and practical conferences. (See the list at CONCLUSION paragraph).

**Publications and personal contributions of the author**. The results of the dissertation were published in 36 printed works (including 5 articles recommended by the Higher Degree Commission for publication of the results of dissertations, 8 articles in journals indexed in Web of Science and Scopus, 3 articles in specialized Russian journals and 20 publications in the works of international and All-Russian conferences).

**Structure and scope of the thesis.** The thesis consists of an introduction, three chapters, a conclusion, a list of literature containing 127 titles, and one annex. The length is 90 pages of the main text, including 29 figures, 1 table.

**Work content**. The introduction justifies the relevance of the thesis work, formulates the object and purpose of the research, determines the scientific novelty and the practical significance of the results.

The first chapter provides an analysis of modern approaches to creating and population ontologies and ontological type knowledge base (KB).

The second chapter describes the METEOR method (Methodology and Technology of Ontology Population based on Integration with Heterogeneous Structured Data Sources) and algorithms of integration of heterogeneous sources of structured data by consolidation for population of databases of ontological type.

The novelty and main idea of the proposed METEOR method is the use of an auxiliary Data Sources Ontology (DSO), which can be extended by adding additional types of data sources. The use of the DSO is complemented by the development of requirements for the creation of display rules that allow linking the DSO and a specific domain ontology (DO) to be populated with copies. The proposed method does not have most of the drawbacks of existing ontology integration solutions.

The application of the developed method can be described by an algorithm consisting of 5 steps, two of which are executed automatically:

1. Identification of data sources.
2. Data source specification by auxiliary DSO.
3. Extraction of data structure from data sources in the DSO.
4. Set Mapping Rules in DO.
5. Population of DO.

The key step of the DO population algorithm is to set the mapping rules. It is done using the annotation mechanism built into the OWL. New annotation properties are introduced: 4 main and 2 auxiliary properties. This approach eliminated the need for additional instrumentation support for the METEOR method, as annotation can be done with any ontology editor.

A new prototype architecture was also developed to implement the proposed METEOR method. The architecture is based on the well-known architecture of ETL-systems. A peculiarity of the proposed architecture is that the receiver of the data is not a repository or database, but ontology or a knowledge base.

The process of population ontology with the METEOR method is illustrated in the Conditional Example Thesis.

The third chapter discusses the application of the developed method by creating and population ontologies from 4 subject areas:

a) Ontology for the integration of EMPIRION empirical research data;

b) Ontology of Ontomorf orphan diseases;

c) Ontology of assembly units for automated car production;

d) Ontology for the administration of the academic activity of the university.

The main scientific results of the thesis are formulated in the conclusion.

The results of the research were obtained in the framework of the RFBR project studies:

- RFBR № 17-07-00228 «Metadology and Technology of formation of ontologies based on integration with heterogeneous data sources (METEOR)»;
- RFBR № 20-07-00854 «Formation of knowledge bases based on empirical research data: ontological approach (EMPIRION)».

Some preliminary results were obtained during the execution of RFBR projects 11-07-00140 2011-2013 «Structuring of knowledge and Content by group design of ontology (COMET)» and 14-07-00294 «Intelligent Support Services for PORTALs of Knowledge Based on Ontology (InC-PORT)».

# CHAPTER 1 CURRENT STATUS OF ONTOLOGICAL ENGINEERING RESEARCH

## 1.1 Challenges in developing ontological-type knowledge bases

### 1.1.1 On the definition of ontology

The digital economy dictates new information laws, and knowledge bases (KB) from rare applications in intellectual systems are becoming the de facto standard for corporate information systems and platforms [Tuzovskiy, 2007]. Despite known design methodologies for the development of KB [Stadnicki, Pietroń, Burek, 2020; Yunianta et al., 2019], there are problems in their practical formation. The problem is particularly acute for ontological knowledge bases, which have been the most popular model of knowledge representation for more than 20 years.

The term «ontology» comes from philosophy, where it means the teaching of being as such; a section of philosophy that studies the fundamental principles of being, the most general entities and the category of being [Dobrochotov, 2010].

At the end of the 20th century the term «ontology» became used in artificial intelligence, in particular in the engineering of knowledge [Studer et al., 2004]. Ontologies were later seen as the basis for building the Semantic Network, a new stage in the development of the WWW network (Word Wide Web). [Berners-Lee, Hendler, Lassila, 2001].

There are many approaches to the definition of «ontology». One of the best-known definitions of ontology was given by Tom Gruber: «Ontology is a specification of conceptualization» [Gruber, 1993]. Nicola Guarino defines ontology as follows: «Ontology is a formal theory limiting the possible conceptualization of the world» [Giaretta, Guarino, 1995]. Both use the concept of «conceptualization», which in turn requires definitions. By «conceptualization» means a strict description of the system of concepts, objects and other entities and relations linking them to each other. [Genesereth, Nilsson, 1987] Conceptualization can be said to be a simplified model of the world created for certain purposes using a systems approach.

A more detailed and practical definition could be given: Ontology is a specification of a domain, or a formal representation thereof, that includes a dictionary of pointers to domain terms and logical expressions that describe what these terms mean, how they relate to each other, and how they may or may not be connected." [Gavrilova, Muromtsev, 2007].

In the work of [Gavrilova, Horoshevsky, 2000] the formal model of ontology means an ordered triple of a species:

$$O=\{X, \Re, \Phi\},$$

where $X$ — a finite set of concepts (concepts, terms) of the subject area that ontology O represents;

$\Re$ — a finite set of relationships between the concepts of a given domain;

$\Phi$ — a finite set of interpretation functions (axiomatization) given on the concepts and/or relation of ontology O.

The work of [Davidovsky, 2011] clarifies this definition as follows:

Ontology is a tuple

$$O = \{C, R, I, T, V, \leq, \perp, \in, =\},$$

where

$C$ — set of concepts (or classes);

$R$ — set of relationships (object properties and data type properties);

$I$ — set of individuals (or instances);

$T$ — set of data types;

$V$ — set of values;

$\leq$ — A reflexive, asymmetric transitive relation to $(C \times C) \cup (R \times R) \cup (T \times T)$, called specialization, which forms partial orders on C and R, called respectively the concept hierarchy and the relationship hierarchy;

$\perp$ — The reflexive and symmetric relation on $(C \times C) \cup (R \times R) \cup (T \times T)$, called the exception;

$\in$ —A relation over $(I \times C) \cup (V \times R)$, called an instrumentation (specification);

$=$ — a relation over $I \times P \times (I \cup V)$, called an attribution;

(sets C, R, I, T, V are pairwise disjoint)».

In general, the ontology structure is a set of elements: concepts, relationships, axioms, individual instances.

Concepts are the conceptualization of the class of all representatives of an entity or phenomenon. Thus, each class describes the totality of individual entities that are combined on the basis of common properties.

Relationships between concepts can be defined that serve both to connect classes and to describe them. In particular, concepts may be related by a functional relation, usually referred to as a verb. For example, when describing an implementation in programming, the ratios «implements» (the class implements the interface) and «expands» or «inherits» can be used (one class expands the other).

The most common type of relationship used in all ontologies is taxonomic relation, i.e., categorization. This type of relationship has other names as well: the categorization relation, the rhodoptic relation, the «class-subclass» relation, Is-a, AKO (a kind of) [Gavrilova, Leshcheva, Rumyantseva, 2011].

Another type of relationship often used is «part-integer» (partonic relation). The «part whole» relation is one of the basic primitives of the structure of the Universe, the definition of this relationship between objects is required in many applications, e.g. in systems for the diagnosis of faults, geographical applications, etc. «part of the whole» is a large independent field of science - "mereology" and "mereotopy".

Axiom sets the conditions that bind concepts and relationships. [http:/www.sci-innov.ru/icatalog_new/entry_68352.htm] They allow the expression of information that cannot be reflected in ontology only by building a hierarchy of concepts and setting different relationships between concepts. Axioms allow further conclusions to be made within ontology. Axioms can also be constraints imposed on any relationship that makes logical inference possible. If ontology is given an axiom, saying that «Teacher» is «Man», who teaches at least one «Discipline», and then it is stated that «Ivanov Ivan Ivanovich» (representative of the class «Man») teaches «probability theory» (where «probability theory» is included in the class «Discipline»), then it will be deduced that «Ivanov Ivanovich» is «Teacher».

Along with the mentioned elements of ontology it also includes so-called «individuals». Individuals are individual representatives of a class of entities or phenomena, that is, specific elements of any category («Ivanov Ivan Ivanovich» in the previous paragraph is an example of an instance of class «Man»). The ontology to which the specimens were added will be called the ontological basis of knowledge.

Basic principles that ontologies should satisfy [Gruber, 1995]:

1. Clarity (Clarity).
2. Consistency (Coherence)
3. Extendibility (Extendibility).
4. Minimal encoding bias (Minimal encoding bias).
5. Minimal ontological commitment (Minimal ontological commitment).

Knowledge structure modelling is the cornerstone of ontological engineering, but for practical use it is important to populate ontology with real data, or individuals (instances of information objects), and the relationships between them.

At the same time, existing methods for population ontology-based knowledge bases have a number of disadvantages:

- Narrow focus on specific project objectives,
- The complexity of the description of data source models and their transformation, the high demand for user skills,
- The ability to integrate data from only one type of source, the lack of modularity and extensibility, and the lack of convenient tools.

This determines the relevance of this study to create new methods of automated population and enrichment of ontologies. The emphasis is on the so-called consolidation of data and knowledge, which makes it possible to reduce the burden of populating and enriching ontologies by using the accumulated data, regardless of the form in which they are stored and presented.

### 1.1.2 Ontology classifications

There are many classifications of ontologies [Ruiz, Hilera, 2006; Berdier, Roussey, 2007; Bullinger, 2009; Gavrilova, Horoshevsky, 2000]. Figure 1 presents commonly accepted criteria and classifications.

The following are distinguished by the degree of formality:

- Informal ontologies suggest a description in any natural language. This category includes: lists of terms (catalogues), glossaries (list of terms with their meanings in natural language), thesaurus (provide additional information on semantic relationships between terms, for example, on synonymy).
- Loosely formalized, it assumes a structure (usually taxonomy), but not a formal logic. This includes structured glossaries, folksonomies (informal taxonomies), formal taxonomies, XML DTDs, database schemes, etc.
- Highly formalized - implies a description in the language of formal logic (first-order predicate logic, descriptive logic, etc.).

The range of different types of ontology by degree of formality is shown in figure 2.

The level of expressiveness of the descriptive means (depth of description of the subject area) of ontology is divided into [Gómez-Pérez, Fernandez-Lopez, Corcho, 2004]:

- "Weighty" ontologies (Heavyweighted) containing axioms and
- "Lightweighted" (Lightweighted) without them.

Depending on the task or subject area:

- Ontology of representation (meta-ontology) — is the conceptualization of formalism of knowledge representation.
- Upper level ontologies — describe general knowledge of the world independent of the subject area.

- Subject Domain Ontologies — Developed jointly with subject area experts to establish common terminology and share information in their field.
- Problem-oriented ontologies are designed to be used by a specific application designed to solve certain problems.
- Applied ontologies — They have the properties of two previous types of ontologies: they contain knowledge of the subject domain and are designed to solve a range of applications.

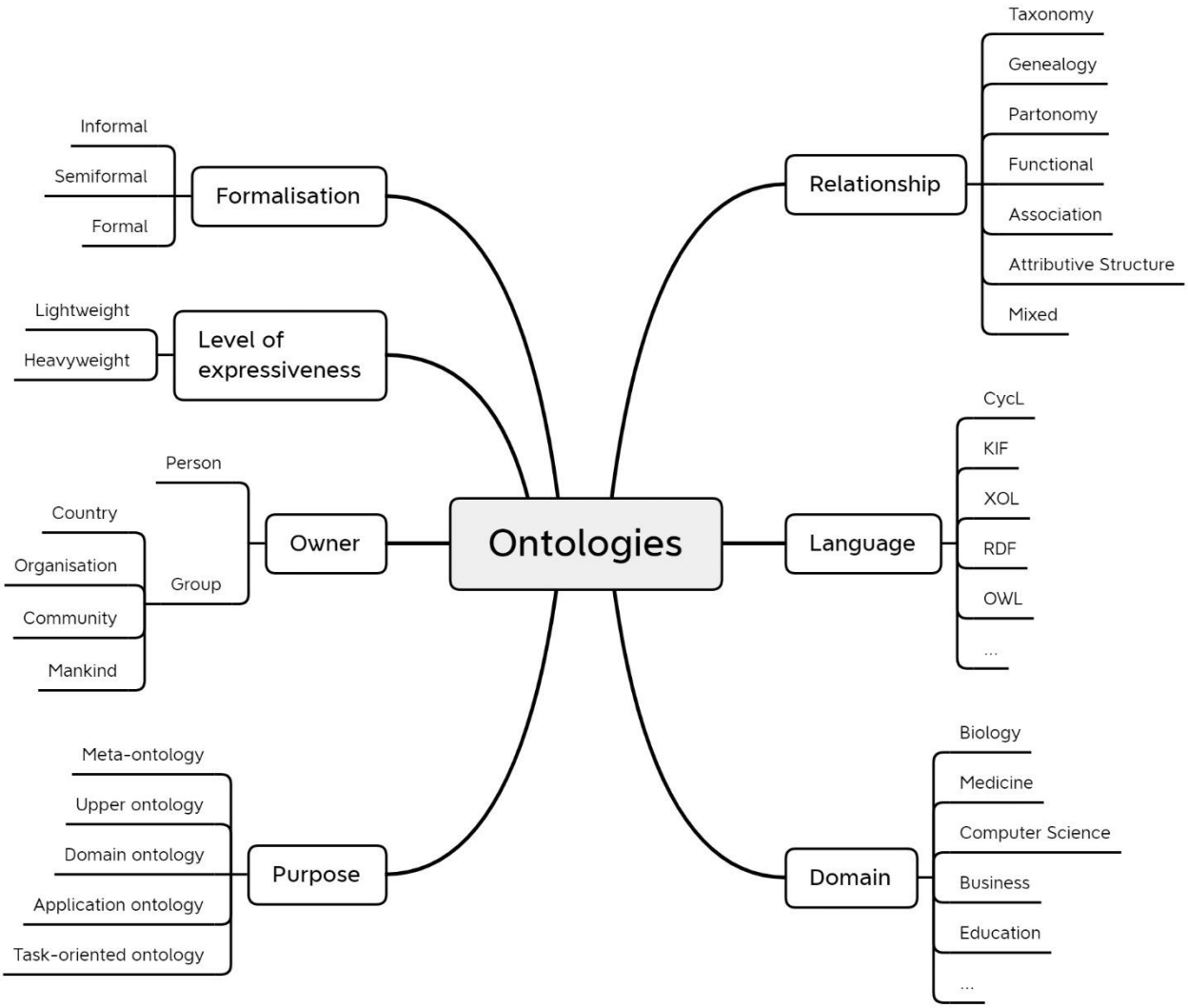


***Figure 1 —*** *Ontology classifications*

[Compiled by: Gavrilova,Horoshevsky, 2000; Gavrilova, Kudryavtsev, Muromtsev, 2020]

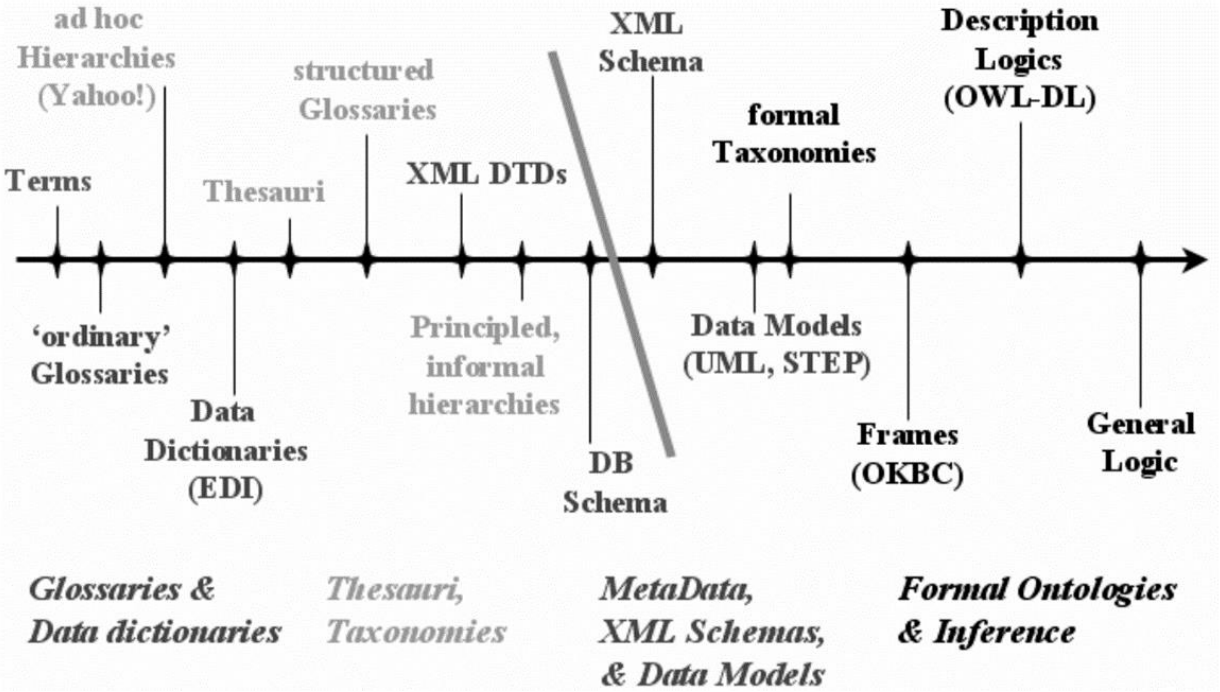


***Figure 2 —*** *Ontology spectrum*

[Source: Uschold, Gruninger, 2004]

According to the type of relationship:

- Taxonomies: The leading relation is rhodospecies («kind-of», «is-a»)
- Partonomics: the leading relation «has part» («consists», «has part»)
- Genealogy: the leading relation of the «progenitor-descendant» («father-son»)
- Functional: relations are expressed by verbs
- Associative: the primary relation is associated with
- Attributes: The Core Ratio has a Property
- Mixed ontologies: ontologies with any type of relationship

### 1.1.3 Applications and examples of ontological-type knowledge bases

The list of subject areas for which ontologies are created is extensive. Knowledge assets are already widely used to manage business processes in various industries. For example, the DSS for Mexico's drylands is based on a knowledge base through which farmers can get advice on how to deal with the situation in order to maximize their harvest. [Sanchez-Cohen et al., 2015].

Perhaps the largest number of ontologies and ontological knowledge bases have been created for medicine, as diagnosis was one of the first applied subject areas for knowledge engineering. These include various dictionaries (Metathesaurus http:///www.nlm.nih.gov/umls/, SNOMED http:/wwww.snomed.org/, MeSH http://wwwww.m.nlnih.gov/mesh.homeml, GALEN

http://www.opengalen.), ontologies for diagnostic systems, such as MIAKT for the diagnosis of breast cancer in women [Patlak et al., 2001] and others.

Information technology and knowledge-based systems can significantly improve the management of complex distributed health systems, where support for interdisciplinarity and essential communication and synchronization between different professionals are essential. For example, the HeCaSe2 multi-agent platform, which is based on an ontological knowledge base, helps physicians collect patient information, provides clinical guidelines for disease diagnosis and treatment, Coordinates all participants in the medical assistance process and monitors all appointments. [Isern, Sánchez, Moreno, 2012]. The DO4MG ontology is designed to support emergency relief, especially when many people are affected at once. Its mission is to ensure timely response and treatment of life-threatening injuries or diseases in the event of mass destruction, effective interaction and coordination between agencies and emergency teams, and real-time access to information. [Haghighi et al., 2010].

But besides medicine, ontologies are used in almost all spheres of human activity: in biology [Stevens et al., 2007], in agriculture [Sanchez-Cohen et al., 2015], in education [Nikitin, 2006; Gavrilova, Leshcheva, Kudryavtsev, 2012; Gavrilova, Leshcheva, Leshchev, 2000; Gavrilova, Leshcheva, Strahovich, 2011], in scientific research [Zagorolko, Borovikova, 2007; Leshcheva, Leshchev, 2014; Gavrilova, Leshcheva, Strakhovich 2015]for urban planning [Berdier, Roussey, 2007], for interacting of heterogeneous devices [Christopoulou, Kameas, 2005], for automated production [Gavrilova et al. 2019] and even knowledge of football [Buitelaara et al., 2008].

## 1.2. Development of ontological-type knowledge bases

The development of large systems (for example, the KB of an organization) is laboriousness in creating a single ontology [Tuzovsky, 2007], and two models of ontology-based KB are proposed:

- Abandoning the deep decomposition of the system and including only the most relevant concepts from the subject areas under consideration in ontology;
- To highlight one of the activities of the organization and to create a detailed but highly specialized ontology for this purpose.

At the same time, the first approach gives an overly rough and generalized model, and the second approach does not allow to use the model for interaction between all units of the company. To solve this problem, it is proposed to create a hierarchy of the organization’s knowledge areas

and create separate ontologies of different sub-areas, which may have different depth depending on the needs for their detailed modelling.

It should be noted that ontologies are subjective [Gavrilova, Leshcheva, 2015] and depend on many factors such as:

- The tasks to be performed by the ontology being developed;
- Methodology used to create ontologies;
- Ontology presentation language
- The software tool that is used to work on ontology;
- The cognitive styles of both experts and analysts working on ontology.

The main results obtained in the study of the connection of cognitive styles of experts with the characteristics of ontologies built by them are described in [Gavrilova, Leshcheva, 2015; Gavrilova, Lescheva, 2016].

The main recent methodologies of ontology development are discussed in the work [Gavrilova, Kudryavtsev, Muromtsev, 2016; Simperl, Luczak-Rösch, 2014]: DOGMA-MESS [De Moor, De Leenheer, Meersman, 2006; De Nheer, Debleeruyne, 2008], DILIGENT [Tempich et al, 2005, 2007], NeOn (www.neon-project.org) [Suárez-Figueroa, Gómez-Pérez, Fernández-López, 2012] etc. The most important modern recommendation for the development of ontologies is to use ready-available ontologies of the subject area that can be found in Swoogle (http:/swoogle.umbc.edu/) or in LOV (Linked Open Vocabularies [http:/lov.ok.org/])or to include them as much as possible in the process of creating ontology [Suárez-Figueroa, Gómez-Pérez, Fernández-López, 2012]. It is also possible to integrate several ontologies in order to obtain a single information space [Bova, 2015], for example by comparing ontological schemes and integrating concepts, with the subsequent creation of a query template for the mutual display of ontologies in each other.

Thus, three basic approaches to creating ontologies and knowledge bases based on them are highlighted [Petasis et al, 2011]:

1. Integration of existing ontologies. In the process of integration, an attempt is made to highlight commonalities in ontologies that describe similar or similar subject areas in order to create a new ontology. Several methods were suggested, such as:
   a. Merging (merging) ontologies to create a single agreed ontology;
   b. Aligning (alignment) ontologies by linking them, allowing them to reuse each other's information;
   c. Mapping (mapping) of ontologies by finding a correspondence between ontology elements.

2. Constructing ontology «from zero» or extending (populating and enriching) existing ontology, usually based on information extracted from subject-oriented content.
3. Specialization of general ontology to adapt it to a specific subject area. Ontology content (ontology population) is the addition of specimens with their properties to ontology, and enrichment (ontology enrichment) is the addition of new relationships and axioms that use these relationships. The development of the ontological knowledge base includes 4 steps [Pan et al, 2017]:

   1. Identification of requirements for the system being created and its knowledge base.
   2. Ontology or ontology networks.
   3. Content of ontology or ontology networks.
   4. Publication of the resulting database for further use. The following is a brief description of ontology presentation languages and ontological engineering tools.

### 1.2.1 Genealogy of ontology languages

Ontology description language is a formal language used to encode ontology. Most notable among them are: OWL (Ontology Web Language) is standard of W3C (World Wide Web Consortium, https:/wwwww.w3.org/), a language for semantic statements, developed as an extension of RDF (Resource Description Framework) and RDFS (RDF Schema); KIF (Knowledge Interchange Format is a knowledge sharing format) is S-based syntax for logic; CycL is an ontological language used in the Cyc project, based on predicate calculus with some higher order extensions; DAML+OIL (DARPA Agent Markup Language, Ontology Inference Layer).

Figure 3 presents the genealogy of the languages of ontology representation, linked to the time scale.

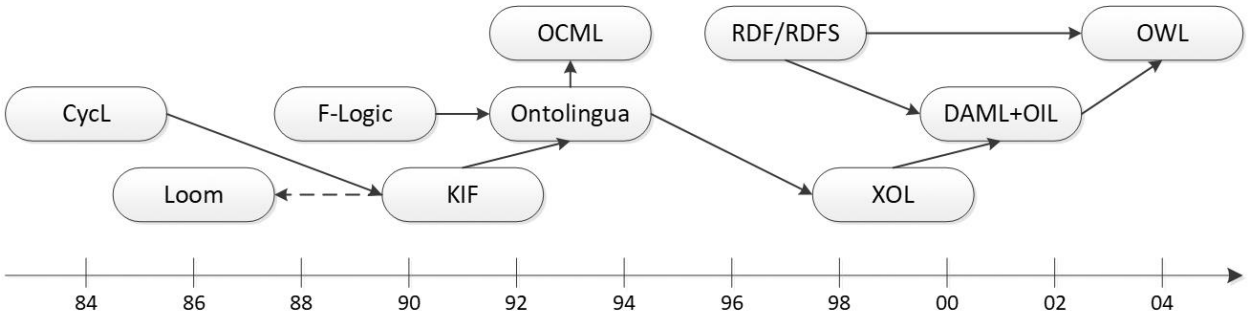


***Figure 3 —*** *Genealogy of ontology languages*

[Composed of: Kazekin, 2008]

The time scale indicates the start of language development (or first release). In general, the time scale is conditional because languages have evolved at different speeds. For example, although work on Loom began earlier, it was influenced by the KIF language.

***CycL***

CycL is a declarative language used to represent well-known knowledge in the commercial knowledge base of Cyc [http:/wwww.cyc.com/]. The project to create this knowledge base was started in 1984 by Douglas Lenat (Douglas Lenat). CycL is a hybrid language that combines first-order framework properties and predicate logic. CycL syntax is similar to Lisp syntax. Despite the unusual syntax, CycL has attractive tools such as nesting (in particular quantum nesting), scolemization, microtheory, error diagnostics, and five levels of truth for statements. Microtheory is a context that contains a body - a set of statements - and assumptions that are true of all statements. However, the truth or falsity of a statement in CycL depends on the context represented as a microtheory. Scolemization is the elimination of quantors of existence and their transformation into quantors of commonality. CycL supports the expression of first-order predicate logic and is strong enough to represent ontologies on it [Lenat, Guha, 1991; Reed, Lenat, 2002].

***Loom***

Loom is a language for constructing intelligent applications. It was developed by the ISI (Information Sciences Institute) at the University of Southern California with the support of DARPA under the leadership of Robert McGregor (Macgregor) in 1986. The Heart of Loom is a knowledge representation system that is used, to provide deductive support for the declarative part of Loom. A deductive inference mechanism called a classifier uses inverse inference, semantic unification, and object-oriented technologies to compile declarative knowledge into a network designed to effectively support the processing of deductive online queries. Loom declarative knowledge consists of definitions, rules, facts and default rules. A high degree of integration between the declarative and procedural constituents of Loom allows programmers to use logical programming, products, and the object-oriented programming paradigm in one application [http://wwwww.isi.edu/isd/LOMOM/LOMOMOH-OMEEhtml.] [MacGregor, 1999].

***F-Logic***

F-Logic was developed by Michael Kifer (Michael Kifer) and Georg Lausen (Georg Lausen) and is a formalism for knowledge representation. F-Logic is as relevant to object-oriented programming as classical predicate calculus is to relational database programming. Functionality includes, among others, object identifiers, complex objects, inheritance, polymorphism, query methods, and encapsulation. The strengths of F-Logic are extensibility and the ability to directly represent fundamental concepts that come from object-oriented programming languages and frame-based languages. The main weakness of F-Logic is related to the mathematical and logical concepts required for programming in this language [Kifer, Lausen, 1989].

***KIF***

KIF is a special language designed to share knowledge between different computer systems (created by different programmers, at different times, in different languages, etc.). The language was originally created by Michael Genesereth (Michael Genesereth) and other Exchange Project members

Defense Advanced Research Projects Agency (DARPA) is an agency of the United States Department of Defense responsible for developing new technologies for use in the armed forces. KIF was designed to be used as a syntax for first-order predicate logic, which is convenient for computer processing. Like CycL, KIF inherits the syntax from Lisp. KIF was conceived as a proxy language, it is not intended as the main language for interaction with the user, nor is it intended for internal representation of knowledge in computer programs (though it can be used for this purpose). Usually when the program reads the knowledge base in KIF, it transforms the data into its internal format. All calculations are performed using this internal format. When a program must communicate with another program, it displays internal data structures in KIF. The disadvantage of this language is its computational complexity [Genesereth, Fikes, 1992].

***Ontolingua***

The original Ontolingua language was developed by KSL (Knowledge System Laboratory) at Stanford University's Computer Science Department. Ontolingua uses the principles of object-oriented approach and is an extension of the computer-oriented KIF language for knowledge sharing and interaction between programs [Gruber, 1993]. Gruber expanded the KIF definition extension to provide additional idioms that are often found in ontologies, and added frame designs (in line with the OKBC (Open Knowledge Base Connectivity) protocol is an application-based programming interface to access knowledge bases.) to enable pseudo-object-oriented ontology to be specified using familiar relationships and functions such as Class, Subclass-Of, Slot, Slot-Value-Type, SlotCardinality and Facet. The disadvantage of a language is that it does not in itself provide a logical inference.

***OCML***

OCML (Operational Conceptual Modeling Language) was developed in the context of the VITAL project [Shadbolt, Motta, Rouge, 1993]. The main purpose of the language is to support knowledge-level modelling. OCML allows you to specify functions, relationships, classes, instances and rules. It also includes tools to describe ontologies and problem-solving methods - the core technologies developed in the field of knowledge representation. About a dozen projects at KMI (Knowledge Media Institute) used OCML to develop models in areas such as knowledge

management, ontology development, electronic commerce, and knowledge processing systems [Motta, 1998].

