ПЛОВДИВСКИ УНИВЕРСИТЕТ „ПАИСИЙ ХИЛЕНДАРСКИ“

ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

ДИМИТЪР БЛАГОЕВ БЛАГОЕВ

СИСТЕМИ ЗА СЕМАНТИЧЕН АНАЛИЗ И ИНФОРМАЦИОННО ТЪРСЕНЕ

ДИСЕРТАЦИОНЕН ТРУД

за присъждане на образователната и научна степен „Доктор“
в област на висше образование 4. Природни науки, математика и информатика;
професионално направление 4.6 Информатика и компютърни науки;
докторска програма Информатика

Научен ръководител:
проф. дмн. Георги Тотков

гр. Пловдив
2012 г.

СЪДЪРЖАНИЕ

УВОД

Актуалност на темата

Търсенето на информация е относително утвърдена технология, която се използва за получаване на множество от свързани материали от огромна колекция. Типичен пример за търсене на информация е търсенето по ключова дума в Интернет, при което резултатът е страници, съдържащи тази ключова дума. Броят на намерени страници, обаче, може да бъде много голям, а и по всяка вероятност, не всичко на дадена страница ще е важно за потребителя. Оттук – преглеждането на всяка страница – резултат на търсенето, с цел откриване на необходимата информация, много често е трудоемка задача, изискваща значителен разход на време. Ето защо се налага процесът на търсене да се допълва с *извличане на информация*, при което в намерените документи се търсят определени фрази (представящи данни, факти и др.), които биха представлявали интерес за потребителя. С други думи, документи с неструктурирано съдържание се трансформират в структури от определен тип.

Създадени са стотици системи за информационно търсене и хиляди функции за търсене, вградени в различни приложения. Във всички тези случаи (независимо от математическия модел), идеите, софтуерните програми и структурите от данни са достатъчно прости. Съществува мнение, че всяко ново поколение програми е по-съвършено от предходното. Друг краен възглед се състои в това, че „всичко ново – това е добре забравено старо“. По всяка вероятност, истината, в случая на софтуерните системи за информационно , е по средата.

Цели и задачи на дисертационния труд

Основна цел на дисертационното изследване е – да се моделира и създаде система за бързо проектиране на процеси в областта на информационното търсене и семантичният анализ.

Изведени са ***4 (четири) основни*** цели на дисертационното изследване:

1. Изследване на съществуващи методи и системи за информационно търсене и семантичен анализ;
2. Предлагане на общ модел за изграждане на системи за моделиране и процеси решаващи разглежданите задачи;
3. Изграждане на конкретна система, базирана на предложеният модел;
4. Създаване и експериментиране с конкретни описания на процеси, създадени с реализираната система.

За постигане на поставените цели в изследването последователно се решават следните задачи:

1. Да се проучи общата теория за информационно търсене и семантичен анализ, съществуващите системи и методи;
2. Да се създаде концептуален модел за описание на процеси за информационно търсене и семантичен анализ.
3. Да се проектира система за описание и управление на работни процеси и да се реализира прототип, базиран на съвременни технологии. Подзадачи:
   3.1. Да се проектира система за управление на работни процеси, базирана на описаният модел от точка 2;
   3.2. Да се разработи прототип на СУРП (система за управление на работни процеси);
   3.3. Да се опишат инструментите, които СУРП предоставя
4. Да се създадат приложения чрез разработения прототип на система за управление на работни процеси, които да докажат приложимостта му за решаване на конкретни практически проблеми.
   4.1. Създаване на система за извличане на контактна информация от документи на български език;
   4.2. Създаване на методи за автоматизирано генериране на метаданни за учебни обекти;
   4.3. Създаване на модул за моделиране и управление на виртуални адаптивни курсове, базирани на moodle;

Структура и обем на дисертацията

Работата се състои от увод, четири глави и заключение. Приложен е списък с използвана литература.

В първа глава е направен обзор в областта на информационното търсене и извличането на информация, който представя основните модели, задачи, методи и средства за тяхното прилагане и решаване, формулира най-важните изисквания към съвременните информационно-търсещи системи (ИТС), дискутира нерешени проблеми и систематизира съответната специализирана терминология. Специално внимание е обърнато на лингвистичните и статистически методи за анализ и оценяване на съдържанието на текстови документи. Във връзка с моделите за информационно търсене са разгледани основните методи за тематичен анализ и обработка на текстова информация.

Във втора глава е представен концептуален модел, който може да бъде използван за описание и бързо прототипизиране на лингвистични процеси.

В трета глава е проектирана система за описание и управление на работни процеси използваща предложеният концептуалният модел. Извършен е анализ на функционалните и нефункционалните изисквания. Описани са основните архитектурни компоненти и реализацията на системата.

В четвърта глава са разгледани приложения на прототипа на система за управление на работни процеси. Обърнато е особено внимание на широката приложимост на разработеният модел. Описани са реализации на система за извличане на контактна информация от документи на български език, подходи за автоматизирано генериране на метаданни за учебни обекти и модул за моделиране и управление на виртуални адаптивни курсове, базирани на moodle, детерминирани математически модели и модул за получаване на неатрактивни мрежи.

В заключението е направено обобщение на архитектурата и реализираната системи. Отбелязани са основните резултати и са очертани възможностите за бъдещо развитие.

Списъкът на използваната литература съдържа 127 заглавия, от които 10 на кирилица, 105 на латиница и 12 интернет-източници.

Списък на съкращенията

БД – база данни;

ИИБ – инженери на изграждащи блокове;

ИРП – инстанция на работен поток;

ИТС – информационно-търсещи системи;

КП – крайни потребители;

МС – мениджъра на събития;

ОРП – описание на работен поток;

СУРП – система за управление на работни процеси;

СУБД – система за управление на бази от данни;

ХОРП – хранилище с описания на работни потоци;

ACM – Association of Computing Machinery;

BPM – Business Process Model;

HITS – Hyperlink Induced Topic Search;

IDF – Inverse Document Frequency;

IR – IInformation Retrieval;

LMS – Learning Management System;

LO – Learning Object;

LOM – Learning Object Metadata;

LSA – Latent Semantic Analysis;

LSI – Latent Semantic Indexing;

NIST – National Institute of Standards and Technology;

SIGIR – Special Interest Group on Information Retrieval;

SMART – Salton's Magical Automatic Retriever of Text;

TF – Term Frequency;

TREC – Text Retrival Evaluation Conference;

WMS – Workflow Management System.

ГЛАВА 1. ОБЗОР НА СЪВРЕМЕННИ МЕТОДИ И СИСТЕМИ

Търсенето на информация е относително утвърдена технология, която се използва за получаване на множество от свързани материали от огромна колекция. Типичен пример за търсене на информация е търсенето по ключова дума в Интернет, при което резултатът е страници, съдържащи тази ключова дума. Броят на намерени страници, обаче, може да бъде много голям, а и по всяка вероятност, не всичко на дадена страница ще е важно за потребителя. Оттук – преглеждането на всяка страница – резултат на търсенето, с цел откриване на необходимата информация, много често е трудоемка задача, изискваща значителен разход на време. Ето защо се налага процесът на търсене да се допълва с *извличане на информация*, при което в намерените документи се търсят определени фрази (представящи данни, факти и др.), които биха представлявали интерес за потребителя. С други думи, документи с неструктурирано съдържание се трансформират в структури от определен тип.

В настоящата глава е направен преглед в областта на информационното търсене и извличането на информация, който представя основните модели, задачи, методи и средства за тяхното прилагане и решаване, формулира най-важните изисквания към съвременните информационно-търсещи системи (ИТС), дискутира нерешени проблеми и систематизира съответната специализирана терминология. Специално внимание е обърнато на лингвистичните и статистически методи за анализ и оценяване на съдържанието на текстови документи. Във връзка с моделите за информационно търсене са разгледани основните методи за тематичен анализ и обработка на текстова информация.

1.1 Въведение в информационното търсене

Във връзка с бурното развитие на Интернет и на приложните системи, свързани с обработка на големи масиви (колекции) от документи, все по-актуални стават проблемите за организация на ефективен достъп до информация. В днешните условия, решенията на задачи, свързани с информационно търсене и извличане на информация, стават не само приоритетни, но и жизнено необходими за осигуряване на своевременен достъп към информация, интересуваща потребителите. За около 4 години изследвания, списъкът на решаваните задачи значително се разшири, и сега включва въпроси за моделиране,

класификация и кластеризация на документи, проектиране на архитектурата на търсещи системи и потребителски интерфейси, езици за заявки, и др.

1.1.1 Задачи

Централният проблем на информационното търсене се формулира просто – да се помогне на потребителя да намира информацията, от която се интересува. За съжаление, да се опишат информационните потребности на различни групи потребители, съвсем не е проста задача. Обикновено това описание се формулира като заявка, представена като набор от ключови думи, характеризиращи най-общо тези потребности.

Класическата задача на информационното търсене, с която и започва развитието на научната област, е търсене на документи, удовлетворяващи заявката в рамките на някаква статическа (към момента на изпълнение) колекция от документи. Например, подобна задача се решава в рамките на повечето съвременни справочни системи, такива като справочната система за операционната система Windows.

Целта на *класификацията* (*кластеризацията) на документи* е автоматичното откриване на група семантично подобни документи сред зададено фиксирано множество от документи [24, 42]. Ще отбележим, че групата се формира само на базата на сходство на описанията на документите, и предварително не се задават каквито и да било характеристики на тези групи.

Както и в задачата за класификация, цел на *задачата за филтрация* е разделяне на множеството от документи на категории. Тези категории обаче са само две -- документите, които удовлетворяват даден критерий, и другите - които не го удовлетворяват.

През последните години, вниманието на изследователите се привлича от въпросите за *определяне на тематичната ориентация на документа по неговото съдържание*. Задачата за *тематична класификация* е частен случай на задачата за класификация. Тук всяка категория е някаква тематика, а цел на класификацията – да се определи тематиката на документа. Класическата задача за класификация (кластеризация) на документи по зададен набор от тематики е разгледан в [12, 46, 52, 108, 125]. Задачата се състои в причисляване на всеки постъпващ, в системата, документ към една (или няколко) предварително зададени

категории (тематики), т. е. автоматично определяне на тематиката на документа по зададено множество от възможни тематики. Ще отбележим, че за разлика от задачата за филтрация на документи [35], тук се подразбира, че в системата не постъпва ‘боклук’, т. е., че всеки от разглежданите документи се отнася поне към една от зададените тематики. Като следствие, методи прилагани с успех в задачи за филтриране, показват не много добри резултати в областта на класификацията.

В термините на изкуствения интелект, и в частност на машинното самообучение, задачата за класифициране е вид обучение с учител (англ. supervised learning), докато тази за кластеризация представлява обучение без учител (англ. unsupervised learning). Подобна е задачата за *тематична филтрация* на документи, т. е. автоматично определяне на документи, съответстващи на дадена тематика, за сметка на ‘отсейване’ на излишните документи [18, 35].

Методите за класификация се основават на един и същи обобщен алгоритъм, който се състои от следните етапи:

- задаване/построяване на описания (модели) на всички тематики;
- построяване на описание на разглеждания документ;
- изчисляване на оценки за близост между описанията на тематики и описанието на документа, и избор на най-близки тематики.

Въпреки известна прилика при формулирането на горните задачи на информационното търсене, по същество, те силно се различават. Като следствие, методи, успешно прилагани при решаване на една задача, често показват не много добри резултати при директното им прилагане за решаване на друга задача.

Повечето подходи за *моделиране* и *описание на тематики и документи* са основани на предположението, че тематиката на документа се определя от неговия речников запас. От разглеждане се изключват т. нар. *стоп-думи*, т. е. най-използваните думи, които могат да се използват в документи от произволна тематика – предлози, местоимения и др. Освен това се счита, че различни форми на една и съща дума не се отразяват на общата тематика на документа, следователно могат да се представляват от една базова словоформа (*терм*). В

качеството на описание на документа се използва цялото множество от срещащи се в документа терми, с изключение на употребяваните повсеместно.

При *построяването на модели (описания) на тематиките* в дадена колекция от документи, също се използват набори от терми, но тези набори съдържат не всички думи, употребявани в съответните документи, а само неголямо тяхно подмножество, което се подбира автоматично. Тематиката се моделира чрез задаване на (относително неголямо) множество от отнасящи се към нея документи. В резултат от анализа на това множество от документи, а така също и на множеството от документи, задаващи останалите разглеждани тематики, автоматично се строи описание на тематиката под формата на набор от терми. Целта на подобен анализ е откриване на отличията на тази тематика от другите и избор на терми, които най-добре подчертават особеностите на съответната тематика.

Обикновено, изборът на думи за описание на всяка тематика се извършва с помощта на алгоритъм, съставен от 4 (четири) стъпки.

1. Построяване на общ речник от терми W.

В този речник се включват всички терми, които се използват поне в един от документите, задаващи тематиката.

2. Изчисляване на вероятностни оценки

За всеки терм $w \in W$ се изчислява оценката на вероятността за негово използване в документите d на дадена тематика C:

$$P(w|C) = \frac{|\{d: d \in C, d \supset w\}|}{|C|}$$

3. Построяване на ‘тематични’ речници

За всяка тематика C се строи ‘тематичен’ речник, в който се включват терми, вероятността от използване на които в тази тематика е по-голяма от вероятността за тяхното използване във всяка друга тематика $C_i \in \Omega$, т.е.

$$P(w|C) \geq \frac{\sum_{c_i \in \Omega} P(w|C_i)}{|\Omega|}$$

За всеки от избраните терми се изчислява неговата *значимост* в рамките на дадена тематика съгласно емпирични формули, например:

$$TermValue(w, C) = \frac{P^3(w|C)}{\sum_{c_i \in \Omega} P(w|C_i)^2}$$

4. Избор на терми за описание

Значимостта на термите, получени на етап 3., задава отношение на порядък във всеки от 'тематичните' речници. Използвайки това отношение, от 'тематичния' речник се избират няколко терми в качеството на описание за съответната тематика.

Оптималното количество терми за включване в описанието зависи от конкретната задача. Експерименти в някои системи показват, че с увеличаване на броя на термите, качеството на класификация отначало се подобрява, а след това започва да намалява. Обикновено, оптимум се достига при неголям размер на описанието — от 10 до 30 терми.

Близка е и задачата за *кластеризация* на документи [25], т. е. автоматично откриване на група семантично (смислово) приличащи си документи сред зададено множество от документи. Тук, за разлика от задачата за класификация, тематичната ориентация на групата не се задава предварително.

В тази посока, все по-голямо внимание привличат и по-сложни подходи [12, 35, 108]. Основната идея на много от предлаганите методи е снижаване на размерността на пространството от 'признаци', по които се извършва класификация на документите. Начално пространство на признаците обикновено е пространството на термите, което се свива на основата на резултати от анализ на голяма група документи. За провеждане на анализа се използват различни подходи – кластеризация на терми на основата на тяхното вероятностно разпределение по документите [12], прилагане на методи за откриване на знания в данните (*data mining*) при определяне на правилата за класификация [24], и др. Ще

отбележим, че въпреки подобряването на качеството на класификация, практическото прилагане на подобни подходи често се усложнява поради голяма изчислителна трудоемкост, водеща до ниска производителност.

Сложността от формиране на информационни заявки към колекции от текстови документи се обуславя от:

- непознаване на набора от ключови думи, еднозначно определящи търсения(те) документ(и);
- липса на достатъчен опит и квалификация за формиране на такива заявки;
- отсъствие на приета и недвусмислена терминология в областта, и др.

Много често, осъществявайки търсене, някои потребители имат само бегла и схематична представа за интересуващата ги тематика. Всичко това обуславя актуалността и значимостта на изследвания, насочени към решаване на един от ключовите проблеми на информационното търсене – проблемът за *адекватно изобразяване на информационните потребности* на потребителите.

Един от вариантите за решаване на този проблем е *търсене на документи по образец*, когато човек задава документ като образец, а система, реализираща съответен вариант на търсене, подбира документи, подобни (по съдържание и тематика) на даденият.

Анализ на провеждани изследвания, посветени на решението на задачи за търсене на документи по образец, извежда твърде незначителен брой на готови и апробирани решения, което в голяма степен се дължи на липсата на достатъчно добре разработени теория и практики на решаване на задачата за тематически анализ на неструктурирана, естественоезикова текстова информация с произволно съдържание.

1.1.2 Модели

Ключово понятие, характеризиращо избор на един или друг метод за анализ и обработка на текстова информация, а също и реализацията на конкретен вариант за информационно търсене, е моделът на търсене [7, 15, 115, 119].

Моделът на търсене включва следните елементи [2]:

- начин за представяне на документите;
- начин за представяне на заявките за търсене;
- вид на критерия за релевантност на документи.

Вариации на горните елементи определят голям брой възможни реализации на системи за текстово търсене. Ще разгледаме някои от тях, най-популярни в днешно време.

Прости модели за търсене. Това са модели, в които документът се представя под формата на набор от асоциирани с него външни атрибути. Към най-простите елементи на модели за търсене се отнасят моделът за дескрипторно търсене и моделът, основан на т.нар. Дъблинското ядро.

В най-простите системи за *дескрипторно търсене*, представянето на документа се описва с множество от думи или словосъчетания от лексиката на конкретната предметна област, които характеризират съдържанието на документа. Тези думи и словосъчетания се наричат *дескриптори*. Индексирането на документа в такива системи се реализира с определяне на характерната за него съвкупност от дескриптори. При това дескрипторите могат да се приписват към даден документ на базата на неговото съдържание или неговото название (индексиране по съдържание и индексиране по заглавие).

В някои дескрипторни системи, индексирането на документи се осъществява ръчно от експерти в предметната област на системата, в други то се извършва автоматично.

Дескрипторните системи може да се отнесат към класа на системите, ориентирани към библиографско търсене или търсене ‘по каталог’.

Дъблинското ядро (Dublin Core) [2, 26] представлява набор от елементи на метаданни, смисълът на които е фиксиран в спецификации, определени от съответния стандарт. В термините на множеството от стойности на тези елементи може да се описва съдържанието на различен род текстови документи.

Първоначалната версия на Дъблинското ядро е предложена в 1995 г. на състоялия се в Дъблин (САЩ) симпозиум, организиран от Online Computer Library Center (OCLC) и

National Center for Supercomuting Applications (NCSA) за описание на информационните ресурси на библиотечните системи.

В моделите на информационно търсене, основани на Дъблинското ядро, представянето на k-ия документ е чрез множество от двойки $D_k = \{(N_{ik}, V_{ik})\}$, където:

N_{ik} – името на i-тия елемент на метаданни на Дъблинското ядро в описанието и съдържанието на k-ия документ;

V_{ik} – стойността на този елемент на метаданни.

Представянето на заявката за търсене също се представя с множество от двойки на някои елементи на Дъблинското ядро и техните стойности $Q = \{(N_j, V_j)\}$, където:

N_j – име на j-ия елемент на метаданни на Дъблинското ядро в описанието на потребителската заявка за търсене;

V_j – стойността на този елемент на метаданни.

Модели, основани на класификатори. Това е разновидност на най-простите модели за търсене. В дадения модел, документът се представя като множество от асоциирани с него атрибути. Атрибутите са идентификатори на класовете, към които се отнася дадения документ. Класовете формират йерархическата структура на класификатора.

Заявката може да се представи по два начина:

1. Прост вариант – заявката е идентификатор на някой клас от дадения класификатор. Критерий за релевантност на документа относно заявката – класът на документа съвпада с класа в представянето на заявката или е негов подклас.

2. Сложен вариант – в заявката може да бъдат указани няколко класа от класификатора. Критерий за релевантност на документа относно заявката – класът на документа съвпада с някой от указаните в заявката класове или е негов подклас.

Булеви модели. В булевите модели за търсене потребителят може да формулира заявката като булев израз, използвайки оператори И, ИЛИ, НЕ. Термите на заявката зависят от

конкретния вариант на модела на търсене. В булевия модел, ориентиран на търсене ‘по текста’, термите ще бъдат съответно думи, критерий за релевантност ще бъде условието за участие на някоя дума или словосъчетание в текста на документа. В булевия модел, ориентиран към търсене по класификатори, терми на изрази ще бъдат идентификаторите на класовете от класификатора. В булевия модел на търсене с използване на Дъблинското ядро, терми ще бъдат стойностите на елементите на метаданни. Документ, който има съвпадащи стойности на елементи на метаданните със стойности, зададени в заявката, се счита релевантен.

В общия случай, критерият за релевантност на документа към заявката в булевите модели на търсене е истинността на булевия израз, представен в заявката.

Безспорно достойнство на булевия модел на търсене е простотата на реализация. За основни недостатъци на модела се считат:

- отсъствието на възможности за аранжиране на намерените документи по степен на релевантност, тъй като липсват критерии за нейното оценяване;
- сложността на използване – далеч не всеки потребител може свободно да оперира с булеви оператори при формулиране на своите заявки.

Ще отбележим, че са предприемани опити за усложняване на булевия модел на търсене с цел осигуряване на възможности за аранжиране на множеството документи, получавани в резултат на заявката. В т. нар. разширени булеви модели [115], се въвеждат специални обобщения на булевите оператори, позволяващи да се припише по-високо тегло на документи, точно удовлетворяващи булевия израз на заявката, и понижено тегло на всички останали документи [2].

Векторни модели. Днес, векторните модели са най-разпространените и използвани на практика модели за търсене. Векторните модели, в отличие от булевите, без труд позволяват аранжиране на множеството от документи – резултат на заявката.

Същността на тези модели се свежда до представяне на документи и заявки под формата на вектори.

На всеки терм t_i в документа d_j и заявката q се съпоставя някакво неотрицателно тегло w_{ij} (w_i за заявката). По този начин, всеки документ и заявка могат да се представят като k-мерен вектора [6]:

$$\vec{d_j} \stackrel{\text{def}}{=} (w_{1j}, w_{2j}, w_{kj}),$$

където k е броят различни терми във всички документи.

Съгласно векторния модел, близостта на документа d_i към заявката q се оценява като корелация между векторите на техните описания. Корелацията може да се изчисли, например, като скаларно произведение ([1]) или разстояние (напр. на Минковски, евклидово, Манхатън, и др.) между съответните вектори.

Използват се различни подходи за избор на теглата. Един от най-простите подходи използва нормализираната честота на дадения терм в документа:

$$w_{ij} = \frac{n_{ij}}{N_j},$$

където n_{ij} – броят на повторения на дадения терм в документа; N_j – общия брой на всички терми в документа.

По-сложни варианти за пресмятане на теглата отчитат честотата на използване на дадения терм в другите документи на колекцията от документи, т. е. отчита се и дискриминантната сила на терма [6]. Всички тези варианти са възможни само при наличие на статистика за използване на термите в колекцията.

Вариациите в начините за изчисляване на теглата на термите и за оценяване на близостта между векторите, определя широкия спектър от модификации на всеки конкретен модел за търсене.

Вероятностни модели. За първи път идеи за подобни модели са предложени през 1960 г. [104]. В основата им стои *принципът за вероятностно аранжиране* (Probabilistic Ranking Principle, PRP). Същността на този принцип се заключава в следното – най-висока обща

ефективност на търсенето се постига в случай, когато резултатните документи се аранжират по намаляване на вероятността за тяхната релевантност към заявката. Отначало за всеки документ се оценява вероятността, че той е релевантен на заявката, а след това, според тези оценки, се извършва аранжиране на документите.

Известни са различни начини за получаване на тези оценки, а също така и на допълнителни предположения и хипотези, на основата на априорни сведения за документите в колекцията, които и определят конкретната реализация на вероятностния модел на търсене.

Например, тази оценка може да се изчисли, в съответствие с теоремата на Байес, с използване на някаква функция за вероятността на срещане на термите в даден документ. С помощта на заявката за търсене се определя вероятността за присъствие на дадения терм в релевантните документи, а по пълната колекция от документи се определя вероятността за присъствие на терма в нерелевантните документи [2].

Мрежи за извод. Също както и вероятностните модели, мрежите за извод се базират на принципа за вероятностно аранжиране на документите – резултат на търсенето [119, 124]. Основното им различие от вероятностните модели се заключава в това, че се използва не оценка за вероятността на релевантност на документа към заявката, а вероятността за това, че той удовлетворява информационните потребности на потребителя.

В рамките на дадения модел, процеса на търсене на документи се представя като процес на разсъждение в условия на неопределеност. В процеса на това разсъждение се оценява вероятността, че информационните потребности на потребителя, изразени с помощта на една или повече заявки, са удовлетворени.

Мрежа за извод, основана на Байесова мрежа, включва възли от четири типа. Възли от първия тип са документите на колекцията, изучени от потребителя в процеса на търсене. Възли от втория вид са терми, които се описват от съдържанието на документите. Възли от третия вид са заявки, състоящи се от терми, които се описват от съдържанието на документите. Възел от четвъртия тип в мрежата е само един, и той съответства на информационните потребности на потребителя, които не са известни на търсещата система.

Всички възли от първия и втория вид се формират предварително за дадената колекция. Възли от третия вид и техните връзки с възлите на терми, описващи документи, и възли на информационните потребности се формират за всяка конкретна заявка.

След като мрежата е построена се осъществява оценка на документите от колекцията. Това се реализира с разпространяване по мрежата на оценки за вероятностите на възлите за конкретния документа. Резултат от разпространяването е изчисляване на вероятността на възела на информационните потребности. При това оценката за всеки документ се построява независимо от оценките на другите документи, с отчитане на матриците, описващи връзките ‘възли от документи’ – ‘възли терми’, ‘възли терми’ – ‘възли заявки’.

Процесът на оценки се повтаря за всеки документ, след което те се аранжират на основата на изчислените оценки за вероятността на възела на информационни потребности [2].

Приблизително 60% от търсещите системи и модули функционират без използване на сериозни математически модели на информационното търсене. По-точно казано, съответните разработчици не поставят пред себе си задачата да реализират абстрактни модели, а в много случаи дори не подозират, че съществуват. На практика, в редица информационно-търсещи системи е реализиран много прост принцип: най-важното е търсещата програма да намира нещо, а потребителят да се оправя сам по-нататък.

Когато обаче става дума за повишаване качеството на търсене в голяма колекция от документи или за случаи на голям брой потребителски запитвания, освен емпирически предложените методи, се оказва полезно ползване на някакъв, дори и прост теоретичен апарат.

Моделът на търсене – това е вид опростяване на реалността, на основание на което се получава определена формална схема (сама по себе си ненужна никому), но на базата на която алгоритмите/програмите могат да вземат решение - кой документ да се счита за полезен (намерен) и как да бъде аранжиран спрямо други (намерени) документи. След приемане на теоретичния модел, свързан с изчисляване на тегловите коефициенти, например, често (впоследствие) на съответните формули се приписва и физически смисъл,

с който (евентуално) стават - за добро или не, естествени и обясними за разработчици и потребители.

Многообразните модели на традиционното информационно търсене (IInformation Retrieval - IR) е прието да се делят на 3 (три) вида: *теоретико-множествени* (булеви, размити множества, разширени булеви), *алгебрически* (векторни, обобщени векторна, латентно-семантически) и *вероятностни*.

Булевото семейство модели, по същество е първото, за което се досеща програмистът, който реализира пълнотекстово търсене. Принципът е прост: думата присъства – документът се счита за намерен, ако не – документът не е намерен. С две думи, класическият булев модел свързва теорията на информационното търсене с теорията на търсене и обработване на данни. Критиките към булевия модел (напълно справедливи) се свързват с неговата прекомерна 'определеност' и непригодност за аранжиране на намерените документи.

Ето защо, още през 1957 г., Joyce и Needham предложили да се отчитат честотните характеристики на думите с цел „... операцията на сравнение да е отношение на разстоянията между вектори" [47]. *Векторният модел* е успешно реализиран през 1968 г. от основателя на науката за информационно търсене Gerard Salton в търсещата система *SMART* (Salton's Magical Automatic Retriever of Text).

Аранжирането на документи в този модел е основано на емпиричното статистическо наблюдение, че колкото по-голяма е локалната честота на термина в документа (*TF*) и колкото по-рядко се повтаря термина в документите на колекцията *(IDF)*, толкова по-голяма е тежестта на дадения документ по отношение на термина. Означението *IDF* въвежда Karen Sparck-Jones през 1972 г. в статията [8] за *различителната сила* (*term specificity*). От този момент терминът *TF*IDF* широко се използва като синоним на векторния модел.

През 1977 г. Robertson и Sparck-Jones [112] обосновават и реализират вероятностния модел (предложен още през 1960 г.), който поставя началото на цяло семейство модели. *Релевантността* в този модел се разглежда като вероятност на това, че даденият документ може да се окаже интересен за потребителя. При това се подразбира наличие на вече

съществуващ първоначален набор от релевантни документи, подбран от потребителя или получен автоматично по опростена процедура. Вероятността да се окаже релевантен, за всеки следващ документ, се пресмята в зависимост от отношението на броя срещания на термини в релевантния набор и в останалата 'нерелевантна' част от набора. Въпреки че вероятностните модели имат някои теоретични преимущества (защото разполагат документите в намаляваща градация според вероятността да се окажат релевантни), на практика не получават широко разпространение.

Тук ще пропуснем подробностите и няма да изписваме обременителните формули за всеки модел (вж. [11]). Важно е да се отбележи, че във всяко от представените семейства, най-простият модел произлиза от предположението за взаимна независимост на думите и притежава просто условие за филтрация: документи, които не съдържат ключовата дума, никога не се намират. Усложнените (алтернативни) модели на всяко от семействата не считат ключовите думи на запитването за взаимно независими, дори позволяват да се намират документи, които не съдържат нито една дума от запитването.

Способността да се намират и аранжират документи, които не съдържат ключови думи от запитването, често се счита за признак на изкуствения интелект или на *търсенето по смисъл* и се отнасят към преимуществата на модела.

За пример ще представим само един, може би най-популярния модел. В теорията на информационното търсене е прието този модел да се нарича *латентно-семантично индексиране* (LSI), с други думи – *откриване на скритите смисли/значения*). Този алгебричен модел е основан на сингулярно разложение на правоъгълна матрица, което асоциира думите с документите. Елемент на матрицата се явява честотната характеристика, която отразява степента на връзка между думата и документа, например, *TF*IDF*. Вместо изходна матрица с размерност от порядъка на милион, авторите на метода [23, 37] предлагат

използване на 50-150 'скрити' смисъла[1], съответстващи на първите *главни компоненти на нейното сингулярно разлагане*.

