ХИМИКОТЕХНОЛОГИЧЕН И МЕТАЛУРГИЧЕН УНИВЕРСИТЕТ
ФАКУЛТЕТ ПО ХИМИЧНО И СИСТЕМНО ИНЖЕНЕРСТВО
КАТЕДРА:“ АВТОМАТИЗАЦИЯ НА ПРОИЗВОДСТВОТО”

Д И П Л О М Н А Р А Б О Т А

Тема: **“РАЗРАБОТВАНЕ НА ОНТОЛОГИЧЕН МОДЕЛ НА ПРОДУКТИ ОТ ПРОКАТНИ ПРОИЗВОДСТВА”**

Дипломант: Костадин Янкулов фак. № МС-0529

София, 2011 г.

Специалност:*“Информационни технологии”*

Образователно-квалификационна степен:*МАГИСТЪР*

Дипломант: Костадин Янкулов фак. № МС-0529

/. . . . . /

Ръководител на дипломната работа: . . . . . .

/ гл. ас. Д. Гочева /

За рецензент определям: . . . . . .

Ръководител на катедра: . . . . . .

/ доц. д-р В. Цочев /

UNIVERSITY OF CHEMICAL TECHNOLOGY AND METALLURGY - SOFIA
FACULTY OF CHEMICAL AND SYSTEM ENGINEERING
DEPARTMENT OF INDUSTRIAL AUTOMATION

DIPLOMA THESIS

Title: **DEVELOPMENT OF ONTOLOGY MODEL OF ROLLED STEEL PRODUCTS**

Specialty: **Information Technology**

Student: **Kostadin Yankulov Fac. № MC 0529**

/. . . . . /

Supervisor: **. . . . . .**

/ Assist. Prof. D.Gocheva/

HeadofDepartment**……**

/Assoc. Prof. V. Tzotchev, PhD/

Sofia, 2011

ХИМИКОТЕХНОЛОГИЧЕН И МЕТАЛУРГИЧЕН УНИВЕРСИТЕТ

ФАКУЛТЕТ ПО СИСТЕМНО И ХИМИЧНО ИНЖЕНЕРСТВО
Катедра: "АВТОМАТИЗАЦИЯ НА ПРОИЗВОДСТВОТО"

Специалност: *"Информационни технологии"*

Образователно-квалификационна степен: *МАГИСТЪР*

Утвърдил:
Ръководител на катедра АП:.....
/доц. д-р В.К. Цочев/

З А Д А Н И Е
ЗА ДИПЛОМНА РАБОТА

на студента: Костадин Янкулов фак. № МС-0529

Тема на дипломната работа

РАЗРАБОТВАНЕ НА ОНТОЛОГИЧЕН МОДЕЛ НА ПРОДУКТИ ОТ ПРОКАТНИ ПРОИЗВОДСТВА

1. Запознаване с областта на онтологиите и разработката на онтологични модели.
2. Запознаване и работа с програмния продукт за създаване на онтологии Protege.
3. Създаване на онтологичен модел на прокатните подукти, използвани в металургичната промишленост в среда на Protege.
4. Изводи

Дата на получаване
на заданието 16.09.2011 Студент:.....

Срок за предаване
на дипломната работа
22.11.2011.... Ръководител:.....
/ гл.ас. Д. Гочева /

СЪДЪРЖАНИЕ

УВОД

Важно условие за успешното прилагане на информационните трхнологии в предприятията е осигуряването на качествени комуникации, обмен и споделяне на информация, координация и оптимизация на решения и процеси с цел постигането на по-висока производителност, гъвкавост и качество.Все по-важно значение придобиват методите и средствата за формулиране, структуриране и извличане на информация във вид на данни и знания. Онтологиите са модели за описание на знания. Именно онтологиите формират най-общата представа за обектите на изследване. За съвременното общество е характерно постоянно усложняване и изменение на средата, в която работят и се адаптират организациите и хората. Различните групи потребители, които се занимават с анализ и обработка на специализирана информация, използват различна терминология, като често се срещат различни означения за едни и същи понятия. Всичко това значително усложнява взаимното разбиране. Това налага разработване на модели за представяне на знания, които да обезпечават автоматизираната обработка на информацията на семантично ниво.

Онтологиите представляват формални, явни описания на термините в различни предметни области и връзките между тези термини. Различните онтологии определят семантиката на конкретна предметна област и способстват за установяването на връзките между значението на нейните елементи. Проектирането и разработката на приложни онтологии представлява жизнено необходим етап в процеса на създаване на информационни и управляващи системи. Използвайки онтологии и базирани на тях системи бази данни и системи бази знания могат да се систематизират и многократно използват знания, да се извличат знания когато и където е нужно, като по този начин се съкращава времето за разработка и цената на информационните системи, а именно те са основните параметри, които всеки желае да ограничи до минимум.

Един от големите проблеми при въвеждането на интегрирана информационна система SAP/R3 в “Стомана Индъстри” АД (собственост на SIDENOR – Steel Products Manufacturing Company S.A.) през 2003 година е системата за управление на данните за продуктите. Номенклатурата на готовата продукция включва най-различни означения за типа на готовите продукти, качеството на стоманата, в зависимост от пазара и производителя, за размерите, за класовете на точност и за други показатели.

Предложен е онтологичен модел на продукти от прокатни производства, на базата на който се илюстрира подход за създаване на система за класификация на продуктите от стоманодобивната промишленост.

Изпплзвани са данни за продуктите на SIDENOR – Steel Products Manufacturing Company S.A и справочни данни за продукти от прокатни производства.

ПЪРВА ГЛАВА

1.КРАТЪК ОБЗОР НА РАЗРАБОТКАТА НА ОНТОЛОГИЧНИ МОДЕЛИ

1.1. Видове бази знания, онтологично базиран подход, предимства на онтологичния подход

Системите базирани на знание са системи основани на използването на методите и средствата на изкуствения интелект. Техни основни компоненти са базата знания и механизъм за извличане на изводи. Съществуват два основни типа бази знания:

- **Машинно читаеми бази знания** – те съхраняват знанието във вид удобен за четене от компютъра, така че да е възможно да се разсъждава дедуктивно по фактите.Те съдържат набор от данни обикновено представен като логически правила, които описват знанието по логически съгласуван начин. Структурата на записаните данни, типовете и връзките между тях може да се дефинират чрез онтология.Обикновено се използват логически оператори за описание на фактите в базата като И (съединение), ИЛИ (допълнение), импликация и отрицание.След това се използва логическа дедукция за „разсъждаване" върху фактите в базата. Такива бази знания се използват в различни области като: Изкуствен интелект, експертни системи, семантични мрежи и други.

- **Човеко-читаеми бази знания.** Те са проектирани за да могат потребителите да търсят в тях нужното им знание и да го извличат или допълват. Обикновено се използват за направата на списъци със съвети, често задавани въпроси и споделяне на информация между служителите в организациите или с клиентите. В тях се съхраняват статии, публикации, официални документи, техническа документация или подобни ориентирани към клиентите документи. Информацията в тях обикновено се търси с помощта на търсеща машина или по категории.

Една от най-трудните задачи в областта на изкуствения интелект е да се създаде база от знания, която да обхваща общоприетите знания, тоест тези които се знаят от нормалните хора. Подобна база знания ще намира приложение в различни области, изискващи човешка експертиза като машинен превод, разпознаване на естествен език, разпознаване на обекти и други дейности изискващи интелигентна реакция.

Тъй като обхватът и разнообразието на тези знания са много големи, обикновено се прилага онтологично базиран подход, тоест изграждат се мета-онтологии за различните области, които в последствие се разширяват със знания за областите и се свързват в една обща онтология, използвана в базата знания.Предимствата на такива онтологии са: логическа съгласуваност между термините, яснота и прозрачност на значенията на термините и ясно съответствие между потребителските изисквания и дизайна на софтуера [1].Онтологичният подход е доказал своите качества в разработката на големи и сложни бази знания в редица сфери като например правосъдие, автоматично генериране на софтуер,застрахователно дело и други.За целите на този труд ще се ограничим до разглеждането на онтологиите, тъй като те се явяват основна част на базите знания и в някои аспекти може да се каже че са неразличими от тях.

1.2. Обзор на онтологиите

1.2.1. Определение, основни елементи и приложение на онтологиите

Онтологията като понятие е възникнала преди много години. От философска гледна точка, онтология е почти всеки начин на класифициращо описание на нещата, които ни заобикалят. От инженерна гледна точка, или от гледна точка на хората, работещи със знания, онтологията е: „явна формална спецификация на споделена концептуализация“ [2] или по-общо „онтологията може да има много форми, но тя непременно включва речник на термините и спецификация на техния смисъл.

Последното включва дефиниции и определяне на релации между понятията, които заедно структурират разглежданата област и ограничават възможните интерпретации на термините...“ [3].

Съгласно Грубер [4], знанието в онтологиите може да се специфицира посредством 5 компонента: понятия, релации, функции, аксиоми и екземпляри. Най-често понятията са организирани в таксономии. Аксиомите, изразяващи твърдения, които винаги са истинни, са включени в онтологията с цел ограничаване на информацията в нея, верифициране на нейната коректност и за извличане на неявна, нова информация.

Сферите на приложение на онтологиите са най разнообразни като на пример: по-лесна комуникация между хора, интероперативност между компютърни системи, като основа за формално кодиране в софтуерното инженерство, като структура или мета-данни за търсене в хранилищата на информация. По-долу даваме някои предимства от употребата им:

- Онтологията улеснява комуникацията между различните страни участващи в процеса на обмен на информация, чрез точното дефиниране на семантиката на термините и предоставянето на общ речник, който да служи за посредник между тях.
- Възможностите за класификация и логически изводи разширяват възможностите за складиране, обработка и повторната употреба на данните.
- Информацията е по-лесно разбираема и недвусмислена и може да се използва както от специалисти в съответните области (технолози), така и от не специалисти.

Употребата на онтологично-базиран подход при разработването на информационни системи улеснява по-лесното намиране на информацията и повторната

й употреба в последствие. Той служи, както за източник на справочна информация, така и като вид справочник подпомагащ работата на технолозите.Онтологичният подход предоставя на различните специалисти удобен интерфейс, чрез който те да обменят информация по между си, да я събират по удобен начин за последващата многократна употреба. Използването на онтология, която задава връзки и ограничения между класовете термини, позволява извличането и използването на информация, която не е зададена в явен вид.

1.2.2. Методологии за разработка на онтологии

Обзорите в литературата на тема “онтологии” разграничават няколко типа методологии: методология на Ушолд и Кинг [5], методология на Гомез-Перез (**Methontology**) [6], методология на Грюнингер [7].

Методология	**Формализация**	Използван подход за изграждане	Приложимост за V&V
Uschold &King	Няма насоки за формализиране	От вътре на вън	Слаба(текстови дефиниции)
METHONTO LOGY	Използва инструмента ОДЕ	От вътре на вън	Да
Gruninger & Fox	Използва логика от първи ред(KIF)	От вътре на вън	Да
Bernaras	Няма насоки за формализиране	Отгоре - надолу	Не
SENSUS	Използва семантични мрежи	Отдолу - нагоре	Да

Първите две методологии са по-близки помежду си и включват 9 основни фази на разработката: планиране, спецификация, извличане на знания, концептуализация, интеграция, имплементация, оценяване, документиране и поддръжка. Основните различия в двете методологии са във фазите на концептуализацията и имплементацията. Във фазата на концептуализация, Ушолд и Кинг използват като средство за междинно представяне естествения език, докато Гомез и Перез предлагат използването на таблици на термините и атрибутите, и бинарни таблици на релациите. Дефинициите в междинното представяне могат да бъдат трансформирани автоматично чрез продукта **ODE** в език **Ontolingua** [8]. Във фазата на концептуализация могат да бъдат използвани различни подходи, в зависимост от предпочитанията или конкретната цел.