***RDF and RDF Schema***

RDF (Resource Description Framework) is the standard model for network data exchange [http://www.w3.org/RDF/]. RDF submits resource claims in a form suitable for machine processing. The resource in RDF can be any entity — both information (such as a website or image) and non-information (such as a person, river or some abstract concept). The structure underlying any expression in RDF is a collection of triplets, each consisting of a subject, predicate and object. The set of such triplets is called an RDF graph. The nodes of this graph are objects and subjects. The link is always directed from subject to object. Thus, the RDF graph forms a semantic network. The statement of the «blue helicopter» in RDF terminology can be represented as follows: subject — «helicopter», predicate — «has color», object — «blue». RDF uses URI [http:////www.w3.org/TR/2004/RECrdf-concepts-20040210/] to denote subjects, relationships and objects. URI (Uniform Resource Identifier) is a unified (uniform) resource ID [https://tools.ietf.org/html/rfc3986]. The RDF language is less expressive than those based on predicate logic calculus, such as CycL and KIF, but is much easier to implement. Since its main purpose is to annotate the resources of the World Wide Web, this is not a disadvantage or limitation. RDF itself does not provide mechanisms to describe properties or relationships, so it was necessary to complement what became RDF Schema, a dictionary description language. In fact, RDF Schema adds data types to RDF. The first version of the W3C recommendations is dated March 1999 [http://www.w3.org/TR/1999/PR-rdf-schema-19990303/].

***XOL***

XOL (XML Based Ontology Exchange Language) as the name implies is a language designed to exchange ontology descriptions based on XML syntax. A study of ontology representation languages found that none of the existing ontology languages met the requirements of the bioinformatics community. It was suggested that a language should be created with object-oriented knowledge systems, but with XML syntax. To meet these two goals, the XOL language was developed. Its authors are Peter Karp, Vinay Chaudhri, and Jerome Thomere. XOL semantics are based on OKBC-Lite, which is a simplified form of knowledge model for OKBC [Karp, Chaudhri, Thomere, 1999].

***DAML+OIL***

DAML+OIL [http://www.w3.org/Submission/2001/12/], proposed by W3C as a standard for the presentation of ontologies on the Internet, builds on earlier W3C standards such as RDF and RDF Schema, and extends these languages with richer modeling primitives. OIL (Ontology

Inference Layer) uses a set of primitives derived from framework languages of knowledge representation (XOL is the basis of the language) and a inference mechanism within descriptive logic. In OIL, ontology is a structure consisting of several components organized into three layers: the object level (for specimens), the first meta-level (which contains ontology definition) and the second meta-level (which contains information about the properties of ontology, such as its authors). A semantic web markup language based on OIL was created for DAML+OIL. DAML stands for DARPA Agent Markup Language. The language has a clear and well-defined semantics.

***OWL***

The OWL language is designed to be used by applications that must process the content of the information, not just deliver it to people. OWL provides a more complete machine processing of Web content than that supported by XML, RDF, and RDF Schema, providing an additional terminology dictionary along with formal semantics. The floor is recommended and developed by the World Wide Web Consortium (W3C [http:/wwww.w3.org/]). There are two versions of OWL (OWL 1 and OWL 2), OWL 1 has three dialects (in order of increasing expression): OWL Lite, OWL DL and OWL Full, OWL 2 has two dialects: OWL 2 Full and OWL 2 DL, the latter with three sub-languages (profiles): OWL 2 EL, OWL 2 QL and OWL 2 RL. Each of the profiles is traded with different aspects of the expressive power of OWL in exchange for other computational and/or implementation advantages.

OWL assumes that the world is open, i.e. that the truth of the claim does not depend on whether there is information about its loyalty. In other words, it cannot be said that a claim is false unless it is clearly stated. This is contrary to the assumption that the world is closed, which implies that any claim that is not known to be true is false.

Next, OWL 2. Several syntax are used to serialize OWL ontologies (RDF/XML, OWL/XML, Functional Syntax, Manchester Syntax, Turtle), of which only the first is mandatory, i.e. any software that works with OWL ontologies, must be able to open and preserve ontologies using this syntax.

The main elements of a language are classes, individuals and properties. OWL classes are interpreted as sets whose elements are individuals. They are described using formal constructs that declare requirements for class membership. By default, classes can overlap unless otherwise explicitly stated.

Properties in OWL represent binary relationships. Properties, like classes, can be organized hierarchically. There are two main types of properties: object properties (Object Property) and property-values (DataType Property). An object property is a relationship between two individuals. Value-properties bind individuals to a value (e.g., a numerical value, a string, a date, etc.). OWL also has a third type of property - annotation properties.

An object property can be the opposite of another property. The properties of P and Q are inverse to each other if for any individual x and y P(x, y) are equivalent $Q(y, x)$, i.e. $\forall\ x,y \in I\ P(x,y) \Leftrightarrow Q(y,x)$. You can specify that the properties do not intersect (disjoint), i.e. $P,Q \in R$ do not intersect if $\forall\ x,y \in I\ P(x,y) \Rightarrow \neg Q(x,y)\ \&\ Q(x,y) \Rightarrow \neg P(x,y)$. The property of an object can also have the following characteristics:

- Transitivity ($P \in R$ is transitivel, if $\forall\ x,y,z \in I\ P(x,y) \& P(y,z) \Rightarrow P(x,z)$)
- Symmetry ($P \in R$ is symmetrical if $\forall\ x,y \in I\ P(x,y) \Leftrightarrow P(y,x)$)
- Asymmetry ($P \in R$ is asimmetrical, if $\forall\ x,y \in I\ P(x,y) \Rightarrow \neg P(y,x)$)
- Functionality ($P \in R$ is functional if $\forall\ x,y,z \in I\ P(x,y) \& P(x,z) \Rightarrow y=z$)
- Inverse functionality ($P \in R$ back functional if $\forall\ x,y,z \in I\ P(y,x) \& P(z,x) \Rightarrow y=z$)
- Reflexivity ($P \in R$ is reflexive if $\forall\ x \in I\ P(x,x)$)
- Irrreflexivity ($P \in R$ is reflexive if $\forall\ x \in I\ \neg P(x,x)$)

Possible definition of composition (chain) properties. Property $S \in R$ is a composition of the properties P and Q if $\forall\ x,y,z \in I\ P(x,y) \& Q(y,z) \Rightarrow S(x,z)$.

Classes can be defined using property constraints. OWL restrictions are divided into 3 categories:

- Quantification limitation;
- Induced capacity constraints;
- Constraint of variables.

The qualifier's limitations are in turn subdivided into existential and universal. Existential constraints describe classes of individuals that participate in at least one relation on a given property with a representative of some given class. For example, the class «Teacher» can be defined as the totality of individuals belonging to the class «Man» and linked by the relationship «Teach» with at least one individual from the class «Discipline». Universal constraints describe classes of individuals that are related only to members of a given class. For example, it is possible to distinguish a class of teachers who give lectures only on elective disciplines, as a set of individuals belonging to the class «Teacher», and connected by the relationship «teaches» only with individuals of the class «Elective discipline».

The power limits shall determine the minimum, maximum or exact number of said ratios (properties). They can, for example, describe classes of teachers who conduct exactly one discipline or at least three disciplines.

A constraint on a variable allows you to describe a class of individuals that are associated with an individual relation or value. For example, the class of teachers working on a certain

department can be described as the class of individuals who are connected by the relation «works on the department» with the individual representing this department.

***SWRL Rule Representation Language***

One of the major drawbacks of OWL is that it is not possible to naturally determine property properties. This does not allow to model property-values in object relationships, n-ary relationships, and property-values in property-values. It is also not possible to perform any operations over properties-values (for example, comparing, computing formulae, etc.). Various additional means are used to remedy or circumvent these shortcomings. One such tool is SWRL (Semantic Web Rule Language), which allows you to add to ontology rules that cannot be represented on «pure» OWL.

The SWRL language is a merger of OWL and RuleML technologies (Rule Markup Language, which, as specified in the specification [RuleML Primer http://emrull.org/papers/Primer], in the future should become Rule Modeling Language or even Rule MetaLogic). In turn, RuleML is based on Datalog, which is a syntactic subset of Prolog-a.

A rule in SWRL consists of two parts: an antecedent and a consequent. Both parts are conjunctions (possibly empty) of so-called atoms. Informally, the rule can be interpreted as follows: if all atoms that are part of the antecedent are fulfilled (they are true statements), then all the atoms of the constitutive must be fulfilled. Atoms may take the form of:

- C(x): Here C can be some OWL description (most often a class) or data range. In the first case, x is an individual or an object variable, and in the second it is a value or variable. It holds if x belongs to C.
- P(x, y): P is an OWL property, both an object property and a value property. x is an individual or object variable, y denotes an individual or a value depending on the type of property P. It holds if x has a property P whose value is y.
- sameAs (x, y): sets the equivalence of x and y. This holds if x and y are the same person.
- differentFrom (x,y): states that x and y are different individuals. This holds if x and y represent different individuals.

In addition to these atom forms, SWRL offers a range of built-in formulas (built-ins) that allow for various operations over objects and data. The presence of these built-in structures is precisely the strength of the SWRL language. At the time of release, the SWRL specification [http://wwwww.w3.org/Submission/SWRL/] is defined as built-ins to perform comparison, mathematical operations, working with strings, date, duration and lists.

A few examples of rules in SWRL (a question mark is inserted before variables):

- *Person(?x) ∧ teach(?x, ?y) ∧ Discipline(?y) ⇒ Teacher(?x)*

This rule records the previous example, stating that any individual belonging to the class Person, who is bound by the relationship «teaches» with at least one discipline, is also a member of the class Teacher.

- *Teacher(?x) ∧ teaches (?x, ?y) ∧ Discipline (?y) ∧ forms (?y, ?z) ∧ Competence (?z) ⇒ forms (?x, ?z)*

If the teacher teaches a certain discipline that forms the competence, then the teacher can be said to form the competence of the students. It should be noted that both of the above rules do not require the use of SWRL, but can be written in OWL.

- *Rectangle (?x) ∧ length (?x, ?a) ∧ width (?x, ?b) ∧ swrlb:equal (?a, ?b) ⇒ Square (?x) ∧ side_length (?x, ?a)*

If the length and width of a rectangle are equal, it is a square with a side length equal to the length (or width).

### 1.2.2 Ontological engineering software systems: analysis and comparison

According to research of XML.com website [http://ww.xml.com/pub/a2004/07/14/onto.html] in 2002 the list of tools for work with ontologies consisted of 52x, in 2004 — 93th products. In 2008, http://wwww.hozo.jp/OntoTools/ already listed more than 150 systems. Currently, the website http://wwww.w3.org/2001/sw/wiki/Tools contains 340 different tools. The growth in the number of tools demonstrates a rapidly growing interest in ontology development.

The following is a comparison of several free systems according to the following criteria:

- Formalism underlying the presentation of ontological knowledge;
- Ontology presentation language
- The functionality of the ontology editor, in particular:
  - Ability to view and edit ontologies using a web interface;
  - Display and editing of ontology in visual form;
  - Logical inference mechanism
  - Request language support
  - extensibility of the system (i.e. possibility to connect additional modules created by third-party developers).

Additional modules for alignment (alignment), mapping (mapping), merging (merging) and evaluation (evaluation) can be included in ontology creation and support tools ontologies, but their presence or absence is not of interest in this work.

***Ontolingua Server***

Ontolingua Server [http://wwww.ksl.stanford.edu/software/ontolingua/] was the first tool to co-create, view, edit and use ontology in WWWW [Farquhar, Fikes, Rice, 1997]. It was developed in 1995 by KSL (Knowledge System Laboratory) at Stanford University's Computer Science Department.

The ontology editor is implemented on the basis of HTML forms. The system includes a graphical browser that allows you to view a hierarchy of concepts, including instances, but does not allow you to edit the taxonomy of concepts presented in a graphic form.

One of the functions of Ontolingua is integration with other knowledge systems. This function is performed by the OKBC subsystem based on the KIF Machine-to-Machine Knowledge Exchange Language. Ontolingua has recently adopted OWL as the standard language for the exchange of ontologies.

In addition to the ontology editor and subsystem, OKBC Ontolingua includes the network application Webster (defining concepts) and Chimaera (analyzing, combining, integrating ontologies). All applications except the OKBC server are implemented on the basis of HTML forms. The knowledge representation system is implemented on Lisp. The formalism underlying knowledge representation is first-order framework and logic.

The server supports the collaborative development of ontology by multiple users using user and group concepts. Ontolingua Server also provides an ontology archive that includes a large number of ontologies from various subject areas, allowing the creation of ontologies from existing ones.

Disadvantages include lack of ontology testing for non-performance. This is rooted in Ontolingua, which is the main language of knowledge representation in the system.

***OntoSaurus***

OntoSaurus [http://wwww.isi.edu/isd/ontosaurus.html] was implemented around the same time as Ontolingua. It was created as a web editor and browser for Loom ontologies. OntoSaurus provides automatic compatibility control, deductive reasoning support and some other features [Swartout et al., 1996]. It also provides limited editing tools, so you have to master Loom to build complex ontologies.

The server part is also written in Lisp. Descriptive logic is used as the knowledge representation model.

***WebOnto***

WebOnto [http://projects.kmi.open.ac.uk/webonto/] is an editor for sharing, creating and editing ontologies using the OCML language [Domingue, 1998; Domingue, Motta, Garcia, 1999].

It was developed in 1997 for Tadzebao, a tool that allows knowledge engineers to conduct synchronous and asynchronous discussions about ontology. Tadzebao recognizes that dialogue is an integral part of collaborative ontology development, and because development is often done remotely, a tool is needed to support such dialogue.

Unlike the two previous tools, the WebOnto client is a Java applet. OCML is used to model ontologies.

The user can create different structures, including classes with multiple inheritance, and editing can be done in a graphical environment. The tool provides scaling tools for building large ontologies and has a number of useful features, such as a separate view of relationships, classes, and rules. WebOnto also validates re-entering data with OCML code integrity control. The formalism underlying knowledge representation is first-order framework and logic.

***OilEd***

OilEd is a stand-alone graphic editor of ontology developed by On-To-Knowledge [Bechhofer S. et al., 2001]. OilEd was developed in 2001 as an editor of OIL ontology. With the advent of DAML+OIL, OilEd was adapted to DAML+OIL ontology and later adapted to OWL.

OilEd combines frame structure and expressiveness of descriptive logic with reasoning services, which allowed for an understandable and intuitive user interface style and the benefits of reasoning support (discovering logically contradictory classes and hidden subclass relationships).

OilEd is written in Java. It is freely distributed under a public GPL license, but is no longer supported.

OilEd was never finished and does not provide a complete ontology development environment. It lacks version control, does not support the development of large scale ontologies and, due to the use of FaCT's reasoning mechanism, limits support for class instances.

***Protégé***

Protégé [http://protege.stanford.edu/] is a local free ontology editor developed by the Stanford University Medical Informatics Group (first released in 1987, now version 5) [Gennari et al., 2003; Knublauch et al., 2004]. It is now the most widely used and freely available tool.

Protégé is designed to create applied ontologies from scratch, allows for the quick and relatively simple construction of a small subject ontology, and has a scalable architecture that allows it to be easily integrated into applications. The ontology structure is similar to the hierarchical directory structure. Based on the ontology formed, Protege can generate forms of knowledge acquisition for the introduction of class and subclass specimens. The tool has a graphical interface, easy to use by inexperienced users, with help and examples.

Protégé is an open-source Java program. It is possible to extend this tool with plugins. The Protégé platform supports two ways of modelling ontologies - based on the OKBC knowledge representation framework model and with OWL. Accordingly, the main expressive capabilities of OWL can be realized - distinguishing individuals, classes and properties; hierarchy of classes and properties; specifying the scope and range of values for object properties, etc. It is also possible to indicate that a pair or group of classes are mutually exclusive. Classes can not only be described but also defined by specifying its necessary and sufficient properties. An individual may belong to several classes simultaneously or not belong to any class. Ontologies can be exported in various formats, including RDF, RDFS, OWL and XML Schema.

Web version - WebProtege [https:///webprotege.stanford.edu/] is being developed.

***Fluent Edidor***

Fluent Editor [http://wwww.cognitum.eu/semantics/FluentEditor/], an ontological editor using a controlled natural language (Controlled Natural Language), is a universal tool for creating and processing complex ontologies. Fluent Editor provides a more user-friendly alternative to the OWL editor based on XML. The main feature is the use of controlled English as a knowledge modeling language. It is supported by a special Predictive Editor that prohibits the input of any sentence that is grammatically or morphologically incorrect and actively assists the user during sentence writing.

Another feature of Fluent Editor is the use of so-called active rules (Active Rules). The active rules are a Fluent Editor mechanism that ensures the execution of the user's mandatory code on C# under certain criteria described as SWRL rules. With active rules you can insert knowledge into the knowledge base, remove it from there, and print messages into the output window.

Fluent Editor can be extended with different plugins. In particular, by installing a special plugin, you can instantly synchronize the ontology representation between different Fluent Editor and Protégé windows in one place. The interoperability of powerful ontology editors such as Protégé and Fluent Editor offers great opportunities to use the strengths of each editor to work with a single ontology.

***Comparison of ontological engineering tools***

Table 1 presents a comparison of ontological engineering tools by previously selected criteria.

**Table 1** — Comparison of ontology editors

| Comparison criteria | Ontolingua Server | OntoSaurus | WebOnto | OilEd | Protégé | Fluent Editor |
|---|---|---|---|---|---|---|
| Formalism | Frames First order logic | Description logic | Frames First order logic | Description logic | Frames First order logic | Description logic |
| Language | Ontolingua | Loom | OCML | DAML+OIL | OKBC, OWL, SWRL | OWL, SWRL |
| User interface | HTML | HTML | Java applets | Local program | Local and WebProtégé | Local program |
| Visual editing | – | – | + | – | + | – |
| Logical inference mechanism | JTP | Classifier Loom | OCML | FaCT | HermiT, FaCT++, Pellet | HermiT |
| Query Language Support | – | – | – | – | + | + |
| Extensibility | – | – | – | – | Plugins | Plugins |

The table shows that Protégé and Fluent Editor are the most developed and convenient to solve this problem, as they provide editing capabilities for both textual and visual ontologies, extensions of functionality using plugins and supports Rule Description Languages (SWRL, Active Rules) and SPARQL queries. If you need to access ontology via a web interface, you can go to WebProtégé. In addition, Protégé and Fluent Editor have proved to be among the few free-to-distribute tools that continue to grow. But the Fluent Editor tool has a disadvantage: it is focused on class names, relationships, etc. exclusively in English, which limits its application environment.

## 1.3. Problem Statement on Ontology Population and Ontology Life Cycle Research

There are many theories, methodologies and tools for modelling different types of semantics using ontologies. However, once the organization has completed the ontology structure simulation, this ontology must be populated with individuals (specimens) and relationships for practical use. This process can be quite laboriousness, especially if it is produced «manually», so it needs to be automated. Methods are required that can be applied to integrate heterogeneous data into the ontology of the domain, both at the design stage and at the operational stage.

The population of ontologies in practice presents a number of difficulties, mainly due to the heterogeneity of data sources. This heterogeneity may be different in nature: data presentation formats, data models, terminology used, differences in data types, units or sets of tolerable values, etc. Even if all data are in the same source, The interpretation of how to interpret data semantics and ontology content depends not so much on the data source model as on the model embedded in the subject area ontology.

In order to analyse existing ontology population methods and identify limitations or deficiencies, consider the ontology life cycle in terms of population and related problems.

The life cycle of ontologies can be considered in two ways. In a broad sense, it covers stages from the conception to the implementation of ontology and its dissemination [Farquhar et al., 1995; Fernandez, Gómez-Pérez, Juristo, 1997; Gandon, 2002; Li, Raskin, Ramani, 2007]. In a narrower sense, it is regarded as a direct development of ontology [Uschold et al., 1998; Noy, McGuinness, 2001; Nanda et al., 2006]. An attempt to reconcile these visions was made by the authors of the NeOn methodology [Suárez-Figueroa et al., 2012], which understand the life cycle in a broad sense.

In the life-cycle model, several general stages are broadly distinguished, each of which may lead to problems in population the ontology with specimens. A number of such problems are described in [Petasis et al, 2011; Celjuska and Vargas-Vera, 2004]. Classify the problems by stages in the ontology life cycle:

1. Ontology Specification is determination of purpose and content boundaries. At this stage, the inclusion of «sensitive» (for example, personal) data in ontology should be solved.
2. Development of ontology, which includes information gathering; conceptualization is formation of classes, properties, their hierarchies and relationships, etc. At the same time, the ontology is initially populated with data. However, a number of population problems can be identified. These include:
    a) Object recognition is identifying parts of data that are instances;
    b) Assigning objects to classes;
    c) Synonymy of objects is the presence of different values related to the same object;

d) object homonymy is similar objects belong to different classes.

3. Evaluation is checking ontology before use. In the case of automatic or semi-automatic population, the adequacy, uniqueness, consistency, completeness and redundancy of the copies of the categories shall be checked.
4. Support in use process. At this stage, evolution - the development of ontology, including the population of new terms and enrichment - can take place. The development of ontology raises problems of consistency between new data and older data (from syntactic, such as changes in input formats, to semantic and structural data).
5. Documentation, including control of changes and consistency of versions. The documentation process should describe the ontology population procedure and the techniques used.

As can be seen from the list above, the content of ontology in one way or another affects all stages of the life cycle of the knowledge base, but most influences the stages of development and support in the use process. Thus, the research task is to develop a method and algorithms for population ontological-type knowledge bases to integrate heterogeneous sources of structured data. The main features of the method should be:

- The possibility of using it to enrich the knowledge base at different stages of the life cycle.
- Possibility of simultaneous population from several heterogeneous sources.
- Integration with world standards in ontology description languages to enable the development and maintenance of a knowledge base using existing ontological engineering tools.

## Conclusions on chapter 1

Modern information systems make extensive use not only of databases but also of knowledge bases. Ontology is the leading model of knowledge representation. The review and analysis of literature presented in the first chapter shows that among the many problems of industrial applications of ontology, it is important and relevant to automate the population of ontological-type knowledge bases with specimens, which is the subject of this thesis.

Also in the first chapter were defined basic concepts of ontological engineering, basic classifications of ontology, examples of fields of application and knowledge bases.

Existing methods for populating ontology-based knowledge bases have a number of disadvantages, such as the narrow focus on project-specific tasks, the complexity of describing models of data sources and their transformation, High user qualification requirements, possibility of integration of data from only one type of sources, lack of modularity and extensibility, as well as convenient tools.

The main complexity of information modelling is heterogeneity, i.e. heterogeneity of the components, data and knowledge described. Moreover, the heterogeneity of data sources has the greatest influence on the process of populating knowledge bases. Problems encountered in population were classified according to the stages of the knowledge base life cycle. Major difficulties were identified in the development and operation phases.

A review of ontological engineering research over the past 10 years has shown Despite the existence of a large number of completed projects and industrial ontology applications, there are significant gaps in the theoretical understanding of the software tools for developing ontological-type knowledge bases. In particular, there is a need to better understand the content of knowledge bases in the context of subject matter semantics.

On the basis of a literature survey [Simperl, Luczak-Rösch, 2014; Suárez-Figueroa, Gómez-Pérez, Fernández-López, 2012; Tempich C. et al., 2005], the main approaches to the development of ontological-type knowledge bases were identified and their peculiarities analyzed. The genealogy of ontology presentation languages has been expanded and expanded. A brief overview of the rules presentation languages with an emphasis on the SWRL language has been made. The chapter also presents the results of the up-to-date analysis of freely-distributed tools for creating and populating ontologies.

The analysis revealed limitations of existing approaches and methods. In the first chapter, the task of research is to create a method for population of ontological knowledge bases and integrating heterogeneous sources of structured data, taking into account the semantics of the subject area. The main features of the method described below are:

- The possibility of using it to enrich the knowledge base at different stages of the life cycle.
- Possibility of simultaneous population from several heterogeneous sources.
- Integration with world standards of ontology description languages to enable development and maintenance of the knowledge base using existing ontological engineering tools.

Chapter 2 describes the newly developed method of ontology population and the architecture of the software prototype.

# CHAPTER 2 METHOD FOR POPULATION OF ONTOLOGICAL KNOWLEDGE BASES

## 2.1 Ontology population

Classical ontology focuses on the modelling of subject domains in terms of classes and subclasses. Once the modeling of classes, attributes, relationships and hierarchies is complete, it is necessary to populate ontology with instances. In many large organizations, data or knowledge can be taken in different formats, such as relational KB, web pages, documentation, etc. To make this information available for general ontology, converting them from their current presentation to a knowledge presentation format is required.

Ontologies can be populated manually and automatically. The variants of manual population of KB are considered in [Grigoriev, Zalotsky, Kudryavtsev, 2012]. For automated population from the point of view of sources:

- Use of structured data (DB, tables, xml files...) [Song, Zacharewicz, Chen, 2013];
- Use unstructured data (texts). Natural language text analysis is applied. The basic terms/approaches are semantic annotation (semantic annotation), information extraction (information extraction) [Domingue, Fensel, Hendler, 2011].

Approaches to addressing the content and enrichment of ontology depend on the level of integration, the properties of the data sources and the required integration techniques and methods. The classification of approaches is shown in figure 4

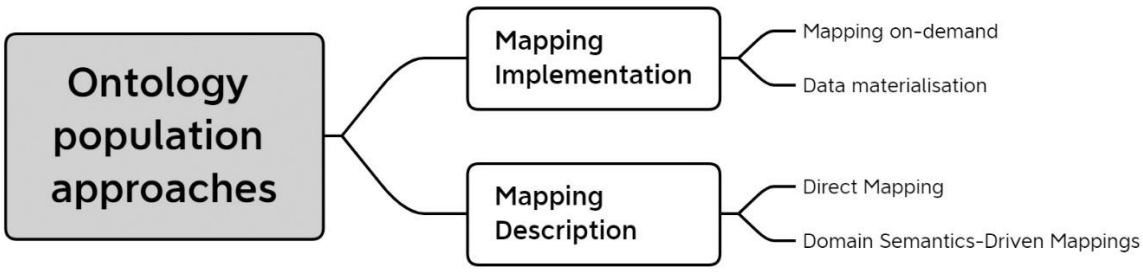


***Figure 4 —*** *Approaches to the population of ontologies*

Depending on the problem to be solved, two approaches can be used to populate the ontology. The first approach, mapping on demand (mapping on-demand), is similar to ontology mapping, but establishes a relationship not between two or more ontologies, but between ontology and data stored in distributed data sources.

The approach worked well in the context of very large amounts of decentralized data. It ensures that the returned values are always relevant, as data is not copied to ontology. It also ensures compliance

with the access management policies implemented in the database management system. On the other hand, if a large number of data sources are integrated or a logical output machine is used to handle complex queries, the performance of queries to databases can become a significant constraint [Sahoo et al., 2009]. It has been suggested [Erling, 2008] that the expressiveness of requests to the knowledge base should be limited, as in particular a variable in the predicate position (resourceA ?r resourceB) or a variable in the position of the object representing the class (resourceA rdf:type ?) can lead to «bomb explosion» effect: exponential increase in the number of generated queries.

The second approach — the inclusion of data (consolidation) in the knowledge base as property values of objects (data materialization) - is used, in particular, in the construction of data warehouses (data warehouse). Once the consolidation process is complete, all necessary data are placed in ontology, making it possible to take full advantage of knowledge bases as compared to databases, namely, logical inference tools and tools for visualizing relationships between objects. But this approach contradicts the principle of separation of ontology code and data, which makes it difficult to develop and, most importantly, to support ontology. If a new person appears or if the properties of the previously described individual are changed, you must create/find the appropriate object in the knowledge base and set/change the values of its properties. In order to deal with data obsolescence, the consolidation process has to be started regularly, leading to the need to find a trade-off between the cost of updating data and the sensitivity of the application to outdated data. A second limitation may be the amount of data, as the resulting knowledge base may exceed the amount of resources available.

Depending on whose structure is primary or more important, the structure of ontology or data sources, distinguish direct integration (Direct Mapping) and integration that takes into account the semantics of the subject area (Domain Semantics-Driven Mappings) [Michel, Montagnat, Faron-Zucker, 2014]. Direct integration is implemented automatically, with the structure of the resulting ontology being completely «inherited» from the structure of the data source. But the structure of the data source can rarely be a good description of the subject area, which is the result of both the process of optimizing productivity and the long history of service, and, in any case, the scheme itself will not represent a complete description of the subject area, as the relevant links often save in the code or in the encoding of various attributes [Dolbear, Goodwin, 2007]. Therefore, either the result of direct integration is used as a starting point to create an ontology that more accurately describes the domain, for example by a method of alignment, or integration that takes into account the semantics of the domain is applied. Using the second approach, the ontology of the domain is taken as the basis, and the rules of how the ontology is populated are set using some mapping language (mapping description language) that is processed by a specialized (pre) processor.

The main types of structured data sources are relational databases, XML documents, spreadsheets, and proprietary structure text files. For the time being, methods for integrating data from different types of sources are being developed in parallel. Consider the main ones.

### 2.1.1 Translation of relational databases into ontology

Relational databases are most commonly used to store structured data sources. In case of direct integration, the layout of the database is directly translated into ontology by means of a special translator program. Mapping rules are set by developers, so using different programs from one database you can get a whole range of similar but not identical ontologies.

When you convert a database schema to ontology using the semantics of the subject area, the mapping rules are set explicitly using the mapping description languages. These languages can be divided into two groups. The first group of languages relies heavily on SQL queries to describe the display of data, which is potentially a disadvantage since complex mapping cases cannot be described with them. On the other hand, the popularity of SQL facilitates the adoption of this approach, as there is no need to learn a new language. The second group languages use specialized mapping description languages, allowing them to be created/extended in a way that satisfies any specific needs, such as searching for keywords and regular expressions. But in practice the expressive capabilities of the second group are still very limited [Michel, Montagnat, Faron-Zucker, 2014].

The main features that mapping languages and tools implementing them should provide are:

1. Generation of unique user-defined identifiers (ability to generate URI resource identifiers other than simply using the primary key values, for example by combining column values, etc.)
2. Logical tables (ability to read tuples not only from tables, but also from representations (SQLview) or SQL.)
3. Field selection (ability to select only a subset of the table columns to translate.)
4. Renaming Fields (ability to map a column into a RDF property with a different name.)
5. Selection condition (possibility to translate only a subset of tuples (an ordered set of elements) of a table by setting the WHERE condition in the SELECT.)
6. Use of existing ontology (possibility of mapping relational objects into copies of existing ontologies.)
7. Mapping one table in n classes (the ability to use column values as a categorization template: the tuples of the table will be translated into examples of different ontological classes based on the value of this field.)