Доказано е [28], че ако разглеждаме само първите k сингулярни числа (останалите се приемат за равни на нула), ще получим най-близката от всички възможни апроксимации на първоначалната матрица от ранг k (в известен смисъл нейната *най-близка семантична интерпретация от ранг k*). Намалявайки ранга, може да филтрираме нерелевантните детайли; увеличавайки, се стараем да отразим все повече нюанси в структурата на реалните данни.

Операцията на търсене или по-точно намиране на *'близки' документи* рязко се опростява, тъй като на всяка дума и всеки документ се съпоставя относително 'къс' вектор, съставен от k *смисъла* (редовете и стълбовете на съответната матрица). Поради недобро осмисляне или може би поради други причини[2], директното използване на латентно-семантичният модел LSI не е получил широко разпространение. За спомагателни цели (автоматично филтриране, класификация, предварително намаляване на размерността на други модели) този метод намира добро приложение.

1.2 Организация и методи

Създадени са стотици системи за информационно търсене и хиляди функции за търсене, вградени в различни приложения. Във всички тези случаи (независимо от математическия модел), идеите, софтуерните програми и структурите от данни са достатъчно прости. Съществува мнение, че всяко ново поколение програми е по-съвършено от предходното. Друг краен възглед се състои в това, че „всичко ново – това е добре забравено старо“. По всяка вероятност, истината, в случая на софтуерните системи за информационно търсене, е по средата.

---

[1] За големи колекции броят на 'смислите' се увеличава до 300.

[2] Експерименти на руски изследователи с LSI, например, установяват, че 'смисъл номер 1' в Рунете имат всички англоезични документи, 'смисъл номер 3' – всички форуми и т.н.

Идеите на 60-те – 80-те години за итеративно уточняване на заявките за търсене, за разбиране на ЕЕ заявки, за търсене по смисъл, за генериране на свързан отговор, и т.н., обаче, все още са неосъществени мечти.

1.2.1 Организация на данните

Системите за информационно търсене оперират със сравнително елементарни структури данни от 4 (четири) типа, които се използват от съответните търсещи алгоритми.

Около 75% от алгоритмите изискват ‘индексиране’ - предварителна обработка на документи, при която се създава спомагателен файл (‘индекс’), призван да опрости и ускори самото търсене. Това са алгоритми за *инвертиране на файлове*, *суфиксни дървета*, *сигнатури*. В крайния ('изроден') случай на *директно/пряко търсене*, предварителният етап на индексиране изобщо отсъства, а търсенето се извършва с помощта на последователно преглеждане на документите от колекцията.

През последните 30 години са развити множество идеи, съкращаващи в 'пъти' търсенето при *директно търсене*. Тези алгоритми подробно са описани и сравнени в литературата ([3, 117]). Въпреки че директното търсене във всички текстове е достатъчно бавно, алгоритмите за пряко търсене се използват дори в Интернет. Норвежката търсеща система Fast (www.fastsearch.com) разполага 256 чипа на една платка, всеки реализиращ логика на пряко търсене на опростени регулярни изрази [33]. Това позволи Fast да обслужва достатъчно много заявки за единица време.

Освен това са създадени алгоритми, комбиниращи индексно търсене за намиране на блок текст със следващо директно търсене вътре в блока (в това число и в Рунет [39]). Директните алгоритми имат принципиално печеливши (в сравнение с други подходи) отличителни черти. Например, неограничени възможности за приближено търсене, тъй като всяко индексиране винаги е съпроводено с опростяване и нормализация на термини, а следователно и със загуба на информация.

Инвертираният файл е най-простата структура данни, интуитивно позната на всеки грамотен човек. Представлява азбучно подреден списък от думи, за всяка от които са изброени позициите/характеристиките, в/с които се среща. Търсещият алгоритъм се състои

в търсене на необходимата дума и обработване на списъкът от позиции. За икономия на памет и ускоряване на търсенето се използват два подхода. *Първо*, може да се икономисват подробности за самата позиция. В най-подробният вариант в инвертирания файл може да се съхранява и номера на думата, и изместването (в байтове) от началото на файла, и цвета и размера на шрифта, и т.н. Често се указват само номера на документа и броя на срещания на думата в него. Именно такава опростена структура се счита за основна в класическата теория на информационното търсене. *Вторият начин* за компресиране: подреждат се позициите за всяка дума по нарастване на адресите и за всяка позиция се съхранява не пълния й адрес, а разликата от предходния. Допълнително към този начин за съхраняване може да се 'наложи' и някакъв начин за опаковане (на практика в системите за търсене се използва рядко – печалбата е неголяма и процесорното време се изразходва неефективно). В резултат на подобни подобрения, размерът на инвертирания файл, като правило, съставлява от 7 до 30 процента от размера на първоначалния текст, в зависимост от подробността на адресацията.

Неднократно са предлагани и други алгоритми и структури от данни, различни от използваните в инвертираното и директното търсене. Преди всичко, това са *суфиксните дървета* [31, 103] и т. нар. *сигнатури* [32]. Първият от тях е функционирал и в Интернет като патентован алгоритъм на системата за търсене *OpenText* [107]. Вторият – метода на сигнатурите, представлява преобразуване на документа към блокови таблици на *хеш-значенията* на неговите думи – ‘сигнатура’ и последователно преглеждане на ‘сигнатурата’ по време на търсене. Нито един от тези два метода, обаче, не е получил широко разпространение.

На проблема за класификация на текстова информация е посветено много внимание [116, 125]. Голяма част от предложените *методи за класификация* [116] се основават на използване на прост векторен модел за описание на документи (Vector Space Model) – класически модел в областта на търсене на информация. В рамките на този модел, документът се описва с вектор, в който всеки използван в документа *терм*, т. е. словоформа, се съпоставя на неговата значимост (тегло) в този документ. Значимостта на терма се основава на статистическа информация за срещанията на терма в този, и (възможно) други документи. Описанието на тематиката също се представя с вектор и за оценка на близостта

на документа и тематиката се използва скаларното произведение на векторите на описание на тематиката и описанието на документа.

Множеството известни методи за тематичен анализ на текст може условно да се раздели на две големи групи – лингвистичен и статистически анализ.;

Първият (*лингвистичен анализ*) е ориентиран на извличане на смисъла на текста по неговата семантична структура. Вторият (*статистически анализ*) – към честотно разпределение на думите в текста. Може само условно да се говори за принадлежност на някой подход към конкретна група; в реални задачи за обработка на текстови документи много често се налага използване на съчетания на методики от двете групи с един или друг акцент.

1.2.2 Лингвистични методи

Малко встрани от статистическите модели и структурите от данни се намира клас алгоритми, които традиционно се отнасят към лингвистиката. Точна граница между статистическите и лингвистичните методи не може да се постави. Условно може да се счита, че лингвистичните методи се опират на лексикални ресурси/речници (морфологични, синтактични, семантични), създадени от човек. Въпреки съществуващи мнения, че за някои езици (напр. английски език) лингвистичните алгоритми не внасят съществено увеличаване на точността и пълнотата [106], все пак в някои случаи (особено за флективни езици като българския език), известна лингвистична обработка може да се окаже много полезна. Без излишни подробности ще приведем списък от задачи, срещани в информационното търсене, които могат да бъдат успешно решавани от лингвистични или близки до тях методи:

- автоматично определяне на езика, на който е написан документа;
- изключване на думи, които не носят информация (т. нар. ‘стоп-думи’);
- сегментиране (tokenisation) – отделяне на думи, дати, имена, изречения;
- лематизация (нормализация, привеждане на словоизменителните форми в основни, вкл. ‘непознати’ за системата);
- разделяне на сложни думи;
- пълно или частично снемане на омонимия (disambiguates);.
- разделяне на именни групи;

- автоматично резюмиране и определяне на ключови думи в текстови документи, и др.

Все още са редки случаите, когато в изследванията и в практиката на построяване на ИТС, се използват алгоритми за словообразувателен, синтактичен или семантичен анализ. При това (погрешно) под семантичен анализ много често се разбира някой статистически алгоритъм (LSI, невронни мрежи, и др.), и ако все пак някъде се използват тълковно-комбинаторни или семантични речници, то те са за тесни предметни области.

Общият лингвистичен анализ може да се представи на четири взаимно допълващи се равнища на анализ.

Лексичен анализ. Заключава се в разбор на текстовата информация на отделни абзаци изречения, думи, определение на езика на изложение, типа на изреченията, и т.н. Даденият анализ не е представлява съществен проблем за реализация.

Морфологичен анализ. Свежда се до автоматично разпознаване на частите на речта за всяка дума в текста (на думата се поставя в съответствие лексико-граматически клас). Често морфологичният анализ се използва като предварителна процедура за обработка на документите при статистическите методи за анализ (привеждане на думи към основна/базова форма).

Морфологически анализ за българския език може да се реализира практически със 100% точност, благодарение на неговата развита морфология. За английския език алгоритмите, приписващи на всяка дума в текста най-вероятният за дадената дума лексико-граматичен клас (синтактическа част на речта), работят с точност около 90%, което е обусловено от лексическата многозначност на английския език.

Синтактичен анализ. Състои се в автоматично отделяне на елементи на изреченията – именни групи, терминологически цели, предикативни основи. Синтактичния анализ позволява да се повиши интелигентността на обработката на тестова информация и е в основата на работата със семантичните елементи на текста.

Семантичен анализ. Състои се в определяне на информативността на текстовата информация и отделяне на нейната информационно-логическа основа. Провеждането на автоматизиран семантичен анализ на текста предполага решаване на задачата за откриване и оценяване на смисловото съдържание на текста. Дадената задача е трудно формализуема вследствие на необходимостта от създаване на апарат за експертна оценка на качеството на информацията.

Реализацията на семантичен анализ на текстова информация предполага задължително използване на експертни системи или системи за изкуствен интелект при откриване на смисловото съдържание на текста. В момента се провеждат само отделни експерименти за провеждане на семантичен анализ на текстова информация, което в голяма степен е обусловено от изключителната сложност на проблема.

Основният проблем на подобен тип търсене е сложността за еднозначна автоматическа интерпретация на съдържанието на текстовите документи. и. Сложността на интерпретация се затруднява от липсата на формално формулиране на информационните потребности на потребителите и трудностите за определяне на степента на съответствие между даден документ и информационните потребности на потребителя].

Проблемите са обусловени от липсата на каквато и да е регулярна структура на текстовите документи, написани на естествен език. Подобни информационни ресурси е прието да се наричат неструктурирани или слабо структурирани. Разработването на методи за анализ на слабо структурирани информационни ресурси изглежда твърде перспективно и многообещаващо направление на изследванията в областта на информационното търсене.

Като правило, търсещите системи, ориентирани към откриване на слабо структурирана информация се използват за търсене на документи в големи и динамични информационни колекции (напр. в Интернет). Особеността на такива колекции е отсъствието на ясно изразени структурни организации, позволяващи подреждане и еднозначно класифициране на съхраняваните документи по тематични направления.

Процесът на търсене на текстова информация, реализиран от типична търсеща система в множество от документи със слабо структурирана информация, включва в себе си следните елементи:

- формализация на заявката за търсене (представяне от потребителите, в един или друг вид, на техните информационни потребности);
- предварителен подбор на документи по формални признаци на наличие на интересуващата ни информация (например, наличие в текста на документа на една от думите в заявката, ако заявката е формулирана на естествен език);
- анализ на подбраните документи (напр. лингвистичен, статистически, и т.н.);

оценка на съответствието между смисловото съдържание на намерената информация с изискванията на заявката за търсене (аранжиране), и др.

1.2.3 Статистически методи

Като правило, статистическият анализ е честотен анализ на колекцията документи, в една или друга форма. Същността на този вид анализ се заключва обикновено в преброяване на повторенията на думите в съответните текстови документи и използване на резултатите за (статистическо) оценяване на конкретни събития (например, оценяване на значимостта на ключова дума чрез изчисляване на теглови коефициенти).

Всевъзможните варианти на различни реализации за броене на думи със следваща обработка на резултатите от преброяването, образуват широк спектър методи и алгоритми.

Ще разгледаме един от най-ефективните статистически подходи - латентно-семантичния анализ.

Латентно-семантичният анализ (LSA - Latent Semantic Analysis) е теория и метод за откриване на структурата на семантичните връзки и извличане на контекстно-зависими значения на думите чрез статистическа обработка на големи набори от текстови данни [4, 6, 51]. Докато повечето статистически техники се опитват да търсят взаимната връзка между отделни думи (или между отделни документи), LSA се концентрира върху двойки от вида дума-документ, а връзките дума-дума и документ-документ разглежда като производни. В основата му стои хипотезата, че между отделните термини (думи или словосъчетания) и

обобщения контекст (изречения, абзаци и цели текстове), в който се срещат, съществуват неявни (латентни) взаимовръзки, обуславящи съвкупност от взаимни ограничения. Откриването и правилното им третиране, дават възможност на LSA да се справи успешно със синонимията и частично с полисемията – двата най-тежки проблема при статистическата обработка на текстова информация. Методът е напълно автоматичен и не използва никакви речници, семантични мрежи, бази от знание, концептуални йерархии, граматични, морфологични или синтактични анализатори и др.

Методът позволява автоматически да се различават смислови отличия на една и съща дума в зависимост от контекста на нейното използване в документа, като откриването на семантичната структура става напълно автоматично, т. е. не изисква ръчно съставяне на речници.

Методът се използва не само в областта на информационното търсене [27, 41], но и в задачи за филтрация и класификация.

Латентно-семантичният анализ се основава на идеята, че съвкупността от всички контексти, в които се среща (или не) дадена дума, задава множество двустранни ограничения, които позволяват да се определи близостта/приликата между смисловите значения на конкретна дума с всяка друга от съответното множество. В качеството на първоначална информация, LSA използва матрица *терми-по-документи*, описваща използвания за обучение на системата набор от данни. Елементите на тази матрица представят честотите на използване на дадена дума (терм) в съответния документ.

Най-разпространеният вариант на LSA е основан на използване на разложение на матрицата по сингулярни стойности (SVD) [22]. Огромната начална матрица се разлага на множество от k (обикновено от 70 до 200) ортогонални матрици, линейната комбинация на които се използва за приближение на първоначалната матрица. Така на всеки документ и на всеки термин се съпоставя вектор с малка размерност от едно и също (латентно) семантично пространство. Степента на близост между два документа/термина се определя като функция на съответните им вектори, най-често чрез косинуса на ъгъла, заключен между тях.

Съгласно теоремата за сингулярно разложение, всяка реална правоъгълна матрица X с размерност $m*n$ може да се разложи като произведение на три матрици (т. нар. SVD разлагане):

$$X = U\Sigma V^T,$$

където U е ортогонална матрица с размерност $m*m$, V е ортогонална матрица с размерност $n*n$, а Σ е диагонална матрица с размерност $m*n$, *за* елементите на която $s_{ij} = 0$, ако i не е равно на j, а s_{ii} са т. нар. сингулярни стойности на матрица X (равни на квадратния корен от съответстващото собствено значение на матрицата XX^T.

Особеността на разлагането се състои в това, че ако в Σ се оставят само k-те най-големи сингулярни стойности, а в матриците U и V - само стълбовете, съответни на тези стойности, то произведението на получените при това матрици Σ_{lsa}, U_{lsa} и V_{lsa}, ще е най-доброто приближение на първоначалната матрица X от ранг k:

$$X \cong \hat{X} = U_{lsa}\Sigma_{lsa}V_{lsa} .$$

Същността на латентно-семантическия анализ се заключава в това, че ако в качество на X се използва матрицата от терми на документите, то матрица $\hat{X}$, съдържаща само k първи линейно независими компоненти на X, отразява основната структура на асоциативни зависимости, присъстващи в първоначалната матрица, като едновременно, информационният шум се отстранява в голяма степен,.

По указания начин, всеки терм и документ може да се представи (моделира) чрез вектори в общо пространство с размерност k (т. нар. *пространство на хипотези*), а близостта между всяка комбинация терми или документи може лесно да бъде представена (изчислена) с помощта на скаларно произведение на вектори.

Едно от най-популярните приложения на LSA е за извличане на информация, подобно на търсещите машини в Интернет: обслужване на потребителски заявки, в резултат на което се връщат електронни документи, които в някакъв смисъл са близки до запитването. В този случай LSA е наричан още *латентно-семантично индексиране.* Емпиричните тестове в това направление са обещаващи. *Дируестър, Дюмаис* и *Харигман* описват два основни теста. При

първия са изследвани 5832 думи, 1033 медицински текста и 30 запитвания към тях. Полученият резултат дава 13% средно подобрение спрямо методи, отчитащи само честотата на срещане на думите. Други изследвания сочат до 30% подобрение.

Интересни експерименти са направени с две разновидности: обратна връзка (англ. *Relevance Feedback)* и крос-езиково извличане на информация (англ. *Cross-Language Information Retrieval).*

При метода на *обратната връзка* потребителят посочва кои от намерените документи удовлетворяват неговите информационни потребности, което се използва от системата, за да предложи ново множество от документи. За целта, обикновено се извършва допълване на заявката с текста на релевантните документи. Тестове показват, че в някои случаи, допълване на заявката с първия релевантен документ води до 33 % подобрение, а ако се включат, например, първите три полезни документа — 67%. Не са ни известни опити за използване на отрицателни примери (нерелевантни) документи, но по всяка вероятност, това би могло също да даде резултат.

В метода за *мултиезиково извличане на информация* се предлага начин за извличане на информация от колекция текстови документи (написани на различни езици), като заявките също могат да бъдат на различни езици. Така например (Ландауер и Литман), за множество от документи, всеки от които има английска и френска версия се провежда LSA върху множеството от двойки документ-превод, разглеждани като единствен документ. В резултат се получава общо латентно пространство за двата езика, което позволява обработване на заявки (и на двата езика) и намиране на документи (също на всеки от двата езика).

Изборът на най-добра размерност за *k* при LSA е нерешен изследователски проблем. В идеалния случай, *k* трябва да е достатъчно голямо за изображение на цялата реално съществуваща структура от данни, едновременно достатъчно малко, за да не включва случайни и несъществени зависимости. Ако избраното *k* твърде голямо, то методът губи своята ефективност и се приближава по характеристики към стандартните векторни методи. Твърде малко *k* не позволява да се ‘улавят’ различията между подобни думи или документи. Изследвания показват, че с увеличаване на *k*, качеството отначало расте, а след това започва да пада.

Основен недостатък на латентно-семантичния анализ, който в значителна степен ограничава неговото приложение, е обстоятелството, че е разчетен за обработване на цялата колекция (т.е. всички документи на колекцията трябва да бъдат достъпни за обработка).

Някои проблеми на LSA произтичат от несъобразяване с граматичната структура на текста (вкл. морфология, синтаксис, словоред, отрицания и др.). Така например, двете изречения ‘Иван написа писмо на Стоян’ и ‘Стоян написа писмо на Иван’ са еквивалентни от гледна точка на LSA. Също така, висока степен на близост (дори 100%, ако думата ‘не’ се разглежда като стоп-дума) имат изреченията ‘Иван написа писмо на Стоян’ и ‘Иван не написа писмо на Стоян’, въпреки че са противоположни по смисъл.

Фундаментален проблем на LSA е и неговата *статистическа несъстоятелност*. LSA се основава на проектиране в пространство с по-малка размерност, което е най-добро приближение на изходната матрица в смисъл на най-малки квадрати. За съжаление, това е вярно само за нормално разпределени данни, а честотите на термините в документите не са такива и за тях са по-подходящи други модели (Пуасоново или отрицателно биномно разпределение).

Най-накрая, LSA е патентован алгоритъм[1], което силно ограничава неговото използване в реални комерсиални приложения.

1.3 Характеристики на търсещите системи

Във връзка с извеждане на основните характеристики на съвременните системи за информационно търсене, накратко ще разгледаме някои ***специфични черти на големи динамични колекции от документи***, до която е предоставен ‘он-лайн’ достъп (до голяма степен същото се отнася и до търсенето в Интернет).

[1] Патенти "Computer information retrieval using latent semantic structure" (U. S. Patent No. 4,839,853, Jun 13, 1989) и "Computerized cross-language document retrieval using latent semantic indexing" (U. S. Patent No. 5,301,109, Apr 5, 1994).

Обем на информацията. Характерна особеност е огромния и динамично нарастващ обем на достъпни информационни ресурси (оценки в терабайтове - за Интернет); за нашия 'правен' случай – брой на документите от порядъка на 10^4. Във връзка с това възникват изисквани към използваните алгоритми (и структури от данни) за организация на търсенето.

Динамика. Дадената особеност значително затруднява използването на общи статистически характеристики на колекцията. Особено поява на нови страници/документи или промяна на тяхното разположение; по някои статистически данни, средното време за живот на 50% от страниците в Интернет не е повече от 10 дни, ежемесечно около 40% от страниците се обновяват, а обемът на данните се увеличава 2 пъти на всеки 2 години.

Взаимосвързаност. Една от особеностите на документалното информационно пространство е взаимната връзка между представените в него документи/страници. Връзката се реализира със система от позовавания/референции, която може да бъде ползвана при реализацията на някои методи за търсене.

Потребители. Колекциите от документи/страници се ползват от потребители, които се различават значително по квалификация и подготовка. Нещо повече, много от тях ме умеят грамотно и ефективно да формулират заявките си за търсене. Статистиката показва, че не по-малко от 60% от заявките за търсене в Интернет са съставени от 1-2 думи (в класическите информационно-търсещи системи това число е 7-9 думи [45]). В резултат се намира огромно количество документи, а резултатите са твърде далеч от реалните информационни нужди на потребителя. Други изследвания върху поведението на потребителите показват, че много от тях не са готови за продължително очакване на резултатите от търсене и анализ на резултатното множество за откриване на необходимите документи: 58% от потребителите се ограничават с изучаване на първия екран с резултати на търсенето, 67% не се опитват да модифицират своята първоначална заявка [45].

Мултиезиковост. Ясна тенденция за създаване на мултиезични информационни среди (Интернет, ЕС), с което на дневен ред се поставят задачите за мултиезиково и кросс-езиково търсене. Решаването на тези задачи предполага реализация на алгоритми за търсене, които са независими от езика на представяне на анализираните документи и от езика на информационните заявки на потребителите.

Други особености (характерни не само за Интернет-пространството, но и за някои колекции от документи), са: *свободното публикуване*; *дублирането и пренасищането* (около 30% от информацията е точно или приближено копие на други документи [118]); *отсъствието на контрол на качеството* на публикуваните документи (от недостоверна информация и непроверени данни, до многочислени орфографически и граматически грешки); времево ограничен достъп (не всички сървъри с документи в Интернет са достъпни в режим 7/24), и др.

При това критериите за качество, използвани в традиционните системи за текстово търсене, стават неадекватни, например, критерият за пълнота на търсенето, т.е. процентът на открити релевантни документи [21].

Появата на Интернет и глобалните системи за търсене, повиши изискванията на потребителите към условията за използване на системите за текстово търсене и изведе на преден план нови изисквания към тях.

Основните изисквания към съвременните ИТС могат да се формулират по следния начин [2]:

- ефективна обработка на много голяма колекция от документи;
- подобрено отразяване на смисловото съдържание на документите и на потребителските заявки за търсене;
- реализация на мултимедийна обработка, т.е. съвместна обработка на документи в различни формати и представяния – текстови документи, изображения, аудио, видео и др.;
- реализация на ефективни методи за търсене в потоци от документи (задачи за филтрация), и др.

Отделно трябва да се отбележи повишаването на изискванията към търсещите системи по отношение на така наричания човешки фактор, определящ ефективността на взаимодействие човек-търсеща система. На първо място, това се отнася към проектирането на потребителски интерфейси и организацията на работа на човек с търсещи системи. В последно време се забелязват определени тенденции към интелектуализация на

информационните системи, ориентация на системите към човека, безусловно засяга и системите за информационно търсене, вкл.:

- мултиезиково и крос-езиково търсене;
- фактографско търсене (реализация на ИТС с отговори на въпроси, зададени от потребителя);
- интерактивно търсене (реализация на диалог с потребителя по време на търсенето, уточняване на заявките и т.н.);
- търсене по заявка, която е под формата ‘документ-образец’;
- търсене на видеоданни (по съдържание на видеоданните, търсене на известни обекти, задачи за разпознаване и т.н.), и др.

Постигнатите резултати в областта на информационното търсене не позволяват, засега, да се говори за безусловна ефективност и качество на съвременните ИТС. Едновременно с това, днес са известни добри практики и решения в областта на текстовото търсене. Ще разгледаме някои от тях в светлината на задачите за тематичен анализ и идентификация на документи.

Съществува широк спектър от предлагани решения и перспективни направления на изследвания в областта на информационното търсене, започвайки от построяване на глобални разпределени информационни структури и търсещи системи, и завършвайки с елементарни на пръв поглед въпроси за анализ на документи. Ще забележим, че именно от методите на анализ в голяма степен зависи ефективността на системите, т.к. те са в основата на всяка ИТС и по този начин определят техните възможности и ограничения.

Освен това, има още един важен фактор, определящ по наше мнение ефективността на всяко информационно търсене – човешкия фактор. Често в болшинството изследвания, отнасящи се към информационното търсене, този фактор или се игнорира, или неговото значение се подценява. Но именно човек в крайна сметка се ползва от разработените информационно-търсещи системи. Отчитането на човешкия фактор, спецификата на неговото действие, предпочитания и очаквания са перспективни и многообещаващи направления на изследванията.

Повечето съвременни търсещи системи и начини за организация на пълнотекстово търсене и методи за анализ на документи не отчитат в достатъчна степен човешкия фактор. а именно, че в голяма степен търсенето се определя от неясни и слабо формализуеми условия, които в значителна степен зависят от опита и предпочитанията на потребителите. Далеко не винаги потребителят на информационно-търсеща система (ИТС) може ясно и еднозначно да формулира именно този набор от ключови думи, който ще доведе до искания резултат. В случая, става дума за вариант на търсене на основата на формиране на информационни заявки, състоящи се от набор ключови думи и някои управляващи елементи на езика на заявки (вариант на търсене, който днес е най-разпространен и сравнително добре разработен).

В зависимост от начина за организация на търсенето (и документите), съвременните системи за информационно търсене (ИТС) се разделят на две големи групи. В първата група попадат ИТС, които осъществяват търсенето/обработката на документите, предполагайки, че последните са организирани в т.нар. ‘каталози’, а във втората – ИТС, които осъществяват т.нар. ‘тематично търсене’ по текстовото съдържание на съхраняваните документи.

Библиографско търсене, или търсене ‘по каталог’. *Каталозите за търсене* в голяма степен са ориентирани към структурна организация на тематични колекции с удобна система на позоваванията и йерархии на документи по тематични колекции. Това позволява на потребителя самостоятелно да намира търсения документ, преглеждайки структурата на каталога, или да използва механизми на търсене, ориентирани към дадения каталог. Във всеки случай, организацията на информацията, нейното структуриране и предварително попълване на тематическия каталог са първостепенен критерий, определящ качеството и ефективността на търсене на ИТС. Попълването на тематичния каталог от документи може да се изпълнява както в ‘ръчен’, така и в автоматичен режим. Най-качествен все още остава ръчният подбор на документи за такива каталози с привличане на експерти по конкретни тематични раздели или полуавтоматичен вариант с предварително ‘грубо’ търсене на документи следващата им селекция.

Информационното търсене, в този случай, осигурява намиране на документа по неговите каталожни данни (реквизити), например, по наименование, по идентификатор на тематика, по име(на) на автора(ите), дата на публикация/модификация, и т.н. Основа на каталога са

предварително зададен модел на представяне на реквизити, реализирани като база данни (БД), в съответствие с който се осигурява запис на отделни елементи на реквизитите и следващо търсене по тях.

Основният проблем и недостатък на подобен вариант на информационно търсене е необходимостта от изпълнение на значителен обем на дейности по предварителната организация и попълване на каталога. Като правило, това е ръчна класификация с привличане на експерти. Подобен подход позволява организиране само на малка част от достъпните информационни ресурси. Имайки предвид огромните обеми от информационни ресурси, натрупани към момента, и тенденцията за ускореното им (дори експоненциално) увеличаване, стават разбираеми проблематичността на структуриране и организация на документалното информационно пространство.

Тематично търсене или търсене 'по текста'. Този вариант на търсене е ориентиран към намиране на документи по тяхното съдържание. Тук се отнася така наричаното пълнотекстово търсене. Общата схема на подобно търсене се заключава във формулиране на някои заявки на потребителите относно съдържанието на документа, и избор от множеството достъпни документи на тези, които удовлетворяват заявката. Такъв вариант на търсене е удобен, преди всичко с това, че отпада необходимостта за предварително разделяне на документите по различни категории. Последното е особено актуално при значителен обем на достъпните документи, висока динамики на тяхното обновяване или отсъствие на някои реквизити (такава ситуация е характерна, например, за Интернет).

1.4 Оценка на качеството

Какъвто и да е моделът на търсещата система, тя се нуждае от оценка на качеството на информационното търсене. Оценката на качеството е фундаментална идея в теорията на търсенето, защото именно благодарение на тази оценка може да се говори за приложимост или не, на един или друг модел, както и да се обсъждат/сравняват техните теоретични

аспекти и характеристики. Обикновено[1] за оценка на качеството се използват две характеристики:

- *точност (precision)* – частта на релевантните материали в получения от ИТС отговор;
- *пълнота (recall)* – частта на релевантните документи в отговора, спрямо общият брой на релевантните документи в колекцията от документи, обект на информационното търсене.

За естествено ограничение на качеството, постигано от конкретни ИТС, служи наблюдението (Donna Harman [14]), че мненията на двама специалисти за релевантност на сравнявани документи съвпадат средно около 40%, като точността и пълнотата са около 65%. От тук следва и примерната горна граница за качеството на системите за търсене, тъй като това качество би могло да се измерва и чрез съпоставяне на мненията на експерти.

Двете характеристики се използват за оценяване на качествата на предлаганите модели за информационно търсене и техните характеристики, в рамките на създадената от Американския институт на стандартите (NIST), конференция за оценяване на системите за текстово търсене (TREC – text retrival evaluation conference))[2]. Стартирала в 1992 г. като консорциум от 25 групи, към момента е натрупала значителен материал, на основата на който се оценяват и усъвършенстват търсещите системи. На всеки пореден конгрес се предлага тема по всяко от направленията, интересуващи изследователите. Всяка тема включва колекция от документи и заявки/запитвания. Примерни теми на конференцията са:

- произволни запитвания (присъства на всички конференции);
- многоезично търсене;
- маршрутизация и филтриране
- високо точно търсене (с единствен отговор, получаван за определено време);

[1] Но не задължително – има и 'алтернативни' метрики.

[2] Материалите на конференциите са публично достъпни на адрес trec.nist.gov/pubs.html.