При разработката на онтологията **SENSUS** [9] е използван подход, който е коренно различен от горните методологии. Той се основава на обобщаване и сливане на вече готови онтологии и речникови данни, използвайки общ скелет за свързването им в една голяма обща онтология. В последствие новите онтологии се изграждат чрез добавяне и свързване към съществуващите термини на такива значими за дадената област и последващо редактиране на дърветата от понятия свързани с тези термини.

Най-общо съществуват три подхода за разработката на онтология: „отгоре-надолу“, “отдолу-нагоре“ и „отвътре-навън“ в зависимост от това в каква посока се осъществява подреждането на термините (от по-общи към по-частни, от по-частни към по-общи, или от група термини, съществени за съответната област, които в последствие биват обобщавани или специализирани според нуждите). Изборът на подход зависи от задачата, избраната методология и начина на изграждане на онтологията (от вече съществуваща база знания или на чисто), но повечето методологии препоръчват използването на последния подход [10] като най-гъвкав.

По отношение на формализацията в процеса на разработка на онтологиите, можем да кажем, че методологията на Грюнингер е единствената широко известна формална методология, която включва следните фази: мотивиращи сценарии, дефиниране на неформални въпроси на компетентност, дефиниране на терминология на базата на логика от първи ред, формално дефиниране на въпросите на компетентност, дефиниране на аксиоми на базата на логика от първи ред и оценяване на онтологията чрез теореми за пълнота.

Може да се каже, че **SENSUS** онтологията също се базира на формален подход, но той се основава на употребата на семантични мрежи, които задават йерархията на термините и връзките между тях. Методологията на **SENSUS** налага общ скелет на всички приложения, които я използват, като по този начин улеснява обмена на информация между тях.

Можем да се заключи, че на този етап на развитие не съществува консенсус относно използването на стандартна методология за изграждане на онтологии. Изборът на подходяща методология се определя в зависимост от вида и целите, които преследва конкретната онтология или приложение. Следователно би било по-добре да не се обвързваме с една конкретна методология, а да използваме набор от методологии, тъй като всяка методология има своите предимства в зависимост от задачата, за която е била проектирана.

1.2.3. Видове онтологии

Онтологиите могат да бъдат класифицирани на базата на два различни показателя – ниво на формалност и ниво на общност [11]. По отношение на нивото на формалност онтологиите биват: неформални, полуформални и формални. Според нивото на общност се разграничават 7 вида онтологии: онтологии от високо ниво, общи

онтологии, онтологии на предметна област, онтологии на задачи, онтологии на методи, приложни онтологии и мета-онтологии.

Онтологиите от високо ниво описват най-общите понятия, като: време, място, обект, материя, събитие, действие и т.н., които не зависят от конкретната задача или предметната област. Онтологиите на предметната област, на задачите и на методите описват терминологичните речници на обхванатата предметна област, решаваните задачи и използваните за целта методи. Те специализират понятията включени в онтологиите от високо ниво и включват общите понятия и релации за конкретната предметна област, задача или метод.

Приложните онтологии са тясно специализирано описание на понятията и връзките на базата на онтологиите от горните две нива. Те включват понятия, специфични както за съответната област, така и за конкретната задача или метод. В много от случаите тези понятия съответстват на ролите, които играят отделните елементи на предметната област, в случаите на конкретна дейност.

Добър вариант за разбиране на онтологиите и техните приложения предлагат Ушолд и Джаспер [3]. Те предлагат три главни категории онтологични приложения:

- неутрално редактиране
- споделен достъп до информация
- индексиране за търсене

Под неутрално редактиране се разбира редактирането не само на онтологията, но и на общи оперативни данни и последващото им транслиране на специфични езици. Тук онтологиите улесняват работата на редакторите, тъй като се налага само редактиране на онтологията и свързаните с нея данни, а транслирането става посредством автоматизирани инструменти.

Споделен достъп до информация имаме когато една и съща информация трябва да се използва от много хора или софтуерни агенти всеки със собствен поглед върху данните. Тук онтологията играе ролята на посредник като подпомага различните агенти да разбират правилно термините и значението им.

При индексирането за търсене, онтологиите се използват като индекси в хранилища за данни, и по този начин осигуряват по висока скорост на търсене на необходимата информация. Онтологиите които се използват в този случай са сравнително прости таксономии задаващи взаимовръзките между термините.

1.3. Език за описание на онтологии - OWL

Развитието на семантичния **Web** и **XML** технологиите доведе до дефинирането на различни **XML**-базирани онтологични езици като **XOL** (**Ontology Exchange Language**), **SHOE** и **OML** (**Ontology Markup Language**). От друга страна, като основни езици за описание на мета-данни в **Web**, се използват езиците **RDF** (**Resource Description Framework**) и **RDF Schema. OIL** е разширение на RDF и е първият език за представяне на онтологии в **Web**, като представлява част от стандартите на **W3C**. **DAML+OIL** се отличава с много добри възможности за описание на ресурси и е планиран като основа за изграждане на **OWL** (**Web Ontology Language**) [12]. **OWL** е разработен като речниково разширение на **RDF**, има три варианта: **OWLLite, OWLDL** и **OWL Full** като е проектиран за използване в приложения, свързани с обработката на информация, а не само с нейното представяне.

- **OWL-Lite**е синтактически прост подезик. Той е предназначен за използване само в ситуации при които се ползва проста йерархия от класове.

- **OWL-DL** представлява разширение на **OWL-Lite** и се основава на **DL-Description Logics** (Дескриптивна логика). Поради това, онтологиите, изградени чрез подлежат на автоматичен логически анализ. Ето защо е възможно автоматично да се определи класификационната йерархия и да се провери за несъответствия в дадена онтология подчинена на **OWL-DL**. Това ръководство е фокусирано именно върху**OWL-DL**.
- **OWL-Full** е най-пълният и изчерпателен **OWL** подезик. Предназначен е за използване в ситуации, при които голямата изчерпателност е много по-необходима от това да се гарантира точността на вземане на решения или пълнотата при определяне в езика. Ето защо е невъзможно автоматичното вземане на решения при **OWL-Full** онтологии.

1.3.1. Дефиниране на класове

- **owl:class**

Класът дефинира група от обекти, които са обединени по това, че имат общи свойства. Например **Ivan** и **Maria** са представители на клас **Person**.Класовете могат да образуват йерархии с помощта на **rdfs:subClassOf**.Има вграден клас **Thing** който неявно е базов клас на всички класове в **OWL**.Също така има вграден клас **Nothing**, който представлява клас от който няма никакви инстанции и всеки друг **OWL** клас го има за подклас.

<owl:Class rdf:ID="Human"/>

- **rdfs:subClassOf**

Класова йерархия може да се изгражда с помощта на едно или повече посочвания, че конкретния клас е подклас на някой друг клас. Например класът **Person** може да се декларира като подклас на класът **Mammal**.И така ако някоя екземпляр е **Person** то той

е и **Mammal**.Представителите на клас **Person** имат всички свойства на класът **Mammal** плюс допълнителните декларирани в класът **Person**.

<owl:Class rdf:ID=“Person">

<rdfs:subClassOf rdf:resource="#Mammal" />

</owl:Class>

- **rdfs:equivalentClass**

Позволява декларацията че някой клас е идентичен със съществуващ друг клас. Има същите свойства като него и неговото описание.

<owl:Class rdf:about="#US_President">

<equivalentClass rdf:resource="#PrincipalResidentOfWhiteHouse"/>

</owl:Class>

- **rdfs:disjointWith**

Позволява декларацията че някой клас няма продължение което да е продължение на съществуващ друг клас.Наследниците на класа **Man** не мога да са наследници на класа **Woman.OWL Lite** не позволява употребата на **rdfs:disjointWith**.

<owl:Class rdf:about="#Man">

<owl:disjointWith rdf:resource="#Woman"/>

</owl:Class>

1.3.2. Дефиниране на свойства

OWL различава две основни категории свойства които този който изгражда онтологията може да декларира.

- **owl:ObjectProperty**– свойства, които свързват обект с друг обект.
- **owl:DatatypeProperty**– свойства, които свързват обект са поле данни. И двете наследяват класа **rdf:Property**.

<owl:ObjectProperty rdf:ID="hasParent"/>

<owl:ObjectProperty rdf:ID="hasRelative"/>

<owl:ObjectProperty rdf:ID="hasSibling"/>

<owl:DatatypeProperty rdf:ID="hasAge"/>

- **rdfs:subPropertyOf**

Йерархия на свойствата може да се създава чрез една или повече декларации, че дадено свойство е подсвойство на друго съществуващо свойство на конкретния клас или негов базов клас.

<owl:ObjectProperty rdf:ID="hasMother"><rdfs:subPropertyOf rdf:resource="#hasParent"/></owl:ObjectProperty>

- **rdfs:domain**

За да се ограничи областта на обектите които могат да имат дадено свойство се използва **rdfs:domain**.Ако за дадено свойство е декларирана област, то може да се прилага само на обекти които наследяват обекти от областта.

<owl:ObjectProperty rdf:ID="hasBankAccount"><rdfs:domain><owl:Class><owl:unionOf rdf:parseType="Collection"><owl:Class rdf:about="#Person"/><owl:Class rdf:about="#Corporation"/></owl:unionOf></owl:Class></rdfs:domain></owl:ObjectProperty>

- **rdfs:range**

За ограничение на обхвата на стойностите на едно свойство се използва **rdfs:range**. Ако за дадено свойство е деклариран обхват на стойностите, то всички негови стойности трябва да наследяват класове от обхвата.

<owl:ObjectProperty rdf:ID="husband"><rdfs:domain rdf:resource="#Woman" /><rdfs:range rdf:resource="#Man" /></owl:ObjectProperty>

- **owl:equivalentProperty**

Две свойства могат да бъдат отбелязани за еквивалентни. Това може да се ползва за създаване на синонимни свойства. Ако две свойства са декларирани за еквивалентни и даден обект X е свързан с друг обект У чрез едното те също са свързани и чрез еквивалентното му.

<owl:ObjectProperty rdf:ID="hasLeader">

<owl:equivalentPropertyrdf:resource="#hasHead"/>

</owl:ObjectProperty>

- **owl:inverseOf**

Свойствата имат посока, но практически на почти всяко свойство съответства някакво в обратна посока например **hasChild** има обратно **hasParent.**Това се декларира с помощта на **owl:inverseOf.**

<owl:ObjectProperty rdf:ID="hasChild"><owl:inverseOf rdf:resource="#hasParent"/></owl:ObjectProperty>

- **owl:FunctionalProperty**

Има свойства които даден обект може да има не повече от веднъж.Всички **objectProperties** и **datatypeProperties** могат да се отбележат като **functionalProperties**

<owl:ObjectProperty rdf:ID="husband">

<rdf:type rdf:resource="&owl;FunctionalProperty" />

<rdfs:domain rdf:resource="#Woman" />

<rdfs:range rdf:resource="#Man" />

</owl:ObjectProperty>

Което е еквивалентно на:

<owl:ObjectProperty rdf:ID="husband">

<rdfs:domain rdf:resource="#Woman" />

<rdfs:range rdf:resource="#Man" />

</owl:ObjectProperty>

<owl:FunctionalProperty rdf:about="#husband" />

Заради наследствеността между FunctionalProperty и ObjectProperty

- **owl:InverseFunctionalProperty**

Свойства чиито резултат може да принадлежи само на един обект се декларират чрез **owl:InverseFunctionalProperty.**

<owl:InverseFunctionalProperty rdf:ID="biologicalMotherOf">

<rdfs:domain rdf:resource="#Woman"/>

<rdfs:range rdf:resource="#Human"/>

</owl:InverseFunctionalProperty>

- **owl:TransitiveProperty**

За деклариране на транзитивни свойства се ползва **owl:TransitiveProperty**.Такива свойства са ако **(x, y)** са в едно и също свойство като **(y, z)** то и **(x, z)** са в същото свойство.