8. Converting the many-to-many relationship into simple triples (the ability to translate a connected table representing many-to-many relations into simple triples, as opposed to a direct map in which the linked tables are transferred to a separate class.)
9. Creation of anonymous nodes (the ability to create anonymous nodes (blank nodes) and refer to them within the graph received during translation.)
10. Translation of data types (possibility of processing relational data types into RDF data types)
11. Data processing (possibility to apply the conversion function to values before forming RDF triples, such as performing a complex transformation of type, computing a value with multiple columns, processing rows, etc.)

The first language of the mapping description was DR2 MAP [Bizer, 2003]. It refers to the first group. The language syntax is based on XML. Using the DR2 MAP language, it is possible to describe only simple mapping rules, so it was necessary to develop it, resulting in the language DR2Q [Cyganiak et al., 2012]. The mapping description for DR2Q is defined by the RDFS schema, but still relies heavily on SQL fragments, for example to use aggregated functions. This language is used in the DR2 server [http:///d2rq.org/; Bizer, Seaborne, 2004] which supports both types of integration and provides different ways of obtaining data. Another follower of D2R was R2O [Barrasa, Corcho, Gómez-Pérez, 2004]. Its syntax is also based on XML, but the expressiveness of this language is much greater, in particular the search for keywords and regular expressions, arithmetic operations, String processing, setting limits on range of values, etc. The language has been used in ODEMapster [http://neon-toolkit.org/wiki/ODEMapster] and DB2OWL [Cullot, Ghawi, Yetongnon, 2007]. Due to the very complex syntax of this language or for some other reason, the developers abandoned it and replaced it with a R2RML-compatible implementation. There are several other projects that have used their mapping languages, such as METAmorphoses [Svihla, Jelinek, 2004], RDBToOnto [Cerbah, 2008], Relational.OWL [http:/source.net/projects/relational-owl/; De Laborda, Conrad, 2006]But their support is long overdue.

In 2012, W3C RDB2RDF issued two recommendations: A Direct Mapping of Relational Data to RDF [https:/www.w3.org/TR/rdb-direct-mapping/] (Direct mapping of relational data to RDF) and R2RML: RDB to RDF Mapping Language (R2RML: RDB mapping language to RDF) [Das, Sundara, Cyganiak, 2012; https:///ww.w3.org/TR/r2rml/]. The first recommendation, as the name implies, regulates the direct mapping of relational databases to RDF. The second document defines the mapping description language from the relational database in RDF, but does not make any recommendations for the implementation of the R2RML processor, so there are a number of R2RML-compatible tools with original implementation approaches such as Ultrawrap (http://capsenta.com/ultrawrap), DB2Triples (https://github.com/antidot/db2triples). R2RML absorbed the best of its predecessors: it arose from their study, was based on their experience and encompassed most of their expressiveness. Therefore, it seems

inevitable to use R2RML when it comes to converting relational data to RDF. Especially since the tools that appeared before R2RML now support it as well. For example, Virtuoso (http://virtuoso.openlinksw.com/dataspace/ doc/dav/wiki/Main/) contains a simple adapter that translates R2RML into the syntax of its own language. However, the authors [Michel, Montagnat, Faron-Zucker, 2014] believe that R2RML cannot be considered an exhaustive solution, so all new proposals will appear. This is because some approaches rely on complex templates that cannot be expressed in R2RML. For example, RDBToOnto [Cerbah, 2008] analyzes redundancy to identify categorization templates. And [Hu, Qu, 2007] uses mining techniques to automatically detect mapping between the database and existing ontology. Finally, R2RML does not provide data manipulation functions, but relies on relational DBMS instead of an internal interface. Disadvantages include learning the syntax of a language or creating a tool to help set the mapping rules in that language. In addition, it is not determined how the database connection is established, how authentication is conducted and, accordingly, how to integrate data from multiple databases at the same time.

### 2.1.2 Data integration from XML documents

XML (Extensible Markup Language) [https://www.w3.org/XML/] is an extensible markup language, a simple flexible text format designed for structuring, storing and transmitting information between applications, oriented towards use on the Internet. XML documents are divided into two categories: data-oriented (data-oriented) and document (content-oriented), but only the first category is considered in the context of transforming XML schemas into ontology.

First methods of converting XML documents into ontology [Ferdinand, Zirpins, Trastour, 2004; García, Celma, 2005] implemented direct integration, then came methods to influence the transformation of the XML scheme into ontology, for example [Anicic, Ivezic, Marjanovic, 2007; Cruz, Nicolle, 2008]. One of the most advanced in terms of capabilities is described in [Rodrigues, Rosa, Cardoso, 2006], in particular, it allows the mapping of multiple XML circuits into one existing ontology, including the creation of individuals. For this purpose, the authors developed their own mapping language, which describes how XML nodes are transformed into ontology elements. All existing approaches can be divided into two groups [Hacherouf, Bahloul, Cruz, 2015]. The first group includes approaches that allow working with XML documents without an XML schema. [O'Connor, Das, 2011; Bohring, Auer, 2015]. The second group includes approaches based on processing the XML schema [Aussenac-Gilles, Kamel, 2009; Bakkas, Jakjoud, Bahaj, 2014; Quix, Kensche, Li, 2007]. It should be noted that none of the proposed methods has become standard and in every project where the transformation of XML documents into ontology is required, a tool is used that is specially developed for the needs of the project and takes into account its peculiarities.

### 2.1.3 Converting Tables to Ontology

Although spreadsheets are the predominant tool for presenting and processing structured data, relatively little work has been done on transforming tables into ontology. It can be assumed that this is due to the fact that tables can be presented in the form of relational databases, for which a number of methods of conversion to ontology have already been developed. However, the need for additional conversion for projects that use multiple disparate tables of different structure can be a significant constraint, so a method for converting tables to ontology is appropriate.

The work of [Tijerino et al., 2005] proposes the approach of TANGO (Table ANalysis for Generating Ontologies), the essence of which is as follows: in the first step of the table «canonizes», i.e. transforms into a standardized form suitable for further analysis; In the second step, a set of «mini-ontologies» is created, each of which describes the structure of one canonized table; then the «mini-ontologies» are merged into one resulting ontology. Thus, the TANGO approach implements direct integration, and the resulting ontology describes the structure of the tables and in general cannot be considered a domain ontology. The main difficulty in obtaining data from ontology-population tables is that the structure of the table can be arbitrary. The first population methods were based on the assumption that the table is in the «standardized» form, i.e. the first row contains column headings and below them are data (similar to the representation of tables in databases), but the structure of the tables, used in practice rarely corresponds to this assumption [O'Connor, Halaschek-Wiener, Musen, 2010]. Later approaches attempted to overcome this assumption by importing cells as separate objects or blocks [Abraham, Erwig M, 2004; Ozturk, 2020] or adding a semantic structure to the table [Nederstigt et al., 2014; O'Connor, Halaschek-Wiener, Musen, 2010; Han L. et al., 2008; Langegger, Wöß, 2009].

## 2.2 Automated Ontology Population Method from Structured Data Sources

This work developed methodology and technology for ontology population based on integration with heterogeneous structured data sources (taking into account their semantics) from structured data sources (METEOR method).

The idea behind the METEOR method is that the user wants to import data from different structured sources (such as relational databases, XML documents, spreadsheets, etc.) into a single subject area ontology (SAO). To do this, it must do two things:

- First, describe the structure of data sources using Data Source Ontology (DSO).
- The second is to formulate mapping rules for relevant entities from data sources and SAO to populate in.

Both actions are performed by a system that takes data, extracts their structure and inserts data into SAO. The architecture of this system is described further in the paragraph «2.2.1 System

architecture». Detailed stages of system operation are described in the paragraph «2.2.2 Process of ontology population of the subject area from structured data sources».

### 2.2.1 System architecture

The proposed approach to the population of ontologies is based on the extraction, transformation and loading technology (ETL: extract-transform-load). Accordingly, the architecture of the system implementing the proposed method of ontology population is similar to that of ETL systems [Paklin, Oreshkov, 2013] as they perform similar tasks (see Fig. 5). The main difference is that the receiver of the data is not a repository or database, but an ontology or knowledge base, which leads to other features. Stages are based on [Vassiliadis, Simitsis, 2009] and adapted to knowledge bases as follows:

**Extraction**: Data sources for insertion into ontology are identified and specified and data extracted.

**Transformation:** Data cleansing and conflict resolution are performed together with the mapping of the data structure to the ontology of the domain (i.e. the structure of the knowledge base).

**Download:** Source data are inserted at the appropriate place in the knowledge base (i.e. SAO is populated).

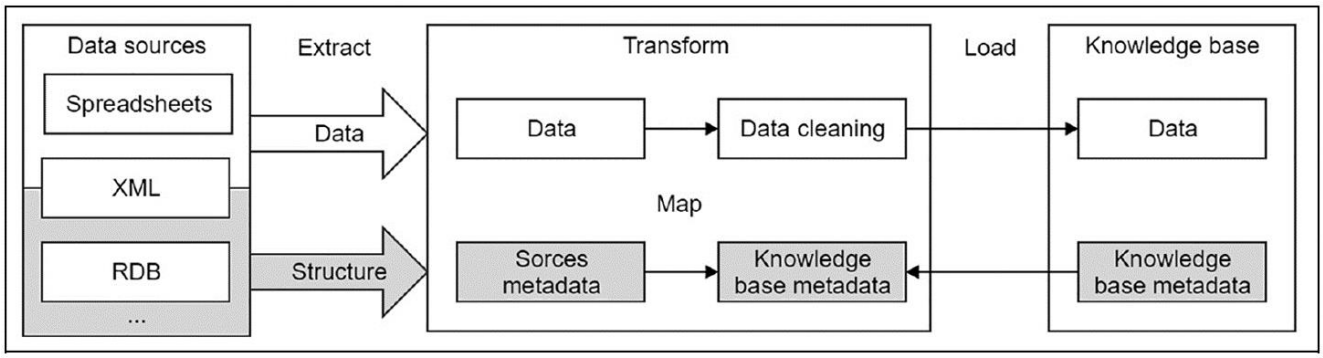


***Figure 5 —*** *Adaptation of ETL technology to populate ontological knowledge bases*

**Generalized structure of knowledge content**

In terms of the ontology population process, the architecture of the system can be represented by five components (Fig. 6), such as:

- Data sources contain structured data in the form of a set of tables and/or files in which, for example, data are ordered into typed columns separated from each other by some separator characters;
- Interim area contains supporting tables corresponding to data sources and an auxiliary knowledge graph, set of triplets based on meta-templates, created solely for the organization of the population process;

- The data recipient is the ontology into which the extracted data should be placed;
- Mapping rules are data flow requirements described by the analyst using a metatemplate;
- Ontology editor is used as a tool in ontology creation and in describing processing or mapping rules.

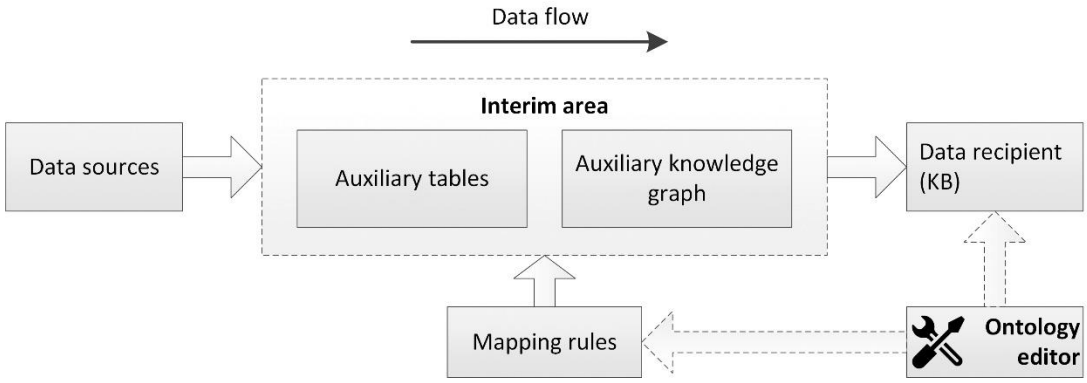


***Figure 6 —*** *Elements of ontological knowledge architecture*

**Data flow during ontology population**

Power Query is a data connection technology available in the classic versions of MS Excel and Power BI.

An auxiliary knowledge graph is created using the ontology of data sources. It contains a data structure that will be processed according to rules established jointly by the knowledge engineer and subject matter expert. The knowledge base is created after the ontology of the subject area is populated.

A peculiarity of the approach is that the processing rules are set by the knowledge engineer when interacting with the subject-matter expert in the form of ontology, which does not require special tools, programming skills or knowledge of specialized languages for describing mapping.

The approach is implemented in the form of four modules, two of which require manual input from the user (Fig. 7). The modules serve the following functions:

**Data Specification Module.** This module creates a data source specification using information on relevant data sources and their location provided by the user.

**Data structure extraction module.** The module refers to the data sources, then extracts their structure and uploads it to the DSO together with the information on the data source from which the data was extracted. The result of this action is a populated-in DSO, an auxiliary knowledge graph that will be used to integrate data from relevant sources into DO.

**The mapping module** establishes the correspondence between the objects in the auxiliary graph and DO, which as a result must be populated. For this purpose, the mapping rules and information on priorities of data sources provided by the user are used.

**The population module** extracts data from data sources according to their structure stored in the supporting knowledge graph, performs consistency and redundancy checks and inserts them in the DO.

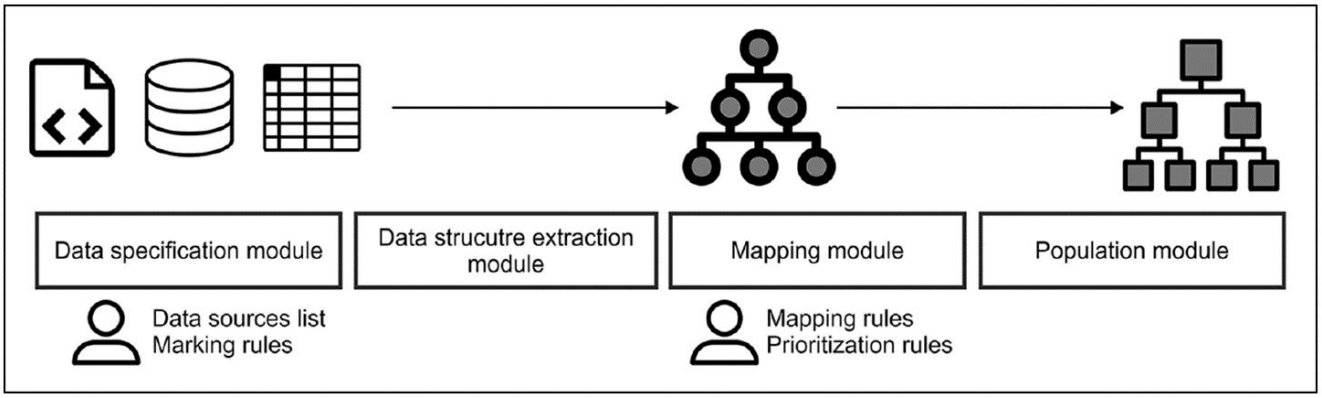


***Figure 7 —*** *Implementing the method with 4 modules*

The main modules are a data structure extraction module and a population module. They are implemented as a working prototype on Python and can serve as an extension for any ontology editor (the author mainly uses Protégé [Musen, 2015; https:/protege.stand.foredu/]). This architecture was chosen because the data specification module and the mapping module require manual input from the user, which can be performed in any ontological editor familiar to the user. In this way it is possible to use software familiar to the user.

### 2.2.2 Process of ontology population of the subject area from structured data sources

**Components necessary for the application of the method:**

1. Subject area ontology.
2. Ontology of data sources.
3. Modules described in the preceding paragraph.

**Domain ontology** used by the organization to solve certain problems or to answer predetermined questions. It is this ontology that will be populated by the method. To create ontology, expert knowledge, existing ontological and non-ontological sources of knowledge, competency issues and other information are used in accordance with the chosen ontology methodology. The development of subject area ontology goes beyond the method.

Basic **Data Source Ontology** is a meta-template for describing the structure and properties of data sources of the following types: spreadsheets, relational databases and XML documents. Figure 8 shows all of its significant classes and relationships (a notation from «Demystifying OWL for the Enterprise» [Uschold, 2018] was used for visualization. This ontology is based on specifications and ontologies found in the public domain. The following are the main classes of this ontology.

**DataSource** class is an upper level class that combines different types of data sources. Any data source has a name and location (path). The DataSource class includes subclasses such as Workbook (for table data sources), Database (for relational databases) and XML (for xml documents).

**Workbook** individuals is bound by a *hasPart* relation to individuals of the **WBSheet** class, which have the same relationship to the instance of the **WBTable** class, and these in turn are bound to the individuals of the **WBlumCon** class. All of these classes have their own names (*hasName* property). With this meta-template you can describe any working book of MS Excel spreadsheet editor.

**Database, DBTable**, **DBColumn** and **Key, PrimaryKey** and **ForeignKey** classes are used to describe the structure of relational databases**. XML, XMLElement** and **Atribute** classes are used to describe the structure of XML documents.

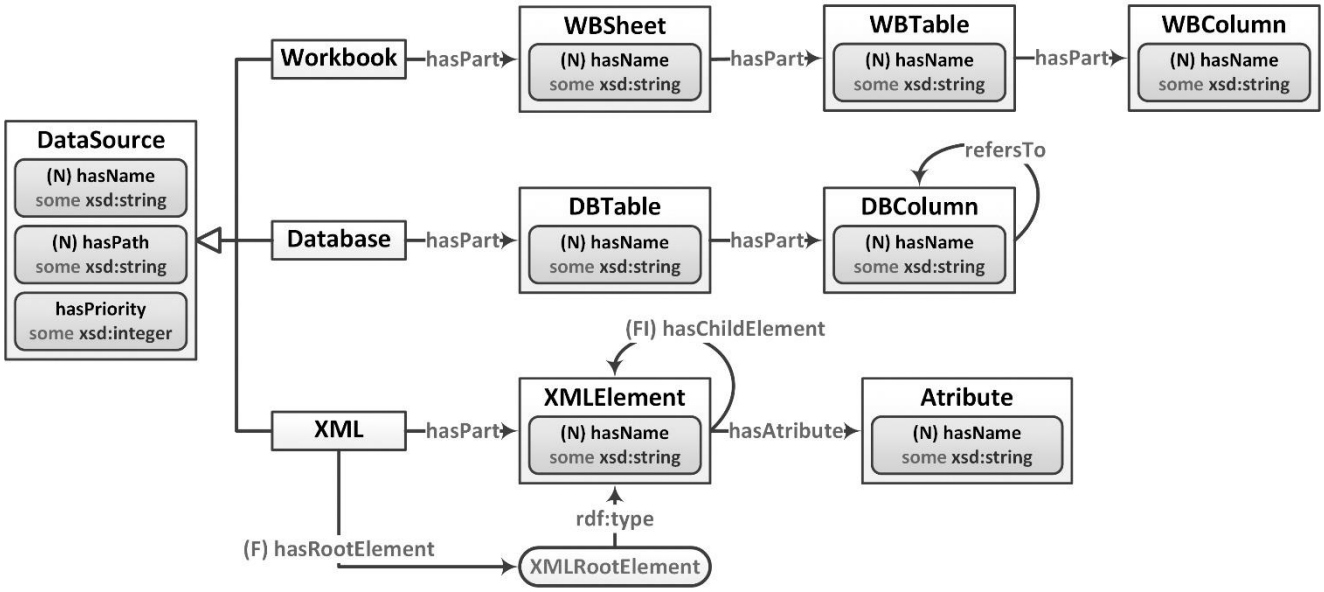


***Figure 8 —*** *Basic ontology of data sources*

**DataSource** subclasses (one for each type of data source), as well as other classes and ratios required to describe meta-structures of other types of data sources This ontology also specifies the necessary annotation properties that are used to describe the mapping of an auxiliary graph in DO in step 4.

**Population process:** the process of the ontology population of the subject area includes five steps (Figure 9). For each step, the main process as well as input and output are briefly described.

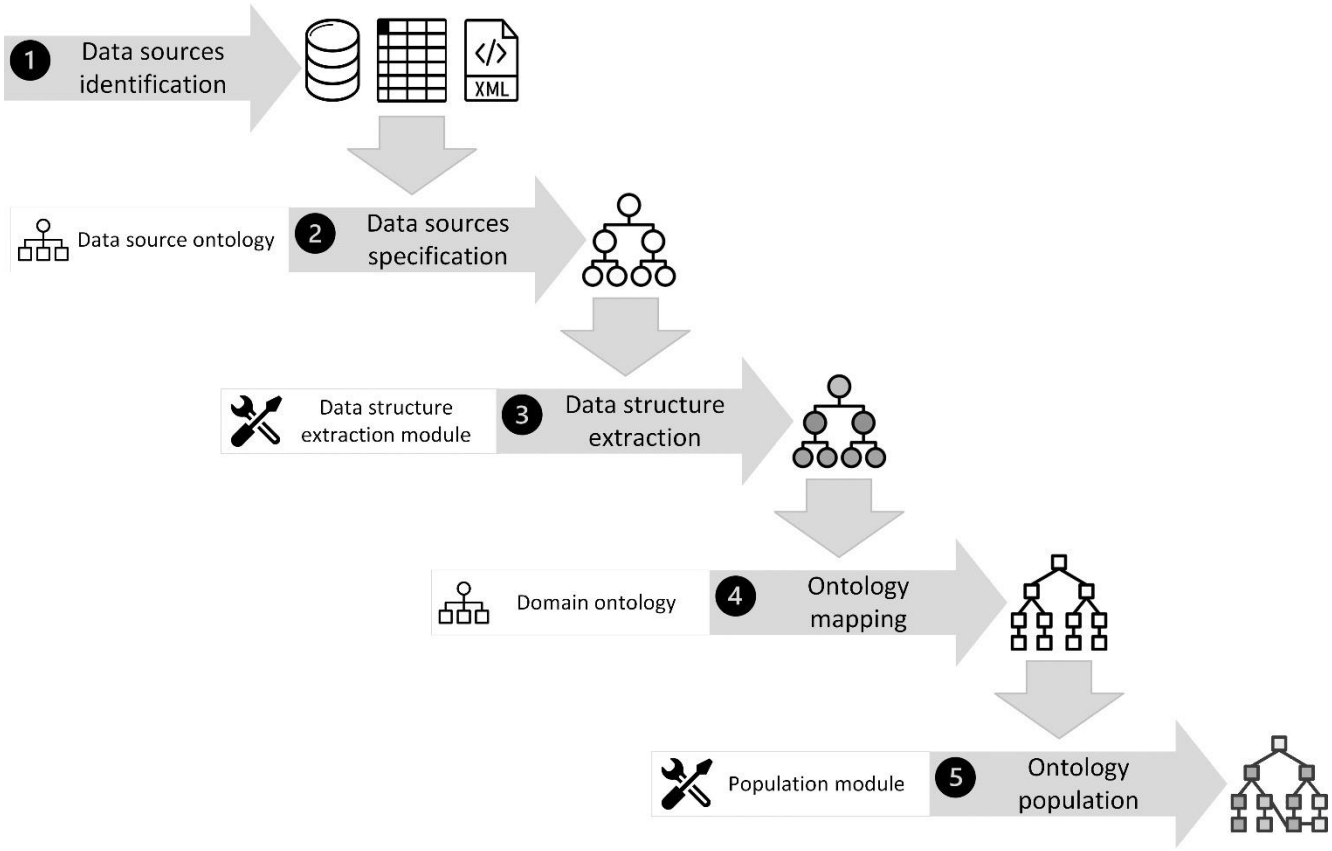


***Figure 9 —*** *Schematic of the process of the ontology population of the subject area*

**Step 1**: Identification of data sources from which ontology will be populated. At this stage, all data sources relevant to the subject area of the populated ontology should be identified (i.e. the data sources needed to solve the problems and answer the questions for which ontology has been created). Their identification is also outside the scope of the method - the method does not limit data types, as it can process data in different structured formats (e.g., XML documents, spreadsheets and databases).

If it is necessary to integrate data from poorly structured spreadsheets or to simplify further operations requiring manual input from the user, supporting tables are created. Power Query, a data connection technology available in the latest versions of MS Excel and MS Power BI, is used to create supporting tables. With this technology, data from different sources can be combined and their structure transformed as necessary.

***Input:*** Expertise, description of organizational processes, data warehouse.

***Output:*** List of data sources from which DO will be populated.

**Step 2**: Data source specification. A specification is a description of data sources. The data specification requires manual input of the user. The ontology of data sources (described earlier) is used as a meta-template to describe the structure and characteristics of data and to further import into the ontology of the subject area - thus there is no need for a specialized mapping language. For each data source it is necessary to create an instance of the corresponding subclass **DataSource**, and *hasName* (data source name) and *hasPath* (data source path) properties must be populated in. This step should

also provide information on the priorities of data sources if they are more than one - the property *hasPriority* (for resolving conflicts during ontology population).

If it is necessary to integrate data from sources of different type from the most widely used (spreadsheets, relational databases and XML documents (already described in the basic DSO), the basic ontology of data sources is extended.

If «non-standard» types are identified in step 1 for which supporting tables cannot be constructed, then **DataSource** subclasses (one for each «non-standard» type) must be added to the DSO in this stepthe copies of which will correspond to the data sources of the relevant type, as well as all classes and properties necessary for the description of all «non-standard» types of data sources identified in the first step. It is also necessary to expand the classes of the data structure extraction module and the population module, which will be responsible for processing data from «non-standard» sources.

***Input:*** list of data sources defined in step 1; base DSO.

***Output:*** Single file with ontological description of identified data sources.

**Step 3**: Extraction of Data Structure. The structure of data from different data sources is extracted and combined into a single structure (automatically by means of the data structure extraction module).

***Input****:* a file describing data sources from step 2; data sources; a data structure extraction module.

***Output*****:** An auxiliary knowledge graph created from source structures and a meta-template to describe them.

**Step 4**: Set Mapping Rules. The auxiliary knowledge graph created in the previous step differs from the ontology of the subject area in at least two aspects. First, the instance names and properties values in the auxiliary knowledge graph are taken from the data and differ from the names in the ontology of the subject area. Second, the structure of the auxiliary graph (and data sources) differs from the structure of the ontology of the domain. To properly import data, a comparison between DO and the DSO based auxiliary knowledge graph is required. This comparison is performed by the Knowledge Engineer together with the Subject Domain Expert by creating mapping rules using a set of annotation properties from DSO. A detailed description of these annotative properties is given in the following paragraph «Annotative properties for the description of DSO mapping in DO».

***Input:*** Subject area ontology; auxiliary knowledge graph.

***Output:*** Mapping rules between the auxiliary knowledge graph and DO (defined in DO).

**Step 5**: Ontology population. In the previous step, classes in DO were related to corresponding entities in the auxiliary knowledge graph by annotation properties and their values. The instances in the knowledge sub-graph reflect the structure of the imported data, but the data itself is stored in the data sources described. To insert values into the DO, the population module processes annotation properties (which set the mapping of data source structures stored in the auxiliary knowledge graph to the DO concepts) and imports data into the DO.

***Input:*** Subject domain ontology containing mapping rules, auxiliary knowledge graph, data sources.

***Output***: A knowledge base; A log file with information about the data selected for insertion in case of conflict and other supporting information.

A hierarchy of ontologies is created during the population process (see fig. 10). A solid frame around a rectangle representing ontology means that inclusion in a higher level ontology is done by importing, and a dashed line shows that a change is being made in ontology itself (for example, adding annotative properties or instances). Since the ontology of the structure of data sources is imported into DO, it can be easily removed from DO by removing the import command.

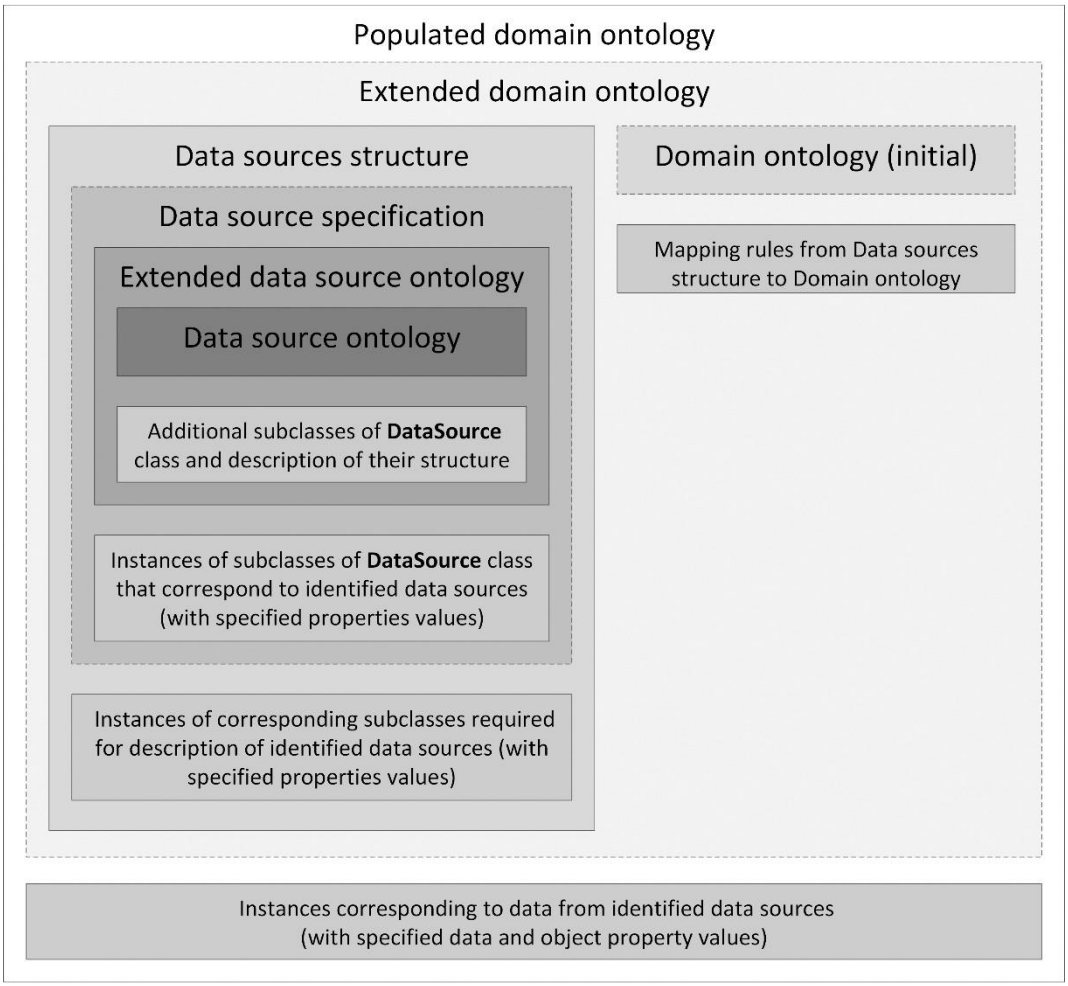


***Figure 10 —*** *Hierarchy of ontologies in the population process*

### 2.2.3 Annotative properties to describe the DSO mapping in DO

Annotative properties are used to describe population rules and are assigned to DO classes. In terms of the resulting ontology, its content is:

- Creating Class Copies

- Setting Object Properties Values (ObjectPropertyAssertion)
- Setting Slot Values (or Attributes) (DataPropertyAssertion)
- Annotation Job (AnnotationPropertyAssertion)

**Creating individuals**

Two annotation properties give instructions to the ontology population module to create instances of class: getInstancesFrom and combineInstancesFrom. Annotations are added to the created instances to indicate the value that caused this instance to be created, as well as the source of the value.