- взаимодействие с потребителите;
- естествено-езикова тема;
- отговори на въпроси;
- търсене в необработени (току що сканирани) текстове;
- гласово търсене
- търсене в много големи бази данни (20 GB, 100GB, и т.н.);
- разпределено търсене и сливане на резултати от търсене в различни системи;
- WEB корпус (напр. с подбор в определен домейн, напр. .gov), и др.

Както може да се установи от списъка на TREC-темите, областта на информационното търсене често се причисляват задачи, които или имат с тях обща идеология (класификация, филтриране, резюмиране, маршрутизация), или са неразделна част от процеса на търсене (кластеризация на резултати, разширяване и стесняване на запитванията, обратна връзка, търсещия интерфейс и езика на запитване). На практика, няма търсеща система, която да не решава, в една или друга степен, някоя от разгледаните задачи.

Съществуват и други авторитетни международни конференции, посветени на обсъждане на въпросите и качеството на информационното търсене: SIGIR (Special Interest Group on Information Retrieval) конференции, провеждани от ACM SIGIR (ACM - Association of Computing Machinery) – международна група от специалисти по IR; WWW (World Wide Web) Conference – конференция по решаване на задачи, свързани с Интернет и др. Високият авторитет на TREC, SIGIR, WWW и участието в тях на водещи изследователски колективи и разработчици на технологии на информационното търсене, до значителна степен определят приоритетните в областта направления.

Осигуряването на средства за решаването на допълнителни задачи е решаващ довод в конкурентната борба между предлаганите на пазара информационно-търсещи системи. Например, кратките анотации, състоящи се от информативни цитати в документа, с които някои търсещи системи съпровождат резултатите на своята работа, им осигуряват в много случаи конкурентни предимства.

За пример да разгледаме т. нар. ‘разширяване на заявката’, което обикновено става с привличане към търсене на асоциирани термини. Решението на тази задача е възможно в

две форми – локална (динамична) и глобална (статична). Локалните техники се опират на текста на заявката и анализират само документи, намерени по нея. Глобалните ‘разширения’ могат да оперират с тезауруси, както априорни (лингвистически), така и построени автоматично по цялата колекция документи. По общоприето мнение, глобалните модификации на заявките чрез тезауруси работят неефективно, понижавайки точността на търсене. Успешният глобален подход е основан на 'ръчно' построени статични класификации, например, Web-директории. Този подход широко се използва в интернет-търсачки при операции за стесняване или разширяване на заявките.

Нерядко реализацията на допълнителни възможности е основана на същите или много подобни принципи и модели, както и самото търсене. Да сравним, например, невронният модел на търсене, в който се използва идеята за предаване на затихващи колебания от думите към документите и обратно към думите (амплитудата на първото колебание – все този *TF*IDF*), с техниката за локално разширяване на заявката за търсене. Последната е основана на *обратната връзка* (relevance feedback), в която се вземат най-много *смислоразличителни* (контрастни) думи от документите.

За съжаление, локалните методи за разширяване на заявката, въпреки ефектните технически идеи от тип ‘Term Vector Database’ [123], все още остават крайно ‘скъпо’[1] удоволствие.

1.5 Извличане на информация

С бързото нарастване на източниците на информация и електронни документи, достъпни през Интернет, все повече изследвания се фокусират върху проблемите на извличане на информация [20, 29].

Методи за извличане на информация първоначално са приложени към идентификация на полезна (за потребител) информация от документи, написани на естествен език, с

[1] В смисъл на 'изчислителни ресурси'.

цел - преобразуване на извлечения текст в определен (например, в табличен) формат. С развитието на световната мрежа Интернет, през последните години, извличането на информация започна широко да се използва и за откриване на текстови фрази във Web-страници, които, за разлика от документите на естествен език, все пак са полуструктирани (имат определена, дефинирана поне с тагове структура).

Според [19], въпреки че *системите за извличане на информация* се различават една от друга, в зависимост от естеството на решаваните от тях задачи, може да се открият поне ***5 (пет) общи функционални елемента*** в тяхната архитектура:

- сегментация и токенизация (необходими при всяка нормализация на входния текст), и поставяне на етикети (за частите на речта и техните семантични характеристики);
- предсинтактичен анализ на изречения (за идентифициране на синтактични фрази като подлог, пряко допълнение, предложна фраза и т.н., чрез провеждане на синтактичен анализ и разбор);
- търсене и извличане на специфични за съответната предметна област фрази[1];
- сливане (за идентификация на кореферентни връзки с други елементи на текста);.
- генериране на шаблони, с попълване на съответните им полета с данни (на базата на извлечените фрази) и създаване на шаблони за съхраняване в база данни, за отпечатване, и др.

Съществуват два основни подхода за построяване на системи за извличане на информация [10] – експертен (knowledge engineering) и автоматично (само)обучение върху анотирани (или не) корпуси.

[1] За разлика от предходните два компонента, които са относително независими от предметната област, извличането е предметно-зависимо.

Първият (експертен) подход, който изисква много човешки усилия, време и средства, разчита на ‘инженери на знанието’ – експерти, които са опитни не само по отношение на системите за извличане на информация, но и притежават висока квалифицирани в съответната предметна област. Експертите е необходимо да изучават множество от свързани с областта документи и да откриват (емпирично) полезни и съществени отношения, с цел създаване на системи от евристични и граматични правила, предназначени за използване в съответната система. В този случай, ефективността и производителността на системата силно зависят от компетентността на привлечените експерти.

‘Ръчно’ проектираните системи обикновено се справят добре с документи от определен тип, но за съжаление, не могат лесно да бъдат пренесени към документи в друг формат. При това, обаче, ‘ръчното’ проектиране изисква много време и е предразположено към грешки. Това е причината, съвременните системи за извличане на информация да използват техники за машинно самообучение и автоматично генериране и поддръжка на извличащи шаблони.

При вторият подход − автоматично обучение, не се изискват експертни знания при адаптиране на системата за извличане на информация към нова област − необходимите човешки дейности са сведени до анотиране на колекция от документи в дадена предметна област, предназначени за провеждане на това обучение. ‘Обучаваният’ алгоритъм се изпълнява върху анотираните документи като автоматично генерира и предлага правилата за извличане на информация от документи в съответната предметна област. Следователно подходи, базирани на системи за автоматично обучение върху колекции от документи, постигат по-бързо резултати и имат по-добра преносимост. Не е за пренебрегване и обстоятелството, че след създаване на системата за автоматично обучение, става възможно – с много по-малко средства (необходими само за анотиране на избрани обучаващи редици от документи) да се извършва ‘пренастройване’ към друга предметна област. Недостатък, разбира се, е необходимостта от ‘ръчно’ анотиране на сравнително голям обем от обучаващи данни (в общия случай, обаче, от изпълнителите не се изисква много висока квалификация в предметната област).

Техниките за машинно самообучение се различават една от друга по вида на обучаващите данни [9] - контролирани (двойки ‘обучаващи данни – резултат’) или неконтролирани (само обучаващи данни).

Контролирано обучение се прилага в доста системи за извличане на информация (*AutoSlog*, *CRYSTAL*, *WHISK*, *SoftMealy* и др.), в които за ‘научаване’ на извличащи шаблони, предварително е необходимо ръчно да се анотират съответни примери (напр., документи от изучаваната предметна, всеки придружен с подходящо резюме). Създаването на подобно резюмирано множество от документи изисква време, не е застраховано от грешки, и понякога не е лесно, особено когато текстовете са в специфична област. С цел да се тушира човешкият фактор, в *WHISK* [121] се редува етикетиране на документи от обучаващата колекция и научаване на извличащи правила, вместо всички примери да се анотират още в самото начало. В *Lixto* [13, 40] се предоставя гъвкав начин за анотиране чрез маркиране на фрази в документите с използване на мишката.

За разлика от горните подходи, в случаите на ***неконтролирано обучение***, се използват допълнителни техники, с цел − да се избегне необходимостта от предварително анотиране на документи. Вместо да се анотират, например, всички контролни примери в документите, тук е необходимо ‘ръчно’ (вж. напр. система *KnowItAll* [30]) само да се именуват класовете за търсените полета. Друга система, която не изисква анотиране е *AutoSlog-TS* [110, 111] - модифицирана версия на *AutoSlog*. Тук всяка дума в контролните документи трябва да бъде определена като значима или незначима (от гледна точка на информационните потребности на потребителите) за конкретната предметна област, като значимите думи по-късно се използват за автоматично конструиране на правила, извличащи полезна информация от текстови документи в областта.

Откриването на правила е основна задача при машинно обучение на системи за извличане на информация. Използват се различни подходи за нейното решаване.

Повечето системи използват индуктивни методи за откриване на общи правила, разпознаващи множество положителни примери. Така например, в система *CIRCUS* най-напред се извършва концептуален анализ на изреченията, след което с помощта на евристики се избира подходящ лингвистичен шаблон и се идентифицират участъци-‘котви’ в текста, заедно с правила за активиране. По този начин, всеки концептуален възел се ‘научава’ автоматично по един положителен пример, без да се взема предвид неговата надеждност спрямо цялото множество обучаващи данни. От тук, много от построените концептуални върхове могат да бъдат прекалено общи и да идентифицират незначима

информация. Поради тази причина е необходима следваща обработка за отстраняване на подобни ‘лоши’ върхове. В други системи (напр. *AutoSlog* и *AutoSlog-TS* [109, 110, 111]) липсват автоматични индуктивни стъпки: *AutoSlog* търси за първата клауза, която съдържа маркирания низ, докато *AutoSlog-TS* търси клауза, която включва текст класифициран като значим. В някои случаи се оказва ефективно машинно самообучение, използващо прости евристики (*WIEN* [49, 50]).

Въведените в [17] ***алгоритми за индуциране на правила*** могат да бъдат класифицирани като базирани на принципите на компресиране и на покриване. *Компресиращите алгоритми* започват с множеството от най-специфични правила − по едно за всяка инстанция, и се опитват да компресират това множество чрез образуване на по-общи правила, които изместват по-специфичните, компресията може да бъде прилагана. ***Покриващите алгоритми*** ([38]) се основават на техника, която широко се използва за индукция на правила в системите за извличане на информация. Алгоритъмът ‘открива’ в даден момент правило, което ‘покрива’ подмножество от положителни примери и отхвърля отрицателни примери (по същество реализира стратегията ‘разделяй-и-владей’). Всяко новосъздадено правило води до отделяне на ‘покритите’ от него положителни инстанции от обучаващата колекция данни, и процесът на обучение/откриване продължава върху останалите примери. Процесът на генериране на правила приключва, когато всички положителни инстанции са ‘покрити’ от създадените правила. В общия случай, покриващите алгоритми са по-ефикасни от компресиращите. От своя страна, компресията предотвратява ‘научаване’ на прекалено специфични правила, като позволява обобщение на вече индуцирани правила (т.напр., в система *RAPIER* [16, 17], случайно се избира двойка правила, след което се прави опит за построяване на по-общо правило).

Според ‘посоката’ на търсене се различават възходящи и низходящи алгоритми за конструиране на правила, при които множеството от правила се индуцира, посредством обобщение или специализация. *Възходящи системи* (използващи обобщение) са *CRYSTAL* [122, 120], *RAPIER* [16, 17] и *SoftMealy* [43, 44], докато примери за реализация на *низходящи (специализиращи) алгоритми* са *WHISK* [120], *SRV* [36], *STALKER* [105] и *Lixto* [13, 40].

Преди да се използват създадените правила за извличане или шаблони за идентификация на полезната информация в документи, обаче, е необходимо да се извършат някои

допълнителни обработки (т.нар. предобработка в системите за извличане на информация). Предобработките имат за цел получаване на характеристики (features) на думи и обекти в разглежданите документи. Различават се синтактична и семантична предобработка.

Синтактичната предобработка се използва за отделяне във всеки документ на по-малки елементи с определено смислово значение (обекти като думи, наименования, дати, фрази, и т.н.), и приписване на всеки от тях не само на съответни характеристики (напр. синтактични; наричано в този случай *синтактично етикетиране/маркиране*), но и установяване на зависимости (релации) между тях (напр. на синтактични отношения между думи или фрази в дадено изречение). Методи за синтактично етикетиране се реализират във всеки синтактичен анализатор на текстове (за даден естествен език), като получените синтактични характеристики на елементите на текста са полезни не само при извличане на информация от граматично правилни, но и от граматично неправилни текстове.

Семантичната предобработка обикновено се използва за определяне на семантичния клас, към който принадлежи даден обект - дума или фраза, отделени при синтактичната предобработка. Семантична предобработка е необходима в повечето системи за извличане на информация, които работят със свободно структурирани и полуструктурирани текстове. В този случай, предварително трябва да е създадена съответната онтология (семантична класификация на обектите в предметната област под формата на йерархия от т.нар. семантични класове). За целта е възможно да се прилагат отново двата подхода – експертен, и на базата на самообучение.

Семантичните класове са зависими от предметната област и могат да бъдат определяни на основата на предварително създаден семантичен речник или да се дефинират от потребителя. Например, в актуалната (за съжаление) област на ‘лошите’ терористични новини, биха могли да бъдат дефинирани семантични класове като <Цел>, <Жертва>, <Извършител> и т.н. По този начин, извличащо правило или шаблон, описани в термините на семантични класове, получават универсалност − могат да се прилагат към голяма група от обекти (напр. думи, които споделят едни и същи семантични характеристики), вместо към единични обекти. Един от начините за *активиране на правило за извличане* [109, 111] е откриването в анализирания текст на конкретен обект (напр. опорна дума), наречен ‘тригер’. В допълнение, може да се дефинира и множество от разрешителни (за прилагане на

правилото) условия (напр., ограничения от лингвистичен характер на тригер-думата – в каква точно граматична форма се среща в текста).

Основните методи на системите за извличане на информация са свързани с разпознаване на шаблони, крайни автомати и класификатори. Редица системи за извличане на информация като *AutoSlog*, *CRYSTAL*, *WHISK*, *WIEN*, *RAPIER* и *SRV* използват *разпознаване на шаблони* в текста за извличане на полезната информация. Всички условия, зададени в шаблона (и асоциираните с него правила за извличане), трябва да бъдат изпълнени, за да се говори за успешна идентификация във входния текст. В *SoftMealy* [43, 44] шаблоните в обработваните текстови документи се идентифицират с помощта на *крайни преобразуватели* (вид крайни автомати). Типична *система за класификация* [30], се състои от два основни компонента: екстрактор и оценител. Екстракторът има за задача да идентифицира случаи на попълване на полета, използвайки съответни правила за извличане. Най-напред на търсещата машина се 'подават' ключовите думи от шаблона, след което се задейства попълване на съответните полета, като правилото се прилага върху намерените текстове. Оценителят, от своя страна, се използва като специален тип класификатор, който има за цел да определи дали извлечените стойности за полетата са валидни, базирайки се на статистически изчисления.

1.6 Търсене по документ-образец

Следвайки формулираните по-горе изисквания към системите за текстово търсене, това би означавало осигуряване на подобрени модели на смисловото съдържание на документи, потребителските заявки и информационното търсене, базирани на документи-образци. Анализът на изследвания, свързани с търсене на документи по образец, открива незначителен брой готови и апробирани решения в областта, което е следствие на липса на достатъчно утвърдена теория и сполучливи практики за решаване на задачите за тематичен анализ на неструктурирана естествено-езикова текстова информация. Практическото решаване на задачи за тематичен анализ на документи (дори и в по-тясна предметна област) е актуално не само за реализация на информационно-търсещите системи, но и при проектирането на системи за обработка и анализ на информация. В тази посока могат да се формулират и изследват широк спектър от задачи за интелектуална обработка на

информация, вкл. задачи за извличане, идентификация, разпознаване и използване на смислово съдържание на неструктурирани текстови документи.

Частна задача на информационното търсене е задачата за търсене по документ-образец. Документът-образец е една от формите за представяне на информационните потребности на потребителя. Цел на търсенето е откриване на тематично близки документи. При това, като правило, става дума не за търсене на идентични или синтактически близки документи, а за търсене на документи, близки по съдържание и смисъл [5].

Тематиката отразява съдържанието на документа и включва в себе си множество ключови думи, намиращи се в някаква зависимост една от друга. Един от моделите на такава зависимост са теглови коефициенти, отразяващи значимостта на една или друга дума в конкретната тематика. Основен проблем е създаването на модел за оценяване (изчисляване) на тематичната близост на документи. Като правило, резултатът на търсенето са множество документи, в една или друга степен удовлетворяващи условията на (тематично) търсене. Именно моделът, приет за изчисляване на тематичната близост, в крайна сметка определя релевантността на документите, ‘извлечени’ в резултат на информационното търсене. Изчислявайки стойностите за тематична близост, тези документи може да се аранжират в зависимост от степента на значимост за потребителя.

При създаването на модели(и) и алгоритъм(и) за изчисляване на тематичната близост на документи се предполага провеждане на експерименти в две направления: а) с привличане на лексикални ресурси и средства; б) оценяване на близост/подобие на графи, представящо връзки от тип ‘позоваване’ между документите в колекцията.

В резултат на провеждане на експерименти (с различни модели), би могло да се очаква получаване на резултати, свързани с търсене на документи по образец, а именно:

1. *Модел* (под формата на граф) на структурното представяне на текста на произволен документ от колекцията;

2. *Методи* за честотно-контекстна класификация на ‘значимите’ думи по тематиката на текста (съпроводени с информация за контекстно-свързани с тях думи);

3. *Алгоритми* за изчисляване на различни оценки/мерки на степента на близост (тематична принадлежност) на текста към даден документ-образец .

Най-простият подход към решаване на задачата за търсене на документи по образец е използване на всички думи на документа-образец в качеството на заявка за търсене. Дължината на подобна заявка, обаче, може да се окаже много голяма, което би се отразило отрицателно качеството на търсенето, т.к. резултат на търсенето ще бъдат всички документи, в които присъстват дадените думи, а подобни документи могат да се окажат твърде много. Това ще е негативно както за самата търсеща система – изчислителните ресурси и трафика не са безгранични, и системата може да се окаже претоварени, така и за човек – прегледът и анализът на намерените документи може да отнеме значително време, за което малко потребители са готови [45].

Приемлив вариант в дадения случай е отделяне на тематиката на документа. Под тематика се разбира множество ключови думи, описващи, с някаква степен на адекватност, съдържанието на документа. Тематиката е приближено представяне на документа. За повишаване на точността и адекватността на описанието на съдържанието на документа, ключовите думи се използват с теглови коефициенти, които са измерими с честотата на повторение на тези думи в текста. Въпросите за отделяне на тематиката и изчисляване на тематичната близост на документи по тяхното тематично представяне определят най-вече възможностите и ефективността на търсене по документ-образец.

Ще отбележим също, че към момента не са известни твърде много специализирани изследвания в даденото направление, като отсъства ясно разработена методика за търсене по документ-образец. Съществуващите подходи обикновено са периферни по отношение на разглежданата задача, и не осигуряват в пълна степен нейното решаване. Нещо повече, отсъства формален модел на задачите за подобно търсене.

Ще разгледаме някои разработени, към настоящия момент, методи и подходи, използвани при решаване на задачата за търсене на документи по образец.

В [6] се предлага вариант за търсене на документи по образец, с реализация на следната редица от действия:

1. За всеки документ се определя относително неголямо множество от документи, представляващи негово апроксимиращо тематично обкръжение (околност);

2. Построените тематични околности се анализират с цел формиране на множество ключови думи, характеризиращи тематиката на първоначалния документ относно останалите документи на колекцията;

3. Получените набори от ключови думи се използват за по-следващо изчисление на относителните оценки на тематичното подобие.

Без да влизаме в подробности, ще отбележим един важен момент, определящ спецификата и ограничението на дадения подход. А именно: реализацията на подобен вариант предполага предварителен анализ на колекцията достъпни документи, което ограничава прилагането на подхода в колекции с висока динамика на обновяване и голям брой достъпни документи (напр. като Интернет).

Реализирани са и някои интересни и оригинални методи за търсене на документи по образец в Интернет, които използват информацията за структурата на позоваванията (links) към други документи. В общия случай подобни варианти предполагат анализ на структурата на графи с върхове − документи/страници и ребра – позовавания/връзки).

В случая на Интернет-колекция, ролята на документа-образец се изпълнява от страницата, а позоваванията към/от тази страница се използват от различни алгоритми за локален анализ на структурата на съответния граф от Интернет-страници.

В [48], например, се разглежда подобен алгоритъм – HITS (Hyperlink Induced Topic Search). В рамките на този алгоритъм се определят два класа документи:

- ‘първоизточник’ – документ, на който често се позовават в контекста на някоя тематика (колкото повече се позовават, толкова по-добър е ‘първоизточника’);
- ‘посредник’ – документ, който се позовава на много ‘първоизточници’ (колкото повече позовавания на първоизточници – толкова по-добър е ‘посредникът’).

Алгоритъмът се състои от две стъпки:

1. Избор на подмножество от колекцията (в сл. Интернет) на базата на заявката за търсене;

2. Определяне на най-добрите ‘първоизточници’ и ‘посредници’ по резултатите на анализа на това подмножество.

Подмножеството от т. 1. се построява чрез разширяване на множеството от страници, намерени по заявката на потребителя, за сметка на добавяне на всички страници, свързани с тях чрез път, съставен от едно (понякога две) позовавания. По-нататък, за всеки документ, рекурсивно се изчислява неговата значимост като ‘първоизточник’ и като ‘посредник’.

Същността на алгоритъма HITS се състои в това, че се опитва да отдели ‘съобщество’, съответстващо на тематиката на заявката, и на основата на анализа на това съобщество да се определят най-авторитетните страници [6].

Друг интересен подход към търсенето на документи по образец е използване на документа като допълнение към традиционна заявка]. Често в процеса на търсене възниква необходимост за допълнителна информация, необходима за уточняване на резултатите от търсенето, при което може да се използва информация от изучавания в момента документ или от вече изучени документи. В този случай, заявката за търсене се формира на основата на думи или отделни фрази, взети от тези документи.

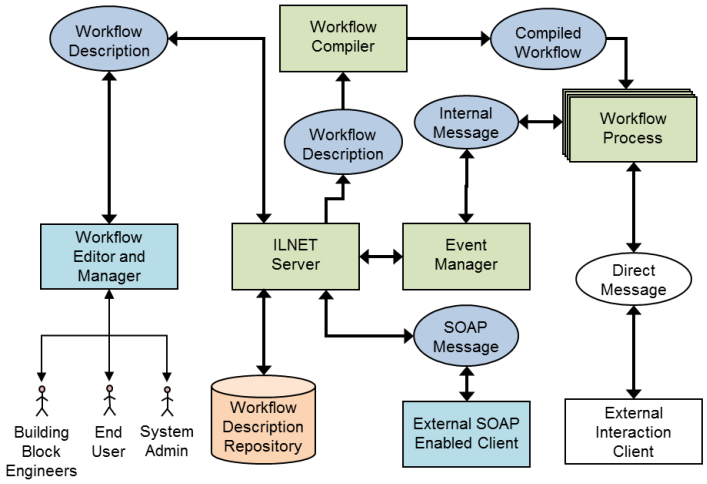
В метатърсещата система *IntelliZap* е реализиран вариант на подобно уточнение, на основата на анализ на контекста, отделен от потребителя текстов фрагмент в изходния документ. Този контекст може след това да се използва както на етапа на формулиране на заявката за търсене, така и на етапа за аранжиране на резултатите, което позволява значително да се повиши точността на търсене за много кратки (1-2 думи) заявки [34].

ГЛАВА 2. МЕТАМОДЕЛ ЗА ОПИСАНИЕ НА ПРОЦЕСИ ILNET

В основата на архитектурата на ILNET седи лека работна рамка за изграждане на модели на процеси и цялостни решения за моделиране на бизнес процеси. Тя е продукт на дългогодишна разработка на серия от инструменти за различни видове лингвистичен анализ. Едно от най-важните й качества е, че въпреки, че е налице фиксиран модел, според който трябва да се описват и изпълняват процесите, съществува вграден механизъм, посредством който може да се адаптират както графичната репрезентация на моделите, така и начина, по който те се изпълняват, за да се дефинират и изпълняват модели на процеси, с които крайният потребител може да е по-запознат. Понастоящем е в тестова фаза моделирането на мрежи на Петри и подмножество от BPML елементите.

Първата версия на системата бе създадена с идеята да се използва основно в областта на компютърната лингвистика [54]. Тя е използвана успешно в системата СЛОГ за преобразуване на текст в говор [58] в етапа за нормализация на входният текст.

В [73] настоящата версия на системата, вече базирана на .NET, с наименование ILNET, се използва за създаването на прототип на архитектура за моделиране на процеси в е-обучението свързани с курсовият работен поток. На фигура **1** е представена архитектурата на системата.



1: Архитектура на софтуерната система ILNET

2.1 Потребителска перспектива

От потребителска перспектива системата *ILNET* се състои от следните компоненти:

- Актьори – те контролират и използват системата:
  - Инженери на изграждащи блокове (ИИБ) – дефинират и имплементират основните изграждащи блокове и работни под-потоци в системата;
  - Крайни потребители (КП) – използват изграждащите блокове, за да дефинират и изпълняват модели на работни потоци в конкретна област;
- Обекти:
  - Хранилище с описания на работни потоци (ХОРП) – централизирано хранилище за работни потоци. Улеснява съвместната работа между различни дизайнери на модели;
  - Описание на работен поток (ОРП) – спецификация за дефиниране на работни потоци, която включва описание на модела на процеса както и на начина на неговото изпълнение;
  - Компилиран работен поток – изпълнима инстанция на работен поток (ИРП);
  - Процес на работен поток – работеща инстанция на работен поток;
  - Вътрешно съобщение – вътрешно представяне на съобщение, което носи информация за конкретно събитие. Използва се за комуникация между работещите ИРП и мениджъра на събития (МС);
  - Външна SOAP услуга – външно за системата приложение или услуга, която може да комуникира с ИРП, посредством SOAP съобщения, които се обработват от МС;
  - Външен интегриран клиент – външен клиент, който може да се свърже и да комуникира директно с ИРП, посредством комуникационни протоколи имплементирани от ИИБ.

Системата е модулна и се състои от пет основни компонента, които комуникират по между си, посредством вътрешни съобщения, или чрез SOAP-базирани уеб услуги. Тези модули са:

- Компилатор на работни процеси, който транслира описанията на процесите в изходен програмен код;
- Сървър за изпълнение и управление на така описаните процеси;
- Графичен редактор за визуално представяне и моделиране на процеси;
- Мениджър на събития, който обработва вътрешната комуникация между процесите и предоставя SOAP интерфейс, посредством който външни приложения могат да обменят съобщения с работещи инстанции на процеси;
- Библиотека от изграждащи блокове, която съдържа дефиниции на помощни под-процеси, които могат да бъдат използвани за по-бързото моделиране процеси. В допълнение, библиотеката предоставя и блокове, които служат като емулатори на структурни елементи в други съществуващи инструменти за моделиране.

2.2 Съставни елементи на модела

Първичният модел, който се разпознава от компилатора, представлява насочен граф, в който всеки връх отговаря на действие а всяка дъга – възможен преход. При успешно изпълнение на действието в даден връх всички изходящи дъги се проследяват паралелно и работният поток автоматично се разделя на толкова под-потока, колкото е броят на тези дъги.

Едно действие може да бъде описано по два начина – чрез програмен език, или чрез изпълняване на друг работен поток. Във вторият случай от основният поток се изпращят входни параметри на под-потока, след което от последният може да върне резултат под формата на изходни параметри. Всеки път, когато се извиква един под-процес от друг работен процес се създава нова инстанция на под-процеса, която се изпълнява.

2.2.1 Постоянна и временна памет

Друга интересна характеристика на предложената система е нейният модел на паметта. Езикът за описание на процеси в ILNET поддържа работа с променливи на три различни вида обхват:

- Локални променливи: те винаги се копират и се достъпват от само една нишка. Могат да описват паметта на една работна нишка, или маркер в мрежите на Петри. Използват се съхраняване на данни и изчисления, които са независими от работата на други едновременно изпълнявани инстанции или нишки на работни процеси. Използването им елиминира риска от това две инстанции или нишки на един процес да се опитат да работят с едни и същи данни в един и същи момент.
- Междинни променливи: те се копират, когато се инициализира процеса (при създаване на нова работеща инстанция). Общи са за всички контролни работни процеси, които произлизат от изначалната инстанция на работният процес. Използват се за синхронизация на активните пътища в рамките на едно стартиране на процеса, както и за имплементация на по-голямата част от синхронизиращите шаблони използвани при моделирането на бизнес процеси.
- Глобални променливи: те никога не се копират и стойността им се споделя във всички инстанции. Тези променливи предоставят механизъм за създаване на временни хранилища, в които различните инстанции на един процес могат да събират и споделят данни. За кратко-временно съхранение на данни, които не са от критичен характер този вид променливи са по-бърз и по-лесен за използване начин отколкото използването на СУБД или други външни хранилища.

Чрез комбиниране на трите вида обхвата модела предоставя лесен начин за реализация на различни шаблони като И, ИЛИ и др. Като се дава възможност за деклариране на структури от данни налични в използваният на ниско ниво език за програмиране, променливите могат да бъдат толкова сложни или прости колкото е необходимо (например: числа, низове, списъци, таблици, логически файлове, връзки към СУБД, и т.н.).

2.2.2 Сървър за изпълнение

Зареждането и изпълняването на моделираните процеси и трансфера на вътрешни и външни съобщения се реализира от ILNET сървъра. Сървъра обработва събитията за работа със съобщения и служи за осъществяването на връзка между работещите инстанции на процеси и външната среда. Той дава възможност за:

- зареждане на процеси;
- стартиране на процеси;

- получаване и изпращане на съобщения;
- терминиране на инстанции на процеси.

Предоставя се хранилище за съхранение и изпълняване на работни потоци. Използвайки една точка за вход в системата се постига както лесна администрация на всички работещи потоци, така и възможност за споделяне на модели на работни потоци между потребителите на системата. Изпълняването на потоци от отдалечен сървър или клъстер от сървъри позволява безпроблемната кооперация между различни потоци.