<owl:TransitiveProperty rdf:ID="subRegionOf">

<rdfs:domain rdf:resource="#Region"/>

<rdfs:range rdf:resource="#Region"/>

</owl:TransitiveProperty>

Еквивалентно на

<owl:ObjectProperty rdf:ID="subRegionOf">

<rdf:type rdf:resource="&owl;TransitiveProperty"/>

<rdfs:domain rdf:resource="#Region"/>

<rdfs:range rdf:resource="#Region"/>

</owl:ObjectProperty>

Заради наследствеността между **TransitiveProperty и ObjectProperty**

- **owl:SymmetricProperty**

Свойства които са двупосочни се декларират чрез **owl :SymmetricProperty.**Такива са свойства за които ако е изпълнено (**x, y**) то и (**y, x**) е изпълнено.За такива свойства Областта на действие(**domain**)и обхвата на резултатите(range)съвпадат**.**

<owl:SymmetricProperty rdf:ID="friendOf">

<rdfs:domain rdf:resource="#Human"/>

<rdfs:range rdf:resource="#Human"/>

</owl:SymmetricProperty>

1.3.3. Ограничения на свойствата

- **owl:Restriction**

OWL позволява да бъдат налагани ограничения.На това как свойствата могат да се ползват от инстанциите на клас**.**Чрез **owl:onProperty**се указва за кое точно свойство се отнася ограничението.Ограничения могат да бъдат налагани на всички видове свойства.

<owl:Restriction>

<owl:onProperty rdf:resource="(име на свойство)" />

(спецификация на ограничението)

</owl:Restriction>

- **owl:allValuesFrom**

Това ограничение налага на всички стойности на едно свойство да са от определен тип.Ограничението с налага за конкретен клас и ако имаме ограничение за свойството **hasDaughter** да са от тип **Women** за класа **Person**, то за класа **Cat** нямаме такова ограничение.Ако **x** е **Person** и **(x, y)** са в **hasDaughter** то **y** представител на **Women**.

<owl:Restriction>

<owl:onProperty rdf:resource="#hasDaughter" />

<owl:allValuesFrom rdf:resource="#Women" />

</owl:Restriction>

- **owl:someValuesFrom**

Това ограничение налага, че поне една от стойностите на дадено свойство в някой клас е от определен тип. Например поне един от родителите е Физик.

<owl:Restriction>

<owl:onProperty rdf:resource="#hasParent" />

<owl:someValuesFrom rdf:resource="#Physician" />

</owl:Restriction>

- **owl:hasValue**

Това ограничение налага че поне една от стойностите на дадено свойство в някой клас е с определена стойност. Например поне един от родителите е Ivan.Това ограничение не е включено в **OWL Lite.**

<owl:Restriction>

<owl:onProperty rdf:resource="#hasParent" />

<owl:hasValue rdf:resource="#Ivan" />

</owl:Restriction>

- **owl:maxCardinality**

Указва максималния брой стойности на определено свойство за определен клас.Примерно ограничението обект от клас **Person** да има най-много 2 родителя**.**

<owl:Restriction>

<owl:onProperty rdf:resource="#hasParent" />

<owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2

</owl:maxCardinality>

</owl:Restriction>

- **owl:minCardinality**

Указва минималния брой стойности на определено свойство за определен клас. Примерно ограничението обект от клас **Person** да има най-малко 2 родителя**.**

<owl:Restriction>

<owl:onProperty rdf:resource="#hasParent" />

<owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">2

</owl:minCardinality>

</owl:Restriction>

- **owl:minCardinality**

Указва минималния брой стойности на определено свойство за определен клас.Примерно ограничението обект от клас **Person** да има най-малко 2 родителя**.**

<owl:Restriction>

<owl:onProperty rdf:resource="#hasParent" />

<owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">2

</owl:minCardinality>

</owl:Restriction>

- **owl:sameAs**

За деклариране на еднаквост се използва **owl:sameAs**.Чрез него се декларира че два обекта са един и същ обект**.**

<rdf:Description rdf:about="#William_Jefferson_Clinton">

<owl:sameAs rdf:resource="#BillClinton"/>

</rdf:Description>

- **owl:differentFrom**

За да се покаже че два обекта са различни обекти се използва **owl:differentFrom.**

<Opera rdf:ID="Don_Giovanni"/>

<Opera rdf:ID="Nozze_di_Figaro">

<owl:differentFrom rdf:resource="#Don_Giovanni"/>

</Opera>

<Opera rdf:ID="Cosi_fan_tutte">

<owl:differentFrom rdf:resource="#Don_Giovanni"/>

<owl:differentFrom rdf:resource="#Nozze_di_Figaro"/>

</Opera>

- **owl:differentFrom**

За да се покаже че два обекта са различни обекти се използва **owl:differentFrom**

- **owl:allDifferent**

Множество от обекти може да се декларират като различни помежду си чрез използването на **owl:allDefferent**.

- **owl:oneOf**

Един клас може да бъде представен чрез списък на всички обекти които могат да го представляват (дните от седмицата).

1.3.4. Сечение, обединение, допълнение на класове

Чрез тези три конструкции(**Intersection, union, complement**) в **OWL** са представени трите стандартни оператора **И, ИЛИ** и **ОТРИЦАНИЕ (AND, OR,**

NOT) . Те съдържат вложени описания на класове, един за **ОТРИЦАНИЕТО** и повече за **И** и **ИЛИ**.

- **owl:intersectionOf**

Описва клас чиито продължения притежават точно тези обекти които са членове на всички продължения в листа с описания.

- **owl:unionOf**

Описва клас чиито продължения притежават тези обекти които са членове на някое от продълженията в листа с описания

- **owl:complementOf**

Описва клас за чиито продължения съдържат точно тези обекти коитоНЕ принадлежат на продълженията на класа описан като обект на **owl:complementOf.**

ВТОРА ГЛАВА

2. СОФТУЕРНА СРЕДА ЗА СЪЗДАВАНЕ НА ОНТОЛОГИИ PROTÉGÉ (OWL)

2.1. Въведение в Protégé/OWL

Protégé е цялостен софтуерен инструмент, използван от системни разработчици и областни експерти при разработката на системи, базирани на знания. Известни са многобройни приложения, разработени с **Protégé**, използвани за решаването на разнородни задачи и вземане на решения в различни области.

Protégéе известен и широко разпространенсвободно достъпен редактор за онтологии с отворен код. Продуктът позволява лесното разширяване и адаптиране към други среди и езици, като моделите може да бъдат редактирани и записвани в различни формати включително **CLIPS, RDF, XML, UML** и релационни бази данни.

OWL компонентата на **Protégé**, може да бъде използвана за разработване и редактиране на **OWL** онтологии, за логически анализ на онтологиите посредством външни за продукта логически анализатори (**reasoner**), за въвеждане на индивиди в онтологията, за извършване на различни заявки за търсене, аналогично на класическа система база данни. Като разширение на **Protégé, OWL** компонентата използва предимствата на широк кръг потребители, библиотеки от компоненти и гъвкава архитектура.

2.2. Създаване на проект с Protégé/OWL

Създаването на проект в **Protégé/OWL** обикновено изисква преминаването през следните етапи:

- Дефиниране на йерархия от класове и подкласове, разделяне на класовете
- Дефиниране на йерархия от обектни свойства
- Дефиниране видовете обектни свойства

- Определяне на област и обхват на свойствата
- Дефиниране на свойства тип данни и техните характеристики
- Въвеждане на екземпляри
- Редактиране на формуляри
- Създаване на заявки

2.2.1. Стартиране на Protégé

2.2.2. Създаване на класове

Менюто **"OWL Класове"** показва йерархията на класовете на онтологията, позволява на разработчиците да създават и редактират класове и показва резултата от класификацията.Съществуват няколкото възможни подхода за развиване на йерархията на класовете:

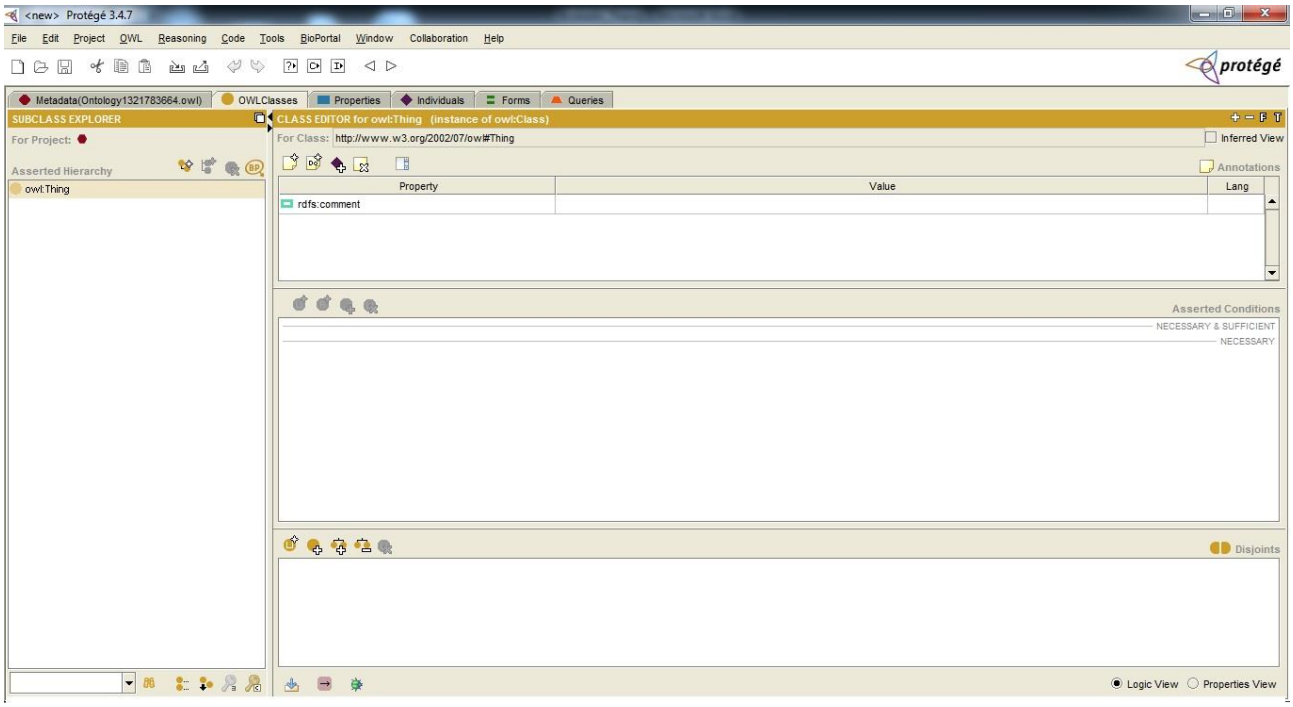
- Процесът на **низходящо** развитие започва от определянето от най-общите понятия на областта с последваща конкретизация на понятията.
- Процесът на **възходящо** развитие започва от определянето на най-конкретните класове, клони на йерархията, с последващо групиране на тези класове в по-общите понятия.
- Процесът на **комбинирано** развитие е комбинацията на низходящия и възходящия подходи: първо се определят по-забележителните понятия, а след това, съответно, се обобщават и ограничават.

Никой от тези три метода не е по-добър от другите два по своята същност. Избирането на подход зависи от личния възглед на разработчика за областта. Ако разработчикът е склонен да разглежда областта от горе надолу, тогава за него, вероятно, по-подходящ ще е низходящият метод. Често за много разработчици на онтологии най-подходящ се оказва

комбинираният метод, т.е. понятия, които, са разположени „в средата”, имат тенденция да бъдат най-нагледните понятия в областта.

Ако разработчикът на онтологии е предразположен да направи първо най-обща класификация на понятията, тогава за него по-подходящ ще е низходящият метод. Ако започне с даването на конкретни примери, тогава възходящият метод е по-подходящ.