***getInstancesFrom***

The value of the property can be any instance from the DSO that belongs to the class that describes the structure of data sources objects that directly contain homogeneous data (such as WBColumn or Atribute). Each unique value generates a new instance in the annotated class, and its value becomes the value of the rdfs property:label of the generated instance. There are no precepts.

***combineInstancesFrom***

It usually has two or more values. The value of the property can be any class whose instances must be combined to create the object of the class to be described. As a result of processing this property, copies are created using the Cartesian product of copies of the corresponding classes. Created instances bind to source instances of properties specified by the nested annotation of useObjectProperty. Precepts: Copies of combinable classes must be created beforehand.

**Setting Object Properties**

The makeReferenceTo property is used to describe the binding of instances by a relation (using an object property). The value of the property can be any class containing instances previously created by the getInstancesFrom or combineInstancesFrom properties, which must become the value of the object properties created. These object properties are annotated by the rdfs:label property, the value of which indicates the data source containing the generated relation information. An object property reference is set with the useObjectProperty embedded annotation property. Precepts: Examples of bound classes must be created beforehand.

**Setting slot values**

The getValuesFrom property is used to specify attributes of instances of class. The value of the property can be any instance from DSO that belongs to a class that describes data source structure objects that directly contain homogeneous data (such as WBColumn or Atribute). Properties are annotated with the rdfs:label property that indicates the data source containing the generated relation information. The

property reference is set with a nested annotation property useDataProperty. Precepts: Copies of the annotated class must be created beforehand.

### 2.2.4 Illustration of the ontology population process by example

Prerequisites:

- Ontology of the subject area. Creating DO is outside the scope of the method. An example of DO to populate is shown in Figure 11.

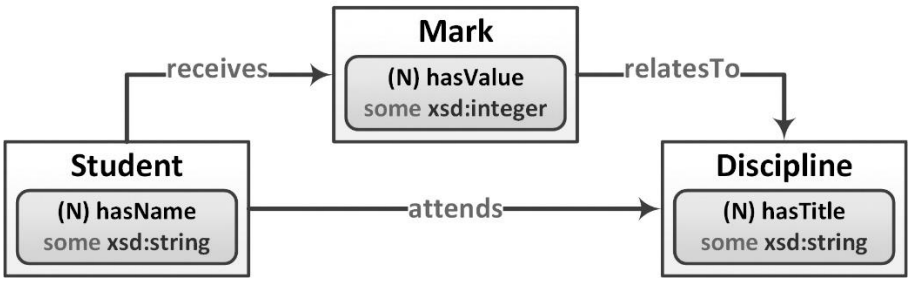


***Figure 11 —*** *Domain Ontology Fragment*

- The basic DSO is shown in figure 11.

**Step 1**: Identification of data sources. This step goes beyond the method and must be performed by the administrator according to his or her knowledge of the task. The example will use two different sources of data on student achievement: the table in figure 12 and the database fragment, the chart and content of which are shown in figure 13.

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | Educational program | Management | | | | | | | |
| 2 | | | Year of admission | 2017 | | | | | | | |
| 3 | | | | | | | | | | | |
| 4 | | | | | | Semester 1 | | | Semester 2 | | |
| 5 | № | Surname | Name | Middle name | Group | Maths | Business History | Macroeconomics | Statistics | Marketing | Microeconomics |
| 6 | st061234 | Aristov | Ivan | Ivanovich | 17.B01 | 5 | 3 | 4 | 5 | 4 | 5 |
| 7 | st061345 | Koval | Elena | Olegovna | 17.B04 | 4 | 5 | 5 | 4 | 5 | 4 |

***Figure 12 —*** *Example of student performance table*

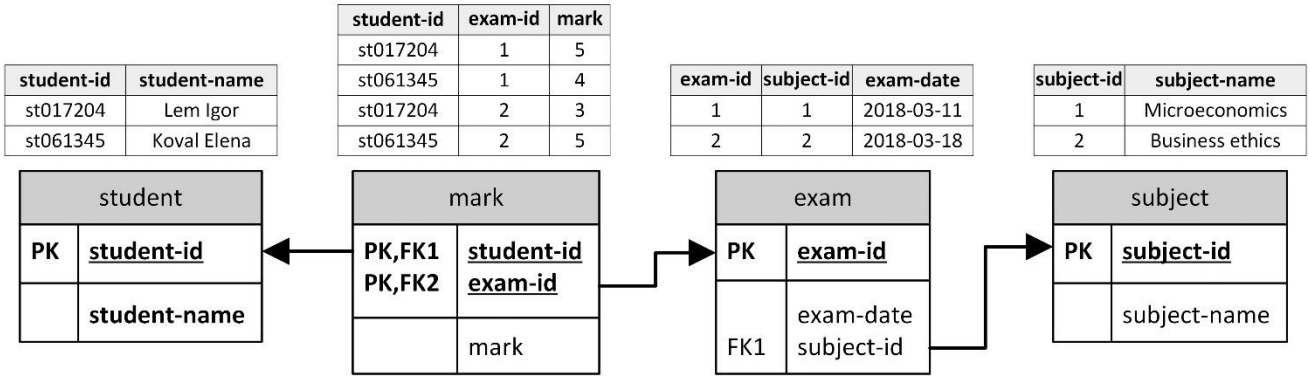


***Figure 13 —*** *Student Achievement Database Fragment*

Using PowerQuery technology, the table shown in Figure 12 has been converted to the «direct» view shown in Figure 14. **The Education Program** and **Year of Admission** columns were not added because they do not provide essential information for the example, but if the objective was to compare the performance of students with different programs or streams, adding these columns would be easy to implement. Data from the database can also be converted to a similar table if desired for illustration, but it is not necessary.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | st | Name | Subject | Mark |
| 2 | st061234 | Aristov Ivan Ivanovich | Maths | 5 |
| 3 | st061234 | Aristov Ivan Ivanovich | Business History | 3 |
| 4 | st061234 | Aristov Ivan Ivanovich | Macroeconomics | 4 |
| 5 | st061234 | Aristov Ivan Ivanovich | Statistics | 5 |
| 6 | st061234 | Aristov Ivan Ivanovich | Marketing | 4 |
| 7 | st061234 | Aristov Ivan Ivanovich | Microeconomics | 5 |
| 8 | st061345 | Koval Elena Olegovna | Maths | 4 |
| 9 | st061345 | Koval Elena Olegovna | Business History | 5 |
| 10 | st061345 | Koval Elena Olegovna | Macroeconomics | 5 |
| 11 | st061345 | Koval Elena Olegovna | Statistics | 4 |
| 12 | st061345 | Koval Elena Olegovna | Marketing | 5 |
| 13 | st061345 | Koval Elena Olegovna | Microeconomics | 4 |

***Figure 14 —*** *Table of learning achievement after conversion*

**Step 2**: Data source specification. This step adds copies describing the data sources identified in the first step. For the example described, an instance of the **Workbook** class and an instance of the **Database** class are added, with values for hasName, hasPath and hasPriority attributes specified. You can also set a queryText property value for a **Database** instance, which will be used to extract data from a database if you decide not to create a supporting table for it in the first step. The value of the property queryText for the example described:

SELECT * FROM mark LEFT JOIN student USING(`student-id`) LEFT JOIN exam USING(`exam-id`) LEFT JOIN subject USING(`subject-id`);

**Step 3**: Extracts the data structure. Executes automatically. In this step, the structure (for example, column names) of the marked data sources are inserted into the DSO as individual instances of the corresponding classes. As a result, all copies and the relationships between them are created, describing the structure of the data sources specified in the previous step. It is important to note that the data itself is in the original data sources and is not imported into the IIAs. However, the links between the structure (such as the column name) and the data (values in the column) are preserved and will be used during the ontology population phase. The output of the data structure extraction module is the structure shown in Figure 15. The values in the figure are the structure of the data sources (names of tables and columns) and not the data itself.

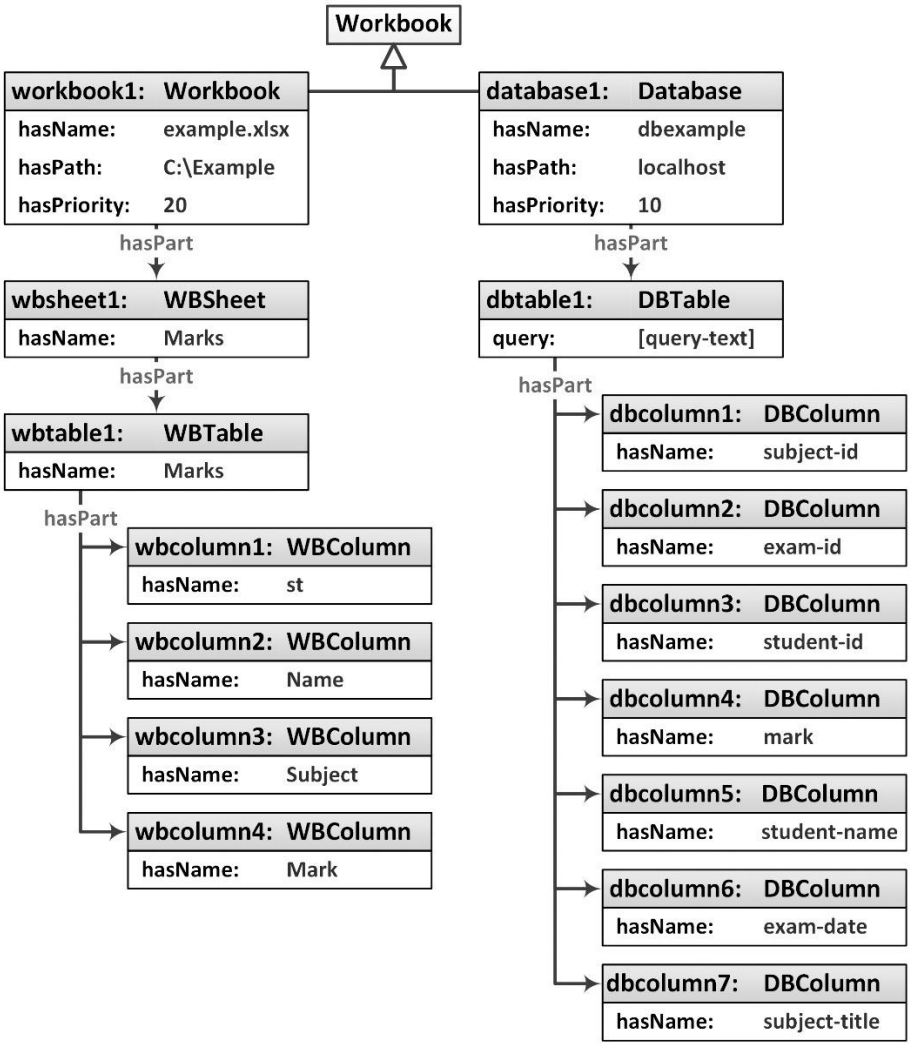


***Figure 15 —*** *Step 3 result: Description of the data structure*

**Step 4**: Set the mapping rules. Set the following annotation properties:

Class **Student**:

getInstancesFrom: wbcolumn1

getValuesFrom: wbcolumn2

useDataProperty: hasName

getInstancesFrom: dbcolumn3

getValuesFrom: dbcolumn5

useDataProperty: hasName

makeReferenceTo: Discipline

useObjectProperty: attends

makeReferenceTo: Mark

useObjectProperty: receives

Class **Discipline**:

getInstancesFrom: wbcolumn3

getValuesFrom: wbcolumn3

useDataProperty: hasTitle

getInstancesFrom: dbcolumn7

getValuesFrom: dbcolumn7

useDataProperty: hasTitle

Class **Mark**:

combineInstancesFrom: Student

combineInstancesFrom: Discipline

getValuesFrom: wbcolumn4

useDataProperty: hasValue

getValuesFrom: dbcolumn4

useDataProperty: hasValue

These properties fully describe the transformation of the structure of data sources into the structure of ontology. The values of the properties correspond to the instances from the DSO (that is, the names of the data structure elements that must be loaded into the subject domain ontology).

**Step 5**: Ontology population.

The population process specified in the previous step by the annotative properties is performed sequentially in the sequence shown in Figure 16. The ontology result is shown in fig.17.

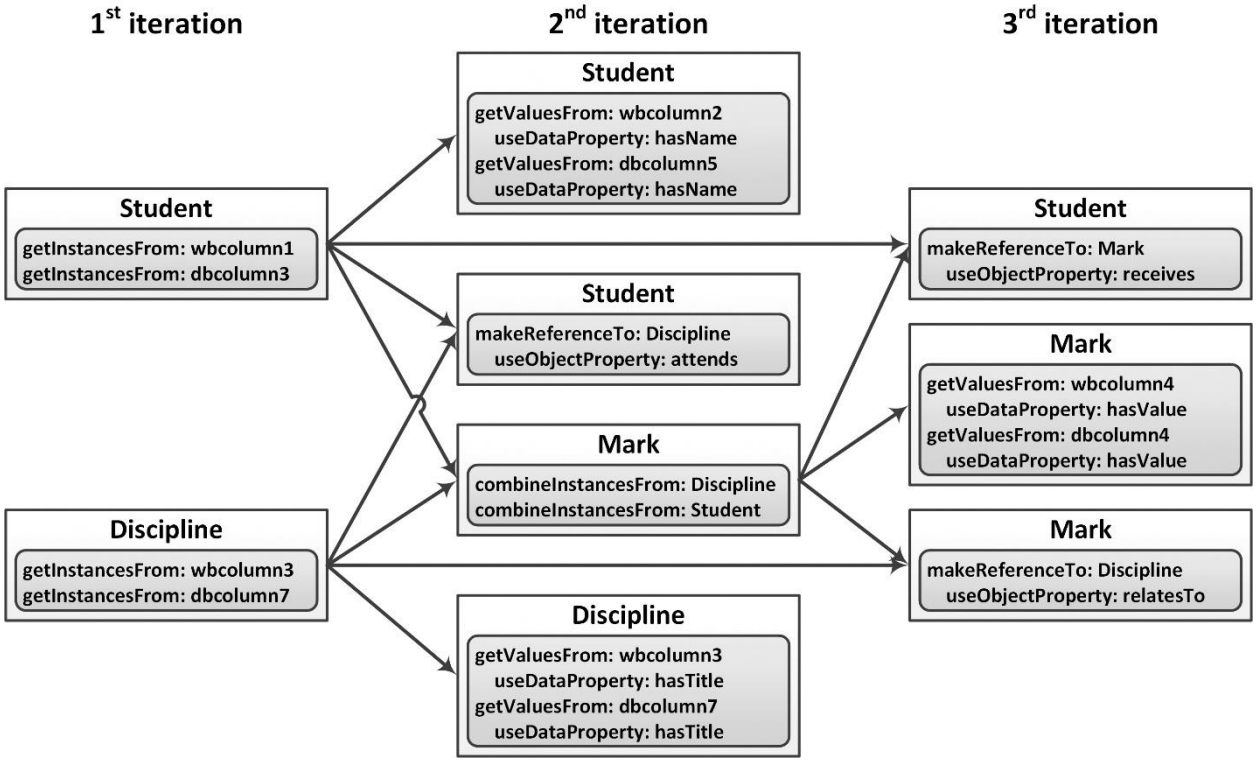


***Figure 16 —*** *Order of ontology population for the example described*

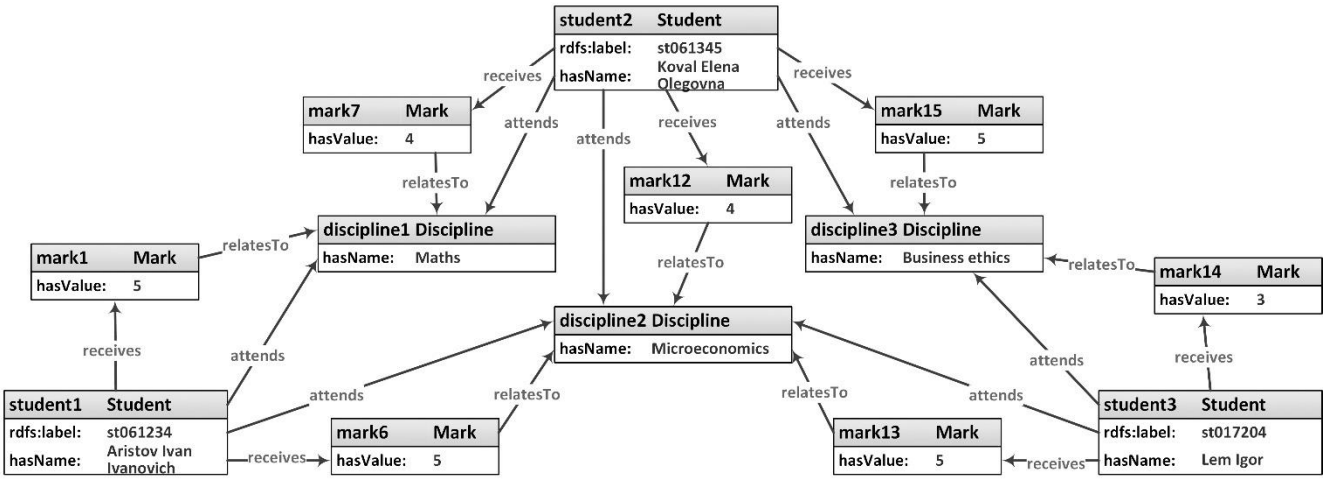


***Figure 17 —*** *Result of ontology population*

## Conclusions on chapter 2

Chapter 2 deals with the development of a new method of ontology population and the architecture of the software prototype. Although modern research offers many approaches to data integration in ontology [Lubani, Noah, Mahmud, 2019; Nederstigt et al., 2014; Petasis et al, 2011; Valarakos et al., 2004], most of them have a number of limitations. Thus, the main shortcomings of the existing solutions are:

1. Need to use and therefore learn syntax of several specific languages to describe mapping. In order to work effectively, it will be necessary to create a convenient tool for each of the languages used.
2. Narrowness of methods: a large number of different methods have been proposed to integrate data into ontology, but each method works for only one type of data source.
3. Lack of guidance on integrating data from multiple sources at the same time. If there is a need to integrate data from, for example, a database and an XML document into a single ontology, then using existing methods will have to act sequentially. In this case the first step is the population of the initial ontology with data from the database, the second is the population of the received ontology with data from the XML document. Since the data in the sources are subject to change, this procedure needs to be repeated regularly, resulting in unnecessary costs.
4. Inconsistencies at the integration stage of data from different sources. Since the authors of the above methods focus on the integration of data from a single source, the issue of resolving such contradictions is not addressed.
5. Ambiguities how the connection to the data source is established, how authentication takes place, etc..
6. Lack of software guidance if companies can use proprietary (private) storage formats and new ones may appear in the future.

The analysis of these problems led to the conclusion that a new method should be created, allowing the integration of data from different types of sources (for example, databases, spreadsheets and XML files). Creating such a method reduces the laboriousness of populating and enriching ontologies. The method uses accumulated data sets regardless of their form of storage and presentation.

The proposed methodology is designed to perform the same type of operations or to perform the same tasks requiring knowledge of rules and procedures, as well as to compare information from different sources. Such tasks arise in enterprises, educational institutions and organizations with different departmental affiliations within the framework of digital documentation. The proposed methodology could form the basis of systems to support knowledge-based decision-making.

For example, in e-commerce systems, recommendatory systems have become widespread. Different approaches are taken to the list of recommendations: collegial recommendations, content-based recommendations, and knowledge-based recommendations. The latter approach could be implemented, inter alia, by the proposed method. Ontology includes knowledge of selection criteria, object compatibility etc. The characteristics of existing objects can be stored in a database or separate files. Information about user preferences can be provided in the request. Upon request, data about objects

are «dragged» into ontology, analyzed according to the laid down rules and on the basis of the analysis made a recommendation for a specific user is formed.

The novelty of the proposed method is the use of auxiliary Data Source Ontology (DSO), which can be extended by adding additional types of data sources. The use of DSO is complemented by the development of requirements for creating mapping rules that allow to bind the DSO and a specific population of ontology of a subject area (DO) to be populated with instances. The proposed method is devoid of all the defects at the beginning of the paragraph (the first four are not available due to the algorithm, the last two problems are solved by a mechanism).

The application of the developed METEOR method (MetaDology and Technology of Ontology Population based on integration with heterogeneous structured data sources) can be described by an algorithm consisting of 5 steps, two of which are performed automatically:

1. Identification of data sources.
2. Data Source Specification by auxiliary DSO.
3. Extraction of data structure from data sources in DSO.
4. Set Mapping Rules in DO.
5. Population of DO.

The key step of the DO population algorithm is to set mapping rules. It is done using the annotation mechanism built into the OWL. New annotation properties have been introduced: 4 basic and 2 auxiliary properties (see 2.2.3 for more details). This approach eliminated the need for additional instrumentation support for the METEOR method, as annotation can be done with any ontology editor.

A new prototype architecture was also developed to implement the proposed METEOR method. The architecture is based on the well-known architecture of ETL-systems. A peculiarity of the proposed architecture is that the receiver of the data is not a repository or database, but ontology or a knowledge base.

The developed prototype implements the created METEOR method and allows to populate the ontological knowledge base from heterogeneous sources. The software prototype consists of 4 modules supporting processes for data source specification, data structure extraction, mapping rules, and DO content. The entire ontology population process using the METEOR method is illustrated by a conditional example. The next chapter focuses on the practical application of the METEOR method to populate 4 ontologies from different subject areas (medical diagnosis of orphan diseases, assembly production, empirical research data retrieval and educational administration).

# CHAPTER 3 PRACTICAL APPLICATION OF ONTOLOGY POPULATION METHOD

## 3.1. EMPIRION Research Processing Ontology and Features

The method of ontology population proposed in the thesis has been tested in the framework of the project «Formation of knowledge bases on the basis of empirical research data: ontological approach (EMPERION)» (RFBR 20-07-00854). Ontology was developed to describe empirical research data (hereinafter, empirion) [Leshcheva, Begler, 2020].

- This ontology ensured that the basic traditional principles of storage and access to data were respected [Jacobsen et al. 2020a; Jacobsen et al. 2020b; Wilkinson, 2016]:
- Findability: Ensure that data and metadata are found;
- Accessibility: provide access to data;
- Interoperability: enable data integration with other data using commonly accepted metadata schemes;
- Reusability: enable reuse of data.

Empirion ontology stores both the general properties of the data set, including date of creation, name, author, description, etc., and its built-in metadata, such as column headers with variables, their range of values and coding methods, units of measurement, etc.

The basis of empirical data sets are the variables usually represented as data columns. In terms of ontology structure, all variables can be divided into 3 types:

a. Variables whose values are given by some list such as{correct, incorrect};

b. Variables with numerical values in some units of measure, with the units themselves specified in the metadata file;

c. Variables whose values are any (non-fixed) rows or dimensionless numbers that are labels rather than actual numeric values (for example, the respondent's number).

EMPIRION ontology uses a three-level representation of variables:

1. Instances of class **Variable** are variables from the data sets described.
2. Instances of class **Representation** describe variants of representation of the described variables. For variables of type (a), an instance of class **Representation** is created for each set of variable values associated with each of the possible values. The values themselves are instances of the **RepresentationValue** class. For variables of type (b) an instance of class Representation is created associated with units of measure that are instances of class **Metadata -> Variable Sensemaker -> Measurement Unit**. For variables of type (c), the

instance of class Representation is of a formal nature and is created for uniformity of queries to the resulting knowledge base.

3. Instances of the **RepresentedVariable** class are some idealized measurable variable, such as sex or age (independent of its representation in specific data sets). For variables of type (a) some «reference» set of values is selected and links are created with these values, which are instances of the class **RepresentedVariableValue**. This is necessary in order to subsequently match the values of a variable encoded differently in different data sets.

Figure 18 presents the structure of the main classes of EMPIRION ontology in the form of a conceptual map.

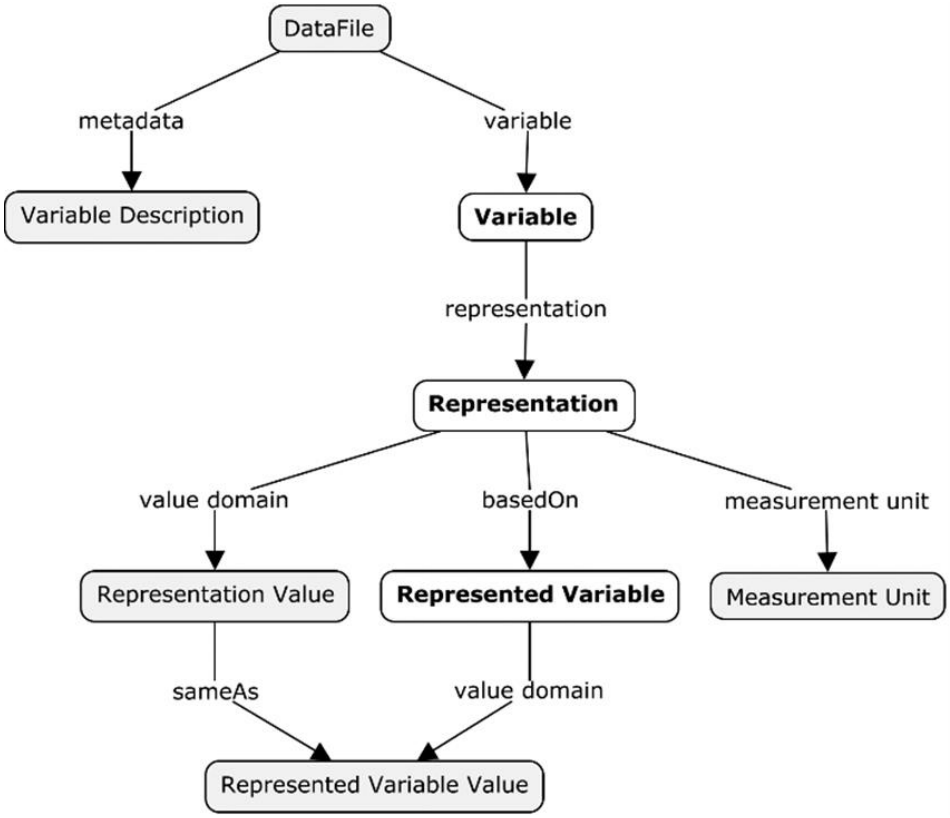


***Figure 18 —*** *Conceptual map of the main classes of EMPIRION ontology*

The new METEOR method, described in Chapter 2, was used to populate ontology EMPIRION. The pre-requisitions of the method are the creation of two ontologies: populated DO (EMPIRION) and DSO (the basic DSO is also named METEOR). The population process method METEOR (see chapter 2) was used. It consists of 5 steps.

**Step 1:** Identification of data sources. Consider that the data set contains the following files:

1. The actual test table file where each column represents the values of a variable;

2. Metadata file is description of experiment and variables.

On the basis of these files, a file was created for each data set containing information on which classes of empirion ontology the variables described are to. All files created in step 1 were placed in one folder and converted to a table view with Power Query. The resulting table has the following structure:

1. DataFile column contains the name of the file with the results of the experiment. The unique values from this column will be reflected in the instance of class Data -> Data File.
2. VariableDescription column contains the name of the metadata file. The unique values in this column will be reflected in the instance of class Data -> Metadata File -> Variable Description.
3. Variable column contains variable names.
4. Representation column contains all possible values for variables of the form (a), whose options are limited to some list.
5. RepresentedValue column contains «reference» values for variables of type (a).
6. MeasureUnits column contains measurement units for variables of type (b). Unique values from this column will be reflected in Metadata -> Variable Sensemaker -> Measurement Unit.
7. A block of columns corresponding to subclasses of Variable whose instances will be the variables from the data set.
8. A block of columns corresponding to subclasses of Representation, whose instances will be different representations of variables from the data set.
9. A block of columns corresponding to the subclasses of RepresentedVariable whose instances are variables independent of the specific data sets and the representation of variables there.

**Step 2:** Data source specification. In this step, the auxiliary ontology empirion_struct was created, importing meteor IEDs. Next, an instance of the Workbook class was created with hasPath attributes (path to file from step 1) and hasName (file name from step 1).

**Step 3:** Extracting a data structure. With the help of the data structure extraction module, the ontology empirion_struct was populated with instances describing the file structure from step 1.

**Step 4:** Set the mapping rules. At this stage, *empirion_struct* ontology was imported into empirion ontology, followed by the following mapping rules: for class Data -> Data File:

- getInstancesFrom DataFile, where DataFile is an instance of the WBColumn class, representing the corresponding column from the table created in the first step.
- RemakeferenceTo Variable Description useObjectProperty metadata, where Variable Description is the name of the class and metadata is the name of the relationship that must be used to create links.
- Several rules of the form makeReferenceTo <Variable SubClass Name> useObjectProperty variable, where instead of <Variable SubClass Name> the name of a

specific subclass of Variable is specified, and variable is the name of the relation that must be used to create a connection.