Мениджъра на събития е съставна част от модела на системата и може да бъде използван, посредством следната група от вградени функции:

- *raiseEvent (Съобщение)* – генерира събитие носещо съобщение С и прекратява изпълнението на процеса докато в МС не постъпи съобщение-отговор;
- *queryEvents (Запитване)* – получава от МС списък с идентификатори на активни събития чиито съобщения пасват на подаденото запитване;
- *fetchEvent (Идентификатор)* – получава от МС цялото съобщение зад събитието със съответният идентификатор;
- *answerEvent (Идентификатор, Отговор)* – изпраща към МС съобщение-отговор на активно събитие със съответният идентификатор. Повече от един отговора могат да бъдат изпратени докато събитието е активно, като за всеки отговор се изпълняват паралелно алтернативни потоци в отделни нишки;
- *closeEvent (Идентификатор)* – деактивира събитие, премахвайки го от списъка с активни събития и освобождавайки заетите от неговото съобщение ресурси;
- *listenEvent (Филтър)* – вместо постоянно да се изпраща запитване към МС за активни събития от даден вид, даден връх от поток или процес може да се закачи към МС със свой филтър на събитията. Всеки път, когато в МС постъпи събитие със съобщение, което пасва на подаденият филтър ще се активира върха от процеса, който е прикачен;
- *closeListener (ListenerId)* – премахва прикачен наблюдател и съответният му филтър от МС.

2.2.3 Графичен редактор

Графичният редактор на модели предоставя лесен и безпроблемен начин за моделиране на процеси в рамките на ILNET работната рамка. Фигура **4** показва работният екран на графичният потребителски интерфейс на модула за визуално редактиране на процеси GEFI. Средата позволява няколко процеса да бъдат редактирани едновременно. При запис, моделите се синхронизират със сървъра, в който са заредени, където се компилират.

Средата поддържа използването на различни стандарти за моделиране, посредством дефиницията на съответните изграждащи блокове. Визуалната информация описваща един модел се прикача към базовите елементи на процеса под формата на метаданни.

Един иноваторски подход използван в графичната среда е използването на хибриден клиент-сървър метод за визуализация. Визуалното изчертаване на основната част от модела, една част от блоковете и връзките между тях се извършва от клиента, докато конкретни блокове могат да бъдат изчертани на сървъра или да се използват данни взети в реално време от сървъра по време на изчертаването. Докато повечето системи предоставят или изцяло клиентско или изцяло сървърно изчертаване (за уеб клиенти) GEFI използва клиентско изчертаване за модела, и в същото време позволява сървърно изчертаване на специфични елементи (изграждащи блокове) от него. Тъй като изграждащите блокове се създават и редактират със същата визуална среда, това означава че средата може да се използва за редактиране начина, по който блоковете се изчертават в редактора. В допълнение, изчертаването на сървърните блокове се обработва от модела на самите блокове, който е зареден и се изпълнява на сървъра. Това прави възможно да се моделират интерактивни много-потребителски елементи в рамките на графичният редактор или да се показват статистики от сървъра в реално време.

Друго разширение на модела е абстракцията на изпълнителният слой. По време на изпълнението на работният поток, след всяко успешно изпълнение на действие, описано във връх, всеки от възможните следващи действия се активира от потребителски-дефиниран шаблон за изпълнение. Чрез този вид шаблони може да се имплементират различни механизми за наблюдение и промяна начина на изпълнение на работният поток според вътрешни или външни (потребителски вход) решения. Всеки работен поток може да има зададен специфичен шаблон за изпълнение.

2.3 Съвместимост на ILNET модела

Процесите в ILNET се разглеждат като комбинация от параметри, работни данни и действия. Те могат да се определят формално с наредената n-торка

$ILNET = (N, F, V, C, V_0, n_{IN}, n_{OUT}, V_{IN}, V_{OUT})$, където:

- N е крайно множество от върхове, отговарящи на дейности или под-процеси (извикването на под-процес е специален вид дейност)
- $F \subseteq N \times N$ е множество от дъги, което определя последователността от изпълнение на дейностите,
  - $n^{\bullet}(x) = \{y | (x, y) \in F\}$ е множество на изходните върхове на връх n
- $V = V_g \cup V_i \cup V_l$ е множество от променливи, като:
  - V_g е крайно множество от глобални променливи,
  - V_i е крайно множество от междинни променливи,
  - V_l е крайно множество от локални променливи,
- $C: N \to T$ е функция на трансформациите на дейностите. Трансформациите са процедури описани със средствата на конкретен език за програмиране, които променят състоянието на системата
- $V_0: V \to D$ е изображение на променливите върху областта на техните типове[1], което определя началните им стойности.
- $n_{IN} \in N$ е началната дейност, която се изпълнява,
- $n_{OUT} \in N \cup \emptyset$ е крайната дейност,
- $V_{IN} \subseteq V$ е множество от променливи, определящо входните параметри на процеса
- $V_{OUT} \subseteq V$ е множество от променливи, определящо изходните параметри на процеса

Един начин за формално определение на функциите описващи трансформациите е:

[1] С цел опростяване на дефиницията е избегнато изричното дефиниране на типове и техните области

$T: S \rightarrow (R \times S)$, където:

- $R = \{r_{success}, r_{fail}\}$ е множество на възможните резултати от изпълнение на трансформацията, според които се определя дали да се продължи към изпълнение на следващите дейности, при резултат $r_{success}$, или не, в противен случай,
- $S: V_M \rightarrow D$ е състояние на системата, съпоставящо стойност от областта D на всяка една от променливите в общото множество на променливите V_M,
- $V_M = V_g \cup (V_i \times P) \cup (V_l \times P \times B)$ е общо множество на променливите в системата по време на нейното изпълнение, където P е множеството на всички инстанции на процеса, а B е множеството на всички възможни нишки[1] на процесите. Тъй като всяко разклонение на пътя на модела води до създаването на нови негови нишки, то $B \subseteq N^*$, където N^* са всички пътища в насоченият граф $G = (N, F)$

Семантиката на изпълнението на модела е дефинирана както следва:

2.3.1 Мрежи на Петри

Мрежите на Петри могат да бъдат изразени използвайки елементите на предложеният метамодел. Ще използваме формалната дефиниция за маркирани мрежи на Петри:

Мрежа на Петри (маркирана) е наредената четворка (S, T, W, M_0), при която:

- $S = \{s_1, s_2, \dots, s_n\}$ е крайно множество от позиции,
- $T = \{t_1, t_2, \dots, t_n\}$ е крайно множество от преходи,
- $S \cap T = \emptyset$
- $W : (S \times T \cup T \times S) \rightarrow \mathbb{N}$ е мултимножество от дъги, (съпоставя цяло неотрицателно число на всяка дъга)

[1] Отделните нишки на инстанциите на процесите могат да бъдат изпълнявани и последователно в рамките на една реална нишка в ОС, или паралелно в отделни реални нишки на ОС. При последователно изпълнение се използва по една опашка на чакащите за изпълнение нишки за всяка отделна инстанция на процеса

- $M_0 : S \to \mathbb{N}$ е началната маркировка на Петри, където под маркировка $M : S \to \mathbb{N}$ се разбира съпоставяне на определен брой ядра (tokens) на всяка позиция от S

Релацията на потока може да се опише и като двойка матрици с размери $|S|$ на $|T|$:

- W^-, определена с $\forall s,t: W^-[s,t] = W^-(s,t)$,
- W^+, определена с $\forall s,t: W^+[s,t] = W^-(t,\mathrm{s})$,

Един преход t е в готовност в една маркировка M, тогава и само тогава, когато

$$\forall s \in S: M(s) \geq W(s,t)$$

Ще се опитаме да създадем общ ILNET метамодел, за описание на мрежи на Петри. Един начин за описание на тази мрежа с ILNET метамодел, е като създадем моделът $ILNET = (N, F, V, C, V_0, n_{IN}, n_{OUT}, V_{IN}, V_{OUT})$ със следните елементи:

- $N = T \cup \{n_{init}\}$
- $F = \{(x,y)|x = y; x \neq n_{init}\} \cup \{(x,y)|x \neq y; x = n_{init}\}$, за $x, y \in N$
- $V = V_g \cup V_i \cup V_l$
- $V_g = \{v_{in}, v_{out}, v_{w^-}, v_{w^+}\}$
- $V_i = S_v \cup T_v$
- $V_l = \emptyset$
- $V_0 = \{S_v \mapsto M_0\} \cup \left\{T_v \mapsto \left(t_{v_1}, t_{v_2}, \dots t_{v_k}\right) \middle| k = |T|; t_{v_i} = true\right\}$
  $\cup \left\{v_{in} \mapsto \{t \mapsto s|W(s,t) > 0; t \in T\}\right\}$
  $\cup \left\{v_{out} \mapsto \{t \mapsto s|W(t,s) > 0; t \in T\}\right\}$
  $\cup \{v_{w^-} \mapsto W^-\} \cup \{v_{w^+} \mapsto W^+\}$.
- $n_{IN} = n_{init}$
- $n_{OUT} = \emptyset$
- $V_{IN} = \emptyset$
- $V_{OUT} = \emptyset$
- $C(n_{init}) = \{V_M \mapsto (r_{success}, V_M)\}$

Тъй като всички променливи в модела принадлежат на V_i, то $V_M = V_i \times P$, инстанциите на процесите се изпълняват напълно независимо, а нишките на една инстанция могат да се

разглеждат като система от напълно симетрични паралелно протичащи под-процеси, които работят само върху глобални (за тях, междинни в системата на ILNET модела) променливи. Можем да опишем дейностите $C(n)$ за $n \in T$ чрез псевдокодът:

if $T_v[n] = true$ and foreach s in $v_{in}(n) : S_V[s] \geq v_w[s, n]$ then

foreach t_{v_i} in T_v set $t_{v_i} = true$

foreach s in $v_{in}(n)$ set $S_V[s] = S_V[s] - v_w^-[s, n]$

foreach s in $v_{out}(n)$ set $S_V[s] = S_V[s] + v_{w^+}[s, n]$

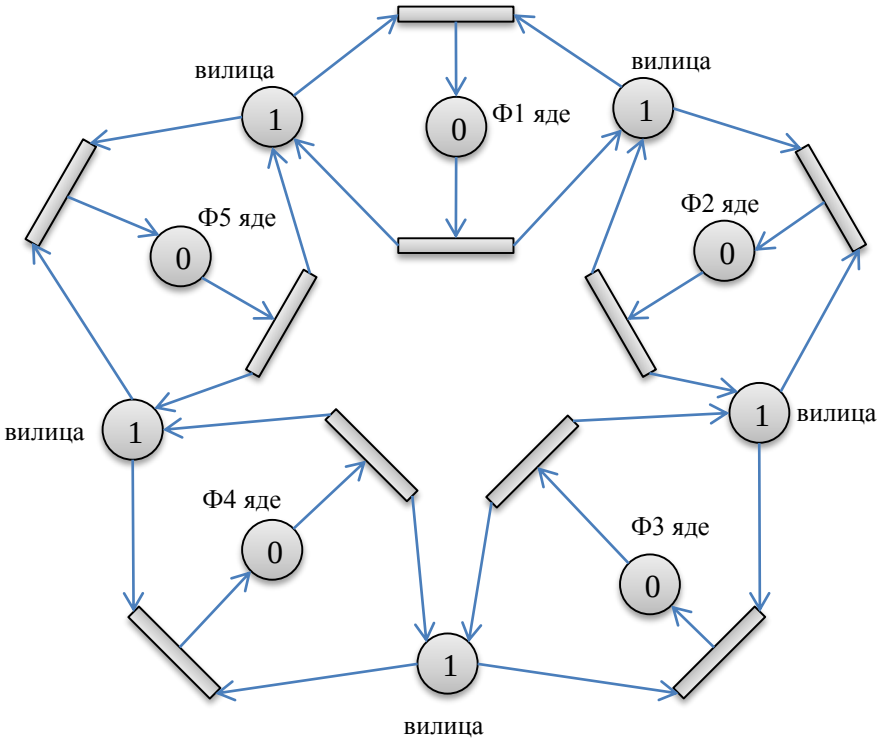
else

$T_v[n] = false$

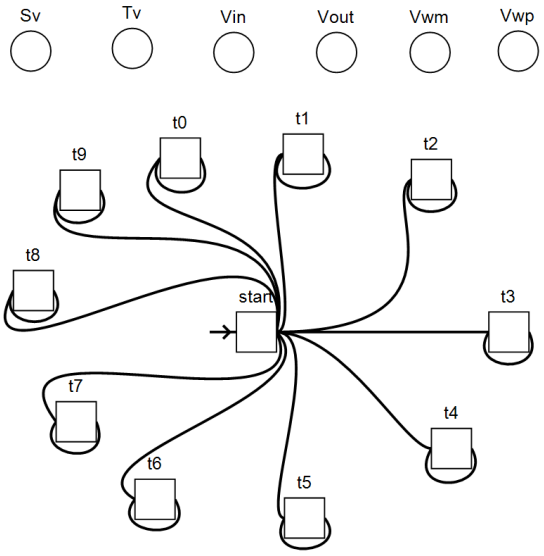
if foreach t_{v_i} in $T_v : \ t_{v_i} = false$ then return false;

return true;

Пример: Нека разгледаме задачата за обядващите философи. Пет философа седят около кръгла маса, в средата, на която има купа със спагети. Пред всеки философ има чиния, а между всеки две чинии има по една вилица. За да може да яде спагети, всеки философ се нуждае от две вилици, като има право да използва само тези, които са от двете страни на чинията му. Философите прекарват времето си като размишляват, в резултат на което огладняват и се опитват да вземат двете вилици около своята чиния. Когато конкретен философ успее да вземе две вилици, той се храни за известно време, след което връща вилиците. Тази задача може да бъде симулирана със мрежата на Петри показана на фигура **2.** На фигура **3** е показан модел на същата задача, построен следвайки горните определения.



Фигура 2: Мрежа на Петри за симулиране на задачата за обядващите философи



Фигура 3: Един начин за представяне на задачата за обядващите философи в ILNET процес

2.4 Употреба и приложение на модела

Следните примери показват някои различни употреби и приложения на ILNET модела. Първо ще представим селекция от изграждащи блокове и как те могат да се конструират, последвани от едно реално приложение в система за моделиране и изпълнение на процеси в е-обучението чрез работни потоци.

2.4.1 Имплементиране на шаблони за работни потоци

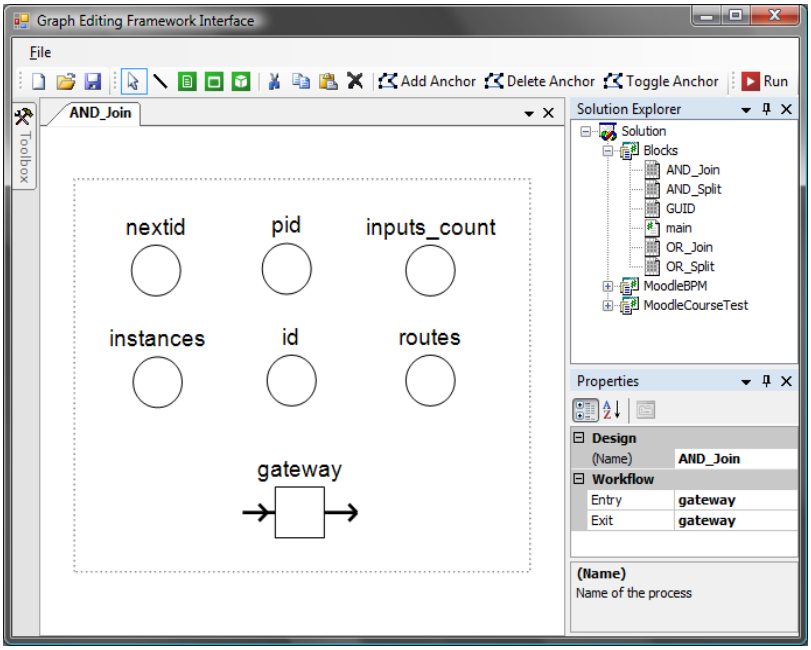
Използвайки функционалността на *ILNET* модела за работни потоци, можем да създаваме блокове, които имплементират различни шаблони. По време на изпълнението под-потоците имат достъп до извикващият ги поток, ако има такъв, както и до контекста, в който те са извикани. Използвайки тази информация могат да се моделират сложни контекстно-зависими блокове, които да изпълняват действия на базата на решения върху акумулирани серии от събития. Един такъв пример е шаблона за синхронизирано сливане (AND-Join, паралелно събиране) (Фиг. 2) имплементиран с използването на един единствен връх с действие описано в кратка C# функция и няколко глобални променливи за акумулиране на данни.

Всеки път, когато се извика върха gateway, от даден поток, за първи път, под-потока посредством рефлексия върху модела на извикващият поток намира броят на всички входящи дъги към върха, който извиква И-блока. От този момент, всеки следващ път, който навлезе в изпълнение на същият И-блок, gateway върха проверява коя входяща дъга в извикващият поток е била проследена и акумулира входа. В момента, в който е налице сигнал от всяка една от възможните входящи дъги, И-блока вместо само да акумулира сигнала, го пуска (генерира сигнал за успешно изпълнение на върха) и се изтриват акумулираните данни, позволявайки на следващи разклонения (цикли) на над-потока да използват отново същият И-блок (при положение че над-потока е правилно моделиран).

Понастоящем шаблона за изпълнение на действията, който се използва по подразбиране реализира паралелно разделяне (AND-Split) всеки път, когато едно действие има повече от една изходящи дъги. Фигура 5 показва как AND-Join блока може да бъде използван в работен поток. Действие D ще се изпълни едва след като и двете действия A и B се изпълнят. Няма

нужда от допълнителна конфигурация на блока (освен ако не искаме по-голяма сложност или функционалност).

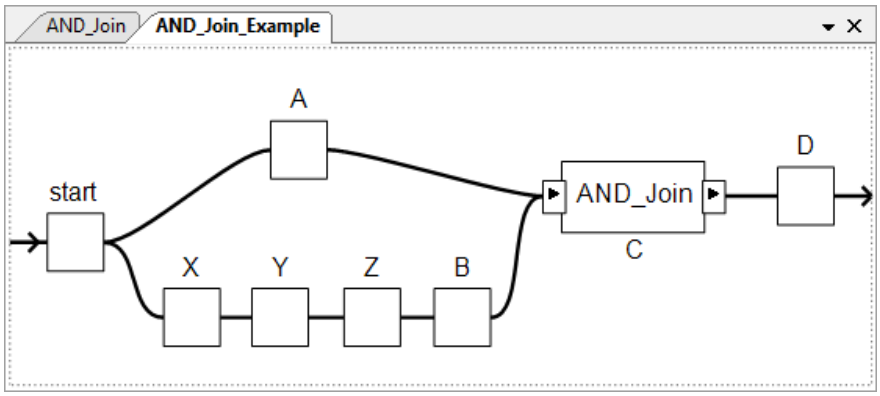
Следвайки тази процедура, може да се реализират редица от разнотипни изграждащи елементи за достъп до данни, обработка на съобщения, използването на уеб-услуги.



Фигура 4: Примерна реализация на AND-Join блок

2.4.2 Блокове за работа с бази от данни

Друго примерно приложение е имплементацията на блокове за осъществяване на достъп и работа с СУБД. Чрез позволяване на зареждането на външни библиотеки могат да се конструират блокове, които използват външен сървър или слой за достъп до данните. На фигура **6** е показан примерен блок за съхранение на данни в MySql база от данни. Функцията на *storeResult* върха имплементира връзката с базата от данни и логиката за съхранение. В последствие блокът може да бъде извикван като обикновен под-поток.



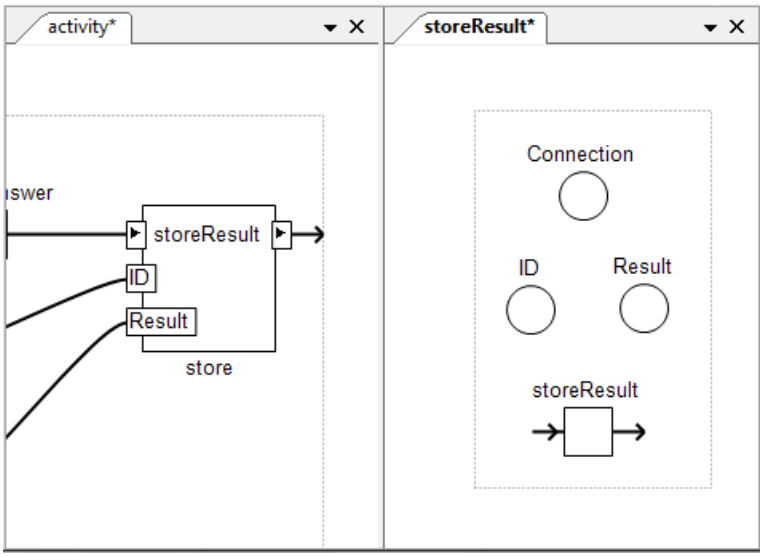
Фигура 5: Примерно използване на И блока

2.4.3 Приложения в практиката

Предишната версия на системата [71] е използвана успешно като инструмент за описание на процеси при нормализацията на текст в системата за преобразуване на текст в говор SLOG [72].

В [73] е използвана ILNET системата за моделиране и управление потоците на курсовете в система за е-обучение.

Системата *ILNET* е интегрирана и в приложения, целящи разширяването на популярни системи за е-обучение с възможности за описание на работни потоци [74]. Решението използва уеб услугата на *ILNET*, за да свърже съществуваща система със сървъра за изпълнение на работни потоци.



Фигура 6: Примерен параметризиран блок за достъп до база от данни

2.5 Изводи

Метамоделът *ILNET* предоставя метод за бързо имплементиране на прототипи на различни видове модели на работни потоци и шаблони. Интегрирайки обектно-ориентиран програмен език в първичният модел, и давайки възможност на потребителските функции да получават информация за структурата и данните на изпълняваният модел, могат да се реализират компактни и интелигентни изграждащи блокове.

Вграденият МС, комбиниран с приложен програмен интерфейс за използване като уеб услуга, позволява интеграцията на модели на работни потоци в почти всяка една съществуваща клиентска система.

Сред възможните бъдещи развития на разработката са:

- Имплементиране на изграждащи блокове и шаблони за изпълнение, които да позволят моделирането на известни стандарти за описание на процеси/потоци;
- ILNET имплементация на оцветени мрежи на Петри;
- Анализ на процесите с цел подобряване на производителността [57];
- Интеграцията на ILNET системата в други практически приложения (персонализирано и адаптивно обучение [75, 76], е-бизнес [60], административен контрол [61], и др.).

2.5.1 Решени задачи

Основните резултати получени във втора глава могат да се формулират по следния начин:

- Анализиране на предложения метамодел за моделиране и управление на процеси и реализиране на прототип на базата на този модел, който да включва.
- Дефиниране на изискванията на метамодела. Предложени са функционалните изисквания към реализацията на метамодела, както и нефункционалните изисквания.
- Проектиране на софтуерна архитектура и разработка на програмен прототип, базиран на тази архитектура с цел изясняване на изискванията към системата.

ГЛАВА 3. АРХИТЕКТУРА И РЕАЛИЗАЦИЯ НА СУРП

Целта на настоящата глава е да опише архитектурата на софтуерна платформа за моделиране и управление на процеси, следвайки метамодела от глава 2, и да покаже как различни информационни процеси могат да бъдат моделирани, реализирани и интегрирани в други решения.

Самата архитектура ще послужи за изграждане на два софтуерни прототипа. Единият софтуерен прототип, от тип *редактор*, ще позволява създаване на екземпляри на метамодела, а вторият софтуерен прототип – „*среда за изпълнение*", ще позволява проиграване на екземпляри на метамодела, изградени с първия прототип.

3.1 Анализ на изискванията към реализацията на модела

С оглед разработване на софтуерна архитектурна рамка на описания метамодел за моделиране и управление на процеси, бяха съставени функционални и нефункционални изисквания към софтуерните системи, поддържащи този модел.

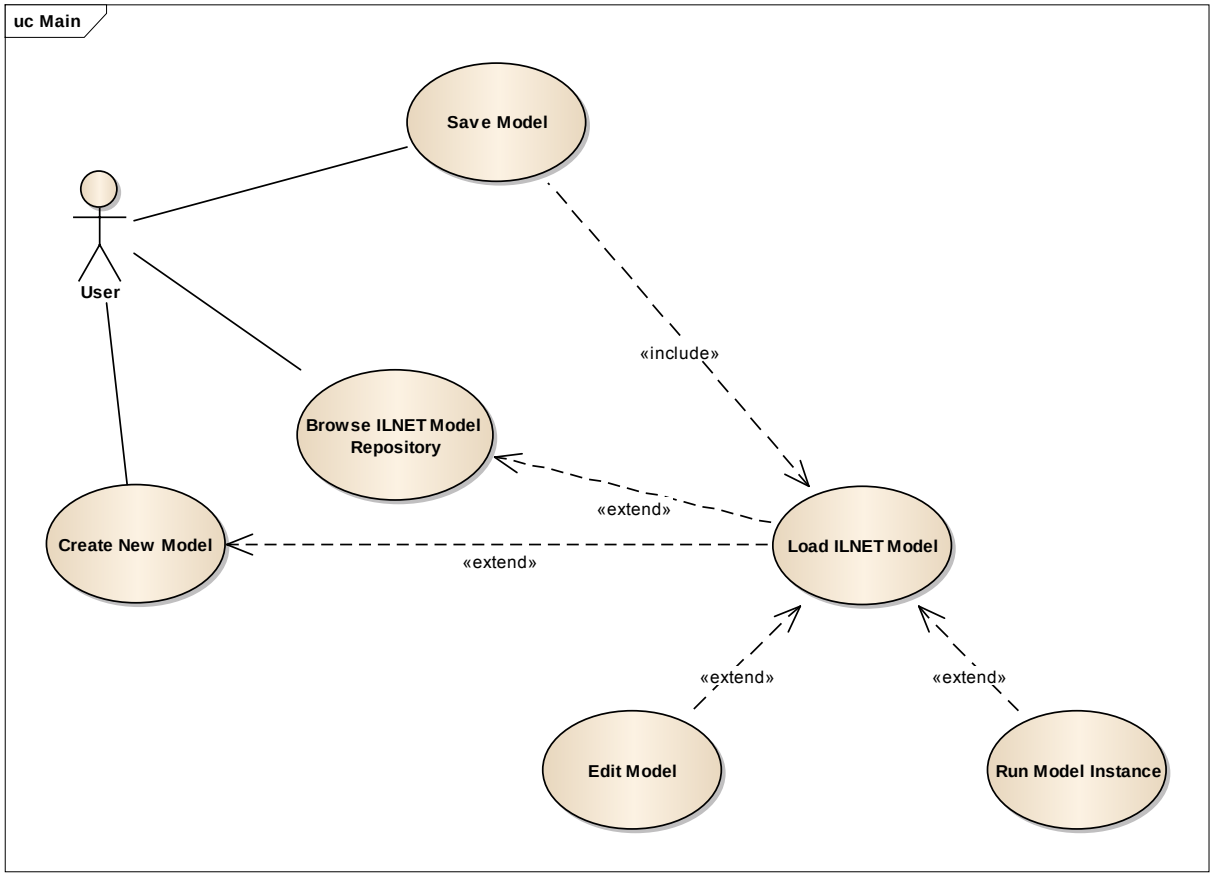
3.1.1 Функционални изисквания

Функционалните изисквания се поставят на база функционалността на отделните компоненти, изграждащи цялостното решение.

Могат да се поставят следните функционални изисквания към реализацията на метамодела:

3.1.1.1 Създаване и редактиране модели на процеси

Потребителският случай „Създаване и редактиране модели на процеси" е показан на **7** и е описан по-долу.



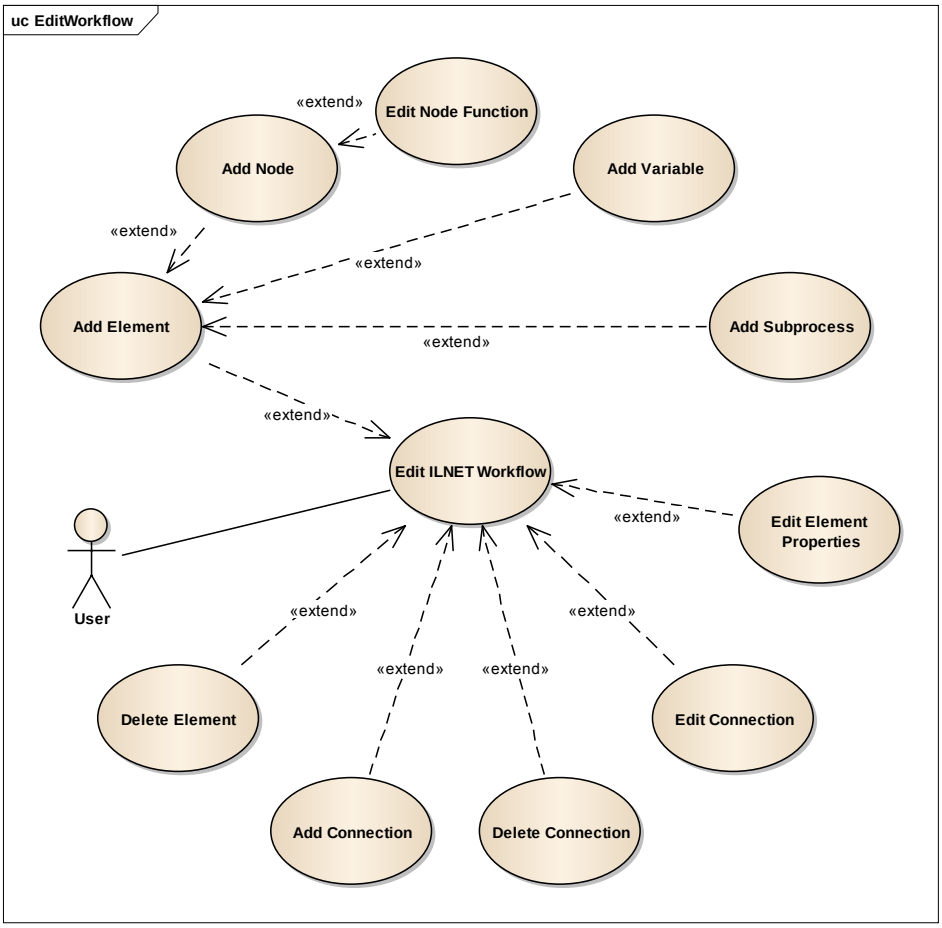
Фигура 7: Създаване и редактиране модели на процеси

Описание	Този случай на употреба описва процесите на създаване и променяне на модели
Участници	Потребител
Пред-условия	Трябва да е осъществена връзка с ILNET сървър
Пост-условия	Няма
Нормална последователност на събитията	1. Потребителят преглежда хранилището на ILNET метамодели. 2. Потребителят създава нов модел в избраната от него директория на хранилището. 3. Потребителят зарежда създадения модел в графичният потребителски интерфейс.