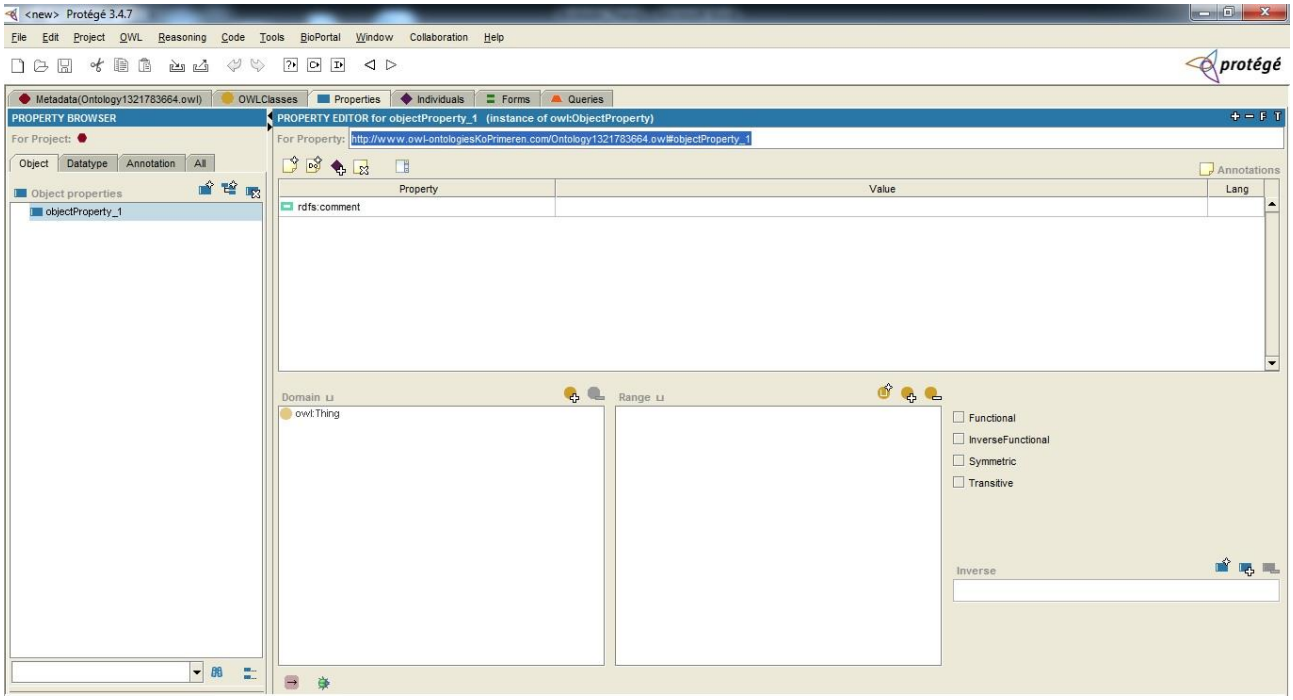
Независимо от това кой метод ще бъде избран, първата стъпка е определянето на класовете. Класовете се организират в йерархична таксономия, като се поставя въпроса: ако обект е екземпляр на един клас, ще бъде ли задължително (т.е. по определение) екземпляр от някой друг клас? Ако клас А е над-клас на класа Б, то всеки екземпляр на класа Б също е екземпляр на А. С други думи, клас Б представя понятие, което се явява разновидност на А.



Фиг.1: Меню **„OWL Classes”**

2.2.3. Създаване на свойства

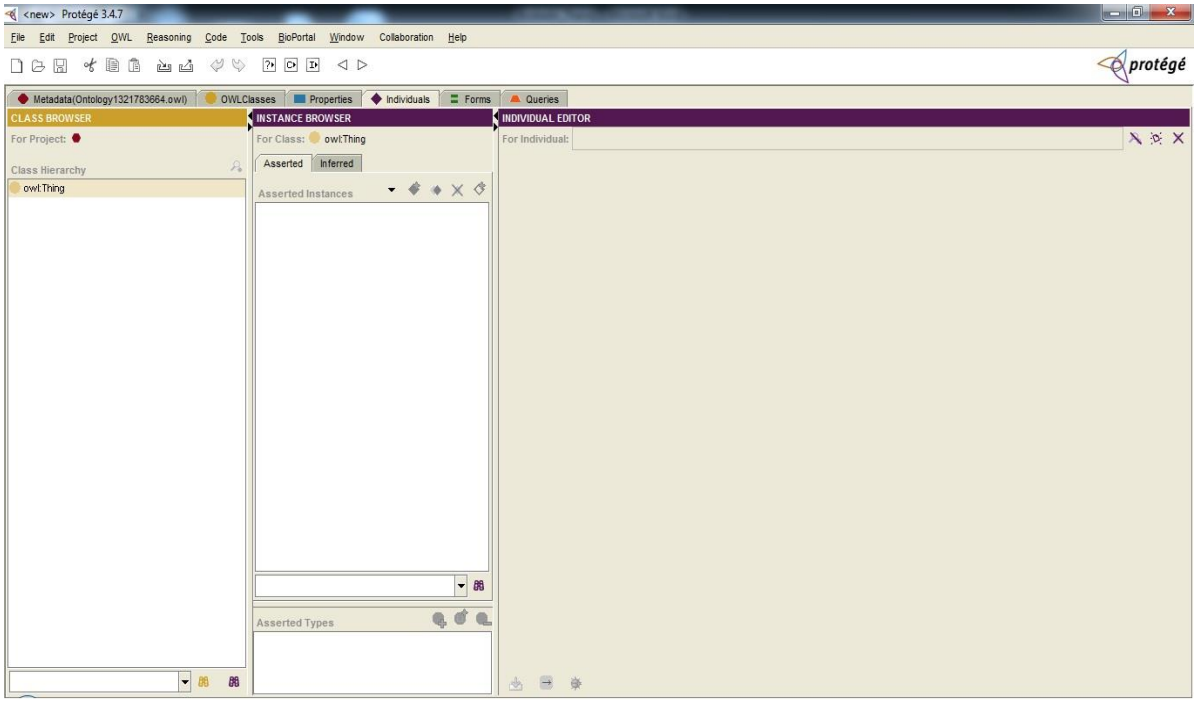
Менюто **„Properties”(“Свойства”)**може да бъде използвано за създаване и редактиране на свойствата в онтологията. **Свойствата**дават възможност да се дефинират общи факти за класовете и по-специфични такива за индивидите. Свойствата са връзки между двойки обекти; за тях също както и за класовете в **Protégé/OWL**създава йерархична структура. Езикът **OWL** дефинирадва типа свойства: свойства оттип данни (data type**)** – описват релации между индивидите на класовете и техните стойности и **обектни свойства(object type)**, дефиниращи релации между класовете. За обектните свойствав **Protégé**се задават**област (domain)** и **обхват (range)**.Обектните свойствата може да бъдат определени и като **инверсни , транзитивни , симетрични,**или**функционални**.



Фиг.2: Меню **„Properties”**

2.2.4. Въвеждане на индивиди

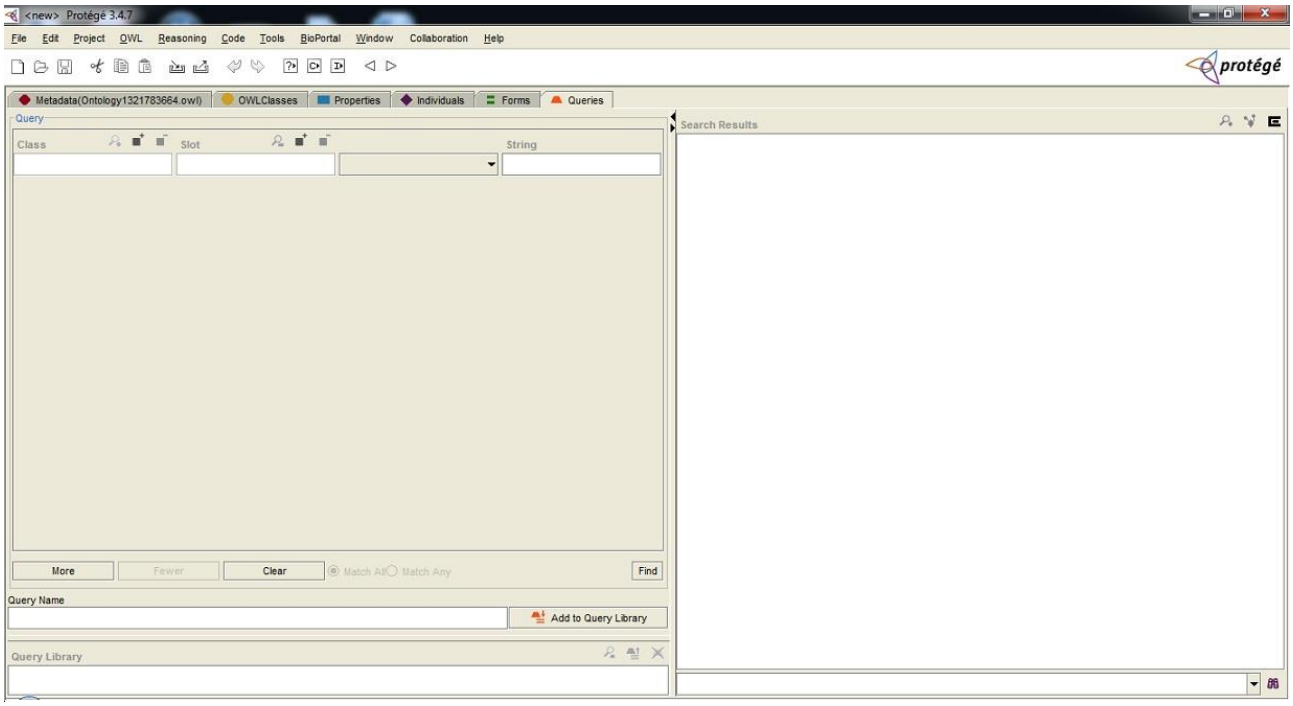
Менюто за създаване на индивиди на класовете в йерархията е представено на фиг.3.3. За определяне на отделния индивид от класа е необходимо (1) да се избере клас, (2) да се създаде отделен индивид от този клас и (3) да се въведат стойностите в свойствата.



Фиг.3: Меню **„Individuals”**

2.2.5. Създаване и записване на запитване

Менюто **`Queries`** представено на фиг.4 позволява да бъдат създавани потребителски заявки към модела за намиране на индивиди, които съвпадат с критерии, определени от потребителя. За да се създаде запитване, трябва да се изберат един или повече класове и/или едно или повече свойства. Запитванията могат да бъдат съхранявани в **Query Library** (библиотеката за заявки) за следващи търсения.



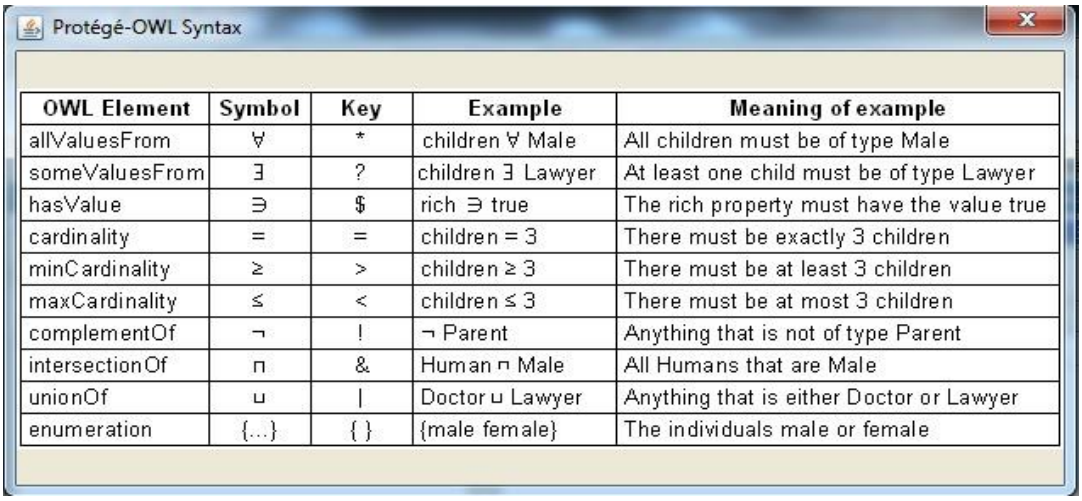
Фиг.4: Меню **„Queries”**

2.2.6. Промяна на формуляри

Формулярите определят начина на въвеждане и представяне на информацията за отделните индивиди. Те се генерират автоматично в продукта за всеки клас в зависимост от вида на конкретните свойства тип данни, които са асоциирани с класа. Потребителят може лесно да променя вида и начина на представяне на данните във всеки формуляр (да размества и скрива полета и др.)