> For Data -> Metadata File -> Variable Description:

getInstancesFrom VariableDescription, where VariableDescription is an instance of the WBColumn class representing the corresponding column from the table created in the first step.

- For **Variable** subclasses:

o getInstancesFrom <WBColumn Variable Instance Name>, where <WBColumn Variable Instance Name> is an instance of the WBColumn class, representing the corresponding column from the table created in the first step.

o RemakeferenceTo <Representation SubClass Name> useObjectProperty representation, where <Representation SubClass Name> is the name of a specific subclass of representation, and representation is the name of the relation to be used to create connections.

- For subclasses of **Representation** class:
  - o getInstancesFrom <WBColumn Representation Instance Name>, where <WBColumn Representation Instance Name> is an instance of the WBColumn class, representing the corresponding column from the table created in the first step.
  - o RemakeferenceTo <Represented Variable SubClass Name> useObjectProperty based on, where instead of <Represented Variable SubClass Name> the name of a specific subclass of Represented Variable is specified, and the name based on is the name of the relation to be used to create connections.
  - o makeferenceTo RepresentationValue useObjectProperty has value domain.
  - o makeferenceTo Measurement Unit useObjectProperty has measurement value.
- For Represented Variable subclasses:
- getInstancesFrom <WBColumn Represented Variable Instance Name>, where <WBColumn Represented Variable Instance Name> is an instance of the WBColumn class, representing the corresponding column from the table created in the first step.
- makeferenceTo RepresentedVariableValue useObjectProperty has value domain.
- For **RepresentationValue** class:
  - o getInstancesFrom Representation, where Representation is an instance of the WBColumn class, representing a corresponding column from a table created in the first step.

- RemakeferenceTo RepresentedVariableValue useObjectProperty sameAs, where RepresentedVariableValue is the name of the class and sameAs is the name of the relationship to be used when creating connections.

- For the RepresentedVariableValue class:
  - getInstancesFrom RepresentedValue, where RepresentedValue is an instance of the WBColumn class representing the corresponding column from the table created in the first step.
- For Metadata -> Variable Sensemaker -> Measurement Unit:
  - getInstancesFrom MeasureUnits, where MeasureUnits is an instance of the WBColumn class that represents the corresponding column from the table created in the first step.

Annotation properties of getInstancesFrom, makeReferenceTo and useObjectProperty, detailed in Chapter 2, were used to define the mapping rules.

**Step 5:** The ontology is populated. This is done automatically with the population module. Figures 19-21 show the fragments of structures created for variables of different types.

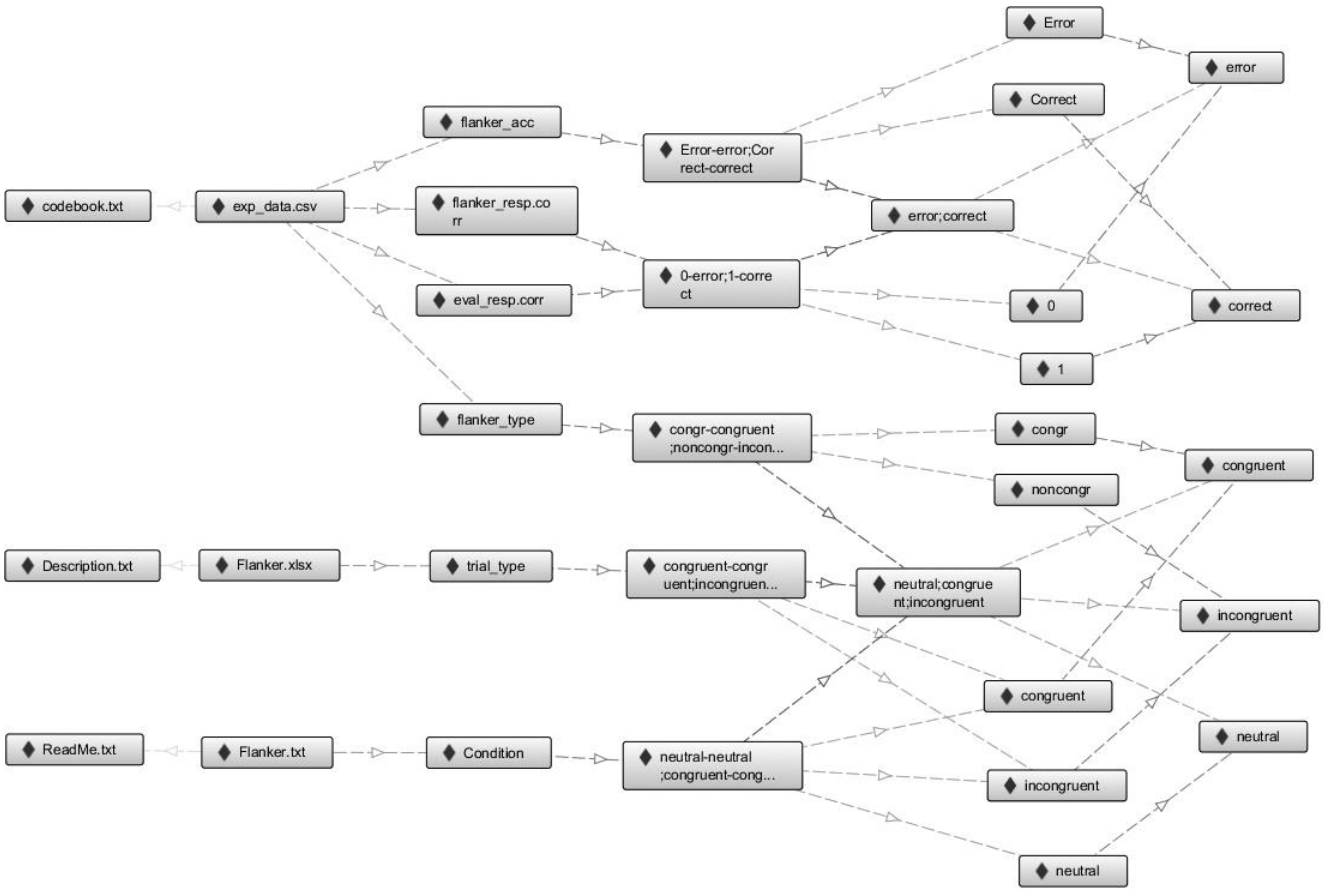


***Figure 19 —*** *Example of a structure created for variables of type (a)*

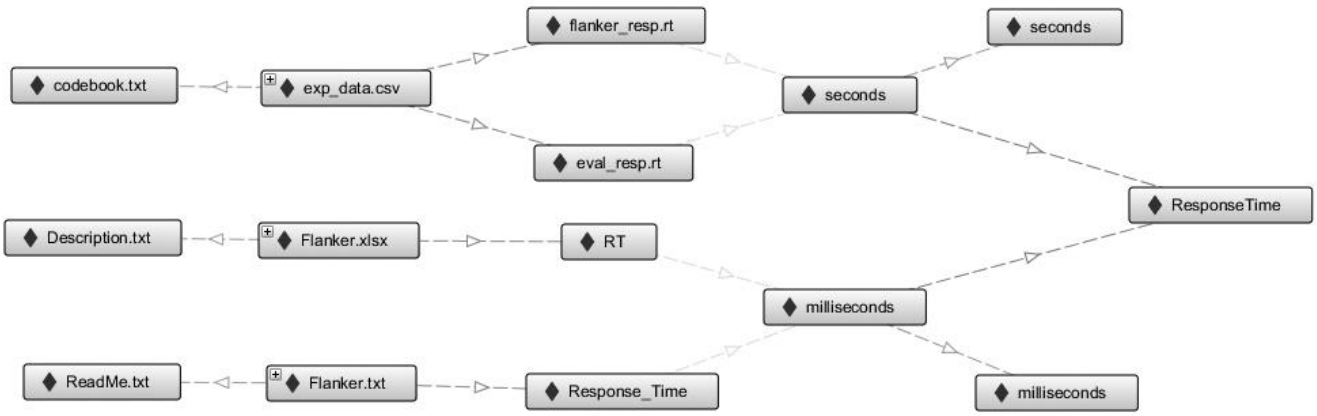


***Figure 20 —*** *Example of a structure created for variables of type (b)*


***Figure 21 —*** *Example of a structure created for variables of type (c)*

Once the ontology is complete, it is possible to execute SPARQL queries to search for datasets with given characteristics. For example, the query below searches for all datasets that measure the variables ResponseTime and StimulusCongnce.

```
PREFIX emp: <http://www.tempuri.org/empirion#>
PREFIX ddi: <http://rdf-vocabulary.ddialliance.org/discovery#>
SELECT DISTINCT ?subject
    WHERE {
        ?subject ddi:variable ?somevar1.
        ?somevar1 ddi:representation ?somerepresentation1.
        ?somerepresentation1 ddi:basedOn emp: responsetimerepresentedvariable1.
        ?subject ddi:variable ?somevar2.
        ?somevar2 ddi:representation ?somerepresentation2.
        ?somerepresentation2 ddi:basedOn emp: stimuluscongruencerepresentedvariable1
    }
```

### 3.2. Ontomorf Ornamental Disease Ontomorf Ontomorf Ontology

Ontological approach is widely used in various medical intellectual systems and applications [Gribova V. V. etc. 2018; Kleshchev, Moskalenko, Chernyakhovskaya, 2005; Ivanović, Budimac, 2014]. Medical ontologies [Kobrinsky, 2018; Kobrinsky et al. 2019] have been proposed for the diagnosis of orphan (rare) diseases. The analysis of the subject area and the work with experts have made it possible to propose algorithms in which the diagnosis is made on the basis of the modality scales of

the features with their coefficients and formulas for the complex assessment of the combination of confidence factors (expression and manifestation).

The analysis identified four age groups of patients:

(1) up to 1 year;

(2) 1-3 years;

(3) 4-6 years;

(4) over 6 years of age.

Then, for each age group, the coefficients of modality, manifestation and expression of each characteristic (symptom) were determined.

Modality ($M_{ik}$, where i is the number of the trait and k is the number of the age period) characterizes the relevance of the traits of disease in each age period. According to the opinion of the experts, for each disease the topics were divided into main, necessary and secondary, and coefficients were determined for each group of topics. If the feature cannot be present in this age range, the modality was assumed to be zero (Mik=0).

Manifestation ($n_{ij}$, where i is the number of the topic and j is the number of the age period) describes the level of trust (confidence) of experts that a given characteristic of a particular disease manifests (is detected) precisely at a given age (age group). I.e. for each age group, experts determined the significance of the manifestation, with the sum of these values for the same disease for all age groups not exceeding 1. Cumulative demonstration confidence factor ($m_{ik}$, where $i$ — feature number, a k is the number of the age period) calculated by formula:

$$m_{ik} = \sum_{j=1}^{k} n_{ij},$$

where $i$ — feature number, a k is the number of the age period.

The expression ($s_{ik}$, where i is the number of the topic and k is the number of the age period) characterizes the experts' confidence that the topic occurs in a particular age group with a certain degree of intensity. The change in age expression indirectly indicates the rate of development of the disease (symptoms).

The following formula was used to provide a comprehensive quantification of the attribute in a given age period:

$$P_{ik} = M_{ik} * m_{ik} * s_{ik},$$

where i is the sign number,

k is the age period,

$P_{ik}$ — Quantification of disease (symptom) for this age period,

$M_{ik}$ — modality of topic for this age period,

$m_{ik}$ — Cumulative demonstration confidence factor for age $k$,

$s_{ik}$ — confidence factor for the same age period.

The formula was used to provide an integrated measurement of topic sum for each case being diagnosed:

$$I = \sum_{i=1}^{n} P_{ik},$$

where n is the number of features,

k is the age of the patient,

I - Integrated Feature Assessment

$P_{ik}$ is the quantification of the disease i for the age period k.

To implement the proposed approach, *ontomorf* ontology was created, the main elements of which are presented in Figure 22.

In the left part are classes in which «constant» information of a knowledge base is contained. This information is assumed to be stable or changing very rarely in the course of the knowledge base.

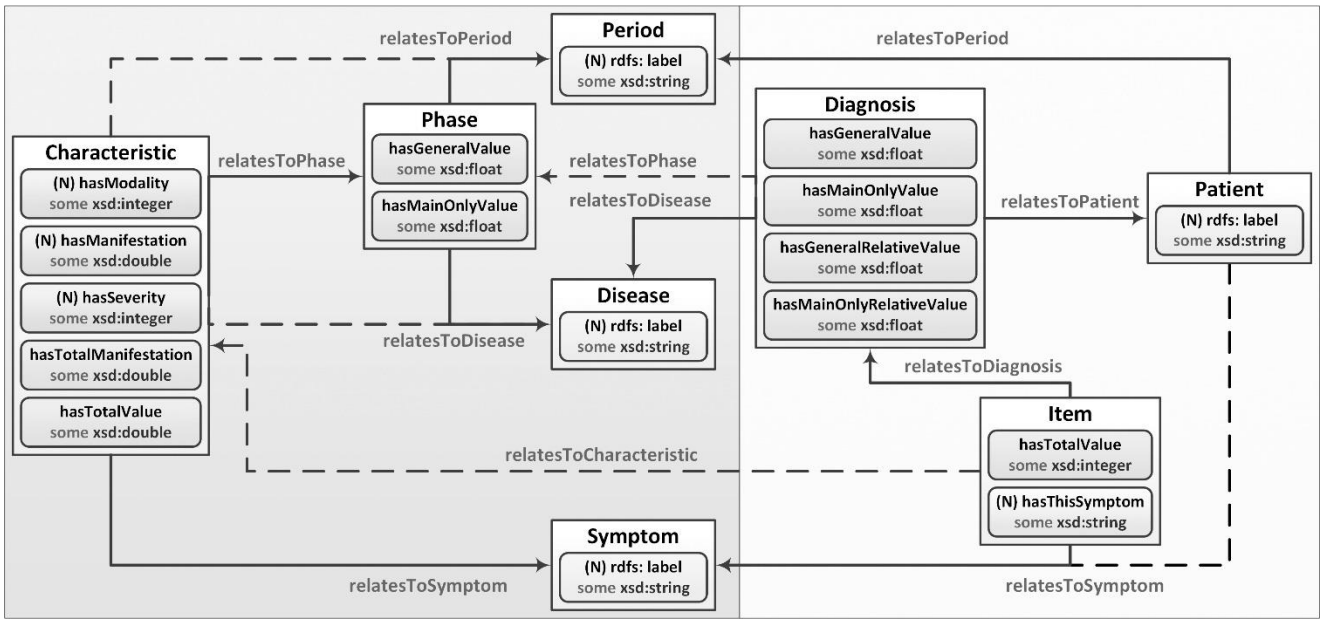


***Figure 22 —*** *Example of a structure created for variables of type (b)*

- The **Disease** class describes the diseases for which the formed knowledge base is intended.
- The **Symptom** class is used to describe the characteristics of the disease of the diagnosed group.
- **Period** class defines age periods: up to 1 year, 1-3 years, 4-6 years, over 6 years.
- Class **Characteristic:** Each element of a class corresponds to one characteristic (symptom) of a particular disease in a given period and contains the values of modality ($M_{ik}$), manifestation ($n_{ik}$) and expression ($s_{ik}$)) of that characteristic for that disease in that

period. Elements of this class are also used to calculate the cumulative manifestation confidence factor ($m_{ik}$), the complex quantification of the feature ($P_{ik}$) and the total values of the elements of **Phase**.

- Class **Phase**: contains the total (reference) values for each disease in a given period.

The right part contains «variable» information about patients.

- **Patient** class contains patient identification data, age, symptoms.
- Class **Diagnosis**: This class produces one copy for each patient for each disease. Contains total values for each disease for a given patient. These values are compared with reference values from a corresponding instance of **Phase** class to form a diagnosis hypothesis.
- Class **Item**: An auxiliary class for calculating the value of instances of **Diagnosis**.

Pink-lit elements and dotted-arrow relationships are determined on the basis of other available information. There are 24 output rules.

The author's METEOR method for population *ontomorf* ontology has been used several times:

1) at the stage of creation of the «basic» (permanent) part of the knowledge base (once);

2) during the exploitation phase to provide information about patients.

The following details describe the process of population of the «basic» part of the knowledge base.

**Step 1**: Identification of data sources. The data for population of the «basic» part of the knowledge base were contained in two Excel files symptoms.xlsx and modalities.xlsx. The structure of symptoms.xlsx is shown in Figure 23.

| | Hurler | | | | | | | | H-Scheie | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | under 1 year | | 1-3 years | | 4-6 years | | ler than 6 ye | | under 1 year | | 1-3 years | | 4-6 years | | ler than 6 ye | |
| **Growth retardation** | 0,7 | 3 | 0,1 | 5 | 0,1 | 7 | 0,1 | 10 | 0 | 0 | 0,7 | 7 | 0,1 | 7 | 0,1 | 8 |
| **Short neck** | 0,1 | 2 | 0,1 | 3 | 0,3 | 7 | 0,4 | 9 | 0,1 | 3 | 0,1 | 3 | 0,2 | 6 | 0,2 | 6 |
| **Macrocephaly** | 0,7 | 2 | 0,1 | 4 | 0,1 | 5 | 0,1 | 8 | 0,4 | 2 | 0,1 | 3 | 0,1 | 4 | 0,1 | 4 |

***Figure 23 —*** *Example of file structure with manifestation and expression coefficients*

The first column contains all the signs of the diseases being diagnosed, and the next block for each disease and for each age group shows the values of the factors of confidence and expression.

The modality is defined by a hierarchical list of the following type (see fig. 24).

| | | | | |
|---|---|---|---|---|
| Mucopolysaccharidosis Type IH (Hurler) | | | | |
| | under 1 year | | | |
| | | (https://rarediseases.info.nih.gov/diseases/12559/mucopolysaccharidosis-type-ih) | | |
| | | | **The main signs (80% -99% of patients have these symptoms):** | |
| | | | | Short neck |
| | | | | Rough facial features |
| | | | | Hepatomegaly |
| | | | | Splenomegaly |
| | | | | Hernia |
| | | | **Necessary signs (30% -79% of patients have these symptoms):** | |
| | | | | Growth retardation |
| | | | | Macroglossia |
| | | | | Hearing loss |
| | | | | Corneal opacity |
| | | | **Secondary signs (less than 30% of patients have these symptoms):** | |
| | | | | Macrocephaly |
| | | | | Scaphocephaly |
| | | | | Hypertrichosis |
| | | | | Thickened skin |
| | | | | Funnel-shaped thorax |
| | 1-3 years | | | |
| | | (https://rarediseases.info.nih.gov/diseases/12559/mucopolysaccharidosis-type-ih) | | |
| | | | **The main signs (80% -99% of patients have these symptoms):** | |
| | | | | Short neck |
| | | | | Rough facial features |
| | | | | Stiffness of large joints |

***Figure 24 —*** *Example of a file structure with feature modalities*

In the first step, the PowerQuery tool merged the data into a single normalized table (see fig. 25). An additional table was needed in which numerical values (Main — 5, Necessary — 4, Secondary — 2) were compared to the text names of the modalities.

| Disease | Period | Symptom | Manifestation | Severity | Modality |
|---|---|---|---|---|---|
| Hurler | under 1 year | Short neck | 0,1 | 2 | 5 |
| Hurler | under 1 year | Rough facial features | 0,7 | 5 | 5 |
| Hurler | under 1 year | Hepatomegaly | 0,5 | 7 | 5 |
| Hurler | under 1 year | Splenomegaly | 0,2 | 3 | 5 |
| Hurler | under 1 year | Hernia | 0,6 | 7 | 5 |
| Hurler | under 1 year | Growth retardation | 0,7 | 3 | 4 |
| Hurler | under 1 year | Macroglossia | 0,3 | 5 | 4 |
| Hurler | under 1 year | Hearing loss | 0,3 | 4 | 4 |
| Hurler | under 1 year | Corneal opacity | 0,2 | 4 | 4 |
| Hurler | under 1 year | Macrocephaly | 0,7 | 2 | 2 |
| Hurler | under 1 year | Scaphocephaly | 0,4 | 3 | 2 |
| Hurler | under 1 year | Hypertrichosis | 0,4 | 5 | 2 |
| Hurler | under 1 year | Thickened skin | 0,3 | 2 | 2 |
| Hurler | under 1 year | Funnel-shaped thorax | 0,6 | 2 | 2 |
| Hurler | 1-3 years | Short neck | 0,1 | 3 | 5 |
| Hurler | 1-3 years | Rough facial features | 0,1 | 7 | 5 |

***Figure 25 —*** *Table Structure after Merge and Transform*

The table contains 6 columns:

- Disease: The names of the diseases being diagnosed.
- Period: Age group.
- Symptom: Names of disease signs.
- Manifestation: Manifestation values ($n_{ik}$).
- Severity: the value of the expression ($s_{ik}$).
- Modality: modality values ($M_{ik}$).

**Step 2**: Data source specification. In this step, an auxiliary *ontomorf_struct* ontology importing meteor ontology was created. Next, an instance of the Workbook class was created with hasPath attributes (path to file from step 1) and hasName (file name from step 1).

**Step 3**: Extracting a data structure. The *ontomorf_struct* ontology was populated with instances describing the file structure from step 1 (see fig. 26).

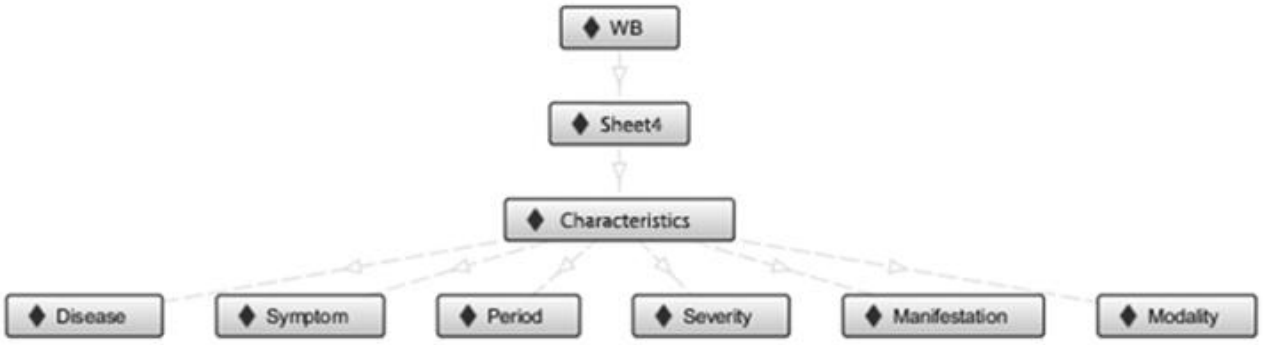


***Figure 26 —*** *Ontology describing the structure of the data source*

**Step 4**: Set the mapping rules. At this stage, *ontomorf_struct* ontology was imported into ontomorf ontology, followed by the following mapping rules:

- For **Period** class:
  - getInstancesFrom Period, where Period is an instance of the WBColumn class, representing the corresponding column from the table created in the first step.
  - makeList (indicates that instances must be organized into a list using the hasNext property).
- For **Disease** class:
  - getInstancesFrom Disease, where Disease is an instance of the WBColumn class, representing the corresponding column from the table created in the first step.
- For **Phase** class:
  - combineInstancesFrom **Disease** useObjectProperty relatesToDisease;

- combineInstancesFrom **Period** useObjectProperty relatesToPeriod (**Disease** and **Period** — class names, a relatesToDisease and relatesToPeriod are the relationship names that must be used to create connections between specimens that represent a combination of disease and period and source specimens of Disease and Period classes).

- For **Symptom** class:
  - getInstancesFrom Symptom, where Symptom is an instance of the WBColumn class, representing the corresponding column from the table created in the first step.
  - makeList (indicates that instances must be organized into a list using the hasNext property).
- For **Characteristic**:
  - combineInstancesFrom **Disease** useObjectProperty relatesToDisease;
  - combineInstancesFrom **Period** useObjectProperty relatesToPeriod;
  - combineInstancesFrom **Symptom** useObjectProperty relatesToSymptom (Disease, Period, and Symptom are class names, and relatesToDisease, relatesToPeriod, and relatesToSymptom are the relationship names that should be used to create connections between specimens that represent a combination of disease, period and feature, and source examples of the Disease, Period and Symptom classes).
  - getValuesFrom Manifestation, where Manifestation is WBColumn instance that represents a corresponding column from a table created in the first step.
  - getValuesFrom Modality, where Modality is WBColumn instance that represents a corresponding column from a table created in the first step.
  - getValuesFrom Severity, where Severity is **WBColumn** instance that represents a corresponding column from a table created in the first step.
- **Step 5:** The ontology is populated. This is done automatically with the population module. Figures 19-21 show the structure fragments created for different types of variables.

After the ontology was populated, a logical inference machine (the rules are given in the annex) was launched, with which missing relationships were constructed and the values required for the diagnostic process were computed. All the output results were exported to ontology, as the withdrawal process took considerable time, and in this case the «basic» part of ontology was populated, i.e. that part of ontology which is not expected to change regularly.

The same algorithm was then applied to download into the resulting knowledge base information about 15 test patients, and then launched a logical inference machine to make a diagnosis. The results are shown in the annex.

Thus, the use of the METEOR method proposed in the thesis made it possible to create a prototype of the medical expert system together with the employees of FITC of RAS.

## 3.3. Specifics of Vehicle Assembly Unit Content

The task of creating and populating of the ontological knowledge base of assembly units of automobiles has arisen in connection with the production need to systematize, organize and re-use the knowledge about assembly units of different automobile units within the automobile company. The elements of the knowledge base of this enterprise are assembly units (AU), parts of various cars (mainly trucks) produced in the company. The knowledge base is the basis for the creation of a reference system in which users (such as an engineer, a procurement manager, an economist, etc.) can find the AU they need by characteristics and compare it with analogues. The relevance of the system is due to the fact that the company manufactures electric vehicles using a component already available on the market, and it takes a lot of staff time to find information about the products of suppliers. The system is supposed to have both information about those AU that are already in use in the company and those that are on the market in the catalogues of suppliers. In the future, the knowledge base is planned to be used in robotic production (assembly), which is also developed in the company. The customer of the knowledge base was the IT-unit, which digitizes all business processes of the enterprise. For the pilot project 50 AU was taken.

The AU knowledge base was developed using VITON (Visual Collegiate Knowledge Graph Development) [Gavrilov et al. , 2019] which is oriented to the solution of the task of describing assembly units (AU) for complex high-tech production and is connected with the formation of a knowledge base of great extensibility (thousands of AU, copies of ontology). A peculiarity of the resulting knowledge base is the manifold and complex structure of the properties of AU, while the number of ontology classes is small.

The AU knowledge base can be divided into four stages: conceptualization, formalisation, implementation and content.

Conceptualization was carried out using intelligence maps. First, a top-down simulation was applied and a mock-up of ontology in the form of an intelligence card was created. An upward simulation was then applied: on the basis of an ontology mock-up, intelligent cards were created for a set of pilot AU provided by the customer. These intelligence maps were evaluated by experts and then the layout was modified to include new properties, units and ranges. These two simulations were implemented

iteratively: the intelligence maps for each AU were based on the mock-up and the mock-up itself was continuously modified according to the new properties identified during the AU analysis.

During the **formalization** phase, electronic tables were used. Once the smart cards for about 50 AU were created and the ontology layout was recorded (i.e. additional AU analysis contributed little to the layout), the properties were formalized using tables. The table with properties served as an intermediate step between intelligence maps and ontology. The main purpose of creating a table was to create a hierarchy of properties and a dictionary. The smart cards for different AU contained characteristics that are child properties of the same high-level property. For example, the ERC series air compressor has the output characteristic Free air delivery («free air flow»), which is related to the property Output («output»), as well as the property Air flow («air flow») of another instance. In addition, the same AU characteristics may be called differently in different suppliers' catalogues, so the tabulation helped to identify synonyms. A similar use of tables to create ontology was mentioned in Rubin's [Rubin, 2008] and Ontorat's [Xiang et al., 2015] approaches. The converted smart cards formed the basis of the table (Figure 27 shows a fragment of the table for AU in the example of the ERC series air compressor).

| Properties | Label | Description | ERC 507L Mattei ERC Series |
|---|---|---|---|
| hasFunctionalCharacteristics | Functional characteristics | "Attribute or characteristic in the use and performance of a product" (Source: ISO 14050:2009(en), 8.3.3.2) | |
| hasFunctionCharacteristics | Function | Estimated behavior of the artifact | |
| hasType | Artefact type | Informal type description (motor, battery, etc.) | compressor |
| hasFunctionType | Function type | Function type according to IEC 81346 | GQ |
| hasOutputCharacteristics | Output | All output that can be provided by the artifact if functioning properly | |
| hasOutputAirCharacteristics | Air outlet specifications | Characteristic group for describing air outlet | |
| hasOutputFreeAirDeliver | Free air supply | | 43 scfm |
| hasRatedPressure | Permissible pressure | | 125 psig |

***Figure 27 —*** *Table fragment for AU air compressor of ERC series*

Conversion techniques similar to those described for mind map transformation [Sure et al., 2002; Křemen et al., 2014], have been applied with some additions:

- AU formed columns in the table (subjects).
- Concepts from ontology prototypes have been transformed into high-level properties, and low-level elements from AU to child properties. In the table they formed rows (predicates). During the transformation, property names were unified according to the chosen naming form, human names were added as labels, as well as a description.
- The lowest level of the mind map was placed in the cells of the table (objects). For each AU, all related properties were added to the table together with their dimensions (if any).

The table was created in three steps:

1. Initial input of properties according to the mind map of ontology.
2. Clarification of the general structure of properties, taking into account their semantic closeness and level of generalization.
3. Enter all CE into the created property structure (specifying the properties and, if necessary, their values), confirm and finalize the structure, add labels and property descriptions.

In the **implementation** phase of the property hierarchy created in the previous stage, it was transformed into a description in the OWL language with the help of the ontology editor (WebProtégé [https://webprotege.stanford.edu/]). The transition consisted of three steps. In the first step, properties were imported to the ontology editor. At the time of their transfer, the general hierarchy of properties was divided into two separate hierarchies for object properties (object property) and property-values (datatype property). At the same time, upper-layer properties were the top-level classification level for both types of properties (for example, a property named hasFunctionalCharacteristics existed in both the property-object hierarchy and the property-value hierarchy). A grade hierarchy was then created with four upper-level classes:

- Artifact («artifact») is example of assembly units and parts thereof;
- Organization («organization») is copy of suppliers' and manufacturers' organizations;
- QuantitativeValue (the quantitative value) is copy of quantitative values of properties;
- Reference («dictionaries») is instance the value of properties imported from external sources (for example, this class included a list of units of measure).