	4. Потребителят редактира модела дефинирайки съставните му работни потоци (процеси) и описвайки техните елементи. 5. Потребителят запазва модела, изпращайки го за съхраняване в хранилището на метамодели. 6. Потребителят стартира работна инстанция на модела.

3.1.1.2 Редактиране на работен поток

Потребителският случай „Редактиране на работен поток" е показан на **8** и е описан по-долу.



Фигура 8: Редактиране на работен поток

Описание	Този случай на употреба редактирането на работен поток в рамките на конкретен ILNET метамодел
Участници	Потребител
Пред-условия	Трябва да е осъществена връзка с ILNET сървър
Пост-условия	Няма
Нормална последователност на събитията	1. Потребителят отваря прозореца за редактиране на метамодела. 2. Потребителят избира дали да добави или да изтрие елемент. 3. При избор за добавяне на нов елемент потребителят избира между: • Добавяне на възел • Добавяне на променлива • Добавяне на подпроцес 4. При избиране на съществуващ възел потребителят има възможност за промяна на неговата функционалност. 5. Потребителят може да добави, изтрие или промени връзка между възли и подпроцеси. 6. Потребителят може да променя свойствата на елементите. 6.1. При промяна на свойствата на възел потребителят има избор между : • Промяна на име • Промяна на кода • Избор дали елемента е начало на работния поток

	• Избор дали елемента е край на работния поток 6.2 При промяна на свойствата на променлива потребителят има избор между: • Избор дали променливата е входен параметър • Избор дали променливата е изходен параметър • Промяна на тип • Промяна на стойност • Промяна на име • Промяна на област на действие : o Локална o Междинна o Глобална 6.3 При промяна на свойствата на подпроцес потребителят има избор между: • Промяна на име • Промяна на името на подпроцеса • Избор дали елемента е начало на работния поток • Избор дали елемента е край на работния поток 7. Потребителят запазва модела, изпращайки го за съхраняване в хранилището на метамодели. 8. Потребителят стартира работна инстанция на модела

3.1.1.3 Изпълняване и управление на моделирани процеси

3.1.2 Нефункционални изисквания

Следните нефункционални изисквания се дефинират към реализацията на метамодела:

1. Разширяемост – да е лесен за адаптиране и да позволява бъдещо разширяване;
2. Скалируемост – възможност за едновременно ефективно обслужване на голям брой конкурентни потребители;
3. Преносимост – трябва да бъде лесно преносим на различни софтуерни платформи;
4. Надеждност – трябва да предоставя надежден достъп до съдържанието;
5. Лесна поддръжка и модификация на компонента: чрез разделяне на бизнес логиката от съдържанието и, от друга страна, на разделяне на съдържанието от представянето му;
6. Минимизиране на цената на разработка чрез използване на безплатни и базирани на отворен код софтуерни технологии и среди за разработка.

3.2 Проектиране на прототип на система за управление на работни процеси

3.2.1 Проектиране на отделните компоненти на прототипа

В тази точка са изброени накратко основните функции на системата (бизнес логиката на приложението), и графичният потребителски интерфейс, който ги реализира. Изброени са отделните визуални компоненти, както и функциите, които ще изпълняват, в това число и действията, които ще се извършват при настъпването на определено събитие.

Проектирането на графичен интерфейс се състои, от една страна в дефинирането на визуалната част, чрез която клиентите ще използват системата (прозорци, изгледи, перспективи, таблици, текстови полета и т.н.), а от друга в дефинирането на конкретните действия, които ще извършва системата, когато клиентът предизвика определени събития, посредством графичните елементи за управление на действията (бутони, връзки и т.н.).

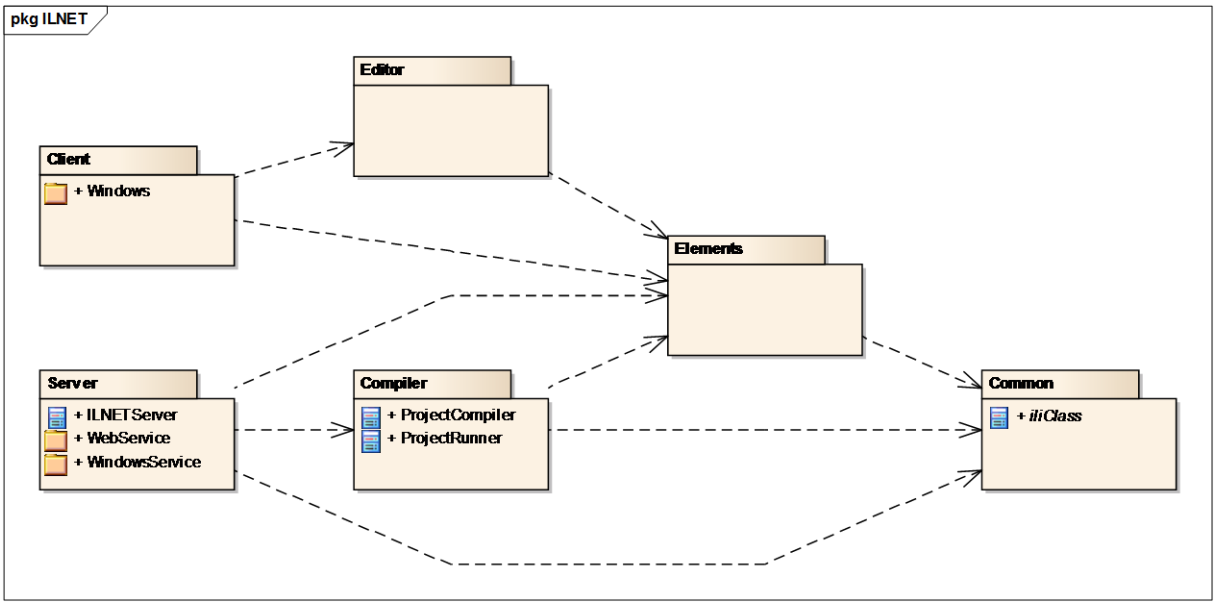
3.3 Описание на реализацията

3.3.1 Пакетна структура на приложението

Приложението е реализирано под формата на няколко пакета. Описанието на реализацията се прави общо за двата софтуерни прототипа.

Във всеки от тях се съдържат класове със строго определена функционална насоченост. На фигура 9 е представена пакетната структура на по-главните пакети. Функционалността реализирана от отделните пакети е следната:

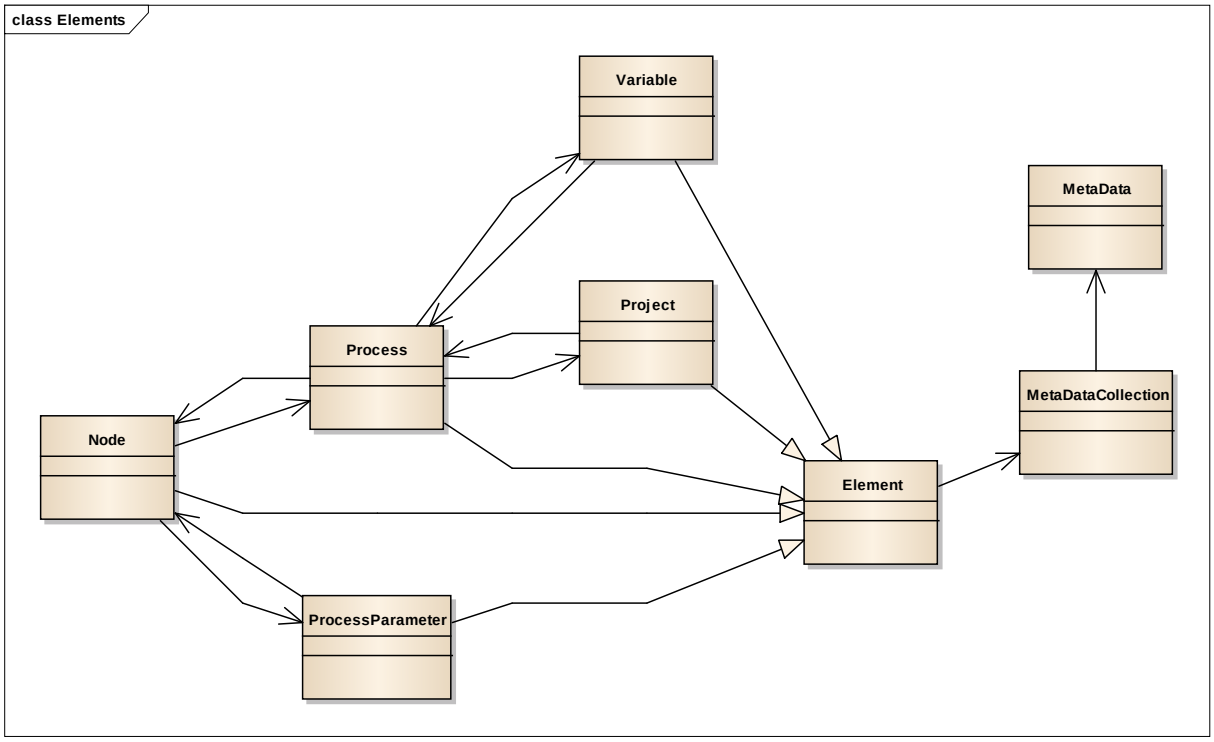
- Пакетът **Elements** съдържа реализацията на ILNET модела под формата на C# класове.
- Пакетът **Common** съдържа класове и дефиниции предоставящи обща помощна функционалност.
- Пакетът **Compiler** съдържа класове, които предоставят методи за компилиране и изпълнение на работни потоци описани чрез ILNET метамодела.
- Пакетът **Editor** съдържа класове, които предоставят методи за визуализация и редактиране на визуалните характеристики на ILNET метамодели.
- Пакетът **Server** съдържа приложения и библиотеки от класове, които реализират сървърната част на архитектурата на приложението.
- Пакетът **Client** съдържа класове, които реализират клиентската функционалност от архитектурата на приложението, в това число и графичният потребителски интерфейс за редактиране на ILNET модели.



Фигура 9: Пакетна структура на приложението

3.3.2 Пакет ILNET.Elements

В този пакет се съдържа реализацията на ILNET модела под формата на C# класове. Имената на класовете отговарят на имената дефинирани в XML схема файлът на модела (приложение 1). Тези класове са с проста структура и служат единствено за предоставяне на лесни методи за създаване и редактиране елементи на модела, сериализация на проектите от и към XML, както и за трансформиране на проектите до изпълним изходен код. На фигура **10** е представена клас диаграма на основните класове на модела. Диаграмата е в съкратен вид (не показва променливите и методите на класовете).

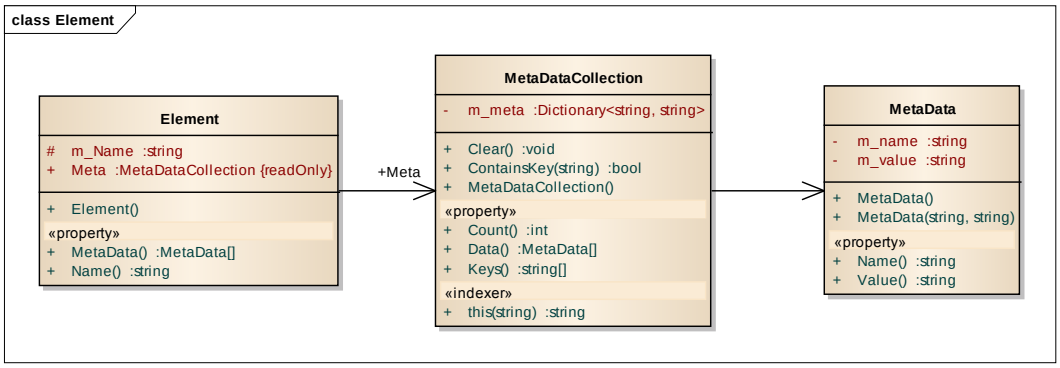


Фигура 10: Клас диаграма на класовете от пакета ILNET.Elements

Класа *Element* съдържа методи за задаване и извличане на метаданните на елементи от метамодела. Всички класове, които описват елемент от метамодела са наследници на този клас. На фигура **11** са показани по-подробно класът *Element* и класовете за представяне на метаданни *MetaDataCollection* и *MetaData*.

Главен клас на пакета е *Project*, в който се съхранява цялото описание на един ILNET метамодел. От него директно или косвено могат да се зададат и извлекат всички елементи

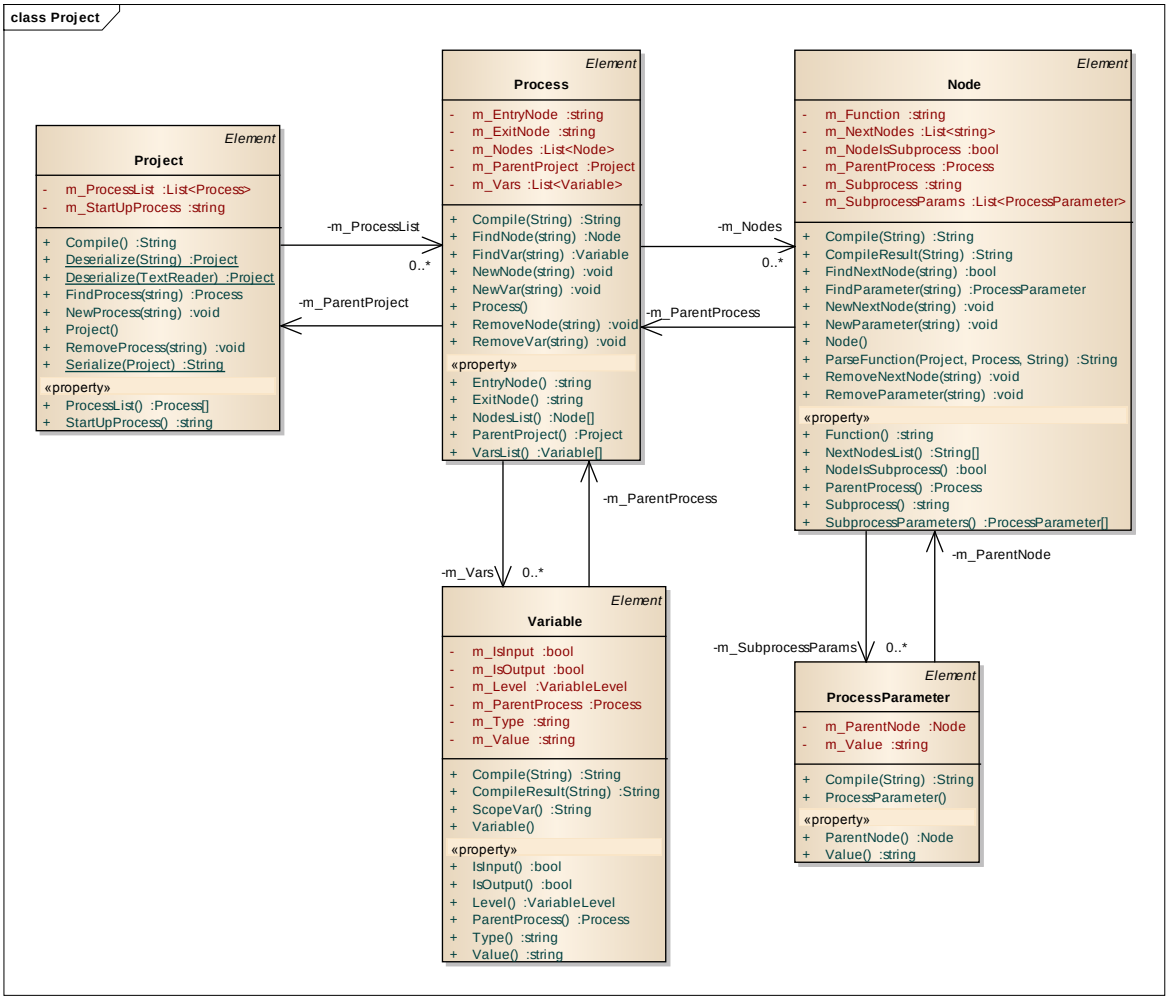
на метамодела. Той съдържа и методите за сериализация и десериализация на метамоделите. На фигура **12** е визуализирана клас диаграма на основният клас и взаимодействията му другите класове наследници на класа *Element*. Всички наследници съдържат метод за компилиране (трансформиране) на описанието на модела до текст, посредством подаден шаблон. Тези методи се използват за генерирането на изпълним изходен код от метамодела.



Фигура 11: Клас диаграма на класа Element и класовете за работа с метаданни на елементите

Ролите на другите класове са следните:

- Класът **Process** представя един работен поток от метамодела и предоставя методи за редактиране на действията и променливите свързани с даден поток.
- Класът **Variable** представя една променлива в работен поток и предоставя методи за редактиране на името, обхвата, типа и стойността по подразбиране на променливата, както и дали е входяща (параметър на потока) и/или изходяща (резултат от изпълнението на работният поток).
- Класът **Node** представя едно действие (функция или под-поток) в работен поток и предоставя методи за редактиране на функцията или името и параметрите на под-потока.
- Класът **ProcessParameter** представя стойността на един параметър при извикване на под-процес от действие (Node) на текущият процес и предоставя методи за редактиране на името (посредством Element.Name) и стойността на параметъра.



Фигура 12: Клас диаграма на класа Project и другите наследници на класа Element

3.3.3 Пакет ILNET.Common

Този пакет съдържа помощни класове, които предоставят обща помощна функционалност и дефиниции на делегати. Този пакет се компилира като самостоятелна библиотека и се използва от другите библиотеки на системата. По този начин става възможно дефинирането на обекти или делегати в една компилирана библиотека или програма, които безпроблемно да се подават към и използват от други библиотеки / програми заредени в рамките на един .Net приложен домейн. На фигура **13** е визуализирана клас диаграмата на пакета.

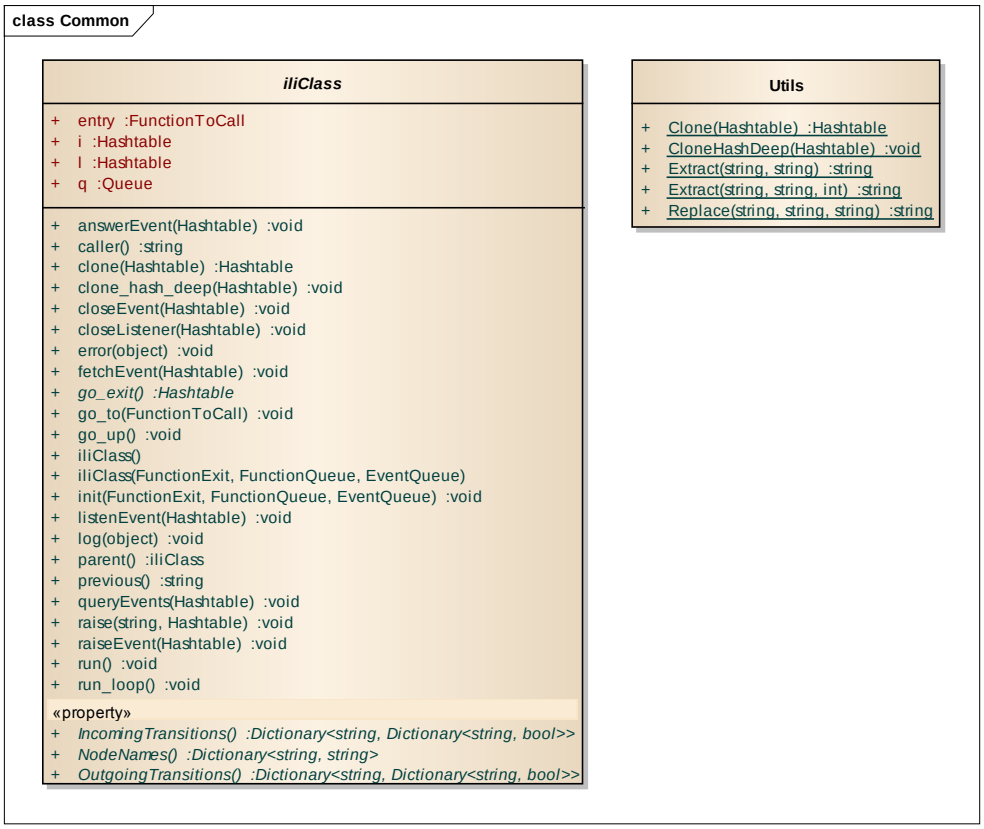
Ролите на класовете в пакета са следните:

- Класът **iliClass** представя един работен поток от метамодела и предоставя методи за редактиране на действията и променливите свързани с даден поток.
- Класът **Utils** предоставя методи за клониране и клониране в дълбочина на динамични структури от данни и методи за извличане и търсене със замяна на текст, които се използват за обработката на шаблоните при компилирането на метамодели.

В допълнение към класовете, в пакета са дефинирани и следните делегати:

```
public delegate void FunctionToCall();
public delegate void FunctionQueue(FunctionToCall f);
public delegate void FunctionExit(Hashtable result);
public delegate int EventQueue(Hashtable data, FunctionExit f);
```

Те служат за осъществяване на обратна връзка между ILNET сървъра и изпълняваните потоци процеси, както и между изпълняваните работни потоци и техните под-потоци.

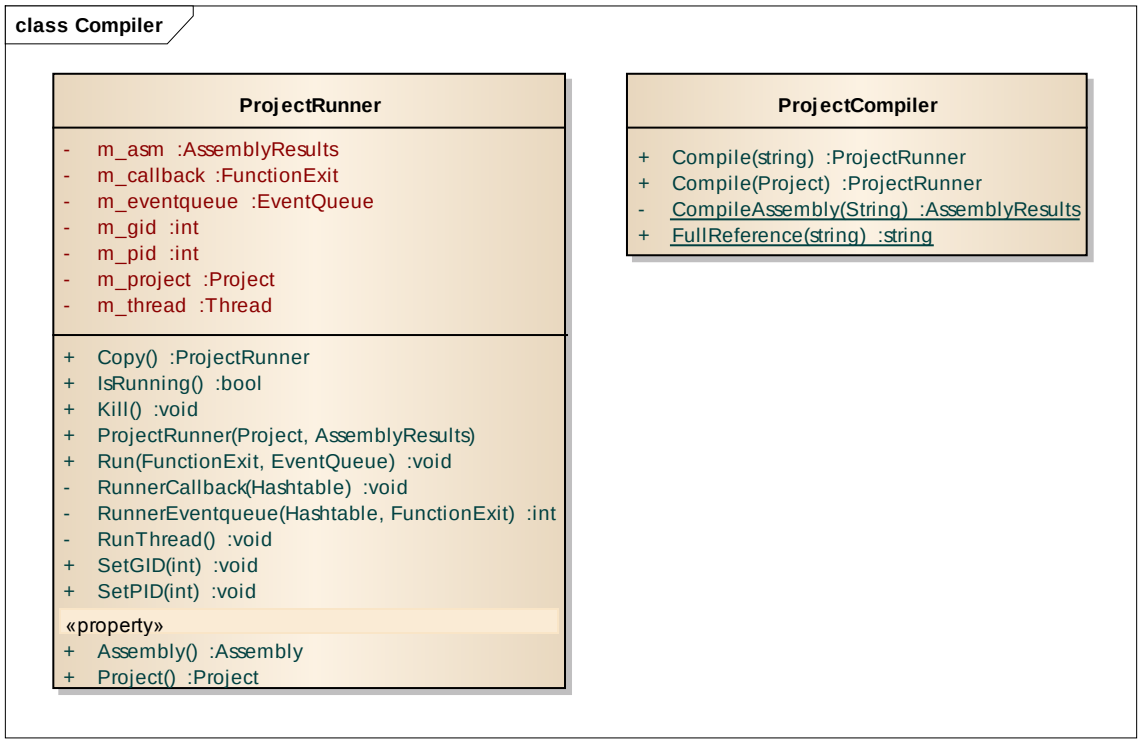


Фигура 13: Клас диаграма на класовете от пакета ILNET.Common

3.3.4 Пакет ILNET.Compiler

Този пакет съдържа класове, които предоставят методи за компилиране и изпълнение на работни потоци описани чрез ILNET метамодела. На фигура **14** е показана клас диаграмата на класовете от пакета. Ролите на класовете в пакета са следните:

- Класът **ProjectCompiler** използва C#.NET компилаторът, за да компилира изпълнима библиотека от генерираният изходният код за даден ILNET метамодел. В резултат създава инстанция на класа ProjectRunner.
- Класът **ProjectRunner** предоставя методи за стартиране и изпълняване на инстанции на описаните в изходният метамодел работни потоци.



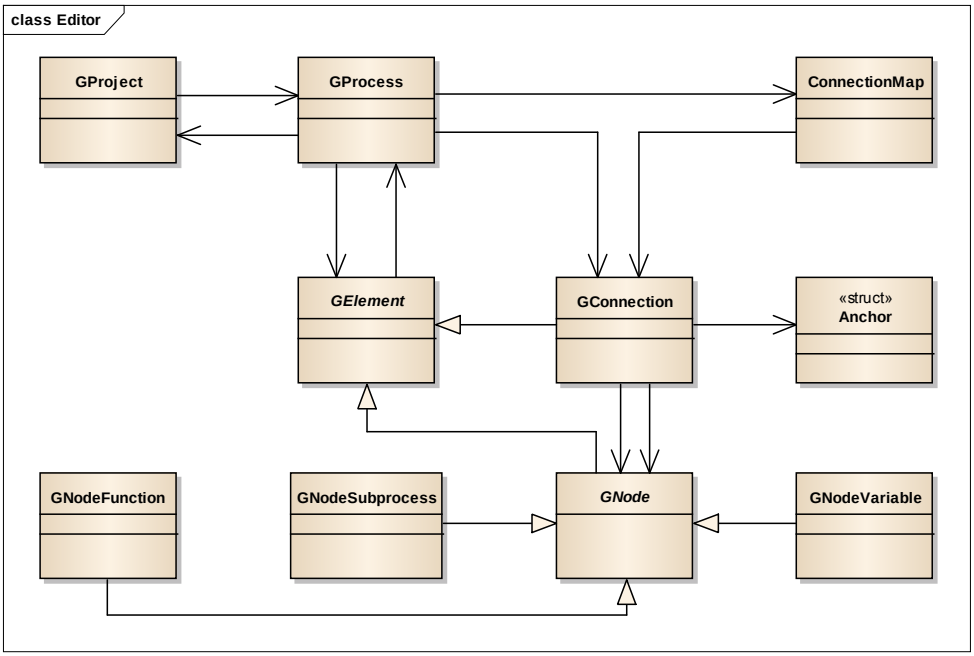
Фигура 14: Клас диаграма на класовете от пакета ILNET.Compiler

3.3.5 Пакет ILNET.Editor

Този пакет съдържа класове, които обгръщат класовете реализиращи ILNET метамодела, и предоставят методи за визуализация и редактиране на визуалните характеристики на ILNET метамодели. По-долу на фигура **15** е представена клас диаграма на основните класове

на пакета. Диаграмата е в съкратен вид (не показва променливите и методите на класовете). Класовете реализират функционалността за визуализиране и взаимодействие с отделните видове елементи на ILNET метамодела както следва:

- Класът **GProject** съдържа списък от всички работни процеси в метамодела. Той няма визуална функционалност, а само редакционна – свързана с добавянето, премахването и преименуването на процеси.
- Класът **GProcess** предоставя методи за визуализация, редактиране и взаимодействие с работни потоци.
- Класът **GElement** е абстрактен клас, който реализира част от общата функционалност за изчертаване и обработка на събития от графичната среда и се наследява от всички класове на библиотеката, които имат визуално представяне.
- Класът **GNode** е абстрактен клас, който наследява **GElement**, и добавя методи и свойства, характерни за елементите от тип действие (Node) или променлива (Variable).
- Класът **GConnection** предоставя методи за визуализация, редактиране и взаимодействие с връзките между елементите.



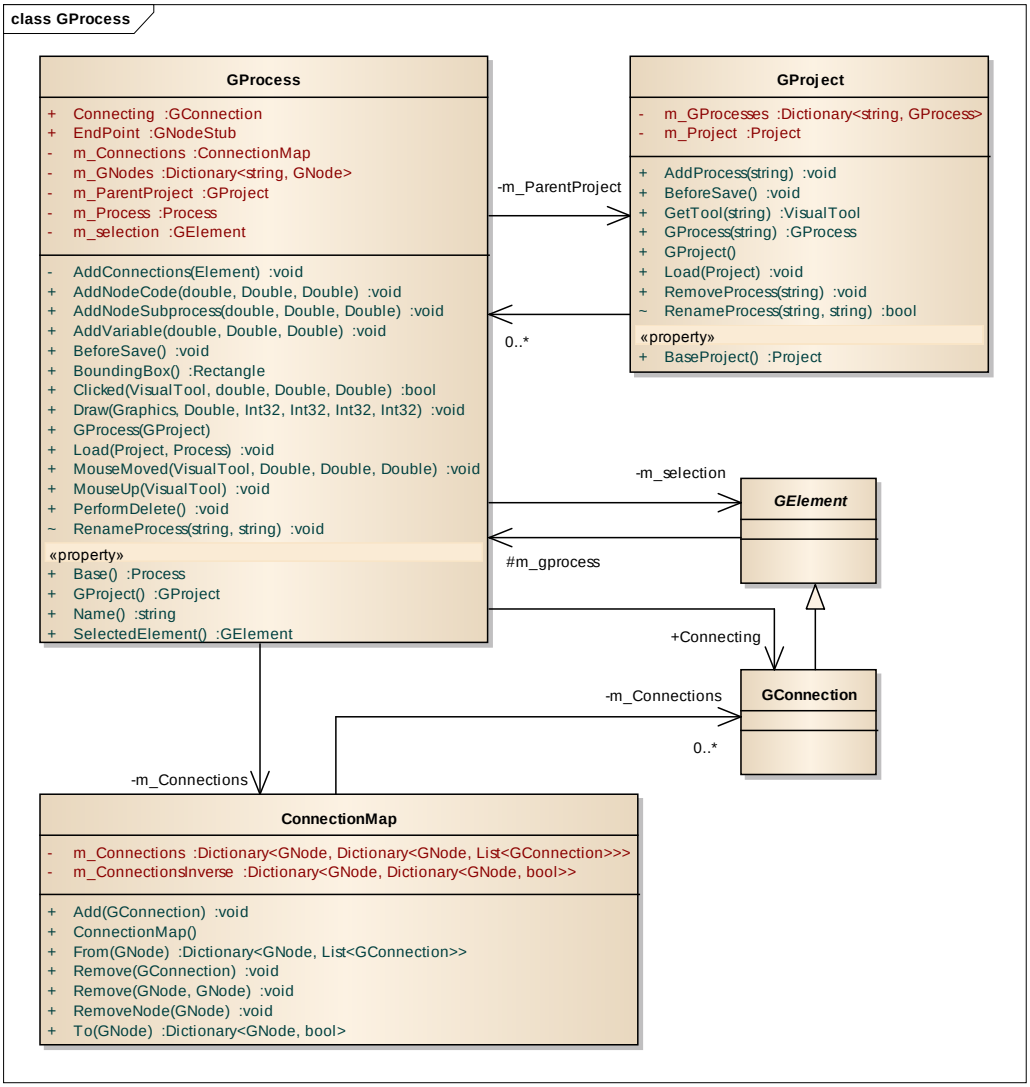
Фигура 15: Клас диаграма на класовете от пакета ILNET.Editor

Главен клас на пакета е *GProcess*, който е основна входна точка за визуализиране и редактиране на описания на процеси. Той делегира заявките за изчертаване и обработка на събития към останалите елементи на модела, ефективно капсулирайки тяхната функционалност. На фигура **16** е показана клас диаграма на класа *GProcess* и свързаните с него класове. Всички връзки между елементите в един процес се индексират в класът *ConnectionMap*, който предоставя методи за добавяне и премахване на връзки, както и за търсене на връзки по начален или краен елемент.