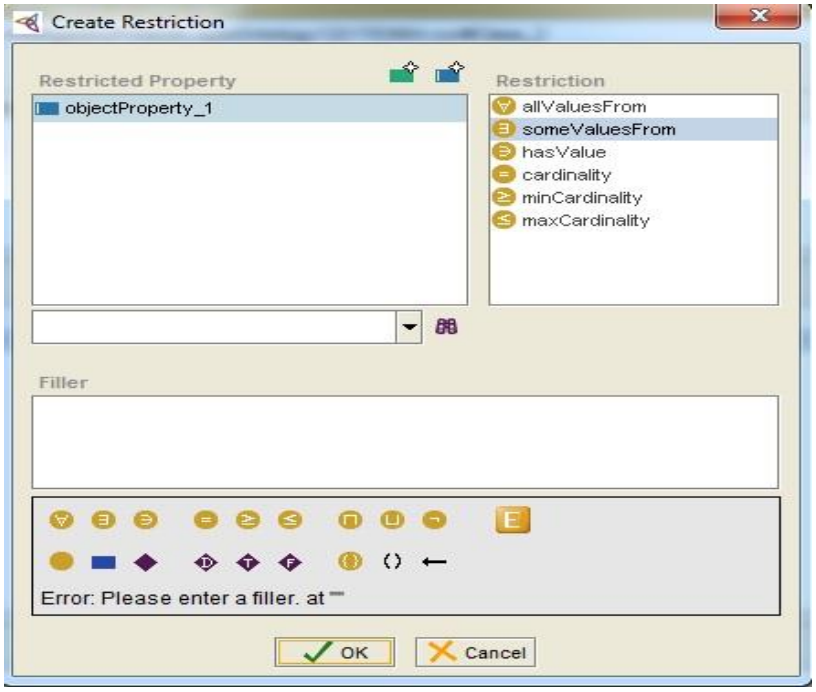
2.2.7. Редактиране на OWL логически изрази

Едно от най-важните средства в **Protégé/OWL** е редактора за логически изрази. Синтаксисът на **OWL** е представен на фиг.5.**OWL** компонентата осигурява удобен редактор на изрази, който позволява на потребителите бързо да събират изрази с мишката или клавиатурата (фиг.6).



OWL Element	Symbol	Key	Example	Meaning of example
allValuesFrom	∀	*	children ∀ Male	All children must be of type Male
someValuesFrom	∃	?	children ∃ Lawyer	At least one child must be of type Lawyer
hasValue	∋	$	rich ∋ true	The rich property must have the value true
cardinality	=	=	children = 3	There must be exactly 3 children
minCardinality	≥	>	children ≥ 3	There must be at least 3 children
maxCardinality	≤	<	children ≤ 3	There must be at most 3 children
complementOf	¬	!	¬ Parent	Anything that is not of type Parent
intersectionOf	⊓	&	Human ⊓ Male	All Humans that are Male
unionOf	⊔	|	Doctor ⊔ Lawyer	Anything that is either Doctor or Lawyer
enumeration	{...}	{ }	{male female}	The individuals male or female

Фиг.5: Синтаксис на **OWL**



Фиг.6: Меню **„Restriction"**

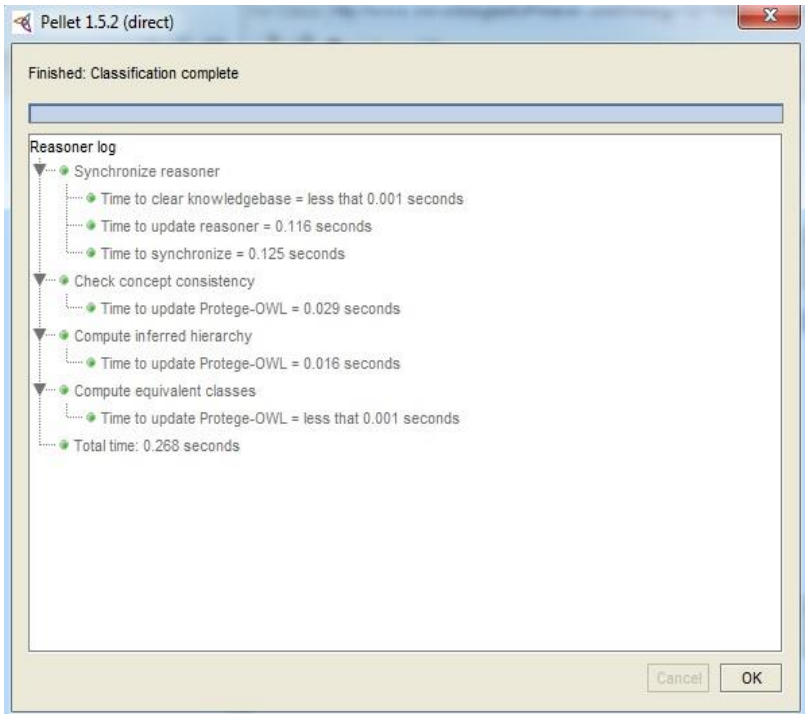
2.2.8. Тестове и автоматична проверка на логиката

Protégé осигурява механизъм за провеждане на серия тестове, които помагат при създаването на онтологичните модели: задаване на област и обхват на свойствата, разделяне

на класовете, голям брой тестове за точно дефиниране на различните видове обектни свойства (транзитивни, инверсни, функционални и др.)

Логическите анализатори, към които **OWL** компонентата на **Protégé** осигурява интерфейс (**Pelet, Fact++, Raser**) осъществяват проверка за консистентност на класовете (дали класът може да има индивиди) и автоматична класификация според дефинираните логически връзки между класовете. Връзката на **Protégé** с **Fact++** е представена на фиг.7. В продукта са предвидени средства (**Wizards**) за бързо и автоматизирано задаване на класове, свойства и индивиди и готови онтологични конструкции.

От особено важно значение е лесния механизъм за интегриране на онтологии, който е гарантиран с концепцията за уникално име на всяка създавана онтология (**Unique ResourceIdentifiers - URIs**).



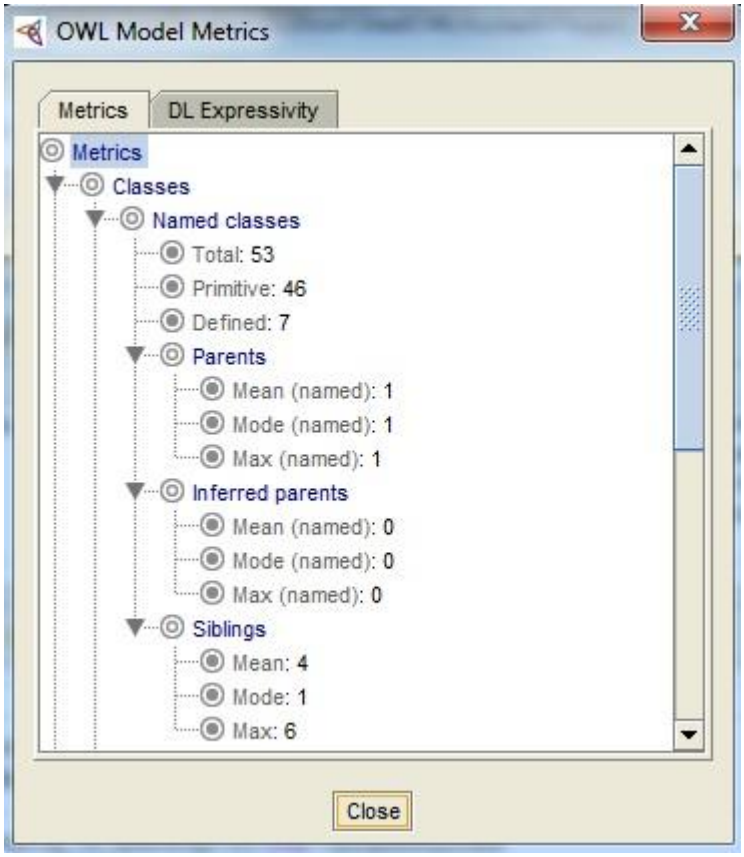
Фиг.7: Логически анализатор **„FaCT++”**

ТРЕТА ГЛАВА

3. СЪЗДАВАНЕ НА ИНФОРМАЦИОННА СИСТЕМА В СРЕДАТА НА Protégé/OWL

3.1 Структура на онтологичния модел

Онтологичният модел се състои основно от 53 класа, три обектни свойства, 15 ограничения върху свойства. Цялата характеристика на изградената онтология може да бъде изведена чрез функцията в **Protégé/ OWL** наречена **Metrics**:



Фиг.8: Резултат от функцията**Metrics**

В онтологията са зададени класове за описание на конкретни продукти на даден завод (конкретно Стомана Индъстри), класове, дефинирани според примерна

класификация на продукти [20] според избрани показатели за класификация: метод на производство (горещо или студено валцуване) и вид на напречното сечение на готовия продукт.

В избраната среда на продукта Protege-OWL в модела лесно могат да бъдат дефинирани и описани като класове продукти от други металургични заводи в рамките на компанията, а също и на други производители.

3.1.1. Задаване на класове

В онтологичния модел са представени два основни класа:

1. клас ***Справочник_Продукти_От_Прокатно_Производство***
2. клас ***Продукти_От_ SIDENOR_Group***

Различните производители на прокатни продукти използват различаващи се наименования за едни и същи изделия. В така предложения модел продуктите се описват според термините на производителя на съответния език и се класифицират според избрана класификация по признаци, дефинирани в избран справочник [20].

Съществуват също и различни класификации в търговските сайтове, които предлагат металургична продукция от много и различни заводи (http://www.metaltrade.ru/index.htm).