In the last stage, ontology was populated with the method proposed in the work.

1. Assembly units have been imported as Artifact class specimens with rdfs:label properties (human name) and rdfs:description properties (if available).
2. Organizations were imported as copies of Organization class with rdfs:label properties (the human name) and hasURL (a link to the organization’s website).
3. Properties values from other dictionaries were imported as instances of the Reference class.
4. Quantitative values were imported as instances of the QuantitativeValue class. Each instance described a value or range of values. For example, the ERC series compressor has the property hasFreeAirDelivery («free air injection») with a range of values from 5.1 to 460 scfm. It is imported as an instance of «5.1-460scfm» class Flow («flow», subclass QuantitativeValue) with properties-values of hasMaxValue (maximum value: 460), hasMinValue (minimum value 5.1; hasUnitCode, unit code: scfm).

The resulting knowledge base demonstrated both the viability of the VITON method and the feasibility of the proposed ontology population method. It should be noted that this example is not intended for the developed method of population of ontologies, as the data were stored in a knowledge

base from a table that could be modified if necessary and specifically designed for convenience, including ontology content.

## 3.4. Content of ontology for teaching administration

The education system in Russia is undergoing significant changes. These changes are due to several factors. The first is the entry of the Russian Federation into the Bologna process on 19 September 2003. Full integration into the Bologna process requires reform of the educational system in general and of higher vocational education in particular. The reform envisages, first of all, the development of educational programmes compatible with European ones and, for their implementation, the corresponding transformation of higher education structures, the normative framework and, finally, teaching practices. The emphasis is shifting from the content of education to learning outcomes, which is achieved through a competency-based approach.

Second, on April 30, 2014, by order of the Government of the Russian Federation, the regular Plan of Measures («Road Map») «Changes in the branches of the social sphere aimed at increasing the efficiency of education and science» was approved. The document states, inter alia, that the main directions of change in higher education include:

- Improvement of the structure and network of State higher education institutions:
    - Annual monitoring of the effectiveness of higher education institutions;
    - Implementation of a program to improve the network of State higher education institutions, including by reorganizing and joining organizations and their subsidiaries.
- Improving the Structure of Educational Programmes:
    - Introduction of an applied bachelor's degree in higher education
    - Monitoring the transition to federal State educational standards for higher-level training and keeping them up to date
- Improving the performance of higher education institutions in accordance with their specialization:
    - Updating federal university development programmes;
    - Support for the development programmes of the network of national research universities;
    - Implementation and monitoring of the development programmes of the leading universities, which receive State support in order to increase their competitiveness among the world's leading scientific and educational centres;
    - Implementation of programmes for the strategic development of higher education institutions.
- Human Resource Development in Higher Education

- Developing and implementing mechanisms for an effective contract with scientific and pedagogical staff of higher education organizations;
- Carrying out measures to certify the scientific and pedagogical staff of higher education institutions;
- Development and implementation of mechanisms for an effective contract with the heads of higher education institutions to establish a link between the quality indicators of State (municipal) educational institutions Services provided by the organization and efficiency of the activities of the head of the educational organization of the higher education system;
- Information and monitoring support for the establishment of an effective contract

Third, on 21 December 2012 the State Duma adopted (and on 26 December 2012 the Federation Council approved) the Federal Law «On Education in the Russian Federation». The federal State educational standards for higher vocational education in the areas of training previously approved are being brought into line with the requirements of the Act.

Moreover, the need for so-called lifelong education is now widely debated. Lifelong education is a lifelong education that is provided by the unity and integrity of the educational system, the creation of conditions for self-education and the full development of the personality, and a set of successions, Coordinated and differentiated educational programs at different levels and levels guaranteeing citizens the right to education and providing opportunities for general education and vocational training; Retraining and lifelong training [Dictionary of agreed terms and definitions in the field of education of the States members of the Commonwealth of Independent States, 2004].


***Figure 28 —*** *Factors influencing higher vocational education in general and higher education in particular*

The following main factors confirm the importance of continuing education:

- The introduction of new technology, the production of modern goods, and the growth of communication facilities create conditions for the elimination or modification of certain types of work. Therefore, the necessary qualifications cannot be guaranteed in basic education.
- The world is becoming a borderless market with a high level of competition between countries. Countries with continuing education programs lead in this competition. They are able to respond to any "challenge" of productivity improvement as soon as possible.
- Change in all areas of life is a central element of modernity. Continuous and rapid changes in technology and other areas require continuous learning;
- It is more efficient and cost-effective for business to increase the productivity of existing employees through continuous learning than to attract new employees

As a result, educational standards, curricula and syllabuses need to be constantly updated to reflect the realities of our time (see fig.28).

At the same time, as noted in the work of [Blagov, Lescheva, Scherban, 2018], there is a gap between the competences on which competency-oriented study plans (COSP) of higher education institutions are based and competences, who are expected to have potential employers as candidates for the position. In order to analyze the existing COSPs and their equalization in terms of professional standards, an ontological model was proposed, containing in addition to classes Teachers, Disciplines,

Competences, etc., classes Work Functions and Professional Standards. Documents such as COSP Basic Education Program (BEP) Bachelor's Degree in «Management», three professional standards «Information Technology Product Manager», «Information Technology Manager» and «Specialist on knowledge management and information objects» in the field of information technologies developed by ACITE Association (Association of Computer and Information Technology Enterprises) [Professional standards in the field of IT [http://www.apkit.ru/committees/education/meetings/standarts.php], as well as a list of additional competences developed on their basis and their compliance with the disciplines considered by COSP.

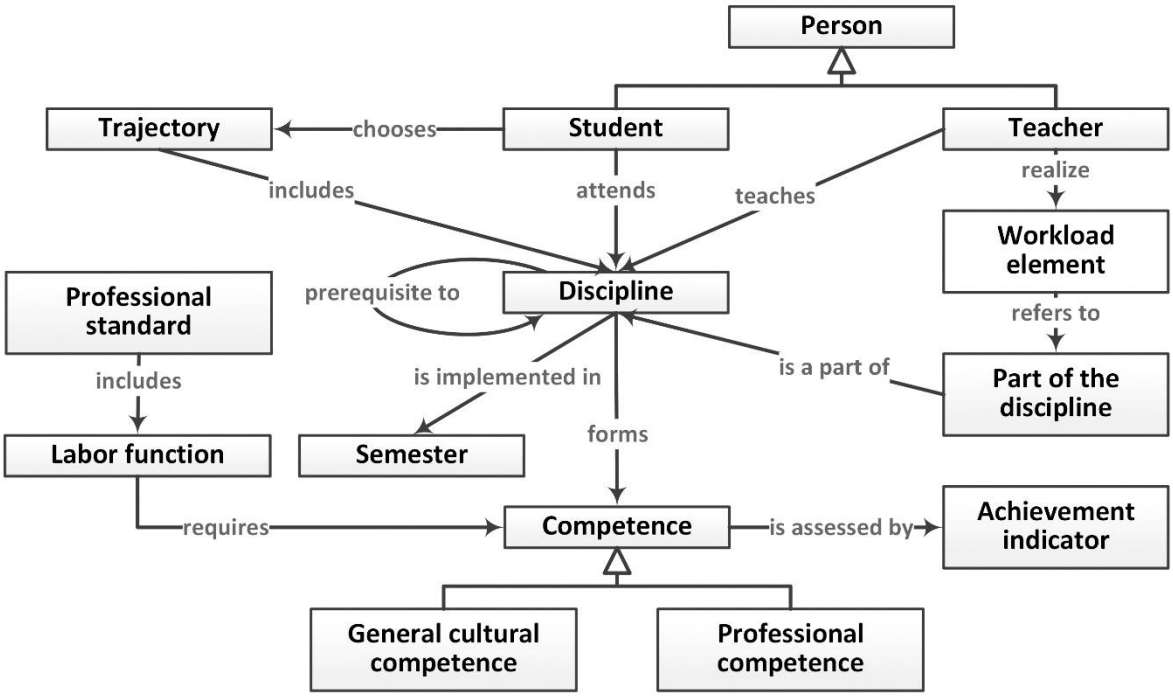


***Figure 29 —*** *Ontology clases for analizis of education programs*

The process of population of this ontology is not described in detail (fig.29), as it is similar to the ontology population processes described in previous paragraphs. A number of questions can be answered, for example:

1. What set of competences has a student received after having listened to some given series of disciplines?
2. What «minimum» set of disciplines ensures mastery of all competences from some set?
3. What competences and disciplines will be affected if some discipline is removed from the curriculum?
4. What subjects of choice may be excluded from the curriculum without prejudice to the acquisition of a particular set of competencies?
5. What competences should a student have (and therefore what disciplines to listen to) in order to take a certain professional position?

Thus, the created knowledge base can be used as an analytical tool by educational program managers and specialist departments to analyze the compliance of existing COSP with employers' requirements and to correct COSP according to the development of relevant fields and professional standards, as well as their work responsibilities.

But this information is not enough for students to build educational trajectories, because the subjects of choice are integrated into blocks, and students do not have the opportunity to master all disciplines from one block. Therefore, ontology has been supplemented with classes describing the structure of the curriculum, including modules of subjects of choice, in which semester a discipline is implemented, possible trajectories of instruction taking into account precepts to disciplines, etc., Information on disciplines and teachers has also been expanded. In order to complete the extended ontology, additional documentation was required to provide information on teachers' classroom loads.

An enhanced knowledge base for competency-based learning allows teachers to be classified according to their workload and competencies, as well as to answer more complex questions such as:

1. What trajectory of learning will ensure that some given set of competences are mastered (In view of the restrictions imposed by the fact that a student may attend only one/more of the disciplines of each block of his choice, as well as by the fact that a number of other disciplines must be pre-qualified in order to enroll in certain courses)?
2. In which semester is it recommended to visit the included training in order to have a minimal impact on the selected set of competences? What competences should be emphasized in the choice of subjects for inclusive learning?

Thus, the created knowledge base can be used not only by the management of the programs or departments, but also by the students of the profile in order to construct trajectories of individual professional development by choosing trajectories, i.e. elective disciplines as professional, and of a specialized nature.

## Conclusions on chapter 3

The third chapter discusses examples of creating ontologies of different subject areas and populating them with the method proposed in the work and described in the second chapter. According to this 5-step canonical ontology population algorithm, the following were implemented and populated:

e) EMPIRION Empirical Processing Ontology;
f) Ontology of Ontomorf orphan diseases;
g) Ontology of automation assembly units;
h) Ontology for the administration of the academic activity of the university.

The most telling example is the population of *Ontomorf* ontology, as it involves not only a single population of ontology prior to operation, but also regular population in the process of its use (input of patient information for diagnosis).

The method of ontology population proposed in the thesis has been tested in the framework of the project «Formation of knowledge bases on the basis of empirical research data: ontological approach (EMPIRION)» (RFBR 20-07-00854). This ontology ensured that the basic traditional principles of data storage and access were respected. It is with this ontology example that the application of the method is demonstrated in detail.

The task of creating and populating the ontological knowledge base of assembly units of automobiles has arisen in connection with the production necessity of systematization, organization and re-use of knowledge about assembly units of different automobile units within the automaker company. The relevance of the system is due to the fact that the company manufactures electric vehicles using a component already available on the market, and it takes a lot of staff time to find information about the products of suppliers. The resulting knowledge base demonstrated both the viability of the VITON method and the feasibility of the proposed ontology population method.

The expanded knowledge base of competency-oriented learning created and populated with the proposed method makes it possible to classify teachers according to workload and competences, and can be used not only by program managers or departments, but also the students of the profile with the aim of constructing trajectories of individual professional development by choosing trajectories, i.e. elective disciplines of both professional and professional character.

All four ontologies created and described relate to different subject areas. The structure of the data sources is also different, it is demonstrated that this is not an obstacle to the use of the developed method. It can be observed that data from one or more spreadsheets were mostly used for population. This is due to the fact that it is the spreadsheets that are widely used in everyday practice in different areas, and all the projects described are practical and implemented for specific clients, both internal and external.

In this way it was demonstrated that the method developed allows to populate ontologies related to different subject areas, regardless of their structure, as well as the structure of data sources, which is a distinctive feature of the proposed method.

The results of the studies were obtained as part of the RFBR project studies:

- RFBR №17-07-00228 MetaDology and Technology of Ontology Formation Based on Integration with Heterogeneous Data Sources (METEOR)
- RFBR № 20-07-00854 Empirical Knowledge Base Formation: The Ontological Approach (EMPIRION)

# LIST OF ABBREVIATIONS

ACITE Association — Association of Computer and Information Technology Enterprises

AKO — a kind of, type of relationship

AU — assembly units

DARPA — Defense Advanced Research Projects Agency

DB — data base

DO — domain ontology

DSO — Data Source Ontology

DTD — Document Type Definition

EMPIRION, project — Formation of knowledge bases based on empirical research data: ontological approach.

ETL — extract-transform-load,

HTML — HyperText Markup Language

IT — information technology

JTP — Java Theorem Prover, www.ksl.stanford.edu/software/JTP

KB — knowledge base

METEOR method and project — Methodology and Technology of Ontology Population based on Integration with Heterogeneous Structured Data Sources

OCML — Operational Conceptual Modeling Language

OKBC — Open Knowledge Base Connectivity

OWL — Ontology Web Language, standard of W3C

RDF — Resource Description Framework

SQL — structured query language

TANGO — Table ANalysis for Generating Ontologies

URI — Uniform Resource Identifier

W3C — World Wide Web Consortium, https://www.w3.org/

WWW — Word Wide Web

XML — eXtensible Markup Language

XOL — ontology exchange language based on XML

# CONCLUSION

Modern information systems are increasingly using knowledge assets in their work. Ontology is generally the primary model for knowledge representation. A review and analysis of the literature presented in the first chapter shows that, among the many problems with ontology applications, one can single out the problem of automating the ontological-type knowledge bases with copies, which this thesis is dedicated to solving.

Existing methods for populating ontology-based knowledge bases have a number of disadvantages, such as the narrow focus on project-specific tasks, the complexity of describing models of data sources and their transformation, High demands on user skills, ability to integrate data from only one type of source, lack of modularity and extensibility, and lack of user-friendly tools.

The main complexity of information modelling is heterogeneity, i.e. heterogeneity of the components, data and knowledge described. Moreover, the heterogeneity of data sources has the greatest influence on the process of populating knowledge bases. Problems encountered in population were classified according to the stages of the knowledge base life cycle. Major difficulties were identified in the design and operation phases.

A review of ontological engineering research over the past 10 years has shown that, despite the large number of completed projects and industrial ontology applications, There are significant gaps in theoretical understanding of the programmatic mechanisms for developing ontological-type knowledge bases.

On the basis of the literature review, the genealogy of ontology languages was expanded and expanded. The analysis revealed limitations of existing approaches and methods. In the first chapter, the task of research is to create a method for population of ontological knowledge bases and integrating heterogeneous sources of structured data, taking into account the semantics of the subject area. The main features of the method described below are::

- The possibility of using it to enrich the knowledge base at different stages of the life cycle.
- Possibility of simultaneous population from several heterogeneous sources.
- Integration with world standards of ontology description languages to enable development and maintenance of the knowledge base using existing ontological engineering tools

The actual method of ontology population is described in the second chapter. The architecture of the software prototype is also described. The proposed method addresses the main shortcomings of existing solutions:

1. Need to use and therefore learn syntax of several specific languages to describe mapping. It also requires the creation of a convenient tool for each of the languages used.

2. Narrowness of methods: a large number of different methods have been proposed to integrate data into ontology, but each method works for only one type of data source.
3. Lack of guidance on integrating data from multiple sources at the same time. If there is a need to integrate data from, for example, a database and an XML document into a single ontology, then using existing methods will have to act sequentially. In this case the first step is population of the initial ontology with data from the database, and the second is population of received ontology with data from the XML document. Since the data in the sources are subject to change, this procedure needs to be repeated regularly, resulting in unnecessary costs.
4. Inconsistencies at the integration stage of data from different sources. Since the methods in chapter 2 focus on the integration of data from a single source, the issue of resolving such conflicts is not addressed.
5. Uncertainty as to how to connect to the data source, how to authenticate, etc.
6. Lack of guidelines for ontology content if companies use their own data storage formats, and for new data formats that may appear in the future.

The proposed METEOR (MetaDology and Technology of Ontology Content Integration Method with Heterogeneous Structured Data Sources) uses accumulated data sets regardless of their form of storage and presentation (for example, databases, spreadsheets and XML files). The application of the method reduces the laboriousness of population and enrichment of ontologies.

The proposed methodology is designed to perform the same type of operations or to perform the same tasks requiring knowledge of rules and procedures, as well as to compare information from different sources. Such tasks arise in enterprises, educational institutions and organizations with different departmental affiliations within the framework of digital documentation. The proposed methodology could form the basis of systems to support knowledge-based decision-making.

The novelty of the proposed method is the use of auxiliary Data Sources Ontology (DSO), which can be extended by adding additional types of data sources. The use of DSO is complemented by the development of requirements for creating mapping rules that allow to bind DSO and population of a specific ontology of a subject area (DO) to be populated with instances.

The application of the developed METEOR method can be described by an algorithm consisting of 5 steps, two of which are performed automatically:

1. Identification of data sources.
2. Data Source Specification by Auxiliary DSO.
3. Extraction of data structure from data sources in DSO.
4. Set map rules in domain ontology.
5. Population of DO.

The key step of the DO population algorithm is to set mapping rules. It is done using the annotation mechanism built into the OWL. New annotation properties have been introduced: 4 basic and 2 auxiliary properties (see 2.2.3 for more details). This approach eliminated the need for additional instrumentation support for the METEOR method, as annotation can be done with any ontology editor.

A new prototype architecture was also developed to implement the proposed METEOR method. The architecture is based on the well-known architecture of ETL-systems. A peculiarity of the proposed architecture is that the receiver of the data is not a repository or database, but ontology or a knowledge base.

The developed prototype implements the created METEOR method and allows to populate the ontological knowledge base from heterogeneous sources. The software prototype consists of 4 modules supporting processes for data source specification, data structure extraction, mapping rules, and DO content. The entire ontology population process using the METEOR method is illustrated by a conditional example.

The work is a continuation and development of numerous research and development in the field of ontological engineering. The results of the thesis are as follows::

1. A new genealogy of the languages of knowledge representation is proposed, which significantly expands and complements the previously proposed genealogy in [Kazekin, 2008].
2. The influence of cognitive styles on the creation of ontological models, where not only qualitative but quantitative research methods were first applied, which allowed new statistically significant relationships between the cognitive type of a person and the characteristics of the ontology he had built, published in [Gavrilova, Leshcheva, 2015; Gavrilova, Lescheva, 2016].
3. A new universal method of ontology population is proposed, which differs in complex approach and for the first time allows simultaneously to integrate data from different types of sources, takes into account their distributed nature, and also proposes a solution to the problem of resolving contradictions in the data to be integrated, which will reduce the laboriousness of populating and enriching ontologies by using the accumulated data, regardless of their form of storage and presentation. Also, the proposed solution is extensible to provide a potential opportunity to work with new data formats in the future. The developed method of implementing the process of transforming data structures into an ontological model does not use additional languages for this, but only costs by ontological modelling tools.
4. Some templates have been developed for building rules for mapping the initial ontology into the knowledge base of the subject area.

5. The criteria for selecting a software platform for implementing the proposed method are formulated.
6. An architecture was developed and a software prototype was implemented, implementing a procedure for consolidating data stored in structured data sources into a knowledge base in the subject area.

   Results were tested in 4 subject areas:

a) EMPIRION Empirical Processing Ontology
b) Ontology of Ontomorf orphan diseases;
c) Ontology of assembly units for automated car production;
d) Ontology for university administration

The results of the studies were obtained as part of the RFBR project studies:

- RFBR № 17-07-00228 MetaDology and Technology of Ontology Formation Based on Integration with Heterogeneous Data Sources (METEOR)
- RFBR № 20-07-00854 Empirical Knowledge Base Formation: The Ontological Approach (EMPIRION)

**The results of the work have been published in the works of international and All-Russian conferences:**

1. The Nineteenth Russian Conference on Artificial Intelligence, RCAI-2021 (Taganrog, Russia, 2021)
2. GSOM Emerging Markets Conference (Saint-Petersburg, Russia, 2018)
3. XXIII scientific conference «Enterprise Engineering and knowledge management», EE&KM –2018 (Moscow, Russia, 2018)
4. IV International Scientific Conference “The Convergence of Digital and Physical Worlds: Technological, Economic and Social Challenges” (Saint-Petersburg, Russia, 2018)
5. BIR 2018 Short Papers, Workshops and Doctoral Consortium, (Stockholm, Sweden, 2018)
6. XXII International Scientific and Practical Conference "System Analysis in Design and Control" (Saint-Petersburg, Russia, 2018)
7. XVI Russian Conference on Artificial Intelligence, RCAI-2018 (Moscow, Russia, 2018)
8. GSOM Emerging Markets Conference 2018 (Saint-Petersburg, Russia, 2018)
9. Enterprise Engineering and knowledge management», EE&KM –2017 (Moscow, Russia, 2017)
10. Systems Analysis and Information Technologies (SAIT-2017) (Svetlogorsk, Russia, 2017)
11. IEEE 30th Neumann Colloquium (Budapest, Hungary, 2017)
12. XV Russian Conference on Artificial Intelligence, RCAI-2016 (Smolensk, Russia, 2016)

13. GSOM Emerging Markets Conference-2016: Business and Government Perspectives (Saint-Petersburg, Russia, 2016)
14. Knowledge-Ontologies-Theories (KONT-15) (Novosibirsk, Russia, 2015)
15. International Conference on Knowledge Engineering and Ontology Development (KEOD 2014) (Rome, Italy, 2014)
16. The 8th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS 2014) (UK, Birmingham, 2014)
17. Federated Conference on Computer Science and Information Systems (Kraków, Poland, 2013)
18. Knowledge-Ontologies-Theories (KONT-13) (Novosibirsk, Russia, 2013)
19. Enterprise Engineering and knowledge management», EE&KM –2013 (Moscow, Russia, 2013)
20. XI Russian Conference "Teaching Information Technology in the Russian Federation" (Voronezh, Russia, 2013)

**List of author's papers on the topic of the thesis:**


1. Leshcheva I., Begler A. A method of semi-automated ontology population from multiple semi-structured data sources //Journal of Information Science. – 2020. DOI: 10.1177/0165551520950243.
2. Kudryavtsev D. V. i dr. Metod kollektivnoy vizual'noy razrabotki ontologicheskogo grafa znaniy/ Kudryavtsev D.V., Begler A.M., Gavrilova T.A., Leshcheva I.A., Kubel'skiy M.V., Tushkanova O.N. //Iskusstvennyy intellekt i prinyatiye resheniy. – 2019. – №. 1. – P. 27-38.
3. Blagov Ye. YU., Leshcheva I. A., Shcherban S. A. Ontologicheskiy podkhod v praktike obrazovatel'noy deyatel'nosti: formirovaniye trayektoriy individual'nogo professional'nogo razvitiya studentov //Otkrytoye obrazovaniye. – 2018. – V. 22. – №. 5. – P. 26-39.
4. Gavrilova T. A., Leshcheva I. A. Ponyatiynyye struktury znaniy i kognitivnyy stil' //Psikhologiya. Zhurnal Vysshey shkoly ekonomiki. – 2016. – V. 13. – №. 1. P. 154-176.
5. Gavrilova T. A., Leshcheva I. A. Ontology design and individual cognitive peculiarities: A pilot study //Expert systems with Applications. – 2015. – V. 42. – №. 8. – P. 3883-3892.
6. Gavrilova T. A., Leshcheva I. A. Building Collaborative Ontologies: A Human Factors Approach //Collaborative Knowledge in Scientific Research Networks. – IGI Global, 2015. – P. 305-324.

7. Gavrilova T., Leshcheva I., Strakhovich E. Gestalt principles of creating learning business ontologies for knowledge codification //Knowledge Management Research & Practice. – 2015. – V. 13. – №. 4. – P. 418-428.
8. Gavrilova T., Leshcheva I. The interplay of knowledge engineering and cognitive psychology: learning ontologies creating //International Journal of Knowledge and Learning. – 2015. – V. 10. – №. 2. – P. 182-197.
9. Leshcheva I., Gavrilova T. How the cognitive features testing can assist in evaluating collective ontology engineering //International Journal of High Performance Computing and Networking. – 2015. – V. 8. – №. 3. – P. 275-284.
10. Leshcheva, I. A., Leshchev, D. V. Analiz dinamiki izmeneniy nauchnoy oblasti metodami ontologicheskogo inzhiniringa //Otkrytyye semanticheskiye tekhnologii proyektirovaniya intellektual'nykh sistem. – 2014. – № 4. – P. 483-486.
11. Gavrilova, T. A., Kudryavtsev, D. V., Leshcheva, I. A., Pavlov, YA. YU. Ob odnom metode klassifikatsii vizual'nykh modeley //Biznes-informatika. – 2013. – № 4(26). – P. 21-34.
12. Gavrilova T., Bolotnikova E., Leshcheva I., Blagov E., Yanson A. Measuring psychological impact on group ontology design and development: an empirical approach //Communications in Computer and Information Science. – 2013. – Vol. 394. – P. 29-43. – DOI 10.1007/978-3-642-41360-5_3.
13. Gavrilova T. A., Leshcheva I. A., Kudryavtsev D. V. Ispol'zovaniye modeley inzhenerii znaniy dlya podgotovki spetsialistov v oblasti informatsionnykh tekhnologiy //Sistemnoye programmirovaniye. – 2012. – V. 7. – №. 1. – P. 90-105.
14. Gavrilova T. A., Leshcheva I. A., Rumyantseva M. N. Knowledge elicitation methods taxonomy: Russian view // Lecture Notes in Computer Science. – 2011. – Vol. 6881 LNAI. – No Part 1. – P. 337-346. – DOI 10.1007/978-3-642-23851-2_35.
15. Gavrilova T. A., Leshcheva I. A., Strakhovich E. V. Ob ispol'zovanii vizual'nykh kontseptual'nykh modeley v prepodavanii //Vestnik Sankt-Peterburgskogo universiteta. Menedzhment– 2011. – № 4. – P. 124-150.
16. Gavrilova T. A., Leshcheva I. A., Leshchev D. V. Ispol'zovaniye ontologii v kachestve didakticheskogo sredstva //Iskusstvennyy intellekt. – 2000. – № 3. – P. 34-39.