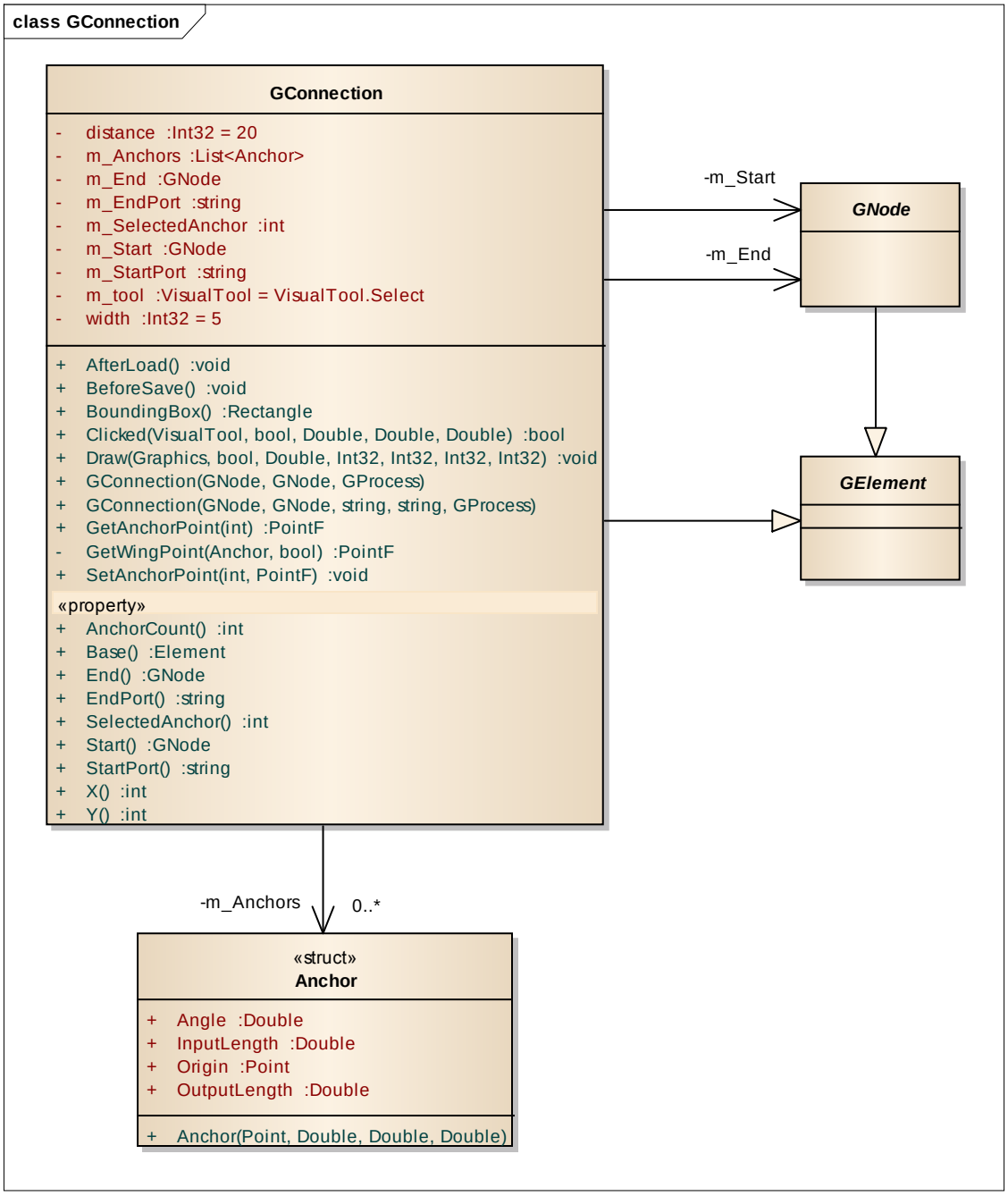
Класът *GConnection* реализира визуализирането на връзките между елементите. Всяка връзка помни елементите (GNode) и местата на прикачване към тези елементи на началото и краят на връзката. Към момента, стандартният начин за изчертаване на връзки използва свързани кубични криви на Безие, чиито контролни точки се съхраняват в структурата *Anchor*, и като метаданни към елемента прикрепен към тяхното начало. На фигура **17** е визуализирана клас диаграмата на класа GConnection и агрегираната от него структура Anchor.

Редакторът позволява създаването на връзки между следните двойки типове елементи:

- *GNodeFunction*/*GNodeSubprocess* → *GNodeFunction*/*GNodeSubprocess* – описва обикновена връзка (преход) от едно действие на процеса към друго.
- *GNodeVariable* → *GNodeSubprocess* – свързва променливата с конкретен входящ параметър на под-процеса (подава стойността на променливата като параметър при извикване на под-процеса).
- *GNodeSubprocess* → *GNodeVariable* – свърза променливата с конкретен резултат на под-процеса (записва в нея стойността на изходящият параметър след приключване работата на под-процеса).

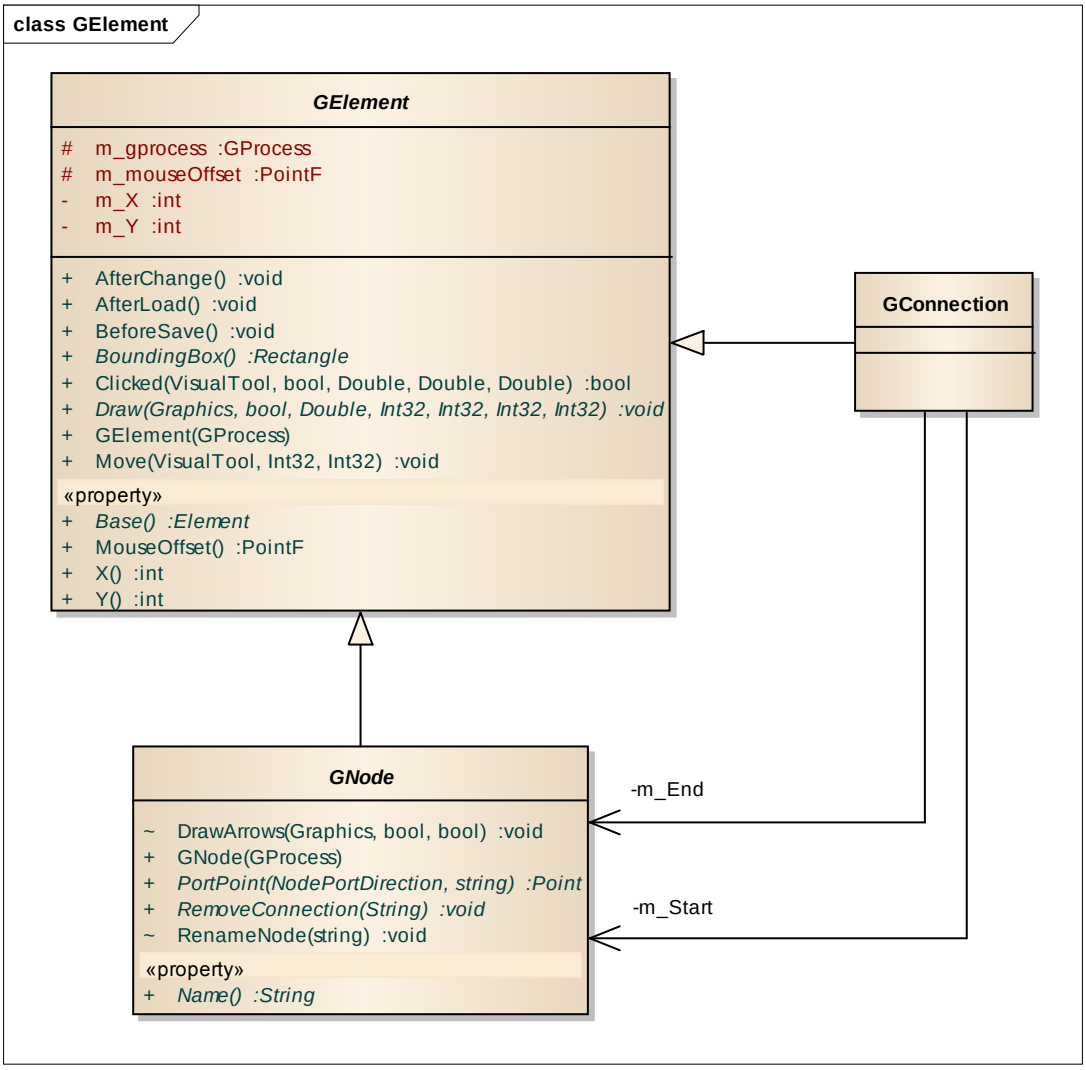


Фигура 16: Клас диаграма на класа GProcess и всички класове от пакета, които той агрегира



Фигура 17: Клас диаграма на класа GConnection

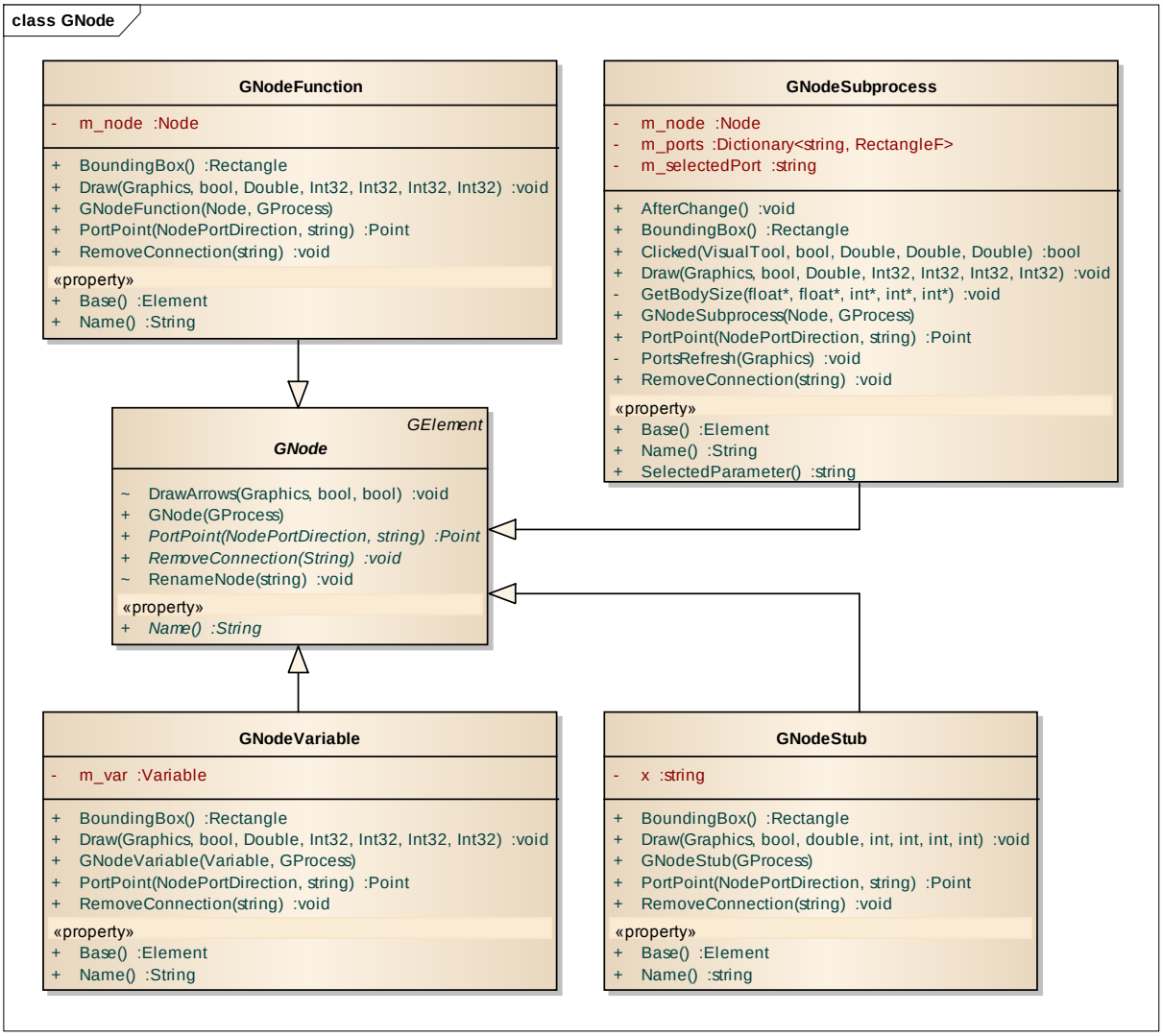
Класът *GElement* реализира част от общата функционалност за изчертаване и обработка на събития от графичната среда и се наследява от всички класове на библиотеката, които имат визуално представяне. Той дефинира задължителните методи свързани със зареждането и записът на метамодела и с визуализацията и интеракцията с него. На фигура **18** е показана клас диаграма на класа *GElement* и неговите наследници.



Фигура 18: Клас диаграма на класа GElement и неговите наследници

Класът *GNode* е абстрактен клас, който наследява *GElement*, и добавя методи и свойства, характерни за елементите от тип действие (*Node*) или променлива (*Variable*), като изчертаване на стрелките за начално и/или крайно действие (*Node*) на работният поток, и получаване на координатите на порт за свързване с действие, който съответства на конкретен параметър на прикаченият към действието под-процес. Класовете наследниците на *GNode* реализират функционалността за визуализиране и работа с различните видове елементи на ILNET метамодела. На фигура **19** е визуализиран класът *GNode* с неговите наследници. Класът *GNodeVariable* обгръща елемент от тип *Variable* и дава възможност за визуалното разполагане на променливи в работният поток, и за свързването им с входящи и изходящи параметри на изпълняваните под-процеси. Класовете *GNodeFunction* и

GNodeSubprocess обгръщат елемент от тип *Node* и реализират визуализирането и редактирането съответно на обикновени действия (функции) и действия, които извикват конкретен под-процес. Класът *GNodeStub* е помощен клас, който няма визуализация и съответстваш елемент от метамоделът, и служи единствено като временен несъществуващ елемент за изчертаването на нови връзки (*GConnection*), когато още не е известен крайният елемент на връзката.



Фигура 19: Клас диаграма на класа GNode и неговите наследници

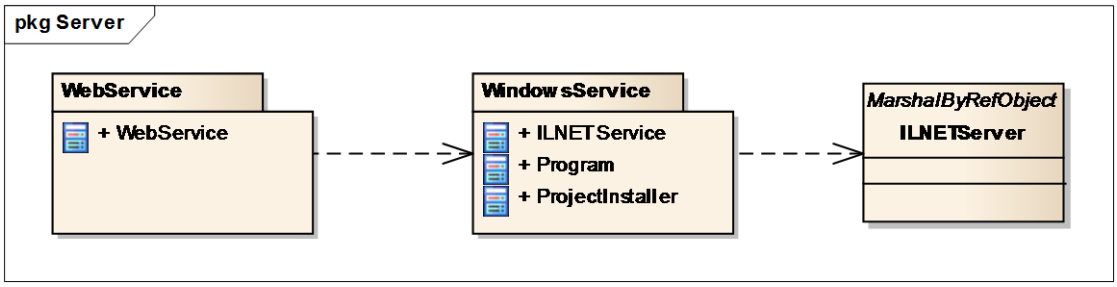
3.3.6 Пакет ILNET.Server

Този пакет съдържа класове, които реализират сървърната част на архитектурата на приложението. Те включват пакети с класове реализиращи различни видове точки за достъп

до сървъра и библиотеката *ILNETServer*. По-долу на фигура **15** е представена пакетната структура на сървърните компоненти. Класът *ILNETServer* предоставя хранилище, средства за изпълнение и средства за обработка на съобщенията от/към и между изпълнявани инстанции на ILNET метамодели. Другите пакети съдържат прокси класове, които предават получените заявки към съответните методи *ILNETServer*.

Функционалността реализирана от отделните пакети е следната:

- Пакетът **WindowsService** съдържа класове реализиращи Windows системна услуга (Service), която хоства инстанция на класа *ILNETServer* и регистрира слушател на TCP/IP порт, посредством което става възможност отдалечено извикване на методите на сървъра от други .NET Remoting приложения. Тази услуга предоставя бърз и удобен начин за свързване и работа със сървъра, но е технологично ограничена само за приложения писани на .NET.
- Пакетът **WebService** съдържа реализацията на ASP.NET Web услуга, която пренасочва всички получени заявки към конкретна ILNET Windows системна услуга. Тази услуга дава възможност на широк набор от приложения да се свържат с ILNET сървъра, посредством предоставянето на WSDL дефиниция и използването на SOAP съобщения при комуникация с услугата.



Фигура 20: Пакетна структура на сървърните компоненти

3.3.6.1 Класът ILNETServer

Класът *ILNETServer* предоставя хранилище, средства за изпълнение и средства за обработка на съобщенията от/към и между изпълнявани инстанции на ILNET метамодели и методи за управление на тези инстанции от другите компоненти. На фигура **21** е представена

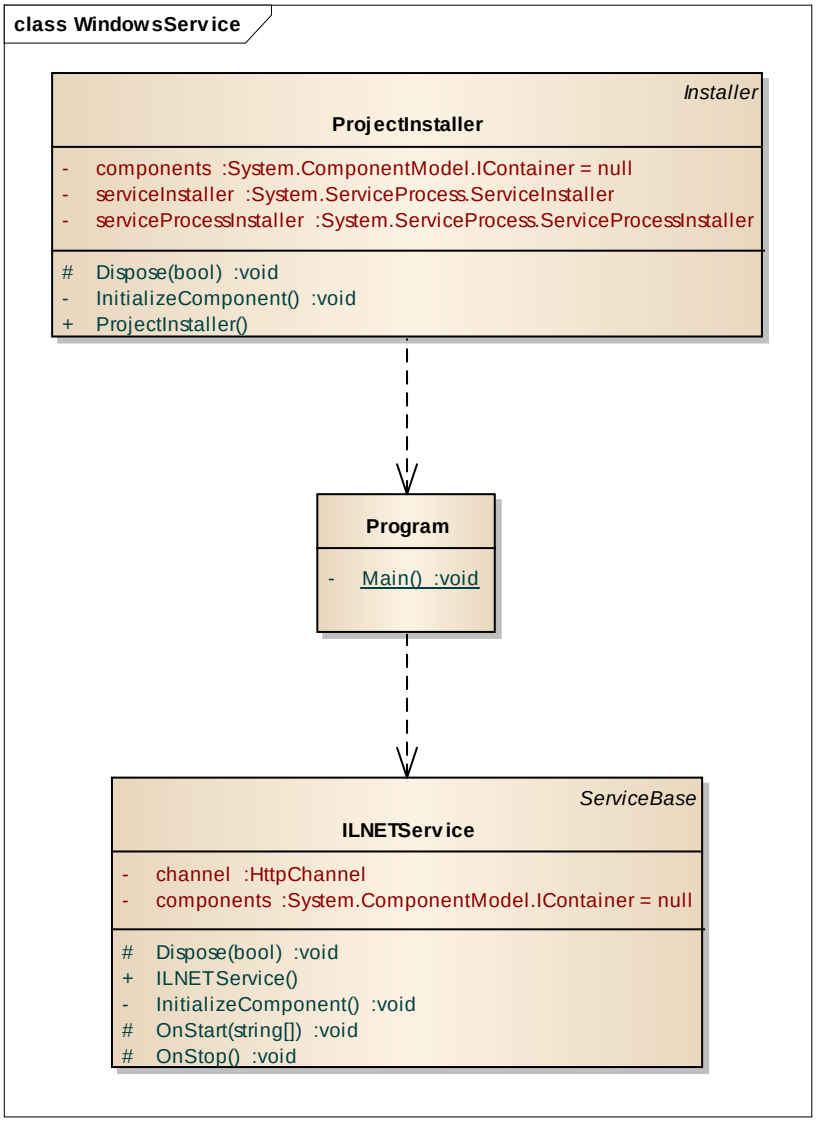
клас диаграма на класа *ILNETServer*. Дефинирани са следните видове методи за работа с инстанции на сървъра:

- Методи за работа с хранилището на ILNET метамодели. Хранилището използва локалната файлова система за съхраняване и извличане на XML описанията на моделите, и позволява групирането им в пакетна (директорна) структура.
- Методи за компилиране, зареждане в паметта и освобождаване на паметта от изпълними ILNET метамодели намиращи се в хранилището на сървъра.
- Методи за стартиране и прекратяване на инстанции на заредените в паметта изпълними метамодели.
- Методи за обработка на съобщенията от събитията, които настъпват в изпълняваните процеси
- Методи за извличане на получените резултати от изпълняваните инстанции на метамодели. Една инстанция може да произведе множество резултати по време на нейното изпълнение (всеки сигнал успешно преминал последното действие на основният работен поток на метамодела произвежда резултат).

3.3.7 Пакет ILNET.Server.WindowsService

Този пакет съдържа класове, които реализират Windows системна услуга, която дава възможност за работата с инстанция на класа *ILNETServer*, посредством .NET Remoting технологията. На фигура **22** е представена клас диаграма на класовете от пакета WindowsService. Услугата регистрира *ILNETServer* класа използвайки шаблона за проектиране Сек (Singleton), като по този начин става възможно хостването на процеси на работни потоци от ILNET метамодели, както и тяхното асинхронно изпълнение в отделни нишки, които се управляват от единствената инстанция на класа *ILNETServer*.

Фигура 21: Клас диаграма на класа ILNETServer



Фигура 22: Клас диаграма на класовете от пакета ILNET.Server.WindowsService

3.3.8 Пакет ILNET.Server.WebService

Този пакет съдържа реализацията на ASP.NET Web услуга, която предоставя достъп до публичните методи на класа *ILNETService*, като пренасочва всички получени заявки към конкретна услуга предоставяща достъп до инстанция на класа *ILNETServer* (работеща инстанция на системната услуга от пакет WindowsService). На фигура **23** е представена клас диаграма на класа. Уеб услугата служи като мост между приложенията, които могат да използват SOAP услуги, и .NET Remoting реализацията на WindowsService компонента. Тази архитектурата на приложението прави възможно уеб услугата и системната услуга да са разположени на различни системи, като по този начин се облекчава основният сървър от

изчислителните ресурси необходими за кодиране и декодиране на SOAP съобщенията, за сметка на по-голямо време за отговор на заявките.

Фигура 23: Клас диаграма на класа WebService

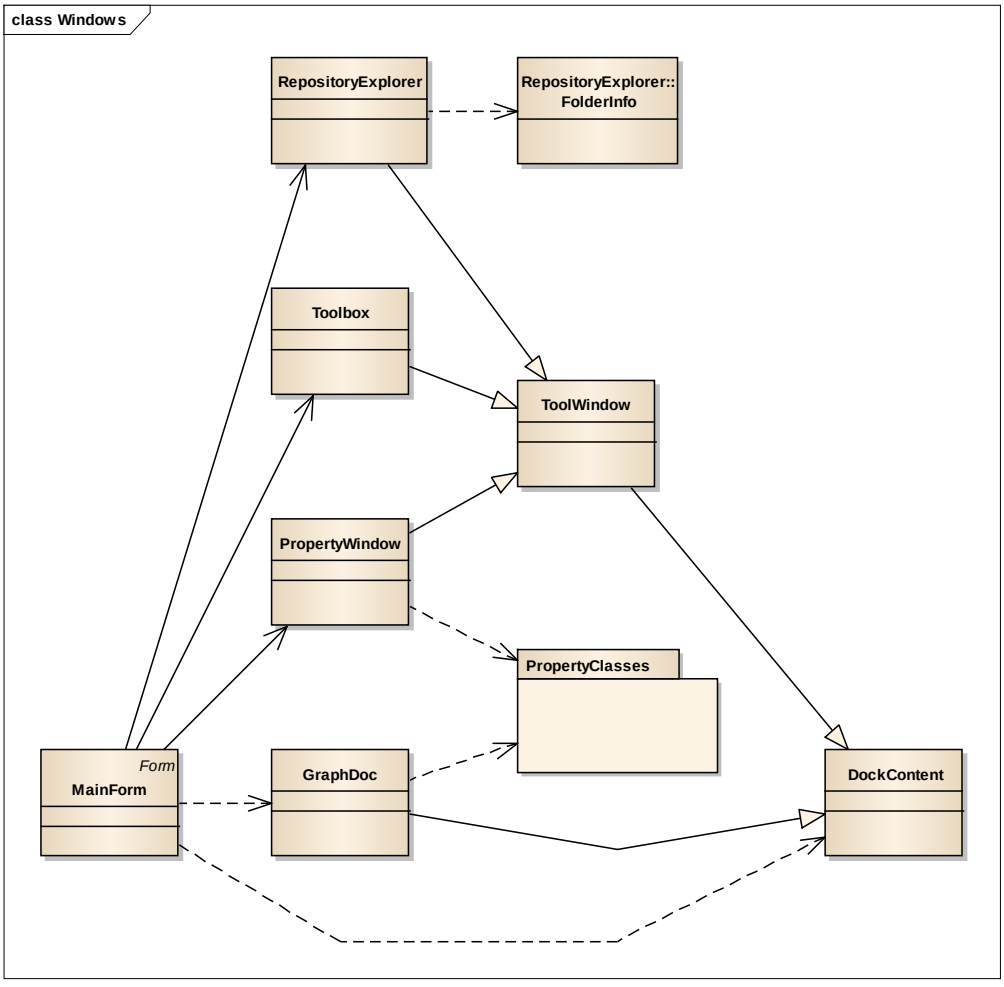
3.3.9 Пакет ILNET.Client.Windows

Този пакет съдържа класове, които реализират графичният потребителски интерфейс на системата. Той е изграден на базата на приложният потребителски интерфейс на .NET библиотеката Windows Forms и с помощта на средствата на Visual Studio за потребителски интерфейс – форми, менюта, ленти с инструменти, събития и др., както и на някои библиотеки, които предоставят разширена функционалност и потребителски удобства при работа с клиента. Сред използваните външни библиотеки са:

- Библиотеката DockPanel Suite предоставя удобна рамка за реализирането на много-прозоречни потребителски интерфейси, които да се доближават по функционалност и гъвкавост до прозоречната система на средата Visual Studio.

- Библиотеката ScintillaNET, която предоставя .NET обвивка на компонентите на библиотеката Scintilla за оцветяване и редактиране на изходен програмен код на различни езици, включително и C#. Използва се за предоставяне на вграден редактор на кода на функциите в работните потоци, като по този начин се улеснява разработката на работни потоци от по-ниско ниво (изграждащи блокове).

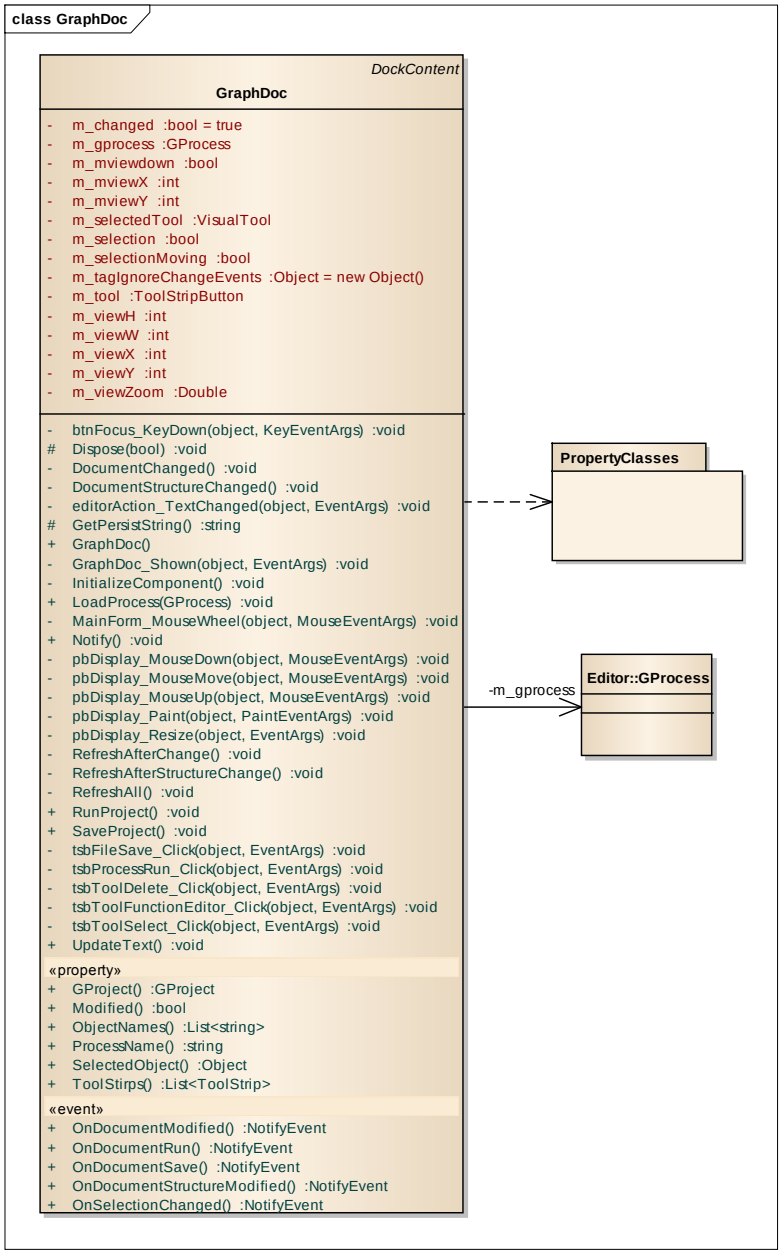
В клиента е използван много-документен интерфейс – Multiple Document Interface (MDI). Това прави възможно едновременното отваряне и редактиране на няколко работни потока от различни ILNET метамодели. На фигура **24** е представена клас диаграма на всички основни класове в пакета и техните връзки. Диаграмата е в съкратен вид (не показва променливите и методите на класовете).