Според сайта http://www.metaltrade.ru/index.htm продукцията на прокатните производства може да се представи чрез следната класификация:

Прокат [rolled products] — Продукция във вид на полуфабрикат или изделие от черни метали и сплави, получени чрез методите на горещо или студено валцуване. Продуктите се разделят на 3 основни групи: сортов, листов прокат и тръби. Продуктите се разграничават и според напречното си сечение: сортов прокат с проста геометрична

форма (кръг, правоъгълник, квадрат, шестоъгълик) и т.нар. фасонни профили (ъгъл, швелер), листов прокат, тръби и т.н

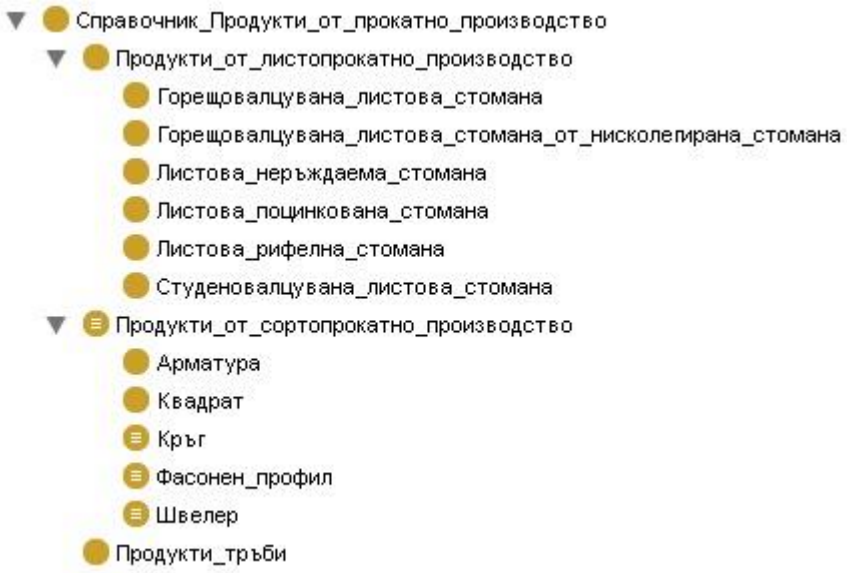
- едросортов прокат [heavy bars & sections] — продукти от горещо валцуване с определени размери
- дребносортов прокат [light sections, merchant bars, small bars & sections]
- сортов прокат [bars & sections, long products]
- листов прокат [flat products, plate/sheet & strip]
- и т.н.

Без да цели обхващането на всички класове продукти от прокатно производство, в онтологичния модел е представена част от примерна класификация на базата на справочните данни [20]

1. Като подкласове на клас ***Справочник_Продукти_от_прокатно_производство*** са представени следните подкласове:

- ***Продукти_от_листопрокатно_производство*** с подкласове ***Горещовалцувана_листова_стомана***, ***Горещовалцувана_листова_стомана_от_нисколегирана_стомана***, ***Листова_неръждаема_стомана***, ***Листова_поцинкована_стомана***, ***Листова_рифелна_стомана*** и ***Студеновалцувана_листова_стомана***
- ***Продукти_от_сортопрокатно_производство*** с подкласове ***Арматура***, ***Квадрат***, ***Кръг***, ***Фасонен профил*** и ***Швелер***
- ***Продукти тръби***

Ограничения могат да бъдат зададени от типа необходими и достатъчни. Подобен е случаят с клас ***Продукти_от_сортопрокатно_производство.*** При него са въведени ограничения от типа allValuesFrom за ***Метод_за_производство***, като е указано, че трябва да бъде само ***ГорещоВалцуване.***Това е предадено и на подкласовете на ***Продукти_от_сортопрокатно_производство.***



2. Вторият основен клас в онтологичния модел е ***Продукти_От_ SIDENOR_Group.*** Основните продукти на Стомана Индъстри, Перник са част от SIDENOR Group, Гърция и са представени както се дефинират в сайта на комбината.

Основните подкласове са:

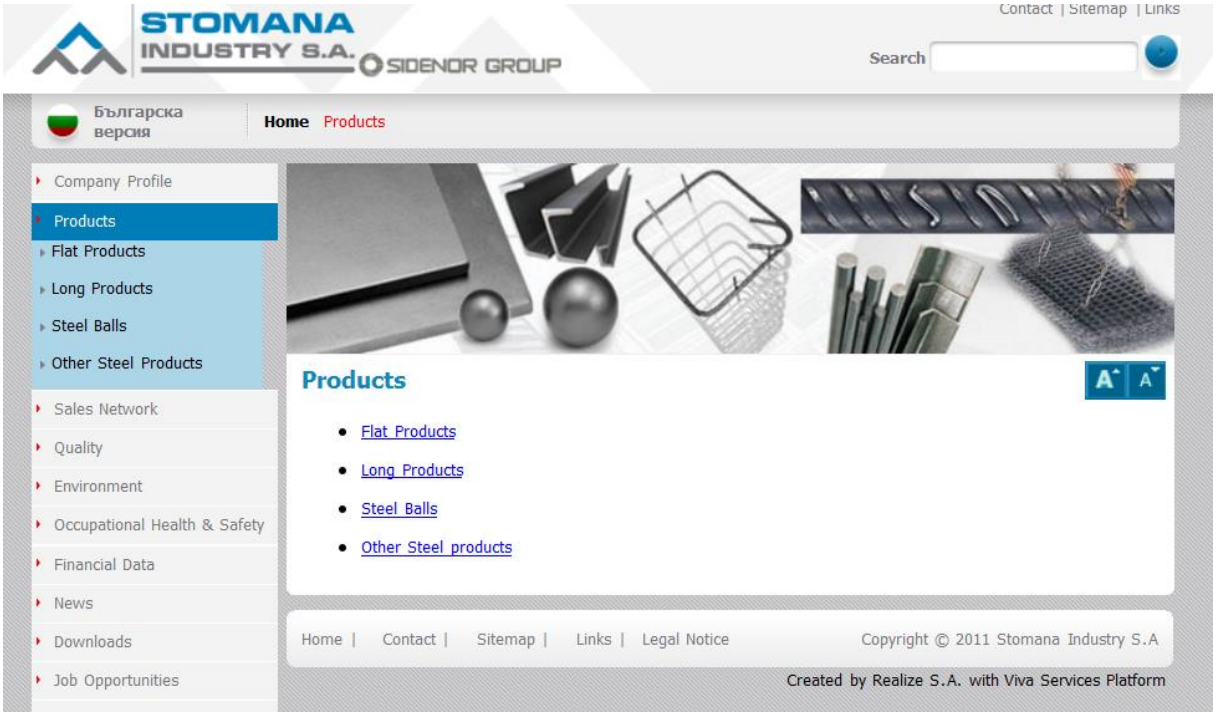
- Flat products с подклас Hot_rolled_steel_plated
- Long products с подкласове Concrete_reinforcing_Products, Merchant_bars, Special_Profiles и Special_Steels
- Other steel products с подкласове Welding_products и Wire_ Poducts
- Steel balls с подклас Special_Steel – Rounds

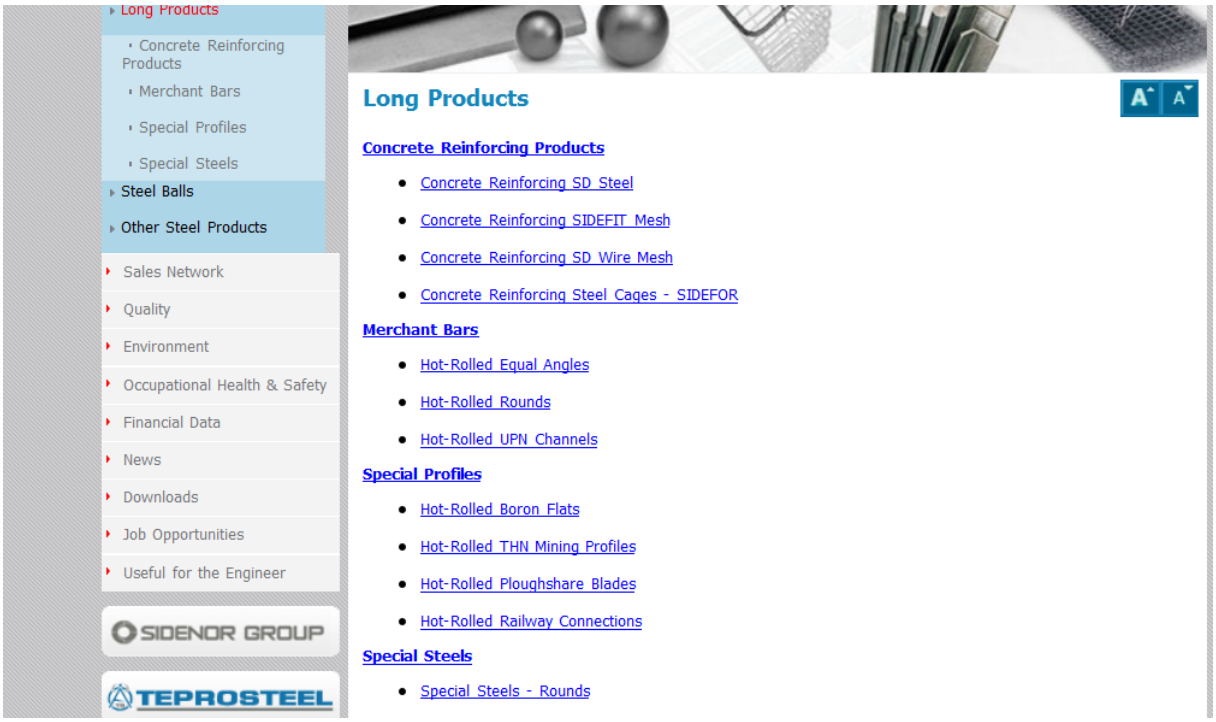
Класът ***Продукти_от_SIDENOR_Group*** се състои от подкласовете ***Flat_products, Long_products, Other_Steel_products***и ***Steel_balls.*** Всеки един от подкласовете е разделен от другите чрез функцията ***disjoint.*** От своя страна всеки един от изброените подкласове има поне по един подклас, който е ограничен чрез функцията ***Create_restriction*** или има свои подкласове. Така например клас ***Flat_products*** има подклас ***Hot_rolled_steel_plated***, който е ограничен с функцията ***Restriction***, а ограничението е от типа ***allValuesFrom***, като е указано, че индивидите от този клас могат да имат ***Vid_na_naprechnoto_sechenie*** единствено ***List.*** Тези ограничения на подклас ***Hot_rolled_steel_plated***са зададени като необходими за индивидите, които ще попадат в тях.

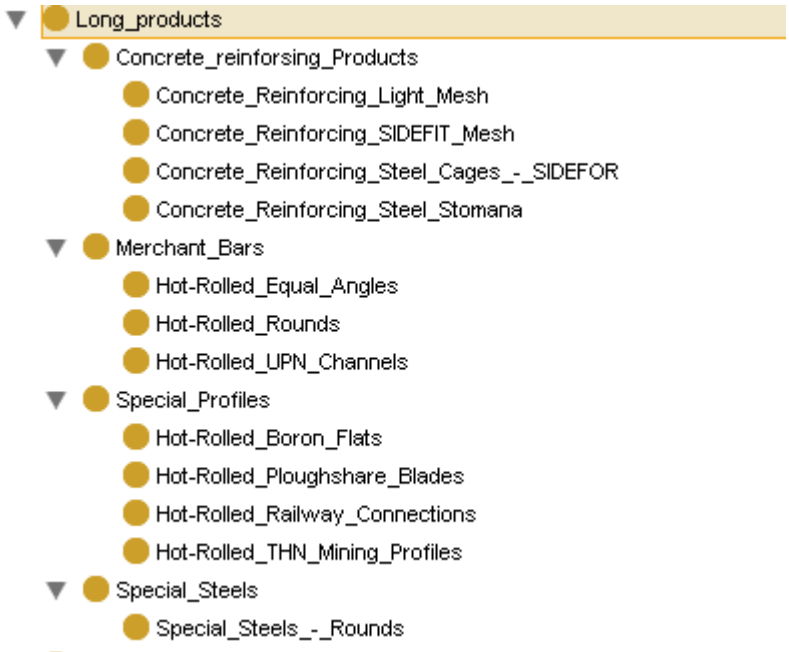
По същия начин подклас ***Long_products*** се състои от подкласове, които от своя страна също се състоят от свои собствени подкласове, към които са добавени съответни ограничения. Например един от тях ***Merchant_Bars*** има зададено ограничение от типа ***allValuesFrom*** за обектното свойство ***hasMetod_za_proizvodstvo*** само ***GoreshtoValtzuvane.*** Това ограничение се наследява от подкласовете ***Hot-Rolled_Equal_Angles, Hot-Rolled_Rounds*** и ***Hot-Rolled_UPN_Channels.*** Към подкласовете ***Hot-Rolled_Equal_Angles***и***Hot-Rolled_Rounds*** са добавени ограничения от тип ***allValuesFrom***за ***Вид_на_напречното_сечние*** съответно ***onlyВинкел***и ***onlyКръгло.***За третия подклас е зададено ограничение от типа ***someValuesFrom*** за ***Вид_на_напречното_сечниеsomeУ-образно.***

За голям брой от класовете са добавени термини на двата езика (английски и български), заедни с описания на конкретните продукти. Тази информация при изграждане на големи онтологии, съдържащи множество на брой класове и подкласове. осигурява лесен и бърз достъп до данните, характеризираща отделните класове.