# LIST OF REFERENCES


1. Begler A. M., Kostousov, S. A. Kriterii vybora instrumenta po rabote s korporativnymi bazami znaniy //Sbornik studencheskikh rabot V Mezhdunarodnoy nauchno-prakticheskoy konferentsii “Upravlencheskiye nauki v sovremennom mire” 2018 / g. Moskva, (2018). – P. 18–23. (In Russian)
2. Blagov Ye. YU., Leshcheva I. A., Shcherban S. A. Ontologicheskiy podkhod v praktike obrazovatel'noy deyatel'nosti: formirovaniye trayektoriy individual'nogo professional'nogo razvitiya studentov //Otkrytoye obrazovaniye. – 2018. – V. 22. – №. 5. – P. 26-39. (In Russian)
3. Bova V. V. Ontologicheskaya model' integratsii dannykh i znaniy v intellektual'nykh informatsionnykh sistemakh //Izvestiya Yuzhnogo federal'nogo universiteta. Tekhnicheskiye nauki – 2015. – №. 4 (165). – P. 225-237. (In Russian)
4. Gavrilova T. A. i dr. Metod vizual'noy kollektivnoy razrabotki ontologicheskogo grafa znaniy //Iskusstvennyy intellekt i prinyatiye resheniy. – 2019. – №. 1. – P. 27-38.
5. Gavrilova T. A., Kudryavtsev D. V., Muromtsev D. I. Inzheneriya znaniy. Modeli i metody. – Sankt-Peterburg : Lan', 2020. – 324 P. (In Russian)
6. Gavrilova, T. A., Kudryavtsev, D. V., Leshcheva, I. A., Pavlov, YA. YU. Ob odnom metode klassifikatsii vizual'nykh modeley //Biznes-informatika. – 2013. – № 4(26). – P. 21-34. (In Russian)
7. Gavrilova T. A., Leshcheva I. A. Ponyatiynyye struktury znaniy i kognitivnyy stil' //Psikhologiya. Zhurnal Vysshey shkoly ekonomiki. – 2016. – V. 13. – №. 1. (In Russian)
8. Gavrilova T. A., Leshcheva I. A., Kudryavtsev D. V. Ispol'zovaniye modeley inzhenerii znaniy dlya podgotovki spetsialistov v oblasti informatsionnykh tekhnologiy //Sistemnoye programmirovaniye. – 2012. – T. 7. – №. 1. – P. 90-105. (In Russian)
9. Gavrilova T. A., Leshcheva I. A., Leshchev D. V. Ispol'zovaniye ontologii v kachestve didakticheskogo sredstva //Iskusstvennyy intellekt. – 2000. – № 3. – P. 34-39. (In Russian)
10. Gavrilova T. A., Leshcheva I. A., Strakhovich E. V. Ob ispol'zovanii vizual'nykh kontseptual'nykh modeley v prepodavanii //Vestnik Sankt-Peterburgskogo universiteta. Menedzhment. – 2011. – № 4. – P. 124-150. (In Russian)
11. Gavrilova T. A., Muromtsev D. I. Intellektual'nyye tekhnologii v menedzhmente: instrumenty i sistemy. – SPb. : Vysshaya shkola menedzhmenta. – 2007. – 488 P. (In Russian)
12. Gavrilova T. A., Khoroshevskiy V. F. Bazy znaniy intellektual'nykh sistem. – SPb. : Piter – 2000. – 384 P. (In Russian)
13. Golenkov V. V. i dr. Ontologicheskoye proyektirovaniye gibridnykh semanticheski sovmestimykh intellektual'nykh sistem na osnove smyslovogo predstavleniya znaniy //Ontologiya proyektirovaniya.– 2019. – V. 9. – №. 1 (31). – P. 132-151. (In Russian)

14. Gribova V. V. i dr. Ontologiya meditsinskoy diagnostiki dlya intellektual'nykh sistem podderzhki prinyatiya resheniy //Ontologiya proyektirovaniya. – 2018. – V. 8. – №. 1 (27). – P. 58-73. (In Russian)
15. Grigor'yev L. YU., Zablotskiy A. A., Kudryavtsev D. V. Tekhnologiya napolneniya baz znaniy ontologicheskogo tipa //Nauchno-tekhnicheskiye vedomosti Sankt-Peterburgskogo gosudarstvennogo politekhnicheskogo universiteta. Informatika. Telekommunikatsii. Upravleniye. – 2012. – №. 3 (150). – P. 27-36. (In Russian)
16. Davidovskiy M. V. Formalizatsiya zadachi soglasovaniya ontologiy v detsentralizovannykh sistemakh // Vestnik zaporozhskogo natsional'nogo universiteta – 2011. – №. 1. – P. 25-29. (In Russian)
17. Dobrokhotov A. L. Ontologiya//Novaya filosofskaya entsiklopediya //M.: Mysl'. – 2010. – P. 149-152. (In Russian)
18. Yerzhenin R. V., Massel' L. V. Ontologicheskiy podkhod k predstavleniyu znaniy o metodologii modelirovaniya slozhnoy sistemy upravleniya //Ontologiya proyektirovaniya. – 2020. – V. 10. – №. 4 (38). – P. 463-476. (In Russian)
19. Zagorul'ko YU. A., Borovikova O. I. Tekhnologiya postroyeniya ontologiy dlya portalov nauchnykh znaniy //Vestnik Novosibirskogo gosudarstvennogo universiteta. Seriya: Informatsionnyye tekhnologii. – 2007. – V. 5. – №. 2. – P. 42-52. (In Russian)
20. Kazekin M.M. Istoriya yazykov predstavleniya ontologiy // Komp'yuternyye instrumenty v obrazovanii. – 2008. – №4 –P. 3-11. (In Russian)
21. Kleshchev A. S., Moskalenko F. M., Chernyakhovskaya M. YU. Ontologiya i model' ontologii predmetnoy oblasti «Meditsinskaya diagnostika» //Vladivostok: Izd-vo IAPU DVO RAN. – 2005. (In Russian)
22. Kobrinskiy B. A. Triyedinstvo faktorov uverennosti v zadachakh meditsinskoy diagnostiki //Iskusstvennyy intellekt i prinyatiye resheniy. – 2018. – №. 2. – P. 62-72. (In Russian)
23. Kobrinskiy B.A., Blagosklonov N.A., Demikova N.S., Gribova V.V., Shalfeyeva Ye.A., Petryayeva M.V. Vozmozhnosti primeneniya ontologicheskogo podkhoda k diagnostike orfannykh zabolevaniy// Semnadtsataya Natsional'naya konferentsiya po iskusstvennomu intellektu s mezhdunarodnym uchastiyem. KII-2019 (21–25 oktyabrya 2019 g., g. Ul'yanovsk, Rossiya). Sbornik nauchnykh trudov. V 2 t. – Ul'yanovsk: UlGTU, 2019. – V.2. – P. 227. ISBN 978-5-9795-1940-1. (In Russian)
24. Kudryavtsev D. V. i dr. Metod kollektivnoy vizual'noy razrabotki ontologicheskogo grafa znaniy/ Kudryavtsev D.V., Begler A.M., Gavrilova T.A., Leshcheva I.A., Kubel'skiy M.V., Tushkanova O.N. //Iskusstvennyy intellekt i prinyatiye resheniy. – 2019. – №. 1. – P. 27-38. (In Russian)

25. Leshcheva, I. A., Leshchev, D. V. Analiz dinamiki izmeneniy nauchnoy oblasti metodami ontologicheskogo inzhiniringa //Otkrytyye semanticheskiye tekhnologii proyektirovaniya intellektual'nykh sistem. – 2014. – № 4. – P. 483-486. (In Russian)
26. Nikitin V.V. Informatsionno-metodicheskoye obespecheniye perechnya formirovaniya napravleniy i spetsial'nostey v oblasti informatsionno-kommunikatsionnykh tekhnologiy // Moskva: MAKS Press, 2006. – 272 P. (In Russian)
27. Osipov G.S. Metody iskusstvennogo intellekta. – M.: Fizmatlit, 2011.
28. Slovar' soglasovannykh terminov i opredeleniy v oblasti obrazovaniya gosudarstv-uchastnikov Sodruzhestva Nezavisimykh Gosudarstv : ok. 100 terminov / M-vo obrazovaniya Ros. Federatsii, Departament soderzh. vyssh. prof. obrazovaniya, Upr. mezhdunar. obrazovaniya i sotrudnichestva, Issled. tsentr problem kachestva podgot. spetsialistov Mosk. gos. in-ta stali i splavov (tekhn. un-ta); [avt.-sost.: O. L. Vorozheykina i dr.] ; pod nauch. red. N.A. Seleznevoy. – Moskva : Issled. tsentr problem kachestva podgotovki spetsialistov, 2004. – 167 P. (In Russian)
29. Tuzovskiy A. F. Razrabotka sistem upravleniya znaniyami na osnove yedinoy ontologicheskoy bazy znaniy //Izvestiya Tomskogo politekhnicheskogo universiteta. 2007. V. 310, № 2. P. 182-185. (In Russian)
30. Abraham R., Erwig M. Header and unit inference for spreadsheets through spatial analyses //2004 IEEE Symposium on Visual Languages-Human Centric Computing. – IEEE, 2004. – P. 165-172.
31. Anicic, N., Ivezic, N., Marjanovic, Z. Mapping XML Schema to OWL, Enterprise Interoperability, Springer London, 2007.
32. Aussenac-Gilles N., Kamel M. Ontology Learning by Analyzing XML Document Structure and Content //KEOD. – 2009. – V. 9. – P. 159-165.
33. Baghernezhad-Tabasi S. et al. IOPE: Interactive ontology population and enrichment guided by ontological constraints //International Conference on Web Information Systems Engineering. – Springer, Cham, 2021. – P. 321-336.
34. Bakkas J., Jakjoud W., Bahaj M. Semantic mapping at the schema level of XML documents to ontologies //2014 International Conference on Next Generation Networks and Services (NGNS). – IEEE, 2014. – P. 165-169.
35. Barrasa J., Corcho Ó., Gómez-pérez A. R2O, an Extensible and Semantically based Database-to-Ontology Mapping Language //In Proceedings of the 2nd Workshop on Semantic Web and Databases (SWDB2004), Toronto, Canada. – 2004.
36. Bechhofer S. et al. OilEd: a reason-able ontology editor for the semantic web //KI 2001: Advances in Artificial Intelligence. – Springer Berlin Heidelberg, 2001. – P. 396-408.
37. Berdier C., Roussey C. Urban ontologies: The towntology prototype towards case studies //Ontologies for Urban Development. – Springer Berlin Heidelberg, 2007. – P. 143-155.

38. Berners-Lee T., Hendler J., Lassila O. The semantic web //Scientific american. – 2001. – V. 284. – №. 5. – P. 28-37.
39. Bizer C. D2R MAP - a database to RDF mapping language. In Proceedings of the 12th International World Wide Web Conference (WWW 2003), Budapest, Hungary, 2003.
40. Bizer C., Seaborne A. D2RQ-treating non-RDF databases as virtual RDF graphs //Proceedings of the 3rd international semantic web conference (ISWC2004). – Hiroshima : Citeseer, 2004. – V. 2004.
41. Bohring H., Auer S. Mapping XML to OWL ontologies //Marktplatz Internet: Von e-Learning bis e-Payment, 13. Leipziger Informatik-Tage (LIT 2005). – 2015 – pp. 147-156.
42. Buitelaara P., Cimianob P., Frankc A., Hartungc M., Racioppa S. Ontology-based information extraction and integration from heterogeneous data sources // Int. J. Human-Computer Studies – 2008. – V. 66. – №. 11.– P. 759–788.
43. Bullinger A. C. Classification—The OntoCube //Innovation and Ontologies: Structuring the Early Stages of Innovation Management. – 2009. – P. 172-195.
44. Cerbah F. Learning highly structured semantic repositories from relational databases. – Springer Berlin Heidelberg, 2008. – P. 777-781.
45. Chasseray Y. et al. A generic metamodel for data extraction and generic ontology population //Journal of Information Science. – 2021.
46. ChristopoulouE., Kameas A. GAS Ontology: An ontology for collaboration among ubiquitous computing devices // Int. J. Human-Computer Studies – 2005. – V. 62. – №. 5 – P. 664–685.
47. Clarkson K. et al. User-centric ontology population //European Semantic Web Conference. – Springer, Cham, 2018. – P. 112-127.
48. Cruz C., Nicolle C. Ontology Enrichment and Automatic Population From XML Data //ODBIS. – 2008. – V. 2008. – P. 17-20.
49. Cullot N., Ghawi R., Yetongnon K. DB2OWL : A tool for automatic database-to-ontology mapping. In Proceedings of the 15th Italian Symposium on Advanced Database Systems, pages 491–494, Torre Canne, Fasano, BR, Italy, 2007.
50. Cyganiak R., Bizer C., Maresch O., Becker C. The D2RQ mapping language v0.8, 2012. URL http://d2rq.org/d2rq-language
51. Das, S, Sundara, S, Cyganiak, R. R2RML: RDB to RDF mapping language, 2012. https://www.w3.org/TR/r2rml/
52. De Laborda C. P., Conrad S. Database to semantic web mapping using RDF query languages //Conceptual Modeling-ER 2006. – Springer Berlin Heidelberg, 2006. – P. 241-254.

53. De Leenheer P., Debruyne C. DOGMA-MESS: A tool for fact-oriented collaborative ontology evolution //OTM Confederated International Conferences "On the Move to Meaningful Internet Systems". – Springer, Berlin, Heidelberg, 2008. – pp. 797-806.
54. De Moor A., De Leenheer P., Meersman R. DOGMA-MESS: A meaning evolution support system for interorganizational ontology engineering //International Conference on Conceptual Structures. – Springer, Berlin, Heidelberg, 2006. – pp. 189-202.
55. Dolbear C., Goodwin J. Position paper on expressing relational data as RDF //W3C Workshop on RDF Access to Relational Databases. – 2007. – P. 25-26.
56. Domingue J. Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web. Proceedings of the 11th Banff Knowledge Acquisition Workshop, Banff, Alberta, Canada, April 18-23, 1998.
57. Domingue J., Fensel D., Hendler J. A. (ed.). Handbook of semantic web technologies. – Berlin, Heidelberg : Springer-Verlag Berlin Heidelberg, 2011. 1035 p.
58. Domingue J., Motta E., Garcia O. C. Knowledge Modelling in WebOnto and OCML: A User Guide //Knowledge Media Institute, Milton Keynes, UK. – 1999.
59. Erling O. Requirements for relational to RDF mapping, 2008. URL http://www.w3.org/wiki/Rdb2RdfXG/ReqForMappingByOErling.
60. Farquhar A., Fikes R., Rice J. The ontolingua server: A tool for collaborative ontology construction //International journal of human-computer studies. – 1997. – V. 46. – №. 6. – P. 707-727.
61. Ferdinand M., Zirpins C., Trastour D. Lifting XML Schema to OWL, in: Koch, Nora and Fraternali, Piero and Wirsing, Martin (Hrsg.): Web Engineering, ICWE 2004, Munich, Germany, July 26-30, 2004.
62. García, R., Celma, O. Semantic Integration and Retrieval of Multimedia Metadata, ISWC, Galway, Ireland, 2005.
63. Gavrilova T., Bolotnikova E., Leshcheva I., Blagov E., Yanson A. Measuring psychological impact on group ontology design and development: an empirical approach //Communications in Computer and Information Science. – 2013. – Vol. 394. – P. 29-43.
64. Gavrilova T. A., Leshcheva I. A. Ontology design and individual cognitive peculiarities: A pilot study //Expert systems with Applications. – 2015. – V. 42. – №. 8. – P. 3883-3892.
65. Gavrilova T. A., Leshcheva I. A. Building Collaborative Ontologies: A Human Factors Approach //Collaborative Knowledge in Scientific Research Networks. – IGI Global, 2015. – P. 305-324.
66. Gavrilova T., Leshcheva I. The interplay of knowledge engineering and cognitive psychology: learning ontologies creating //International Journal of Knowledge and Learning. – 2015. – V. 10. – №. 2. – P. 182-197.

67. Gavrilova T. A., Leshcheva I. A., Rumyantseva M. N. Knowledge elicitation methods taxonomy: Russian view // Lecture Notes in Computer Science. – 2011. – Vol. 6881 LNAI. – No Part 1. – P. 337-346.
68. Gavrilova T., Leshcheva I., Strakhovich E. Gestalt principles of creating learning business ontologies for knowledge codification //Knowledge Management Research & Practice. – 2015. – V. 13. – №. 4. – P. 418-428.
69. Genesereth M. R., Fikes R. Knowledge interchange format reference manual (version 3.0) //Computer Science Department, Stanford University. – 1992.
70. Genesereth M. R., Nilsson N. J. Logical Foundations of Artificial Intelligence // Los Altos, California : Morgan Kaufmann. – 1987. – 405 P.
71. Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., ... & Tu, S. W. The evolution of Protégé: an environment for knowledge-based systems development //International Journal of Human-computer studies. – 2003. – V. 58. – №. 1. – P. 89-123.
72. Giaretta P., Guarino N. Ontologies and knowledge bases towards a terminological clarification //Towards very large knowledge bases: knowledge building & knowledge sharing. – 1995. – P. 25-32.
73. Gómez-Pérez A., Fernandez-Lopez M., Corcho O. Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. – Springer-Verlag London, 2004. – 404 C.
74. Gruber T. R. A translation approach to portable ontology specifications //Knowledge acquisition. – 1993. – V. 5. – №. 2. – P. 199-220.
75. Gruber T. R. Toward principles for the design of ontologies used for knowledge sharing // International Journal of Human-Computer Studies. – 1995. – V. 43. – №. 5-6. – P. 907-928.
76. Hacherouf M., Bahloul S. N., Cruz C. Transforming XML documents to OWL ontologies: A survey //Journal of Information Science. – 2015. – V. 41. – №. 2. – P. 242-259.
77. Haghighi P. D. et al. Ontology-based service-oriented architecture for emergency management in mass gatherings //Service-Oriented Computing and Applications (SOCA), 2010 IEEE International Conference on. – IEEE, 2010. – P. 1-7.
78. Han L. et al. RDF123: from Spreadsheets to RDF //International Semantic Web Conference. – Springer, Berlin, Heidelberg, 2008. – P. 451-466.
79. Ivanović M., Budimac Z. An overview of ontologies and data resources in medical domains //Expert Systems with Applications. – 2014. – V. 41. – №. 11. – P. 5158-5166.
80. Langegger A., Wöß W. XLWrap–querying and integrating arbitrary spreadsheets with SPARQL //International Semantic Web Conference. – Springer, Berlin, Heidelberg, 2009. – P. 359-374.

81. Hu W., Qu Y. Discovering simple mappings between relational database schemas and ontologies. – Springer Berlin Heidelberg, 2007. – P. 225-238.
82. Isern D., Sánchez D., Moreno A. Ontology-driven execution of clinical guidelines //Computer methods and programs in biomedicine. – 2012. – V. 107. – №. 2. – P. 122-139.
83. Karp P. D., Chaudhri V. K., Thomere J. XOL: An XML-based ontology exchange language. – 1999.
84. Kifer M., Lausen G. F-logic: a higher-order language for reasoning about objects, inheritance, and scheme //ACM SIGMOD Record. – ACM, 1989. – V. 18. – №. 2. – P. 134-146.
85. Knublauch, H., Fergerson, R. W., Noy, N. F., & Musen, M. A. The Protégé OWL plugin: An open development environment for semantic web applications //The Semantic Web–ISWC 2004. – Springer Berlin Heidelberg, 2004. – P. 229-243.
86. Křemen P. et al. Ontology-driven mindmapping //Proceedings of the 8th International Conference on Semantic Systems. – 2012. – P. 125-132.
87. Lenat D. B., Guha R. V. The evolution of CycL, the Cyc representation language //ACM SIGART Bulletin. – 1991. – V. 2. – №. 3. – P. 84-87.
88. Leshcheva I., Begler A. A method of semi-automated ontology population from multiple semi-structured data sources //Journal of Information Science. – 2020. DOI: 10.1177/0165551520950243.
89. Leshcheva I., Gavrilova T. How the cognitive features testing can assist in evaluating collective ontology engineering //International Journal of High Performance Computing and Networking. – 2015. – V. 8. – №. 3. – P. 275-284.
90. Lubani M., Noah S. A. M., Mahmud R. Ontology population: Approaches and design aspects //Journal of Information Science. – 2019. – V. 45. – №. 4. – P. 502-515.
91. MacGregor R. Retrospective on LOOM //Information Sciences Institute, University of Southern California, Tech. Rep. – 1999.
92. Michel F., Montagnat J., Faron-Zucker C. A survey of RDB to RDF translation approaches and tools: Research Report. 2014. URL: http://hal.archives-ouvertes.fr/hal-00903568
93. Motta E. An overview of the OCML modelling language //the 8th Workshop on Methods and Languages. – 1998.
94. Musen M. A. The protégé project: a look back and a look forward //AI matters. – 2015. – V. 1. – №. 4. – P. 4-12.
95. Nederstigt L. J. et al. FLOPPIES: a framework for large-scale ontology population of product information from tabular data in e-commerce stores //Decision Support Systems. – 2014. – V. 59. – P. 296-311.
96. O'Connor M. J., Das A. Acquiring OWL ontologies from XML documents //Proceedings of the sixth international conference on Knowledge capture. – 2011. – P. 17-24.

97. O'Connor M. J., Halaschek-Wiener C., Musen M. A. Mapping master: a flexible approach for mapping spreadsheets to OWL //International Semantic Web Conference. – Springer, Berlin, Heidelberg, 2010. – P. 194-208.
98. Ozturk O. OPPCAT: Ontology population from tabular data //Journal of Information Science. – 2020. – V. 46. – №. 2. – P. 161-175.
99. Pan J. Z. et al. (ed.). Exploiting Linked Data and Knowledge Graphs in Large Organisations. – Cham : Springer, 2017.
100. Patlak, M., Nass, S., Henderson, I., Lashof, J. (Eds.), 2001. Mammography and Beyond: Developing Technologies for the Early Detection of Breast Cancer. National Academy Press, Washington, DC.
101. Petasis G. et al. Ontology population and enrichment: State of the art // Knowledge-driven multimedia information extraction and ontology evolution. – Springer-Verlag, 2011. – pp. 134-166.
102. Quix C., Kensche D., Li X. Matching of ontologies with xml schemas using a generic metamodel //OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". – Springer, Berlin, Heidelberg, 2007. – P. 1081-1098.
103. Reed S.L., Lenat D.B. Mapping ontologies into Cyc //AAAI 2002 Conference Workshop on Ontologies For The Semantic Web. – 2002. – P. 1-6.
104. Rodrigues, T., Rosa, P., Cardoso, J. Mapping XML to Exiting OWL ontologies, International Conference WWW/Internet 2006.
105. Rubin D.L. Creating and curating a terminology for radiology: Ontology modeling and analysis // J. Digit. Imaging. 2008. Vol. 21, № 4. P. 355–362.
106. Ruiz F., Hilera J. R. Using ontologies in software engineering and technology //Ontologies for software engineering and software technology. – Springer Berlin Heidelberg, 2006. – P. 49-102.
107. Sahoo S. S. et al. A survey of current approaches for mapping of relational databases to RDF //W3C RDB2RDF Incubator Group Report. – 2009. URL http://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SurveyReport_01082009.pdf. Sahoo, S. S., Halb, W., Hellmann, S., Idehen, K., Thibodeau Jr, T., Auer, S., ... & Ezzat, A.
108. Sanchez-Cohen I. et al. A decision support system for rainfed agricultural areas of Mexico //Computers and Electronics in Agriculture. – 2015. – V. 114. – P. 178-188.
109. Shadbolt N., Motta E., Rouge A. Constructing knowledge-based systems //Software, IEEE. – 1993. – V. 10. – №. 6. – P. 34-38.
110. Simperl E., Luczak-Rösch M. Collaborative ontology engineering: a survey //The Knowledge Engineering Review. – 2014. – V. 29. – №. 1. – P. 101-131.
111. Song F., Zacharewicz G., Chen D. An ontology-driven framework towards building enterprise semantic information layer //Advanced Engineering Informatics. – 2013. – Vol. 27. – №. 1. – P. 38-50.

112. Stadnicki A., Pietroń F. F., Burek P. Towards a Modern Ontology Development Environment //Procedia Computer Science. – 2020. – V. 176. – P. 753-762.
113. Stevens R., Aranguren M.E., Wolstencroft K., Sattler U., Drummond N., Horridge M., Rector A. Using OWL to model biological knowledge // International Journal of Human-Computer Studies. Volume 65, Issue 7, July 2007, P. 583–594.
114. Studer R. et al. Situation and Perspective of Knowledge Engineering //Knowledge Engineering and Agent Technology. – 2004. – P. 237-252.
115. Suárez-Figueroa M. C., Gómez-Pérez A., Fernández-López M. The NeOn methodology for ontology engineering //Ontology engineering in a networked world. – Springer Berlin Heidelberg, 2012. – P. 9-34.
116. Sure Y. et al. OntoEdit: Collaborative Ontology Development for the Semantic Web // Semant. Web — ISWC 2002. 2002. P. 221–235.
117. Svihla M., Jelinek I. Two layer mapping from database to RDF //Proceedings of Electronic Computers and Informatics (ECI). – 2004.
118. Swartout B. et al. Ontosaurus: a tool for browsing and editing ontologies //9th Banff Knowledge Aquisition for KNowledge-based systems Workshop. – 1996.
119. Tempich C. et al. An argumentation ontology for distributed, loosely-controlled and evolving engineering processes of ontologies (DILIGENT) //European Semantic Web Conference. – Springer, Berlin, Heidelberg, 2005. – pp. 241-256.
120. Tijerino Y. A. et al. Towards ontology generation from tables //World Wide Web. – 2005. – V. 8. – №. 3. – P. 261-285.
121. Uschold M. Demystifying Owl for the Enterprise (Synthesis Lectures on Semantic Web: Theory and Technology) //Morgan & Claypool Publishers, 2018. – P. 237.
122. Uschold M., Gruninger M. Ontologies and semantics for seamless connectivity //ACM SIGMod Record. – 2004. – V. 33. – №. 4. – P. 58-64.
123. Valarakos A. G. et al. Enhancing ontological knowledge through ontology population and enrichment //International Conference on Knowledge Engineering and Knowledge Management. – Springer, Berlin, Heidelberg, 2004. – P. 144-156.
124. Vassiliadis P., Simitsis A. Extraction, Transformation, and Loading // In: Liu L and Tamer M (eds) Encyclopedia of Database Systems. New York: Springer – 2009. – V. 10. – pp. 1095–1101.
125. Xiang Z. et al. Ontorat: Automatic generation of new ontology terms, annotations, and axioms based on ontology design patterns // J. Biomed. Semantics. 2015. Vol. 6, № 1. P. 1–10.
126. Yunianta A. et al. OntoDI: The methodology for ontology development on data integration //Int. J. Adv. Comput. Sci. Appl. – 2019. – V. 10. – №. 1. – P. 160-168.

127. Zagorulko Y. A., Borovikova O. I., Zagorulko G. B. The development of a system for automated ontology building based on heterogeneous ontology design patterns //Journal of Physics: Conference Series. – IOP Publishing, 2021. – V. 1715. – №. 1. – P. 012014.

# APPENDIX

List of rules for ascertainment of relations shown as a dotted line on the left side of Figure 22.

```
#Relation Characteristic relatesToPeriod Period
Characteristic(?it), Phase(?ph), relatesToPhase(?it, ?ph),
Period(?p), relatesToPeriod(?ph, ?p) -> relatesToPeriod(?it, ?p)

#Relation Characteristic relatesToDisease Disease
Characteristic(?it), Phase(?ph), relatesToPhase(?it, ?ph),
Disease(?d), relatesToDisease(?ph, ?d) -> relatesToDisease(?it, ?d)

#Value of hasTotalManifestation of first element if Manifestation not
equals -1
Pointer(pointer), Period(?per), hasFirst(pointer, ?per),
Characteristic(?h), relatesToPeriod(?h, ?per), hasManifestation(?h,
?number), notEqual(?number, -1) -> hasTotalManifestation(?h, ?number)

#Value of hasTotalManifestation of first element if Manifestation
equals -1
Pointer(pointer), Period(?per), hasFirst(pointer, ?per),
Characteristic(?h), relatesToPeriod(?h, ?per), hasManifestation(?h,
?number), equal(?number, -1) -> hasTotalManifestation(?h, 0)

#Value of hasTotalManifestation of next element if Manifestation not
equals -1
Period(?p1), Period(?p2), hasNext(?p1, ?p2), Characteristic(?h1),
relatesToPeriod(?h1, ?p1), Characteristic(?h2), relatesToPeriod(?h2,
?p2), Symptom(?sy), relatesToSymptom(?h1, ?sy), relatesToSymptom(?h2,
?sy), Disease(?di), relatesToDisease(?h1, ?di), relatesToDisease(?h2,
?di), hasTotalManifestation(?h1, ?tm), hasManifestation(?h2, ?m),
add(?sum, ?tm, ?m), notEqual(?m, -1) -> hasTotalManifestation(?h2,
?sum)
```

#Value of hasTotalManifestation of next element if Manifestation equals -1

Period(?p1), Period(?p2), hasNext(?p1, ?p2), Characteristic(?h1), relatesToPeriod(?h1, ?p1), Characteristic(?h2), relatesToPeriod(?h2, ?p2), Symptom(?sy), relatesToSymptom(?h1, ?sy), relatesToSymptom(?h2, ?sy), Disease(?di), relatesToDisease(?h1, ?di), relatesToDisease(?h2, ?di), hasTotalManifestation(?h1, ?tm), hasManifestation(?h2, ?m), add(?sum, ?tm, ?m), equal(?m, -1) -> hasTotalManifestation(?h2, ?tm)

#Value of hasTotalValue of next element if Manifestation equals -1

Characteristic(?h), hasTotalManifestation(?h, ?tm), hasSeverity(?h, ?s), hasModality(?h, ?m), multiply(?res1, ?s, ?m), multiply(?res2, ?res1, ?tm) -> hasTotalValue(?h, ?res2)

#Set value of GeneralPreliminary of first element of type Characteristic:

relatesToSymptom(?it, ?sy), hasTotalValue(?it, ?v), Symptom(?sy), hasFirst(pointer, ?sy), Characteristic(?it), Pointer(pointer) -> hasGeneralPreliminaryValue(?it, ?v)

#Calculate General, sum of consecutive elements of type Characteristic

Characteristic(?it2), Symptom(?sy1), Symptom(?sy2), Characteristic(?it1), relatesToPhase(?it1, ?h), relatesToPhase(?it2, ?h), Phase(?h), relatesToSymptom(?it1, ?sy1), relatesToSymptom(?it2, ?sy2), hasGeneralPreliminaryValue(?it1, ?pv), hasNext(?sy1, ?sy2), add(?res, ?pv, ?tv), hasTotalValue(?it2, ?tv) -> hasGeneralPreliminaryValue(?it2, ?res)

#Set General, value to element of type Phase

hasGeneralPreliminaryValue(?it, ?tv), relatesToSymptom(?it, ?sy), Symptom(?sy), hasLast(pointer, ?sy), relatesToPhase(?it, ?h), Characteristic(?it), Pointer(pointer), Phase(?h) -> hasGeneralTotalValue(?h, ?tv)

#Set value of MainOnlyPreliminary of first element of type Characteristic to 0 if its modality is less 3:

```
relatesToSymptom(?it, ?sy), Symptom(?sy), hasFirst(pointer, ?sy), Characteristic(?it), Pointer(pointer), hasModality(?it, ?m), lessThan(?m, 3) -> hasMainOnlyPreliminaryValue(?it, 0)

#Set value of MainOnlyPreliminary of first element of type Characteristic, if its modality is over 3:
relatesToSymptom(?it, ?sy), hasTotalValue(?it, ?v), Symptom(?sy), hasFirst(pointer, ?sy), Characteristic(?it), Pointer (pointer), hasModality(?it, ?m), greaterThan(?m, 3) -> hasMainOnlyPreliminaryValue(?it, ?v)

#Calculate MainOnly, sum of consecutive elements of type Characteristic, if modality of second element is over 3:
Characteristic(?it2), Symptom(?sy1), Symptom(?sy2), Characteristic(?it1), Phase(?h), relatesToPhase(?it1, ?h), relatesToPhase(?it2, ?h), relatesToSymptom(?it1, ?sy1), relatesToSymptom(?it2, ?sy2), hasMainOnlyPreliminaryValue(?it1, ?pv), hasNext(?sy1, ?sy2), add(?res, ?pv, ?tv), hasTotalValue(?it2, ?tv), hasModality(?it2, ?m), greaterThan(?m, 3) -> hasMainOnlyPreliminaryValue(?it2, ?res)

#Calculate MainOnly, sum of consecutive elements of type Characteristic, if modality of second element is less 3:
Characteristic(?it2), Symptom(?sy1), Symptom(?sy2), Characteristic(?it1), Phase(?h), relatesToPhase(?it1, ?h), relatesToPhase(?it2, ?h), relatesToSymptom(?it1, ?sy1), relatesToSymptom(?it2, ?sy2), hasMainOnlyPreliminaryValue(?it1, ?pv), hasNext(?sy1, ?sy2), hasModality(?it2, ?m), lessThan(?m, 3) -> hasMainOnlyPreliminaryValue(?it2, ?pv)

#Set MainOnly value to element of type Phase
hasMainOnlyPreliminaryValue(?it, ?tv), relatesToSymptom(?it, ?sy), Symptom(?sy), hasLast(pointer, ?sy), relatesToPhase(?it, ?h), Characteristic(?it), Pointer (pointer), Phase(?h) -> hasMainOnlyTotalValue(?h, ?tv)
```