Фигура 24: Клас диаграма на класовете от пакета ILNET.Client.Windows

3.3.9.1 Класът GraphDoc

Класът *GraphDoc* реализира визуализирането и редактирането на един работен поток от ILNET метамодел. Той съдържа графични компоненти от тип ленти с инструменти, панели, прозорец за изчертаване на изображения и панел за редактиране на изходният код на функциите. На фигура **25** е представена клас диаграма на класа *GraphDoc*. С цел компактност, не са показани член променливите на визуалните компоненти на формата.



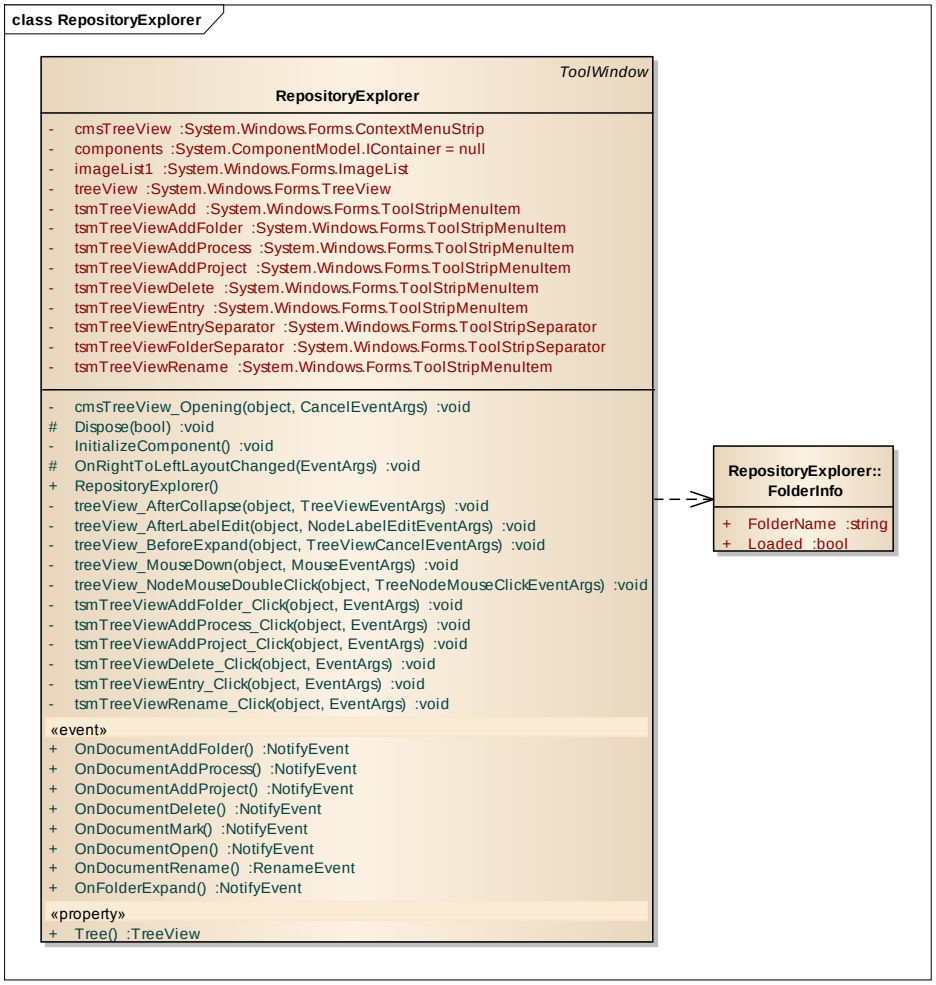
Фигура 25: Клас диаграма на класът GraphDoc

В прозорецът за изчертаване, посредством класовете на ILNET.Editor пакета, се показва избраният работен поток, а събитията за взаимодействие (интеракция) с мишката се предават на инстанцията на класа *GProcess*. Класовете от пакета PropertyClasses служат за показване и редактиране на характеристиките на различните видове визуални елементи.

3.3.9.2 Класът RepositoryExplorer

Класът *RepositoryExplorer* реализира визуализирането и навигирането в хранилището на ILNET сървъра. Той съдържа един графичен компонент от тип TreeView, в който динамично зарежда и показва структура на хранилището и проектите в него. За всеки проект като под-елементи се показват всички негови работни потоци.

На фигура **26** е представена клас диаграма на класа *RepositoryExplorer*.



Фигура 26: Клас диаграма на класът RepositoryExplorer

3.3.9.3 Класът PropertyWindow

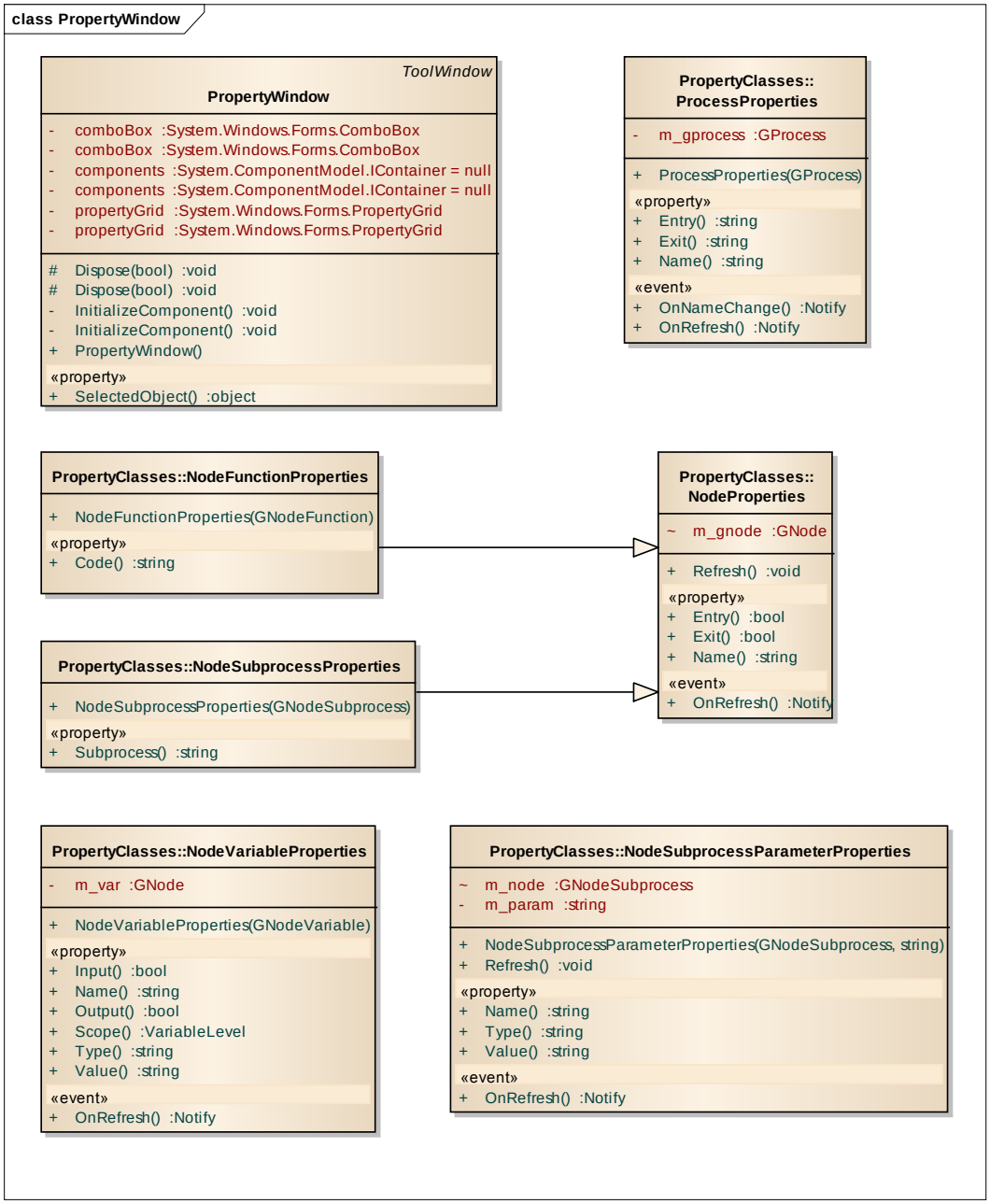
Класът *PropertyWindow* реализира визуализирането и редактирането на характеристики на обекти. Той съдържа компонент от тип PropertyGrid, посредством който се реализира визуализирането и редактирането на полетата от елементите на ILNET метамодела. Дефинирани са помощни класове, които описват данните на отделните елементи, и обработват събитията свързани с промяна и опресняване на данните.

На фигура **27** е представена клас диаграма на класа *PropertyWindow* и помощните класове за описание данните на елементите.

3.3.9.4 Класът MainForm

Класът *MainForm* реализира основният прозорец на графичният потребителски интерфейс на клиентското приложение. Той съдържа графични компоненти от тип ленти с инструменти, и инструментални панели и на свой ред обработва голяма част от събитията, които възникват в отворените документни прозорци. Използва една инстанция на класа RepositoryExplorer за показване на съдържанието на сървърното хранилище, и една инстанция на класа PropertyWindow за показване и редактиране на характеристиките на селектираният в някой от документните прозорци елемент. За всеки отворен работен поток създава нов документ от тип GraphDoc. Реализиран е много-документен интерфейс, посредством панела DockPanel на библиотеката DockPanelSuite.

На фигура **28** е представена клас диаграма на класа *MainForm*. С цел компактност, не са показани член променливите на визуалните компоненти на формата.



Фигура 27: Клас диаграма на класът PropertyWindow

Фигура 28: Клас диаграма на класът MainForm

ГЛАВА 4. ПРИЛОЖЕНИЯ НА РАЗРАБОТЕНИЯ МОДЕЛ

4.1 Извличане на контактна информация от документи на български език

Извличането на структурна информация от неструктурирани текстови документи е древна задача чиито интерес се увеличава с нарастване количеството информация достъпна в неструктуриран формат (напр. Интернет). В настоящата статия е представена система предназначена за улесняване потребителя при създаването на автомати и граматики за анализ на текстове, използвайки регулярни изрази, морфологичен анализатор, връзка с БД и вграден интерпретатор. Тази функционалност позволява софтуерният продукт да бъде използван като средство за описание процеса на извличане на информация. При проведен експеримент с извличане на контактна (адресна) информация са постигнати висока степен на откриване и прецизност на получените данни. Използваният разширен синтаксис на регулярни изрази и граматика спомагат за по-лесното имплементиране на необходимите за подобрението на анализа корекции и нововъведения.

4.1.1 Регулярни изрази

Регулярният израз е низ, който описва или разпознава множество низове, съгласно определени синтактични правила. Те се използват от много текстови редактори и помощни инструменти за търсене и манипулиране на съдържанието на текстове, базирани на определени шаблони. Стандартно средствата за работа с регулярни изрази предоставят следните операции за тяхното конструиране:

- **алтернативи (или)** – избор между два или повече равностойни под-израза.
- **групиране** – използва се за определяне на обхвата и прецедента на операторите.
- **квантификация** – квантификатор след символ или група определя колко често този предишен израз може да се появи. Най-разпространените квантификатори са ‘**?**’ (0 или 1 път), ‘*****’ (0 или повече пъти) и ‘**+**’ (1 или повече пъти)

Тези конструкции могат да бъдат комбинирани, за да се получи израз с произволна сложност. Точният синтаксис за регулярни изрази варира между различните инструменти и области на приложения.

Много реализации на парсери с регулярни изрази добавят допълнителни елементи към синтаксиса с цел улесняване извличането на допълнителни данни след приключване или по време извършването на търсенето. В имплементацията на разглежданата система две разширения на синтаксисът предоставят следните възможности:

- **именувано групиране** – позволява извличане на текстът разпознат от конкретна група (под-израз). За разлика от стандартният достъп до текста разпознат от под-изразите, посредством индекс на група от първо ниво (най-често реализиран като \x или $x, където x е число-индекс) именуването на групите разширява функционалността като предоставя както достъп до вътрешни под-изрази (вложени в основните) така и средство за указване че един или повече под-израза описват един и същ езиков елемент в допълнение към много по-лесното обработване на резултата от търсенето с регулярният израз поради премахнатата необходимост от постоянно броене на скоби
- **външни под-изрази** – за учебни примери и експерименти дължината на регулярните изрази рядко е проблем. Когато обаче се започне описание на дадена езикова конструкция, което подлежи на чести редакция и допълване във времето изразът може да стане необозрим. Използването на външни изрази, които само се указват в основният прави реализацията на промените значително по-лека и с по-малко риск от грешки

4.1.2 Елементи на използваната граматика

Използвана е системата ILI [54] за визуално построяване на граматики.

Структурата на граматиката използвана в настоящата разработка представлява множество от класове, всеки от които съдържа граф описващ процеса на анализът извършван от конкретният клас.

Всяка граматика съдържа един или повече класове. Когато са стартирани, всеки клас обхожда входният текст, събирайки всички възможни начини да се отиде от неговия начален връх, до крайният. Всеки краен автомат може да се представи като се използва един клас. Но за повечето цели един клас не е достатъчен. Много парсери използват BNF нотацията, позволявайки една дефиниция (автомат) да бъде използван много пъти и по този начин

значимо намалявайки както размерът на граматиката, така и времето за написването й. Всеки клас в настоящата система може да бъде считан като дефиниция на едно правило. Но за разлика от дефинициите в BNF, класовете тук могат да получават параметри и да връщат какво са открили. Тези параметрични дефиниции позволяват на потребителя да се фокусира върху граматиката като цяло. Класовете "Регулярен израз", "Низ" и "Заявка до БД " са само няколко примера. Всеки клас има характеристика, която инструктира парсера кога да стартира търсенето за даден клас. За момента тя може да е "никога", "веднъж" или "на всеки символ ". Класът също така запаметява данните на всяко последно успешно разпознаване на подкласовете на върховете. Това позволява условията и операциите, които върховете съдържат, да бъдат по-мощни, използвайки късно оценяване, предикати и т.н.

Върховете на графите от класовете могат да се използват за представяне на обикновени състояния. Допълнително в настоящата системата върхът определя и:

- Какви условия трябва да са изпълнени при влизане / излизане от върха;
- Какво се случва при влизане / излизане от върха;
- Колко символа да се прескочат от входния файл преди влизане / след излизане от върха (може да е отрицателно) (използва се за контрол на главата на четене от входния файл);
- Какво е неговото име – мястото в класа, където се записва резултата от успешно разпознаване;
- Какъв клас (ако е зададен) се използва за анализиране на данните (подклас);
- Какви параметри да се подадат на подкласа, ако се използва такъв;

Парсерът използва търсене с обхождане в ширина, за да намери всички участъци от текста, които се разпознават от граматиката. Понеже класовете са недетерминистични е възможно едно стартиране на клас да върне повече от един резултати. Това изисква парсерът да може да проследява два или повече пътища в един клас, едновременно, без да обърква паметта на различните нишки. За да постигне това, парсерът създава нова инстанция на класа всеки път, когато той се стартира, или когато се достигне разклонение на пътя. За да се направи анализът по ефективен се използва трислойна архитектура на паметта.

Всеки клас има три слоя от променливи:

- Локални променливи: винаги се копират, достъпни са само от една инстанция. Представляват паметта на един път. Могат да бъдат използвани за изчисления, които не зависят от резултатите, постигнати при обхождането на другите пътища. Елиминират риска от объркване на променливите от различните работещи инстанции.
- Посредствени променливи: копират се, когато класът се инициализира. Достъпни са от всеки наследник на първата инстанция (когато е стартиран класът). Може да се използва за синхронизиране на активните пътища при едно извикване на класа. Подходящо за имплементиране на (А без Б) структурата.
- Глобални променливи: никога не се копират, достъпни от всички инстанции.

До момента са разработени модули за разпознаване на низове и регулярни изрази, извършване на морфологичен анализ и работа с БД.

4.1.3 Реализация

В основата си системата се състои от изпълним модул, който съдържа в себе си библиотеките за интерпретиране на разширените граматики и регулярни изрази описани в предишните секции и код за синтезиране на XML код съдържащ откритите при анализ елементи в съответствие с използваната граматика.

Тъй като най-голямото хранилище на свободно достъпна информация е Интернет е добре да се обърне внимание на предварителната обработка на документите преди извършване на по-нататъшни анализи върху тях. На фигура **29** е показан работният поток за анализ на хипертекстов документ използван в настоящата разработка. Това са някои от най-съществените неща, за които трябва да се внимава при работа с хипертекстови документи:

- Кодовата таблица. Следните фактори са полезни при определяне правилната кодировка за документа: средна дължина на думите и брой азбучни символи. Те помагат за различаването на едно-байтови от мулти-байтови кодировки.
- Ненужни данни – това са скриптове, скрити полета, коментари и др. които не се използват по време на анализ или могат да доведат до смъкване прецизността.
- Специални символи – някои символи могат да присъстват повече от веднъж в дадена кодова таблица под различни кодове с различни семантики. Ако анализаторът не

диференцира тези семантики е добре тези символи да се преобразуват до един единствен, за да се улесни писането на изразите.

Фигура 29: Работен поток за анализ на хипертекстов документ

При създаването на граматиката и регулярните изрази е използван процес подобен на подходите за гъвкава разработка на софтуер. Тъй като използваните разширения в граматиката и синтаксисът на регулярните изрази правят анализаторът лесен за разбиране и промяна във всеки един момент от неговото доразработване, се губи необходимостта от създаване на напълно точни изрази и граматики от самото начало. Вместо това е използван подход подобен на принципа на опита и грешката, при който се редуват коригиране на анализатора и тестване върху досегашните страници + нови.

На таблица 1 са показани процентът на разпознаване на различните под-елементи от търсената контактна информация при извършен анализ върху 149 фирмени сайта на български фирми. Всяка от разгледаните редакции представлява модифицирана версия на предишната. Ясно се вижда как с всяка модификация на граматиката и/или регулярните изрази в по-горна редакция се подобрява точността на работа на анализаторът. Това се дължи от части на условието че всяка корекция трябва да не влошава точността при пускане на анализаторът върху вече вкараните в тестовата група документи.

Последната 12-та редакция съдържа регулярни изрази с обща дължина 187 961 символа. По-голямата и дълга част от тях са автоматично генерирани изрази разпознаващи различни видове населени места в страната (184 117), които са използвани като вмъкнати под-изрази в регулярният израз за адрес.

Редакция	Адреси	Телефони	Имейли	Линкове
10	75.07%	85.82%	100.00%	80.00%
11	86.54%	96.65%	100.00%	85.00%
12	89.54%	96.54%	100.00%	90.63%

Таблица 1: Резултати при извличане на контактна информация от хипертекстови документи.

Разработен е подход за извличане на информация от текстови документи. Използваните средства позволяват лесното реализиране на различни модули за извличане на данни. Визуалната среда за описание на граматиката може да се използва и като инструмент за моделиране на процеси на анализ и обработка на извлечената информация а интуитивният интерфейс прави създаването и редакцията на граматиките лесно и бързо. Проведеният експеримент с прогресивно подобряване на анализаторът дава добри резултати.

4.2 Автоматизирано генериране на метаданни за учебни обекти

С навлизането на е-обучението се появява необходимостта от създаване на софтуерни системи за автоматизирано генериране на електронни курсове. Изключително важно за всеки такъв курс е описанието на характеристиките на включените в него учебни обекти (LO - Learning objects), т.е. създаването и/или генерирането на метаданните.

Основен стандарт в областта на метаданните за обекти, използвани в обучението е LOM (Learning Object Metadata) [99], според който метаданните – на брой 58, са разпределени в девет групи: 1. Основни (General), 2. Продължителност на съществуване (Life Cycle), 3. Данни за метаданните (Meta-Metadata), 4. Технически характеристики (Technical), 5. Образователни (Educational), 6. Права (Rights), 7. Връзки (Relation), 8. Анотация (Annotation) и 9. Класификация (Classification). Всяка група съдържа определен брой полета, които я

характеризират в детайли. Освен че описват съдържанието на учебния обект, метаданните се отнасят и до други елементи на процеса на обучение. С използване на метаданни, аташирани към съответните учебни обекти, се улеснява тяхното търсене и обмен, като значително се повишава качеството на администриране на системите за е-обучение. От друга страна, създаването и въвеждането на метаданни е трудоемка и високо квалифицирана експертна дейност, което налага търсене на различни начини за тяхното автоматизирано генериране.

Според [100], автоматизираното генериране на метаданни за даден учебен обект е резултат от машинна обработка, в която участват единствено разработчик, разпространител на софтуер и инициатор на процеса. Последното е валидно само при условие, че метаданните за даден LO са статични и неизменяеми при участие на дадения обект в различни процеси на виртуално обучение. Подобно, между другото, е и разбирането на повечето автори, които предполагат (явно или не), че метаданните, съпровождащи учебните обекти (вкл. тестовите единици), са относително неизменни – независимо от процесите, в които участват. Подобна предпоставка силно стеснява възможностите за удовлетворително решаване на проблема за автоматизирано генериране на метаданни за LO. Например, генерирането на метаданни, свързани с трудността, областта, степента на гранулираност, типична възраст на обучаемите, средното време, необходимо за усвояване на материала, и др. под. на даден LO, няма как да не зависи силно от конкретния контекст на провежданото обучение. Нещо повече, динамичният характер на редица метаданни за LO (напр. трудност), едва ли може да бъде поставен под съмнение (напр. извън зависимост от постиженията на конкретна група обучавани). Очевидно, че за решаване на посочения проблем, трябва да бъдат изследвани и други подходи.

4.2.1 Подходи за автоматично и автоматизирано генериране на метаданни

Съществуват различни методи за автоматично и автоматизирано извличане на метаданни за даден LO в обучението. По-голямата част от тях се основават на стандарта LOM − Learning Objects Metadata (вж. например [95, 97, 105]). Разработени са и гъвкави инструменти, които не следват известни стандарти, а разглеждат връзката на потребителите с учебните обекти при поставена конкретна цел [102].

В системите за автоматизирано генериране на метаданни се използват методи за извличане, които зависят от типа на съответната метаданна (низ, число, елемент от списък, и т.н.) и от вида на съответния LO.

Обикновено се използват ***два основни подхода*** за генериране на метаданни − съответно базирани на ресурси или на контекста на материала [95, 105].

Първият подход е комбинация от следните методи:

- *добиване* (harvest) на метаданни чрез събиране на вече създадени метаданни от съществуващи хранилища;
- *извличане* (extraction) – необходимата информация се открива в съдържанието на материала (напр. търсене на ключови думи);
- *класификация* – определяне на стойностите на метаданните от специално съставени речници, например идентификатори на езици;
- *разпространяване* **(**или наследяване) с използване на връзка на конкретния материал с останалите, и неговата позиция в структурата на курса.

В зависимост от конкретния материал и хранилището, в което се извършва търсенето, дадена метаданна може да бъде генерирана чрез повече от един метод. Съществуват и метаданни, които не могат да бъдат определени чрез нито един от тези подходи. За тях е възможно да се приложи вторият (контекстно базиран) подход.

Метаданните могат да бъдат извличани от различни източници. Например в [95] са разграничени пет подобни източника: системата за управление на обучението (LMS − Learning Management System) и използваната операционна система; log файловете на сесиите; профилите на потребителите; обратната връзка - информацията, която се получава в следствие на работата на потребителите с материалите по време на обучение (напр. време, необходимо за усвояване на материала, резултати от изпитвания и др).; други LO и техните метаданни.

В [97] се изследва подход, който анализира контекста на документа и включва подробно изследване на профила на неговия създател и на информацията за курса, в който е включен материала. Системата за управление на изучаваното съдържание дава информация относно

връзките между обектите, начинът за използване на метаданните за материали, които съдържат даден LO. В системата за управление на е-обучението се съдържа богата контекстуална информация (напр. в кои курсове се използва даден обект, колко пъти е осъществяван достъп до него, колко пъти е ‘свалян’ и др.), която също може да бъде използвана за определяне на метаданни. Едновременно с концептуалния анализ се използват и други методи: анализ на съдържанието на документа (включващо езикови идентификатори, методи за извличане на ключови думи, разпознаване на образи за изображенията); оценка на данни, свързани с използването на документа в системата (напр. време, проведено в изучаване на даден материал; време за решаване на определени задачи и т. н.); анализ на информацията, свързана със сложността на структурата на документа (напр. някои материали − част от учебния обект, понякога се съхраняват отделно и са източник на данни за останалите), и др.

Подобни подходи за генериране на метаданни е възможно да се осъществят в рамките на конкретни среди за обучение (напр. в среда за обучение *AdapWeb* [127]). В основата на реализация [126] е стандартът LOM, като в зависимост от вида на съответните метаданни, процесът на генериране се извършва автоматично или автоматизирано. За автоматично генериране на метаданни се използват 4 (четири) метода:

- анализиране на физическия файл, който съдържа учебния материал и извличане на данни за попълване на метаданни от раздел 4. Технически характеристики на LOM;
- търсене и извличане на метаданни от базата данни с потребители, курсове и материали на системата за е-обучение (в сл. AdapWeb);
- търсене и извличане на метаданни от съответната БД за използваните LO;
- анализиране на XML-документи от БД, извличане и складиране на данни с използване на съответните тагове.

При втория (автоматизиран) подход се предполага диалог с потребителя за потвърждаване и уточняване на данните, получени при автоматично извличане. Използват се два метода:

- определяне на стойности по подразбиране (предположение за възможните метаданни);

- търсене по шаблон на обект, подобен на изследвания (за целта е необходимо шаблонът да е предварително създаден от потребители в XML базата данни).

Общото и в трите подхода е, че са базирани на стандарта LOM (главно − поради неговата популярност в света на метаданните за учебни обекти, използвани в е-обучението). Извличането на метаданни от физическия файл, който съдържа материала, търсенето в хранилища с метаданни, разглеждането на връзките между обектите и тяхното взаимодействие с потребителите, са основните подходи, които се срещат в представените системи. В изследване [95], освен по-горе посочените методи е експериментиран и подход, който използва отговори на потребителите на софтуер за електронни курсове. Опитът е направен за метаданни ‘ключови думи’ (поле 1.5 на LOM) и постига добри резултати - около 90% от получените отговори съвпадат с думите и фразите, посочени от експерта, който въвежда материала в системата.

В настоящата работа се изследва подход, свързан с хипотезата, че определени метаданни за LO − в по-малка или в по-голяма степен, зависят и се определят от съдържанието, историята, субектите и събитията на процеси на обучение, в които участва дадения LO. В случая на автоматизирано генериране на тестови единици и съответни метаданни, първите изследвания в указаната посока, са проведени в [113] с въвеждане на т. нар. ‘акумулативни тестови единици’.

Целта тук е - да се изследва приложимостта на подобни ‘акумулативни’ подходи за автоматично и автоматизирано генериране на метаданни, свързани с LOM.

4.2.2 Към автоматизирана система за генериране на метаданни

В Таблица 2 е представена класификация на методите за автоматично (или автоматизирано) извличане на метаданни от LO. Включването на даден метод към някоя от 6-те групи се определя от неговата същност – основният принцип, на който се основава неговото прилагане.

Група	Същност
1.	Акумулативни тестови въпроси (test items), генерирани при виртуално обучение
2.	Автоматизирано извличане от текстове
3.	Анализ на връзки и взаимодействия
4.	Търсене в хранилище с метаданни (при въвеждане)
5.	Информация за самата система
6.	Все още неопределена

Таблица 2. Групи от методи за извличане на метаданни по стандарт LOM

Първата група методи е свързана с използване на системи от акумулативни въпроси, подходящи за определен вид метаданни, свързани със съответния LO (текст, тестова единица, събитие и т.н.), а така също и с други LO, използвани многократно в контекста на различни процеси на виртуално обучение. Примери на акумулативни въпроси, отнасящи се до метаданни от схемата на LOM, са представени в Таблица 3. Елемент на подобно обучение могат да бъдат не само обучавани, но и преподаватели – експерти в изучаваната предметна област (ПО), като отговорите на последните могат да имат по-съществен принос (напр. по-голяма тежест) в процеса на генериране. Следвайки принципите [103, 104] на акумулиране на отговори и генериране на нови типове тестови единици (вкл. с използване на схеми за оценяване на различни когнитивни равнища [94]), в процеса на виртуално обучение могат да бъдат постоянно натрупвани и оценявани кандидат-данни за този вид метаданни.

Метаданна	Примерни акумулативни въпроси
1.2 Title	Посочете най-подходящото (според Вас) заглавие на учебния обект
1.4 Description	Резюмирайте (до 200 знака) материала
1.5 Keywords	Избройте (поне 5) ключови понятия, свързани с учебния обект.
1.6 Coverage	Посочете периода, за който се отнася материала. Посочете (поне 3) основни тематични области, свързани с материали (желателно в йерархичен ред)

Таблица 3. Примерни акумулативни въпроси по стандарт LOM

Втората група методи (за автоматизирано извличане) включва − от една страна, подходи за търсене на информация в текстовото съдържание на учебния материал, и извличане на данни от физическия файл, в който е съхранен (напр. попълване на технически характеристики според данните във визитката на файла); от друга − методи за класификация спрямо предварително създадени речници, например идентификация на езика.

В [93] е разгледан метод за извличане на данни, посредством специализирани граматики и регулярни изрази. Използваният подход е подходящ за извличането на метаданни, които да с достатъчно ясна и отличаваща ги структура и не изискват сложен контекст за тяхната коректна идентификация. Този подход често дава най-добри резултати за данни, които лесно могат да бъдат отделени и извлечени от текста, но изисква значителни умения за реализирането на необходимите граматики за тяхното извличане. От друга страна методите на принципа на обучението върху входни данни [96] са по-лесни за използване за сметка на допълнителния процес на натрупване и анализиране на обучаващи примерни метаданни.

Методите от ***третата група*** провеждат анализ на връзки, отношения, взаимодействия и данни за потребители, структура на обекти и др. под. в автоматизираната система.

Методите от ***четвъртата група*** са предназначени за търсене и откриване на съответни метаданни във вече създадени хранилища. В [98] е направено проучване на такива хранилища, базирани на LOM. Изследвани са приликите и разликите в подходите за изграждане на такива структури. На първо място е необходимо да бъде определен стандартът на метаданните, използвани в съответната автоматизирана система. Решението на този въпрос зависи най-вече от съдържанието и предназначението на ползваните електронни документи. По принцип е възможно и ‘смесване’ на елементи от различни стандарти.