Подобни описания са въведени за класовете ***Hot-Rolled_Rounds, Hot-Rolled_UPN_Channels*** и***Steel_Balls_for_Grinding.*** За тях са въведени описания от типа: място на производство, видове, предлагани форми и размери, както и процесите където могат да се използват и където техните качества се проявяват в най-добра степен. Казано по-кратко, чрез описанията можем да въведем разнороден вид информация, която да ни бъде от полза в по-късен етап при използване на онтологията.







При построяването на модела, след създаването на класовете и подкласовете са зададени свойства.

3.1.2.Създаване на свойства

OWL Свойствата представляват връзки (взаимоотношения) между два екземпляра. Съществуват два основни типа свойства:обектни свойства (Object properties)и свойства тип данни (Datatype properties). Обектните свойства свързват един екземпляр с друг. Свойствата тип данни свързват даден екземпляр със стойности на данните за него. OWL също притежава и трети тип свойства анотационни свойства (Anotation properties). Анотационните свойства могат да се използват за добавяне на информация (метаданни - данни за самите данни) към класове, индивиди или обекти. Свойствата могат да имат и обратна насоченост. Примерно: има притежател и притежаван е от. Свойствата също така могат да бъдат ограничавани като им се зададе единична (еднопосочна) стойност - с цел по-голяма функционалност. Те също могат да бъдат транзитивни или симетрични.

Задаване на обектни свойства

Обектните свойства, които характеризират подклас ***Продукти_от_ SIDENOR_Group***са:

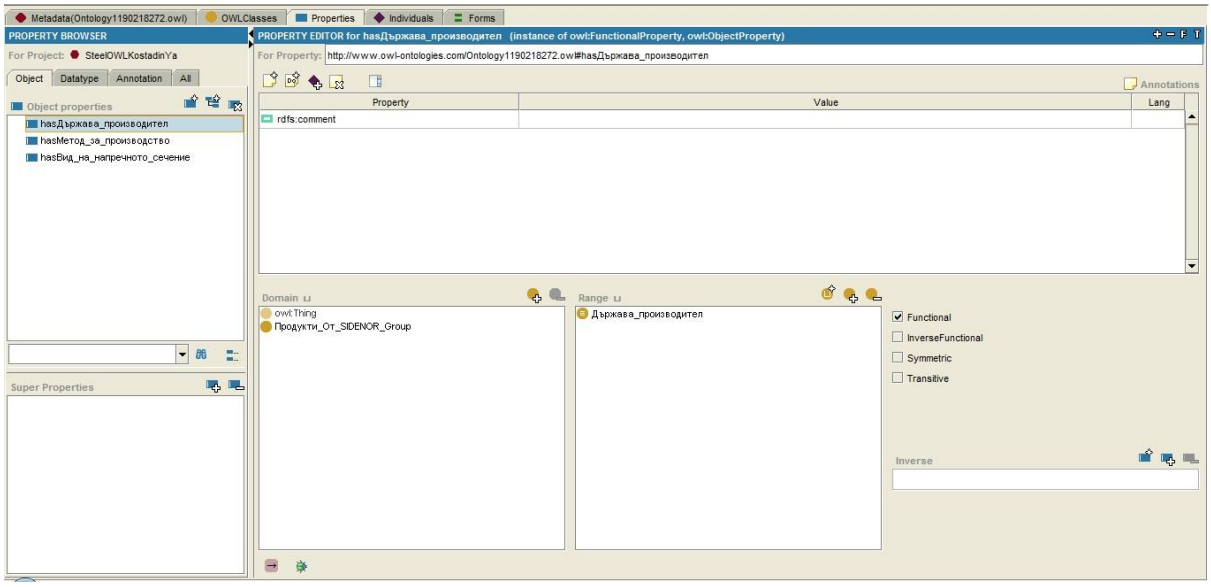
- ***Държава_производител***
- ***Метод_за_производство***
- ***Вид_на_напречното_сечение***

За свойството ***Държава_производител*** е зададена област (*domain)* ***Продукти_от_SIDENOR_Group*** и обхват (range) ***Държава_производител***.

За свойство ***Метод_за_производство*** е зададен обхват ***Метод_за_ производство***, а за свойство ***Вид_на_напречното_сечение***зададен обхват ***Вид_на_напречното_сечение.***

Свойството ***Държава_производител*** е функционално, същото се отнася и за свойството***Вид_на_напречното_сечение.***

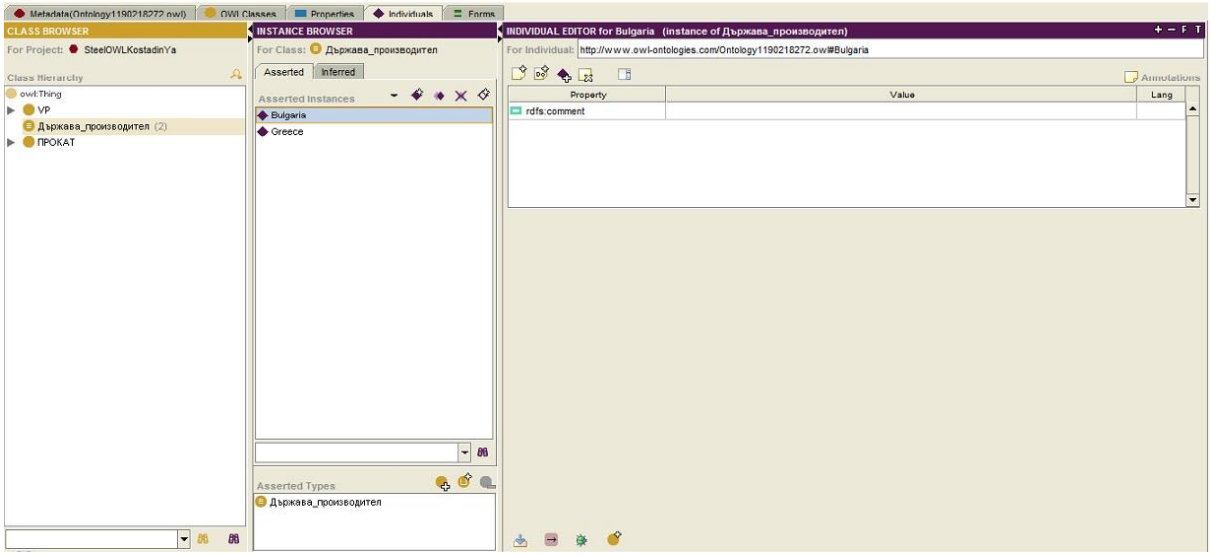
В конкретния случай обектното свойство ***Държава_производител*** свързва индивиди от клас ***Продукти_от_SIDENOR_Group***с индивиди от клас ***Държава_производител***. При разширяване на модела с продукти от други заводи на същата фирма, произведени в различни страни (Румъния, Албания), тази информация е полезна при търсена на конкретнен продукт.



3.1.3. Създаване на индивиди

Индивидите представляват обектите в предметната област, която описваме. Важна разлика между PROTÉGÉ и OWL е че OWL не използва UniqueNameAssumption (UNA, уникалност на имената). Това означава, че две различни имена биха могли да се отнасят за един и същ индивид. Например “държава” и “страна” биха могли да

означават един и същ екземпляр. В OWL изрично трябва да бъде упоменато, че екземплярите са еднакви един с друг, или че са различни – в противен случай те *вероятно биха могли* да бъдат еднакви, или *вероятно биха могли*да бъдат различни. Като представители на класовете, в онтологичния модел се задават т.нар. екземпляри (индивиди). В създадения модел са зададени примерни екземпляри в клас ***Държава_производител***.

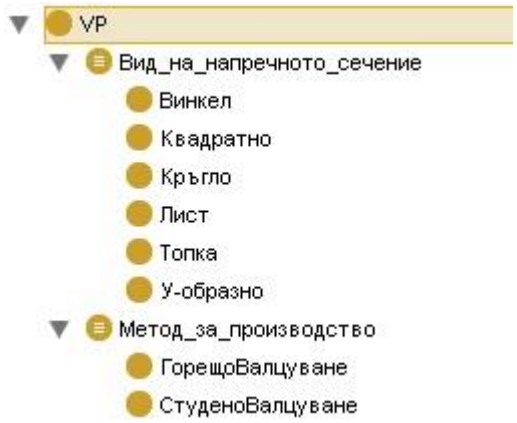


3.1.4. Създаване на ValuePartitions

Използван е модел ValuePartitions, за да допълване на описанията на класовете. ValuePartitions не са нито част от OWL, нито от някой друг онтологичен език, те са "проектантски шаблони" (designpatterns). При онтологиите те са аналогични на тези при обектно ориентираното програмиране – те представляват решения на често срещани и повтарящи се проблеми.

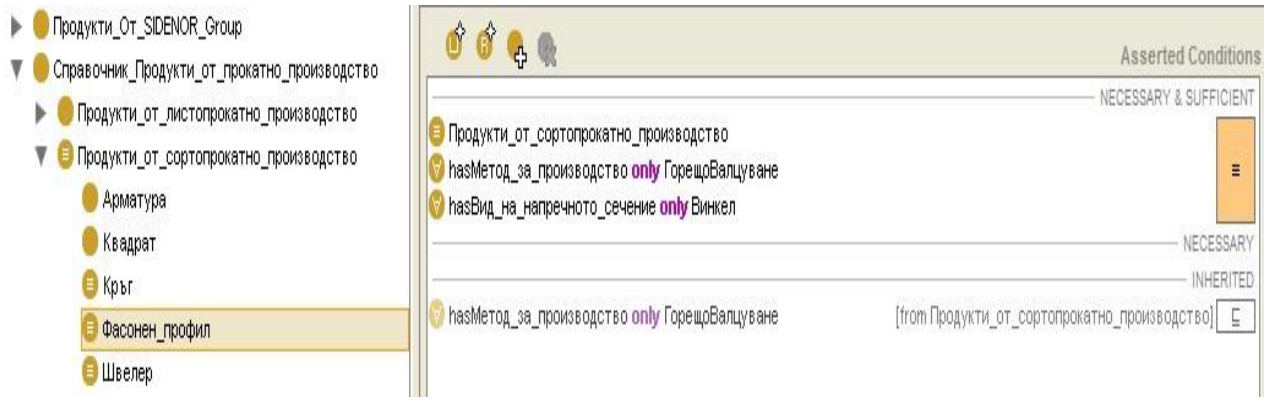
Създаваден е образец (ValuePartition), в нашия случай наречени "***Метод_за_производство***" и "***Вид_на_напречното_сечение***" за определяне

неизползвания метод за производство и вида на напречното сечение. ValuePartitions ограничават обхвата на възможните стойности, например в нашия случай "***Метод_за_производство***" ще ограничи възможните стойности до ***ГорещоВалцуване***и ***СтуденоВалцуване***, а "***Вид_на_напречното_сечение***" ще ограничи възможните стойности до ***Кръгло, Квадратно, Лист, Топка, У-образно*** и ***Винкел.***



3.1.5. Дефинирани класове в йерархията

Част от класовете (някои продукти на сортопрокатно производство) са **дефинирани класове**, т.е. за тях са зададени НДУ. Например, за да принадлежи даден обект към клас Фасонен профил, необходими и достатъчни условия са неговите екземпляри да са произведени чрез горещо валцуване и да имат напречно сечение тип винкел.