List of rules for ascertainment of relations shown as a dotted line on the right side of Figure 22.

```
#Relation Diagnosis relatesToPhase Phase
Diagnosis(?dg), Disease(?di), relatesToDisease(?dg, ?di), Phase(?ph), relatesToDisease(?ph, ?di), Period(?pe), relatesToPeriod(?ph, ?pe), Patient(?pa), relatesToPatient(?dg, ?pa), relatesToPeriod(?pa, ?pe) -> relatesToPhase(?dg, ?ph)

#Relation Item relatesToCharacteristic Characteristic
Characteristic(?ha), Phase(?ph), relatesToPhase(?ha, ?ph), Diagnosis(?dg), relatesToPhase(?dg, ?ph), Item(?it), relatesToDiagnosis(?it, ?dg), Symptom(?sy), relatesToSymptom(?it, ?sy), relatesToSymptom(?ha, ?sy) -> relatesToCharacteristic(?it, ?ha)

#Relation Patient relatesToSymptom Symptom
Item(?it), Diagnosis(?dg), relatesToDiagnosis(?it, ?dg), Patient(?pa), relatesToPatient(?dg, ?pa), Symptom(?sy), relatesToSymptom(?it, ?sy), hasThisSymptom(?it, "да") -> relatesToSymptom(?pa, ?sy)

#Value of hasTotalValue, if hasThisSymptom equals «нет»
Item(?it), hasThisSymptom(?it, "нет") -> hasTotalValue(?it, 0)

# Value of hasTotalValue, if hasThisSymptom equals «да»
Item(?it), Characteristic(?ha), relatesToCharacteristic(?it, ?ha), hasThisSymptom(?it, "да"), hasTotalValue(?ha, ?tv) -> hasTotalValue(?it, ?tv)

#Set of value of GeneralPreliminary of first element of type Item:
Symptom(?sy), Pointer (pointer), hasFirst(pointer, ?sy), Item(?it), relatesToSymptom(?it, ?sy), hasTotalValue(?it, ?tv) -> hasGeneralPreliminaryValue(?it, ?tv)

#Calculate General, sum of consecutive elements of type Item
Symptom(?sy1), Symptom(?sy2), hasNext(?sy1, ?sy2), Item(?it1), Item(?it2), relatesToSymptom(?it1, ?sy1), relatesToSymptom(?it2,
```

?sy2), Diagnosis(?dg), relatesToDiagnosis(?it1, ?dg), relatesToDiagnosis(?it2, ?dg), hasGeneralPreliminaryValue(?it1, ?pv), hasTotalValue(?it2, ?tv), add(?res, ?pv, ?tv) -> hasGeneralPreliminaryValue(?it2, ?res)

#Set General, value to element of type Diagnosis
hasGeneralPreliminaryValue(?it, ?tv), relatesToSymptom(?it, ?sy), Symptom(?sy), hasLast(pointer, ?sy), relatesToDiagnosis(?it, ?h), Item(?it), Pointer (pointer), Diagnosis(?h) -> hasGeneralValue(?h, ?tv)

#Set value of MainOnlyPreliminary of first element of type of Item to 0, if its modality is less 3:
relatesToSymptom(?it, ?sy), Symptom(?sy), hasFirst(pointer, ?sy), Item(?it), Pointer (pointer), hasModality(?ha, ?m), lessThan(?m, 3), Characteristic(?ha), relatesToCharacteristic(?it, ?ha) -> hasMainOnlyPreliminaryValue(?it, 0)

#Set value of MainOnlyPreliminary of first element of type of Item if its modality is over 3:
relatesToSymptom(?it, ?sy), hasTotalValue(?it, ?v), Symptom(?sy), hasFirst(pointer, ?sy), Item(?it), Pointer(pointer), hasModality(?ha, ?m), greaterThan(?m, 3), Characteristic(?ha), relatesToCharacteristic(?it, ?ha) -> hasMainOnlyPreliminaryValue(?it, ?v)

#Calculate MainOnly sum of first element of type of Item, if modality of second element is over 3:
Item(?it2), Symptom(?sy1), Symptom(?sy2), Item(?it1), Diagnosis(?dg), relatesToDiagnosis(?it1, ?dg), relatesToDiagnosis(?it2, ?dg), relatesToSymptom(?it1, ?sy1), relatesToSymptom(?it2, ?sy2), hasMainOnlyPreliminaryValue(?it1, ?pv), hasNext(?sy1, ?sy2), add(?res, ?pv, ?tv), hasTotalValue(?it2, ?tv), Characteristic(?ha), relatesToCharacteristic(?it2, ?ha), hasModality(?ha, ?m), greaterThan(?m, 3) -> hasMainOnlyPreliminaryValue(?it2, ?res)

```
# Calculate MainOnly sum of first element of type of Item, if modality of second element is less 3:
Item(?it2), Symptom(?sy1), Symptom(?sy2), Item(?it1), Diagnosis(?dg), relatesToDiagnosis(?it1, ?dg), relatesToDiagnosis(?it2, ?dg), relatesToSymptom(?it1, ?sy1), relatesToSymptom(?it2, ?sy2), hasMainOnlyPreliminaryValue(?it1, ?pv), hasNext(?sy1, ?sy2), Characteristic(?ha), relatesToCharacteristic(?it2, ?ha), hasModality(?ha, ?m), lessThan(?m, 3) -> hasMainOnlyPreliminaryValue(?it2, ?pv)

#Set MainOnly value to element of type Diagnosis
hasMainOnlyPreliminaryValue(?it, ?tv), relatesToSymptom(?it, ?sy), Symptom(?sy), hasLast(pointer, ?sy), relatesToDiagnosis(?it, ?h), Item(?it), Pointer (pointer), Diagnosis(?h) -> hasMainOnlyValue(?h, ?tv)

#Value of hasGeneralRelativeValue instance of type Diagnosis:
Diagnosis(?dg), Phase(?ph), relatesToPhase(?dg, ?ph), hasGeneralValue(?dg, ?dtv), hasGeneralTotalValue(?ph, ?ptv), divide(?res, ?dtv, ?ptv), greaterThan(?ptv, 0) -> hasGeneralRelativeValue(?dg, ?res)

#Value of hasMainOnlyRelativeValue instance of type Diagnosis:
Diagnosis(?dg), Phase(?ph), relatesToPhase(?dg, ?ph), hasMainOnlyValue(?dg, ?dtv), hasMainOnlyTotalValue(?ph, ?ptv), divide(?res, ?dtv, ?ptv), greaterThan(?ptv, 0) -> hasMainOnlyRelativeValue(?dg, ?res)
```

**Table 2 —** The results of the work of the reasoner making diagnosis

| *Patient* | *Syndrome* | *a* | *b* | *c* | *d* |
|---|---|---|---|---|---|
| *Sample 1.*<br>*Sex: male*<br>*Age: 2 years 10 months*<br>*Symptoms :*<br>*• coarse facial features ,*<br>*• macrocephaly*<br>*• scaphocephaly*<br>*• deformity of hand joints*<br>*• stiffness of large joints*<br>*• hepatomegaly*<br>*• splenomegaly*<br>*• mental retardation*<br>*• cardiopathy*<br>*• kyphoscoliosis*<br>***Diagnosis : Mucopolysaccharidosis type II (Hunter syndrome)*** | Hurler | 125,6 | 47% | 111,0 | 52% |
| | Hurler-Scheie | 43,7 | 35% | 35,5 | 41% |
| | Maroteaux–Lamy quick | 6,6 | 15% | 1,6 | 4% |
| | Maroteaux–Lamy slow | 0,0 | 0% | 0,0 | 0% |
| | Morquio -A quick | 1,8 | 10% | 0,8 | 5% |
| | Morquio -A slow | 0,0 | 0% | 0,0 | 0% |
| | Morquio -B | 0,0 | 0% | 0,0 | 0% |
| | Sly | 30,0 | 61% | 24,6 | 85% |
| | Sanfilippo D | 0,0 | 0% | 0,0 | 0% |
| | Sanfilippo A | 6,4 | 50% | 5,0 | 86% |
| | Sanfilippo B | 3,1 | 36% | 2,5 | 100% |
| | Sanfilippo C | 0,0 | 0% | 0,0 | 0% |
| | Hunter -T | 34,2 | 41% | 29,4 | 39% |
| | Hunter -Л | 11,5 | 33% | 8,9 | 28% |
| | Scheie | 19,6 | 44% | 14,8 | 46% |
| *Sample 10*<br>*Sex: male*<br>*Age: 10 years*<br>*Symptoms :*<br>*• macrocephaly*<br>*• short neck*<br>*• coarse facial features*<br>*• hepatomegaly*<br>*• splenomegaly*<br>*• deformity of hand joints*<br>*• mental retardation*<br>***Diagnosis : Mucopolysaccharidosis type II (Hunter syndrome)*** | Hurler | 247,7 | 41% | 220,5 | 47% |
| | Hurler-Scheie | 126,8 | 38% | 98,0 | 44% |
| | Maroteaux–Lamy quick | 110,9 | 40% | 78,9 | 47% |
| | Maroteaux–Lamy slow | 13,4 | 20% | 6,8 | 16% |
| | Morquio -A quick | 41,0 | 26% | 33,6 | 27% |
| | Morquio -A slow | 12,0 | 22% | 10,8 | 23% |
| | Morquio -B | 24,6 | 25% | 21,0 | 34% |
| | Sly | 85,0 | 47% | 53,4 | 47% |
| | Sanfilippo D | 14,2 | 56% | 8,0 | 55% |
| | Sanfilippo A | 53,0 | 41% | 35,0 | 41% |
| | Sanfilippo B | 55,3 | 48% | 37,5 | 51% |
| | Sanfilippo C | 14,2 | 48% | 8,0 | 55% |
| | Hunter -T | 165,6 | 49% | 140,0 | 51% |
| | Hunter -Л | 53,6 | 53% | 36,0 | 45% |
| | Scheie | 67,2 | 46% | 52,4 | 65% |
| *Sample 11*<br>*Sex: female*<br>*Age: 9 years*<br>*Symptoms :*<br>*• Growth retardation*<br>*• Thickened skin*<br>*• Hypertrichosis (HP)*<br>*• Scaphocephaly*<br>*• Short neck (LP)*<br>*• Kyphoscoliosis*<br>*• Stiffness of large joints (HP)*<br>*• Keeled chest*<br>*• Hernia*<br>*• Hepatomegaly (HP)*<br>*• Splenomegaly (LP)*<br>*• Clouding of the cornea*<br>***Diagnosis : Mucopolysaccharidosis type VI (Maroteaux–Lamy syndrome)*** | Hurler | 346,1 | 57% | 277,5 | 60% |
| | Hurler-Scheie | 223,8 | 67% | 168,0 | 76% |
| | Maroteaux–Lamy quick | 158,6 | 57% | 101,2 | 61% |
| | Maroteaux–Lamy slow | 43,2 | 66% | 27,6 | 63% |
| | Morquio -A quick | 104,8 | 67% | 91,2 | 72% |
| | Morquio -A slow | 37,6 | 69% | 34,0 | 73% |
| | Morquio -B | 73,6 | 74% | 56,0 | 90% |
| | Sly | 116,5 | 64% | 73,9 | 65% |
| | Sanfilippo D | 10,7 | 42% | 2,9 | 20% |
| | Sanfilippo A | 58,1 | 45% | 24,1 | 28% |
| | Sanfilippo B | 51,6 | 45% | 19,6 | 27% |
| | Sanfilippo C | 14,1 | 48% | 2,9 | 20% |
| | Hunter -T | 140,6 | 41% | 103,0 | 37% |
| | Hunter -Л | 78,1 | 77% | 58,9 | 73% |
| | Scheie | 87,4 | 60% | 58,0 | 72% |

| *Patient* | *Syndrome* | *a* | *b* | *c* | *d* |
|---|---|---|---|---|---|
| *Sample 12*<br>*Sex: male*<br>*Age: 8 years*<br>*Symptoms :*<br>*• Growth retardation*<br>*• Coarse facial features*<br>*• Short neck (HP)*<br>*• Kyphoscoliosis (HP)*<br>*• Stiffness of large joints (HP)*<br>*• Hepatomegaly (HP)*<br>*• Splenomegaly (LP)*<br>*• Mental retardation(HP)*<br>***Diagnosis : Mucopolysaccharidosis type VI (Maroteaux–Lamy syndrome)*** | Hurler | 316,5 | 52% | 300,5 | 65% |
| | Hurler-Scheie | 187,2 | 56% | 164,0 | 74% |
| | Maroteaux–Lamy quick | 147,3 | 53% | 119,3 | 71% |
| | Maroteaux–Lamy slow | 30,0 | 46% | 21,6 | 50% |
| | Morquio -A quick | 70,2 | 45% | 58,4 | 46% |
| | Morquio -A slow | 24,4 | 45% | 20,8 | 44% |
| | Morquio -B | 54,0 | 54% | 48,0 | 77% |
| | Sly | 119,4 | 66% | 86,4 | 77% |
| | Sanfilippo D | 17,1 | 68% | 10,9 | 75% |
| | Sanfilippo A | 77,1 | 60% | 59,1 | 70% |
| | Sanfilippo B | 76,7 | 67% | 57,1 | 78% |
| | Sanfilippo C | 17,1 | 58% | 10,9 | 75% |
| | Hunter -T | 169,0 | 50% | 141,0 | 51% |
| | Hunter -Л | 67,8 | 67% | 51,0 | 63% |
| | Scheie | 75,4 | 52% | 52,4 | 65% |
| *Sample 13*<br>*Sex: male*<br>*Age: 1 year 1 months*<br>*Symptoms :*<br>*• Macrocephaly (HP)*<br>*• Coarse facial features (HP)*<br>*• Clouding of the cornea*<br>*• Short neck (LP)*<br>*• Growth retardation*<br>*• Kyphoscoliosis (LP)*<br>*• Hyperlordosis (HP)*<br>*• Funnel chest*<br>*• Stiffness of large joints (HP)*<br>*• Hernia (LP)*<br>*• Cardiopathy(HP)*<br>***Diagnosis : Mucopolysaccharidosis type VI (Maroteaux–Lamy syndrome)*** | Hurler | 124,4 | 46% | 106,8 | 50% |
| | Hurler-Scheie | 66,9 | 53% | 47,5 | 55% |
| | Maroteaux–Lamy quick | 37,4 | 87% | 33,6 | 93% |
| | Maroteaux–Lamy slow | 3,2 | 100% | 3,2 | 100% |
| | Morquio -A quick | 13,5 | 77% | 12,9 | 84% |
| | Morquio -A slow | 2,2 | 92% | 2,2 | 100% |
| | Morquio -B | 15,6 | 90% | 15,6 | 100% |
| | Sly | 33,2 | 68% | 21,2 | 73% |
| | Sanfilippo D | 0,0 | 0% | 0,0 | 0% |
| | Sanfilippo A | 6,4 | 50% | 2,0 | 34% |
| | Sanfilippo B | 4,9 | 56% | 0,5 | 20% |
| | Sanfilippo C | 1,2 | 100% | 0,0 | 0% |
| | Hunter -T | 61,8 | 74% | 60,2 | 81% |
| | Hunter -Л | 30,7 | 87% | 30,5 | 95% |
| | Scheie | 32,9 | 74% | 23,9 | 74% |
| *Sample 14*<br>*Sex: female*<br>*Age: 6 months*<br>*Symptoms :*<br>*• Coarse facial features*<br>*• Clouding of the cornea (LP)*<br>*• Hypertrichosis (HP)*<br>*• Cardiopathy (HP)*<br>*• Hepatomegaly (HP)*<br>*• Splenomegaly(LP)*<br>***Diagnosis : Mucopolysaccharidosis type VII (Sly syndrome)*** | Hurler | 45,2 | 47% | 41,2 | 50% |
| | Hurler-Scheie | 4,0 | 20% | 4,0 | 45% |
| | Maroteaux–Lamy quick | 0,0 | 0% | 0,0 | 0% |
| | Maroteaux–Lamy slow | 0,0 | 0% | 0,0 | 0% |
| | Morquio -A quick | 0,0 | 0% | 0,0 | 0% |
| | Morquio -A slow | 0,0 | 0% | 0,0 | 0% |
| | Morquio -B | 0,0 | 0% | 0,0 | 0% |
| | Sly | 3,0 | 32% | 3,0 | 86% |
| | Sanfilippo D | 0,0 | 0% | 0,0 | 0% |
| | Sanfilippo A | 0,4 | 25% | 0,0 | 0% |
| | Sanfilippo B | 0,4 | 25% | 0,0 | 0% |
| | Sanfilippo C | 0,0 | 0% | 0,0 | 0% |
| | Hunter -T | 2,2 | 16% | 0,0 | 0% |
| | Hunter -Л | 0,0 | 0% | 0,0 | 0% |
| | Scheie | 0,0 | 0% | 0,0 | 0% |
| *Sample 15*<br>*Sex: female* | Hurler | 196,6 | 44% | 185,4 | 53% |
| | Hurler-Scheie | 112,7 | 45% | 100,7 | 60% |

| *Patient* | *Syndrome* | *a* | *b* | *c* | *d* |
|---|---|---|---|---|---|
| *Age: 5 years*<br>*Symptoms :*<br>*• Coarse facial features (HP)*<br>*• Cardiopathy (HP)*<br>*• Hepatomegaly (HP)*<br>*• Splenomegaly (LP)*<br>*• Mental retardation (HP)*<br>*• Hearing loss (HP)*<br>*• Kyphoscoliosis (HP)*<br>***Diagnosis : Mucopolysaccharidosis type VII (Sly syndrome)*** | Maroteaux–Lamy quick | 37,9 | 31% | 30,3 | 36% |
| | Maroteaux–Lamy slow | 10,0 | 45% | 7,6 | 49% |
| | Morquio -A quick | 25,4 | 36% | 20,4 | 36% |
| | Morquio -A slow | 5,2 | 28% | 4,8 | 32% |
| | Morquio -B | 12,8 | 23% | 8,8 | 23% |
| | Sly | 47,9 | 47% | 43,5 | 70% |
| | Sanfilippo D | 3,6 | 95% | 2,4 | 100% |
| | Sanfilippo A | 29,1 | 58% | 24,9 | 77% |
| | Sanfilippo B | 25,1 | 57% | 21,3 | 77% |
| | Sanfilippo C | 4,2 | 68% | 2,4 | 100% |
| | Hunter -T | 54,9 | 31% | 47,1 | 32% |
| | Hunter -Л | 18,1 | 30% | 11,1 | 22% |
| | Scheie | 51,8 | 51% | 35,6 | 63% |
| *Sample 16*<br>*Sex: female*<br>*Age: 6 years*<br>*Symptoms :*<br>*• Mental retardation*<br>*• Hypertrichosis (HP)*<br>*• Coarse facial features*<br>*• Macrocephaly (HP)*<br>*• Growth retardation*<br>*• Funnel chest (HP)*<br>*• Kyphoscoliosis (HP)*<br>***Diagnosis : Mucopolysaccharidosis type VII (Sly syndrome)*** | Hurler | 134,5 | 30% | 92,7 | 27% |
| | Hurler-Scheie | 64,8 | 26% | 43,0 | 26% |
| | Maroteaux–Lamy quick | 32,1 | 27% | 19,5 | 23% |
| | Maroteaux–Lamy slow | 2,4 | 11% | 1,6 | 10% |
| | Morquio -A quick | 16,9 | 24% | 15,3 | 27% |
| | Morquio -A slow | 3,6 | 19% | 3,4 | 23% |
| | Morquio -B | 20,6 | 37% | 18,8 | 48% |
| | Sly | 48,9 | 48% | 35,5 | 57% |
| | Sanfilippo D | 0,0 | 0% | 0,0 | 0% |
| | Sanfilippo A | 21,9 | 44% | 19,5 | 60% |
| | Sanfilippo B | 19,3 | 44% | 16,5 | 60% |
| | Sanfilippo C | 0,0 | 0% | 0,0 | 0% |
| | Hunter -T | 94,9 | 54% | 88,3 | 60% |
| | Hunter -Л | 32,4 | 53% | 30,8 | 60% |
| | Scheie | 28,8 | 29% | 14,0 | 25% |
| *Sample 2*<br>*Sex: female*<br>*Age: 5 years*<br>*Symptoms :*<br>*• coarse facial features*<br>*• Thickened skin*<br>*• stiffness of large joints*<br>*• clouding of the cornea*<br>*• hepatomegaly*<br>*• splenomegaly*<br>*• cardiopathy*<br>***Diagnosis : Mucopolysaccharidosis type I (Hurler syndrome)*** | Hurler | 186,7 | 42% | 178,3 | 51% |
| | Hurler-Scheie | 119,7 | 48% | 113,5 | 68% |
| | Maroteaux–Lamy quick | 32,1 | 27% | 18,3 | 22% |
| | Maroteaux–Lamy slow | 6,0 | 27% | 2,0 | 13% |
| | Morquio -A quick | 21,8 | 31% | 12,0 | 21% |
| | Morquio -A slow | 4,8 | 26% | 3,2 | 21% |
| | Morquio -B | 7,6 | 14% | 2,4 | 6% |
| | Sly | 44,1 | 43% | 39,9 | 64% |
| | Sanfilippo D | 1,2 | 32% | 0,0 | 0% |
| | Sanfilippo A | 15,1 | 30% | 10,9 | 34% |
| | Sanfilippo B | 14,7 | 33% | 10,9 | 39% |
| | Sanfilippo C | 1,8 | 29% | 0,0 | 0% |
| | Hunter -T | 39,1 | 22% | 29,1 | 20% |
| | Hunter -Л | 18,4 | 30% | 10,6 | 21% |
| | Scheie | 64,8 | 64% | 56,6 | 100% |
| *Sample 3*<br>*Sex: female*<br>*Age: 3 months*<br>*Symptoms :* | Hurler | 38,0 | 40% | 38,0 | 46% |
| | Hurler-Scheie | 2,5 | 13% | 2,5 | 28% |
| | Maroteaux–Lamy quick | 0,0 | 0% | 0,0 | 0% |
| | Maroteaux–Lamy slow | 0,0 | 0% | 0,0 | 0% |

| *Patient* | *Syndrome* | *a* | *b* | *c* | *d* |
|---|---|---|---|---|---|
| • *cardiopathy*<br>• *stiffness of large joints*<br>• *hepatomegaly*<br>• *splenomegaly*<br>• *coarse facial features*<br>• *kyphoscoliosis*<br>***Diagnosis : Mucopolysaccharidosis type I (Hurler syndrome)*** | Morquio -A quick | 0,0 | 0% | 0,0 | 0% |
| | Morquio -A slow | 0,0 | 0% | 0,0 | 0% |
| | Morquio -B | 0,0 | 0% | 0,0 | 0% |
| | Sly | 3,0 | 32% | 3,0 | 86% |
| | Sanfilippo D | 0,0 | 0% | 0,0 | 0% |
| | Sanfilippo A | 0,0 | 0% | 0,0 | 0% |
| | Sanfilippo B | 0,0 | 0% | 0,0 | 0% |
| | Sanfilippo C | 0,0 | 0% | 0,0 | 0% |
| | Hunter -T | 1,8 | 13% | 0,0 | 0% |
| | Hunter -Л | 0,0 | 0% | 0,0 | 0% |
| | Scheie | 0,0 | 0% | 0,0 | 0% |
| *Sample 4*<br>*Sex: female*<br>*Age: 2 yearsa*<br>*Symptoms :*<br>• *hepatomegaly*<br>• *mental retardation*<br>• *coarse facial features*<br>• *deformity of hand joints*<br>***Diagnosis : Mucopolysaccharidosis type III C (Sanfilippo C syndrome)*** | Hurler | 85,2 | 32% | 85,0 | 40% |
| | Hurler-Scheie | 27,5 | 22% | 27,5 | 32% |
| | Maroteaux–Lamy quick | 2,4 | 6% | 0,0 | 0% |
| | Maroteaux–Lamy slow | 0,0 | 0% | 0,0 | 0% |
| | Morquio -A quick | 1,2 | 7% | 0,8 | 5% |
| | Morquio -A slow | 0,0 | 0% | 0,0 | 0% |
| | Morquio -B | 0,0 | 0% | 0,0 | 0% |
| | Sly | 14,4 | 29% | 12,0 | 41% |
| | Sanfilippo D | 0,0 | 0% | 0,0 | 0% |
| | Sanfilippo A | 6,2 | 48% | 5,0 | 86% |
| | Sanfilippo B | 3,1 | 36% | 2,5 | 100% |
| | Sanfilippo C | 0,0 | 0% | 0,0 | 0% |
| | Hunter -T | 13,7 | 16% | 10,5 | 14% |
| | Hunter -Л | 4,4 | 13% | 2,0 | 6% |
| | Scheie | 13,6 | 31% | 13,6 | 42% |
| *Sample 5*<br>*Sex: male*<br>*Age: 4 years*<br>*Symptoms :*<br>• *coarse facial features*<br>• *mental retardation*<br>• *keeled chest*<br>***Diagnosis : Mucopolysaccharidosis type III B (Sanfilippo B syndrome)*** | Hurler | 77,3 | 17% | 67,5 | 19% |
| | Hurler-Scheie | 26,0 | 10% | 15,0 | 9% |
| | Maroteaux–Lamy quick | 10,9 | 9% | 7,5 | 9% |
| | Maroteaux–Lamy slow | 1,8 | 8% | 0,0 | 0% |
| | Morquio -A quick | 8,2 | 12% | 7,4 | 13% |
| | Morquio -A slow | 2,3 | 12% | 2,3 | 15% |
| | Morquio -B | 6,4 | 12% | 2,4 | 6% |
| | Sly | 31,5 | 31% | 25,5 | 41% |
| | Sanfilippo D | 0,0 | 0% | 0,0 | 0% |
| | Sanfilippo A | 18,5 | 37% | 18,5 | 57% |
| | Sanfilippo B | 16,5 | 38% | 16,5 | 60% |
| | Sanfilippo C | 0,0 | 0% | 0,0 | 0% |
| | Hunter -T | 27,5 | 16% | 27,5 | 19% |
| | Hunter -Л | 3,0 | 5% | 3,0 | 6% |
| | Scheie | 16,4 | 16% | 14,0 | 25% |
| *Sample 7*<br>*Sex:*<br>*Age: 7 years*<br>*Symptoms :*<br>• *stiffness of large joints*<br>• *clouding of the cornea*<br>• *coarse facial features* | Hurler | 185,2 | 30% | 174,0 | 37% |
| | Hurler-Scheie | 122,8 | 37% | 112,0 | 51% |
| | Maroteaux–Lamy quick | 98,1 | 35% | 58,5 | 35% |
| | Maroteaux–Lamy slow | 22,8 | 35% | 12,0 | 28% |
| | Morquio -A quick | 51,8 | 33% | 40,8 | 32% |
| | Morquio -A slow | 21,8 | 40% | 18,8 | 40% |

| *Patient* | *Syndrome* | *a* | *b* | *c* | *d* |
|---|---|---|---|---|---|
| • *deformity of hand joints*<br>• *growth retardation*<br>• *cardiopathy*<br>***Diagnosis : Mucopolysaccharidosis type VI (Maroteaux–Lamy syndrome)*** | Morquio -B | 32,2 | 32% | 21,0 | 34% |
| | Sly | 63,8 | 35% | 50,0 | 44% |
| | Sanfilippo D | 4,7 | 19% | 2,9 | 20% |
| | Sanfilippo A | 37,3 | 29% | 34,1 | 40% |
| | Sanfilippo B | 39,6 | 34% | 34,6 | 47% |
| | Sanfilippo C | 5,3 | 18% | 2,9 | 20% |
| | Hunter -T | 108,6 | 32% | 79,8 | 29% |
| | Hunter -Л | 31,2 | 31% | 28,8 | 36% |
| | Scheie | 76,4 | 53% | 50,4 | 63% |
| *Sample 8*<br>*Sex:*<br>*Age: 20 years*<br>*Symptoms :*<br>• *deformity of hand joints*<br>• *hernia*<br>• *coarse facial features*<br>• *clouding of the cornea*<br>• *stiffness of large joints*<br>• *cardiopathy*<br>***Diagnosis : Mucopolysaccharidosis type VI (Maroteaux–Lamy syndrome)*** | Hurler | 185,2 | 30% | 174,0 | 37% |
| | Hurler-Scheie | 104,8 | 31% | 76,0 | 34% |
| | Maroteaux–Lamy quick | 104,5 | 38% | 80,9 | 48% |
| | Maroteaux–Lamy slow | 27,2 | 42% | 20,0 | 46% |
| | Morquio -A quick | 51,0 | 33% | 40,0 | 32% |
| | Morquio -A slow | 21,8 | 40% | 18,8 | 40% |
| | Morquio -B | 31,2 | 31% | 20,0 | 32% |
| | Sly | 66,5 | 37% | 62,5 | 55% |
| | Sanfilippo D | 5,8 | 23% | 2,4 | 17% |
| | Sanfilippo A | 44,8 | 35% | 29,6 | 35% |
| | Sanfilippo B | 46,2 | 40% | 34,6 | 47% |
| | Sanfilippo C | 9,8 | 33% | 2,4 | 17% |
| | Hunter -T | 100,6 | 30% | 71,8 | 26% |
| | Hunter -Л | 16,6 | 16% | 14,2 | 18% |
| | Scheie | 75,8 | 52% | 50,4 | 63% |
| *Sample 9*<br>*Sex: male*<br>*Age: 2 years*<br>*Symptoms :*<br>• *hernia*<br>• *mental retardation*<br>• *coarse facial features*<br>• *stiffness of large joints*<br>• *deformity of hand joints*<br>• *hepatomegaly*<br>• *splenomegaly*<br>***Diagnosis : Mucopolysaccharidosis type II (Hunter syndrome)*** | Hurler | 138,2 | 52% | 138,0 | 65% |
| | Hurler-Scheie | 46,7 | 37% | 35,5 | 41% |
| | Maroteaux–Lamy quick | 8,8 | 20% | 6,4 | 18% |
| | Maroteaux–Lamy slow | 1,6 | 50% | 1,6 | 50% |
| | Morquio -A quick | 4,4 | 25% | 4,0 | 26% |
| | Morquio -A slow | 1,2 | 50% | 1,2 | 55% |
| | Morquio -B | 3,6 | 21% | 3,6 | 23% |
| | Sly | 26,0 | 53% | 23,6 | 81% |
| | Sanfilippo D | 0,0 | 0% | 0,0 | 0% |
| | Sanfilippo A | 9,6 | 75% | 5,0 | 86% |
| | Sanfilippo B | 6,3 | 72% | 2,5 | 100% |
| | Sanfilippo C | 1,2 | 100% | 0,0 | 0% |
| | Hunter -T | 22,4 | 27% | 18,4 | 25% |
| | Hunter -Л | 6,7 | 19% | 4,1 | 13% |
| | Scheie | 18,2 | 41% | 14,8 | 46% |