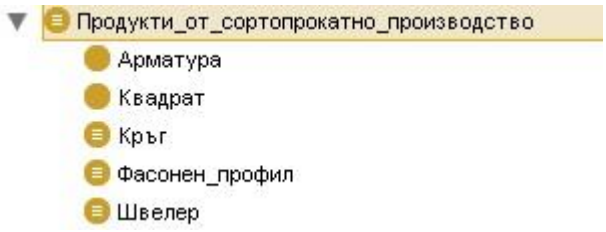
Подобни са и класовете ***Кръг*** и ***Швелер***. За тях са зададени НДУ, които да групират техните екземпляри спрямо метода на производство - горещо валцуване и напречното сечение - съответно кръгло и ***У-образно***.

НДУ за дефинираните класове са зададени с помощта на редактора за ограничения върху свойствата, като е използван шаблонът Value Partition.

Като част от шаблона Value Partition в Protégé е зададена покриваща аксиома (covering axiom). В конкретния случай: ***Вид_на_напречното_ сечение*** може да бъде единствено ***Кръг, Квадрат, Лист, Топка, У-образно*** или ***Винкел***.

Дефинирани в онтологичния модел са следните класове:

- ***Продукти_от_сортопрокатно_производство***
- ***Кръг***
- ***Фасонен_профил***
- ***Швелер***



За тях са зададени НДУ според свойствата ***Метод_за _производство*** и ***Вид_на_напречното_сечение***.

В разработения онтологичен модел клас ***MerchantBars*** се дефинира с помощта на свойството ***Метод_на_производство*** и върху свойството е наложено универсално ограничение (тип∀), т.е. всички екземпляри от клас ***MerchantBars*** са произведени чрез метода на горещото валцуване. Това ограничение се предава и на подкласовете на клас ***MerchantBars***, всички наследяват ограничението. Свойството множествено унаследяване се предава по принципа на йерархията, т.е. свойствата валидни за класовете се предават на техните подкласове.

3.1.6. Логически анализ на модела

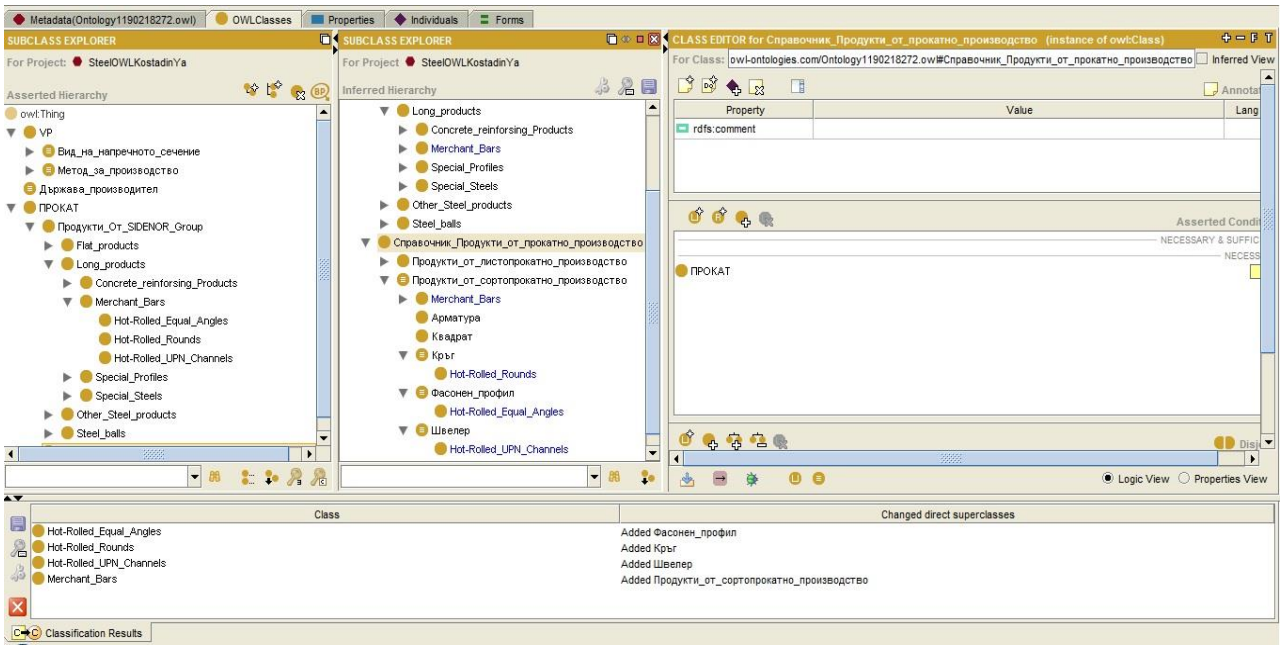
Един от отличителните белези на онтологиите създадени с OWL-DL е, че те могат да бъдат обработени от логически анализатори (Reasoners). Една от основните функции на логическия анализатор е да провери дали даден клас е подклас на друг клас или не. Чрез анализ на всички класове логическият анализатор е в състояние да изчисли предполагаема йерархия (inferred hierarchy) на класовете в онтологията. Друга стандартна функция на анализатора е проверката на логиката. Въз основа на описанието на даден клас, класификаторът може да определи дали той въобще може да има някакви индивиди(екземпляри). Един клас се смята за несъгласуван, ако не може да има екземпляри.

В PROTÉGÉ- OWL ръчното изграждане на йерархията на класовете се нарича asserted hierarchy (заявена йерархия). Йерархията, която се създава автоматично от анализатора се нарича *inferred hierarchy* (предсказана йерархия).

За да се класифицира автоматично онтологията се използва бутона “Classify taxonomy”, а за да се провери съгласуваността, се използва бутона “Check consistency”. Когато автоматично класифицираната йерархия е готова, ще се появи още прозорецът “*inferred hierarchy”* до съществуващия вече “*asserted hierarchy”*. Ако даден клас е бил преквалифициран, то името му се появява в син цвят в прозореца *inferred hierarchy.* Ако даден клас е несъгласуван, то името му е оцветено в червено.

В резултат на работата на логическия анализатор клас ***MerchantBars*** се прехвърля като подклас на клас ***Продукти_от_Сортопрокатно_ Производство*** в йерархията на клас ***Справочник_Продукти_От_Прокатно_ Производство***.

Аналогично, чрез задаване на универсални ограничения (тип∀) за класовете ***Hot_Rolled_Equal_Angles, Hot_Rolled_Rounds*** и ***Hot_Rolled_ UPN_Chanels***, всички техни екземпляри са продукти на горещо валцуване и имат напречно сечение съответно ***Винкел, Кръг*** и ***У-образно***.Класификаторът премества тази класове като подкласове на дефинираните класове ***ФасоненПрофил*** , ***Кръг*** и ***Швелер*** в йерархията на клас ***Справочник_Продукти_От_Прокатно_Производство***. Резултатите от работата на анализатора са представени на фиг.9.



Фиг.9 Предсказана йерархия от логически анализатор.

Четирите класа от продуктите на SIDENOR Group (***MerchantBars, Hot_Rolled_Equal_Angles, Hot_Rolled_Rounds*** и ***Hot_Rolled_UPN_Chanels***) са класифицирани като подкласове в йерархичното дърво на клас ***Справочник_Продукти_От_Прокатно_Производство*** от логическия анализатор на базата на дефинираната в онтологията логика.

След дефинирането на свойства за всички класове, те автоматично биват класифицирани от логическия анализатор в йерархията, която е заложена в онтологията. Допълнителни признаци за класифициране могат да се въведат и използват чрез модела ValuePartition.

Фиг. 10: Класификация на продукти в среда на Protégé – OWL

ЗАКЛЮЧЕНИЕ

Protégé, версия 3.4.7.е един от най-успешните и най-често използвани програмни продукти за разработка на онтологични модели. Последователно са разгледани необходимите стъпки, през които трябва да се премине, за да се създаде проект. Акцентирано е на създаването на класове, подкласове и своиства и индивиди.

Разгледани са причините, които обуславят необходимостта от разработка на онтологии, тяхното значение, видовете онтологични модели и методологии за създаването им. Представени са различните видове езици за описание на онтологиите като основно внимание е отделено на OWL.Направен е кратък обзор на понятията онтология, таксономия и стандарти за мета данни. Разгледани са значението на онтологичните модели, актуалното състояние на използването на онтологии, основните елементи и структурата на онтологичните модели. По-специално внимание е отделено на методологиите за онтологично моделиране.

Дипломната работа е посветена на разработване на OWL онтология. Представен е онтологичен модел на групите от прокатни продукти, използвани в металургична ата индустрия в среда на Protégé/OWL. Последователно са описани етапите на създаването на модела. Създаденият примерен онтологичен модел служи за автоматична класификация на продукти от различни прокатни производства, от различни производители от национални и международни компании.

Бъдещото разширяване и усъвършенстване на модела ще осигури необходимият общ език за комуникация между различните заинтересовани лица, участващи в процеса на доставки, производство и търговия на ппродукти от металургичните производства, и ще улесни разбирането и споделянето на знанията на различни етапи от работата им. Той би могъл да служи както за източник на справочна информация в сайтовете за електронна търговия, така и за създаване на връзки между отделните офиси в различни страни.

ЛИТЕРАТУРА

1. Gangemi A., Borgo S., Catenacci C., Lehmann J.(2004), Task taxonomies for knowledge content, METOKIS Deliverable.
2. Borst, W. N. Construction of Engineering Ontologies. University of Tweenty. Enschede, NL-Centre for Telematics and Information Technology. 1997.
3. Uschold M., Jasper R., A framework for understanding and classifying ontology applications, Proc. of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods, pp. 11.1 – 11.12, Sweden, 1999
4. Gruber, T. R. Towards principles for design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies, 43 (5-6), pp. 907-928. 1995.
5. Uschold M., King M., Towards a methodology for building ontologies, In Proc. of the workshop on Basic Ontological Issues in Knowledge Sharing, International Joint Conference on Artificial Intelligence (IJCAI'95), Montreal, 1995.
6. Gómez-Pérez A., A Framework to Verify Knowledge Sharing Technology, Expert Systems with Application, Vol. 11, No. 4. pp. 519-529, 1996.
7. Grüninger M., Fox M. S., Methodology for the design and evaluation of ontologies, In Proc. of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95, Montreal, Canada, 1995.
8. Gruber T. R. , A translation approach to portable ontology specifications, Knowledge Acquisition, 5 (2), pp. 199-220, 1993.
9. Swartout, B.; Ramesh P.; Knight, K.; Russ,T. Toward Distributed Use of Large-Scale Ontologies. Symposium on Ontological Engineering of AAAI. Stanford (California). Mars, 1997.

10. Lopez, M.F., “Overview of the methodologies for building ontologies”. In V.R. Benjamins, B. Chandrasekaran, A. Gomez-Perez, N. Guarino, and M. Uschold (Eds): Proc. Of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden, August, 1999.
11. Guarino N., Formal ontology and Information Systems, in Proc. of FOIS’98, Trento, Italy, 6-8 june, Amsterdam, IOS press, pp. 3-15, 1998.
12. OWL Web Ontology Language Overview, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/owl-features/.
13. Melville W., Deasy C, JALCA, 1977, 6, 216.
14. Ontology Development 101: A Guide to Creating Your First Ontology, Natalya Fridman Noy and Deborah L. McGuinness. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.
15. http://www.w3.org/
16. М. Нишева, Въведение в системите, основани на знания, учебни записки.
17. H. Knublauch, R. W. Fergerson, N. F. Noy, & M. A. Musen. The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications, Third International Semantic Web Conference, Hiroshima, Japan, 2004.
18. http://www.owl-ontologies.com/assert.owl
19. http://www.owl-ontologies.com/2005/08/07/xsp.owl
20. http://www.metaltrade.ru/index.htm