Петата група от методи определят метаданни, които се формират още при създаването на системата за обучение и схемата за мета-метаданни.

Целта на разработката е да систематизира и изследва подходящи подходи за извличане на метаданни за LO. За провеждане на експерименти е избран стандарта LOM, поради неговото широко използване и факта, че ще се изследват материали от различни области.

Проучването на различни методи за автоматизирано извличане на метаданни показва, че съществуват добри възможности за минимално участие на автора на LOS в съставянето на визитка на учебния обект. Както беше посочено, може да се разграничат 5 (пет) основни групи подходи. Изследването на методи от първата група (акумулативни въпроси, генерирани в резултат на виртуално обучение с различни групи обучавани) се оказва особено продуктивно, и ще бъде предмет на следващи публикации. Особено предизвикателна е задачата по изграждане на курсове за е-обучение 'от нулата'. Последното означава – стартиране на виртуално обучение от набор LO, свързани със съответната предметна област (дори при отсъствие на съпровождащи метаданни и банка от тестови въпроси), и постепенното генериране (на базата на подходящи акумулативни тестови единици и на метаданна за всеки използван в обучението LO, и на тестови въпроси от различен тип (на базата на естествени дистрактори – резултат от 'колекционирани' и оценени отговори на обучаваните в резултат на зададените акумулативни въпроси). Перспективно направление на изследванията в областта на интелигентните системи за е-обучение е свързано с развитие на една друга идея − включване в процеса на виртуално обучение и на акумулативни въпроси, свързани с когнитивни равнища на знанието (напр. по таксономията на Блум).

4.3 Моделиране и управление на виртуални адаптивни курсове, базирани на moodle

Бизнес-процесът може да се представи като набор от относително обособени и структурирани дейности или задачи, предназначен за производство на продукт (или достигане на определена цел), представляващи интерес за конкретен потребител или група от потребители [77]. Технологиите за моделиране на бизнес-процеси (BPM[1]), посредством потокови структури, включващи техни подпроцеси или дейности (workflow), са съвременен начин за реализация, поддръжка и управление на бизнес-процеси с използване на информационни технологии. Съответните софтуерни средства предоставят възможности на

[1] Business Process Modeling

потребителите за моделиране на интересуващите ги бизнес-процеси, за тяхната виртуална реализация, стартиране и изпълнение в реално време, и с възможности за усъвършенстване на съответните модели на базата на получаваните резултати.

Системите за управление на потокови структури (WMS[1]) дават възможност за дефиниране на различни процеси или дейности, в които участват и си взаимодействат множество потребители. При достигане на всяка отделна стъпка на потока, WMS осигурява комуникация и включване на потребителите и ресурсите, необходими за изпълнението. След приключване на съответната стъпка (по един или друг начин – вкл. с обработка на грешки и частично приключване), WMS трябва да подготви необходимите данни и да 'осигури' включване на лицата, отговорни за изпълнението на следващата (в зависимост от потоковата структура) стъпка.

Средствата за моделиране на бизнес-процеси управляват изпълнението на задачи и дейности, свързани със съответната бизнес-среда, вкл. по разпределение, контрол и оптимизация на необходимите ресурси (материални, информационни и човешки). В повечето случаи, задачите и дейностите, отнасящи се до даден бизнес-процес, са взаимно-зависими, т.е. изпълнението на една задача/дейност зависи по един или друг начин от изпълнението на други задачи/дейности. Съществуващите WMS използват различни модели на работните потоци, за да представят тези логически връзки и взаимна обусловеност на дейностите, необходими за протичането на съответния бизнес-процес. Често използвани са модели, базирани на различни видове графи (напр. мрежи на Петри), в които върховете (позициите) представят отделни дейности, а графовата структура се използва за управление на процеса, разпределяне на активности, планиране и координиране на изпълнението на задачите, и др.

Броят на софтуерните продукти, базирани на WFM-технологиите, постоянно нараства. Въпреки това, до момента, няма стандарти, които да позволяват съвместна работа на WFM-

[1] Workflow Management System

продукти, представящи различни процеси. Основаната през 1993 г. коалиция WfMC1 обединява проектанти, консултанти, аналитици, университетски и изследователски групи с интереси в областта на потоковите структури и BPM. Дело на WfMC е създаването и разпространението на стандарти в изследваната област с цел осъществяване на комуникация между иначе несъвместими системи за управление на потоци и интегрирането им с различни ИТ-услуги.

В работата са представени модули на софтуерна система, със създаването на която се цели решаване на някои от посочените проблеми в областта на е-обучението (разглеждано като бизнес-процес). Проектирането на системата е предшествано от теоретично проучване на по-общия въпрос за приложението на технологиите за моделиране и управление на бизнес-процеси в случая на проектиране и реализация на виртуални учебни курсове.

4.3.1 Анализ на състоянието на проблема

Съвременните системи за електронно обучение (CeO) са фокусирани върху осигуряване на отделни виртуални учебни дейности, оставяйки на заден план процеса на обучение (в рамките на който се провеждат) и свързаните с тях изисквания и ограничения от логически и методически характер. Иначе казано, съвременните CeO не са подходящи за моделиране на разнообразните форми на обучение, наблюдавани в практиката. Нещо повече, дори опитите за представяне на адаптивни виртуални учебни курсове в подобни системи среща определени трудности (вж. [78, 79]).

Удовлетворяването на подобни ***изисквания за адаптивност и приспособимост на виртуалния учебен процес*** (едновременно със спазване на определени методически и ресурсни ограничения), обаче, поставя на дневен ред редица проблеми, свързани със създаването на адекватни средства за неговото моделиране, изпълнение в реално време и управление (вкл. адаптивни потребителски интерфейси, средства за автоматизация и др.). В

[1] Workflow Management Coalition

последните години, за решаване на посочените проблеми опити се привличат технологии и средства за WMP.

Така например, в [80, 81] се анализира необходимостта от създаване на средства за по-гъвкаво описание на учебните курсове и на начините за протичане на учебният процес. Като възможно решение е предложено разширение на авторската система за електронно обучение *Flex-eL* с интегриране на *FlowMake* (средство за моделиране управление на работни потоци).

В [82] авторите се фокусират върху конкретен модел на процеси, който прилагат при описание на университетски учебни курсове. Въведеният [83] графичен език за описание на процеси отговаря по спецификация на стандарта на коалиция WfMC [84], като използва само основният набор от шаблони, въведен там, а именно: Sequence, XOR-Split, XOR-Join, AND-Split и AND-Join.

По-късно в [85] е предложен нов подход за проектиране на системи за е-обучение, базирани на [86], описвайки е-обучението като бизнес-процес, и използвайки инструменти за моделиране на бизнес-процеси. В разработката се използват концепции за моделиране на бизнес-процеси, заимствани от UML1 и Rational Unified Process [87, 88], като се предлага методика за описание и анализ на системите за е-обучение, която да подпомогне тяхното следващо разработване.

Друго цялостно решение - от моделиране до изпълнение в реално време, е представено в [89]. Прилагат се корпоративни инструменти, сред които и *Microsoft BizTalk* като средство за виртуално изпълнение на бизнес-процеси. За описание на процесите се използва авторска графична среда, която свежда съответните модели до представяне на езика *XLANG* (използван в *BizTalk*).

В [90] се предлага - задачата за следене и контрол на изпълнението на виртуалните учебни дейности, да се решава с използване на обектно-ориентиран метод за моделиране на

[1] Unified Modeling Language

учебните потоци и изграждане на архитектура на три нива. Практическа реализация на приложение на базата на предложените метод и архитектура не е описана.

В общи линии, ***основните подходи***, използвани за решаване на проблема за приложение на BPM в е-обучението, са:

- създаване на специализирана система за е-обучение с вградени средства за моделиране на работни потоци (‘от нулата’);
- изграждане на CeO на базата на съществуваща WMS – с използване на създадени средства за моделиране на бизнес-процеси (‘нова CeO над съществуваща WMS);
- развитие на функционалността на известна CeO със средства за моделиране и управление на процеси на е-обучение (‘специализирана нова WMS над съществуваща CeO’);
- интегриране на съществуваща CeO и конкретна BMS (специализирана за управление на бизнес-процеси в друга област).

Някои от присъщите, в по-малка или по-голяма степен, недостатъци на посочените подходи са: моделът на е-обучение може да се опише с ‘готови’ средства, но не може да бъде директно реализиран (стартиран, управляван и изпълнен за съответните потребители в CeO – обучавани, преподаватели, настойници и др.); за моделиране на учебните процеси се използват авторски среди, които в повечето случаи не са съвместими със съответната WMS; редица известни и широко използвани в практиката CeO и WMS са затворени (за неоторизирани разширения); проектирането и създаването на сложни софтуерни системи ‘ad-hoc’ е трудоемка и скъпо струваща дейност, и др. В резултат, повечето реализации трудно намират приложение, тъй като не са съвместими с масово разпространените и използвани ‘стандартни’ системи за е-обучение и управление на ‘линейни’ виртуални курсове.

4.3.2 Същност на разработката

В основата на настоящата разработка е общ подход, същността на който се заключава в изследване на възможността за интегриране в *CeO* с отворен код (*широко разпространена*) на елементи на WMS (с *широко предназначение*). Съществено е да се отбележи – какво се разбира тук под ‘WMS с широко предназначение’. В случая става въпрос за WMS, която (за

разлика, например, от изброените по горе основни подходи) притежава средства за моделиране и управление не само на бизнес-процеси, но и на компонентите на самата потокова структура (дейности, подпроцеси, ресурси, ограничителни условия и др.). Иначе казано, в подобни системи могат предварително да се моделират елементите на бизнес-средата (в случая – е-обучението), които се използват при дефиниране, изпълнение и управление на съответните процеси, след което системата започва да действа като специализирана (за целите на е-обучението) WMS.

Предложеният за реализация подход има няколко важни ***предимства***:

- преодолява синдрома ‘още една CeO’ за проектанти и потребители (интерфейса и функционалността на избраната популярна CeO се запазват);
- позволява реализация на ‘бърз прототип’ на CeO с управление на процесите (не е необходимо да се проектира и реализира ‘от нула’ – използват се възможностите на избраната популярна CeO, като специфичните компоненти на е-обучението се моделират със средства на WMS ‘с широко предназначение’);
- разширява функционалността на конкретната CeO в посока на използване на технологии и средства за моделиране и управление на виртуални учебни процеси;
- с развитието на избраната CeO (напр. с включване на нови виртуални учебни дейности и събития) автоматично се обогатява функционалността на системата като цяло (в модела на процеса може да се включват и успешно управляват и новите дейности);
- възможно е мултиплициране на резултатите (избор на друга CeO като основа), и др.

Ще отбележим, че същият подход успешно е използван и за моделиране на адаптивно електронно тестване, като на мястото на Moodle е авторска система за поддръжка на т. нар. акумулативни тестови въпроси [91].

4.3.3 Проектиране и реализация

Конкретната разработка е базирана на *Moodle*1 (вер. 1.9.14) и на авторската система за моделиране и управление на работни потоци *ILNET* с широко предназначение в различни бизнес-среди (детайлно представяне на *ILNET* ще бъде направено в следващи публикации; за някои нейни характерни особености вж. [91]).

Създаденият на тази основа ‘бърз прототип’ на WMS за е-обучение, наречен *BEST2 3.M* (за друг подход в реализацията – *BEST 2.0* вж. [92]) предоставя инструменти за моделиране на конкретни е-курсове под формата на сложни (в общия случай нелинейни и адаптивни) потокови структури от учебни дейности (типични за *Moodle*) и управление на съответното обучението за множество от студенти.

Основните разширения и промени в *Moodle***,** които се оказаха необходими за интегриране на елементи на *ILNET*, са:

- създаване на услуга за достъп до данни (съхранени в Moodle-средата), а именно - списъци от всички създадени тестове и документи, резултати от оценени тестове, и др.;
- създаване на нов тип учебен курс, чиято структура да се моделира със средства на технология на BPM (на обучаваните е осигурен достъп до всички предстоящи и приключили дейности по привичен начин – нелинейната потокова структура остава ‘зад кадър’ [78]);
- модификация на механизма за визуализация на елементите на курса (с разрешаване на достъп само до определени дейности, документи и тестове - не изобщо), и в зависимост от текущото състояние и прогрес на конкретния обучаван;

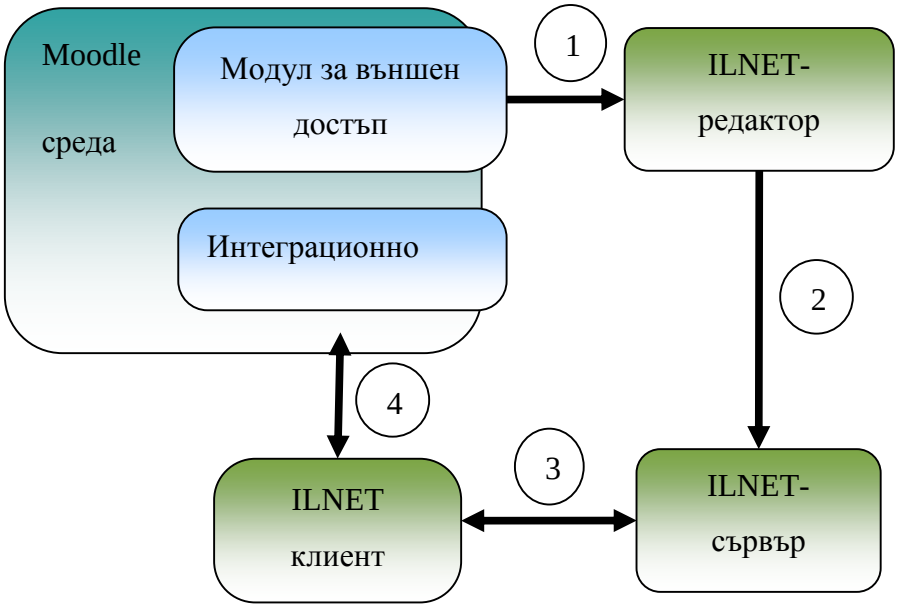
---

[1] Изборът се основава на обстоятелството, че голяма част от проектантите и потребителите на е-обучение са част от т. нар. Moodle-общество.

[2] *BEST* – **B**ulgarian **E**ducational Si**t**e

- добавяне на модул за известяване (обработване на събития), позволяващ осъществяване на двупосочна интерактивна връзка между системата за моделиране и управление на процеси и *Moodle*.

Основните компоненти на системата за моделиране и управление на виртуални курсове *BEST 3.M* са: *разширена Moodle-среда* (включително ILNET-клиент), *ILNET-редактор* и *ILNET-сървър*. Фигура **30** дава обща представа за комуникацията между отделните компоненти на *BEST 3.M*.



Фигура 30: Общ изглед на архитектурата на BEST 3.M

За създаване на виртуален курс, съпроводен с учебни материали, се използват стандартни средства на *Moodle*. Моделът на учебния процес, свързан със съответния курс, се създава чрез ILNET-редактора, в който могат да се използват елементите на курса (учебни материали, тестове и др.). За целта се използва функционалното разширение *Модул за външен достъп*, който позволява на външно (за *Moodle*) приложение да получи достъп до данните, асоциирани с дадения курс. Модулът предоставя поисканата информация под формата на XML-документ, като по този начин реализира универсален програмен интерфейс за извличане на данни, разположени в *Moodle* (връзка, означена с 1. на фигура **30**).

След внасянето на данните, потребителят на *BEST 3.M* има възможност, с използване на средствата за визуално моделиране на ILNET-редактора, да моделира учебния процес, свързан с изучавания курс като потокова структура, използвайки вече асоциираните учебни материали,, дейности, тестове и др. така създаденият модел на процес на е-обучение се подготвя за изпълнение като се зарежда в ILNET-сървъра (връзка 2. на фигура **30**). На този етап моделирането на учебния процес е приключило, и обучението по съответния учебен курс (в случая – виртуална интерпретация на така моделирания учебен процес) може да стартира.

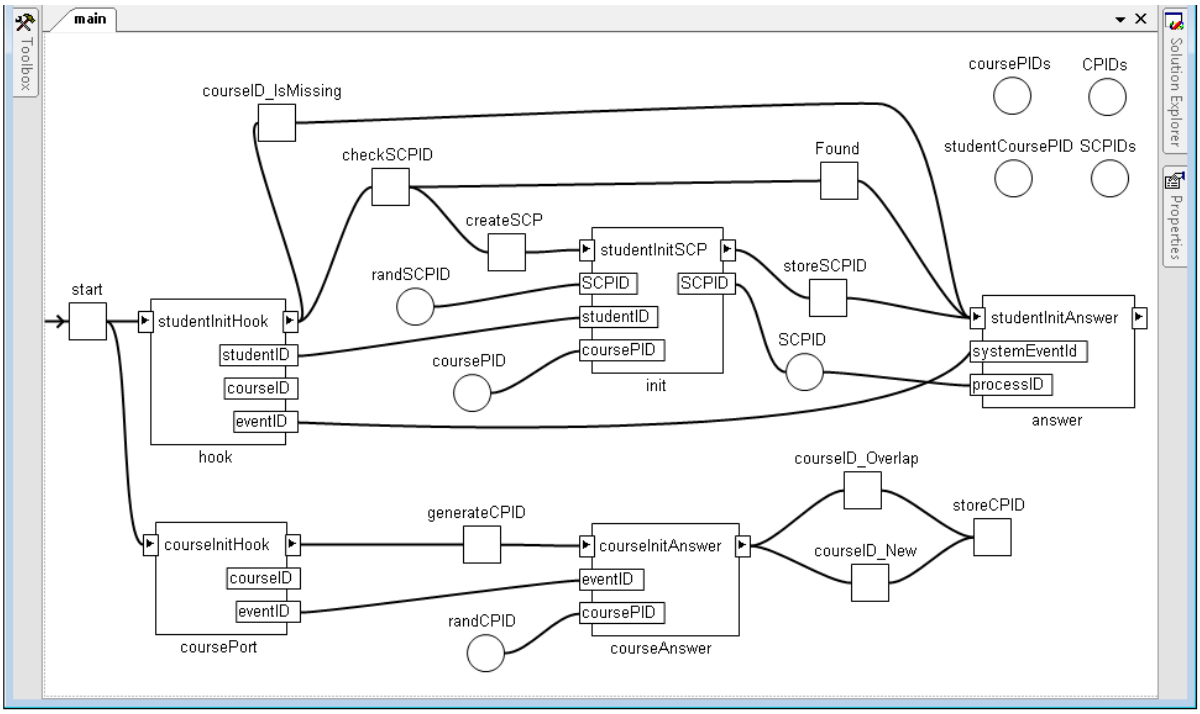
За осъществяване на двустранната комуникация между ILNET-сървъра и средата *Moodle* (връзки 3. и 4. на фигура **30**) отговарят модулите *ILNET-клиент* и *Интеграционно разширение*. ILNET-клиентът, осъществяващ непосредствената комуникация с ILNET-сървъра е изграден на основата на SOAP-библиотека, предоставена в средата *Moodle*. Интеграционното разширение има за цел да предаде контрола върху управлението на учебния поток от средата *Moodle* към системата за изпълнение и управление на адаптивни курсове (реализирано под формата на Plugin).

Управлението и изпълнението на моделираните като работни потоци виртуални курсове се извършва от системата за управление на процеси *ILNET*, която осигурява:

- графичен редактор за моделиране на процеси;
- компилиране и изпълнение на моделираните процеси;
- контрол на различните инстанции на процеси (вкл. създаване, активиране, прекъсване, приключване и др.);
- управление и рутиране на съобщения;
- общи дейности по контрол, администриране, проверка и др.

На фигура **31** е представен модел на помощен процес, необходим за осъществяването на интеграцията между моделираните в системата виртуални курсове и *Moodle*. Процесът управлява регистриране на (адаптивни в общия случай) курсове в един централизиран списък и генериране на уникален идентификатор на дейности в случая, когато обучаван стартира обучението си по даден курс. Идентификаторът се използва за адресация на съобщенията, касаещи съответния обучаван, и свързани със стартиралото обучение

(например - кои учебни дейности, материали и тестове предстоят за изпълнение и/или разглеждане).



Фигура 31: Описание на процеса ‘създаване на инстанция на курс’

В отделните етапи на разработката, модулите на *BEST 3.M* са тествани първоначално с контролни данни, а по-късно - и с реални.

Перспективно изследователско направление за развитие на системата е в посока на създаване на методи, софтуерни средства и дружелюбен потребителски интерфейс за моделиране, изучаване и управление на други аспекти от процеса на обучение (например автоматизирана оценка на качеството, методика на е-обучението, провеждане на адаптивни тестове, и др.).

ЗАКЛЮЧЕНИЕ

В дисертационния труд е направен преглед на типовете системи за семантичен анализ и информационно търсене. Разгледани са съвременните направления в разработката на системи за семантичен анализ и информационно търсене. Изводите от направения обзор са използвани за създаване на концепция на модел за описание на процеси за информационно търсене и семантичен анализ. Изведена е архитектура, удовлетворяваща поставените изисквания. Направена е реализация на архитектурата. Моделът е приложен успешно за разработка на система за извличане на контактна информация от документи на български език, методи за автоматизирано генериране на метаданни за учебни обекти и модул за моделиране и управление на виртуални адаптивни курсове, базирани на moodle.

Поставените в увода задачи са постигнати.

Основните приноси на дисертационния труд са:

1. Анализирани са съвременните концепции за информационно търсене и семантичен анализ. Изследвани са популярни съвременни системи и възможностите им са класифицирани;
2. Изготвена е концепция на модел за описание на процеси за информационно търсене и семантичен анализ;
3. Изготвена е архитектура и е реализиран прототип на система за управление на работни процеси;
4. Разработени са система за извличане на контактна информация от документи на български език, методи за автоматизирано генериране на метаданни за учебни обекти и модул за моделиране и управление на виртуални адаптивни курсове, базирани на moodle.

Връзките между приносите, целите, задачите, мястото на описание в дисертационния труд и направените публикации са описани в следната таблица:

Принос	Вид принос	Цел	Задачи	Параграф	Публикация
1	научно-приложен	2	2	(2.2), (2.3)	1,2,3,5
2	научно-приложен	3	3.1	(3.1)	6,7,9
3	приложен	3	3.2, 3.3	(3.2), (3.3)	
4	научно-приложен	4	4	(4.1), (4.2), (4.3)	4,7,8,9

Перспективи

Предложеният модел може да се развива в следните насоки:

1. Имплементиране на изграждащи блокове и шаблони за изпълнение, които да позволят моделирането на известни стандарти за описание на процеси/потоци;
2. Интеграцията на системата в други практически приложения (персонализирано и адаптивно обучение [75, 76], е-бизнес [60], административен контрол [61], и др.);
3. Реализация на уеб-базиран клиент на системата;
4. Разширяване на архитектурата на сървърният компонент с възможност работа в клъстер.

Апробация

Резултати, получени от изследването, са използвани в следните национални проекти:

1. D002 308, Автоматизирано генериране на метаданни за спецификации и стандарти на Е-документи, 2008-2011
2. МИ-203, Моделиране на учебните процеси и управление на проекти за е-обучение, 2007-2010

Част от резултатите, получени в дисертационния труд са докладвани на следните национални и международни конференции:

1. 3-та научна конференция за студенти, докторанти и млади научни работници, 25.4.2009, Пловдив, България.

2. International Conference on Information Research and Applications (i.Tech 2009), 02.09.-05.09.2009, гр. Мадрид, Испания
3. 10th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing 2009, June 18-19, 2009, Rousse, Bulgaria.
4. Национална конференция "Образованието в информационното общество", АРИО, 26-27 май 2011, Пловдив, България

Публикации по дисертационния труд

1. Totkov G., D. Blagoev, *Regular Expressions Builder and Parser for Unicode Syste*ms, Information Technologies and Control, Year III, №1/2005, 40-45.
2. Тотков Г., Д. Благоев, В. Ангелова, *За озвучаването на компютърен български текст*, Национална научна конференция „10 години катедра Компютърни системи към ТУ – София, филиал Пловдив“, ноември 2003, 119-131.
3. Blagoev D., G. Totkov, *Visual Parser Builder*, Proc. of the International Conference 'Recent Advances in Natural Language Processing' RANLP'05, 21-23 September 2005, Borovetz, 112-116.
4. Благоев Д., *Извличане на контактна информация от документи на български език*, в Сборник доклади на 3-та научна конференция за студенти, докторанти и млади научни работници, 25.4.2009, Пловдив, 339-343.
5. Blagoev D., G. Totkov, M. Staneva, Kr. Ivanova, Kr. Markov, *Indirect Spatial Data Extraction from Web Documents*, International Book Series Information Science & Computing, New Trends in Intelligent Technology, Number 14, ITHEA, 2009, 89-100.
6. Indzhov Hr., D. Blagoev, G. Totkov, *Executable Petri Nets: Towards Modelling and Management of obu Processes*, ACM International Conference Proceeding Series; Vol. xxx, Proc. of the 10th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing 2009, Rousse, Bulgaria, June 18-19, 2009. IIIA.12.1 IIIA.12.6.
7. Благоев Д., Хр. Инджов, Г. Тотков, *Моделиране и управление на виртуални адаптивни курсове, базирани на Moodle*, Сборник материали на конференция на НБУ, В. Търново, 30.9.2010 г. (в печат).
8. Христина Костадинова, Георги Тотков, Димитър Благоев, *Автоматизирано генериране на метаданни за учебни обекти*, Сборник доклади на Национална

конференция "Образованието в информационното общество" Пловдив, АРИО, 26-27 май 2011, стр. 044-052

9. Г. Тотков и кол., *Е-обучението в информационното общество: технологии, модели, системи, достъпност и качество*, ISBN 978-954-423-651-9, българска, първо издание, Университетско издателство „Пасий Хилендарски" – гр. Пловдив, 2010г.

ДЕКЛАРАЦИЯ

за оригиналност

по чл. 27, ал. 2 от ППЗРАСРБ

Получените резултати, описани в заключението на дисертационния труд „Системи за семантичен анализ и информационно търсене” са оригинални.

20.02.2012 г. Подпис:

гр. Пловдив /Димитър Благоев Благоев/

БИБЛИОГРАФИЯ

1. Дубинский А. Г. *Некоторые вопросы применения векторной модели представления документов в информационном поиске* // Управляющие системы и машины. - 2001. - №4. - С. 77-83.
2. Когаловский М. Р. *Перспективные технологии информационных систем*. – М.: ДМК Пресс; М.: Компания АйТи, 2003. – 288 с.
3. Кормен Т., Ч. Лейзерсон, *Алгоритмы: построение и анализ*, Р.Ривест МЦНМО, 2000 http://www.ozon.ru/?context=detail&id=114200.
4. Кураленок И.Е., Некрестьянов И.С. *Автоматическая классификация документов на основе латентно-семантического анализа* // Труды первой всероссийской научно-методической конференции “Электронные библиотеки: перспективные методы и технологии, электронные коллекции”. – СПб., 1999. - С. 89-96.
5. Некрестьянов И., Пантелеева Н. *Системы текстового поиска для Веб* // Программирование. – 2002. – N4.
6. Некрестьянов И.С. *Тематико-ориентированные методы информационного поиска*: Диссертационная работа к.т.н.: 05.13.11 / Санкт-Петербургский государственный университет – СПб., 2000. – 80 с.
7. Сэлтон Г. *Автоматическая обработка, хранение и поиск информации*: Пер. с англ. / Под ред. А.И. Китова. – М.: Советское радио, 1973. – 560 с.
8. *A Statistical Interpretation of Term Specificity and Its Application in Retrieval,* Karen Sparck Jones Journal of Documentation, 1972
9. Alpaydin E. *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, October 2004.
10. Appelt D. E., D. J. Israel. *Introduction to Information Extraction Technology*, 1999. A tutorial prepared for IJCAI-99.
11. Baeza-Yates R., Ribeiro-Neto B. *Modern Information Retrieval*. ACM Press, 1999.

12. Baker L. D., Andrew Kachites McCallum. *Distributional clustering of words for text classification*. In Proc. of the SIGIR'98, pages 96-103, 1998.

13. Baumgartner R., S. Flesca, G. Gottlob. *Visual Web Information Extraction with Lixto*. In The VLDB Journal, pages 119–128, 2001.

14. Bergman M. *The Deep Web: Surfacing Hidden Value*. Journal of Electronic Publishing, 7, Aug. 2001.

15. Brown E.W. *Execution Perfomance Issue in Full-Text Information Retrieval*. Dissertation. University of Massachusetts. Departament of Computer Science. February 1996.

16. Califf M. E., R. J. Mooney. *Relational Learning of Pattern-Match Rules for Information Extraction*. In Working Notes of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing, pages 6–11, Menlo Park, CA, 1998. AAAI Press.

17. Califf M. E.. *Relational Learning Techniques for Natural Language Extraction*. Technical Report AI98-276, 1, 1998.

18. Callan J.. *Learning while filtering documents*. In Proc. of SIGIR'98, pages 224-231, Melbourne, Australia, 1998.

19. Cardie C. *Empirical Methods in Information Extraction*. AI Magazine, 18(4):65–80, 1997.

20. Cowie J., W. Lehnert. *Information Extraction*. Commun. ACM, 39(1):80–91, 1996.

21. Craswell N., Bailey P. *Is it fair to evaluate Web systems using TREC ad hoc methods?* In Proc. of the SIGIR'99, 1999.

22. Cullum J. and R. Willougby. *Real rectangular matrix. In Lanczos algorithms for large symmetric eigenvalue computations*. Brikhauser, Boston, 1985.

23. Deerwester S., S.T. Dumais, G.W. Furnas, T.K. Landauer, R. Harshman. *Indexing by Latent Semantic Analysis* JASIS, 1990 http://citeseer.nj.nec.com/ deerwester90indexing. html.

24. Dietr Merkl. L*essons learned in text document classification*. In Proc. of the Workshop on Self-Organizing Maps (WSOM'97), pages 316-321, Helsinki, Finland, June 1997.

25. Dietr Merkl. *Text data mining*. In A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Language as Text. Marcel Dekker, New York, 1998.

26. *Dublin Core Metadata Element Set Reference Description*, Version 1.1, 1999-07-02. http:/purl.org/dc/documents/proposed_recommendations/pr-dces-19990702.html.

27. Dumais S. *Latent semantic indexing: TREC-3 report*. In Proc. of the Third Text REtrieval Conference, 1995.

28. Eckart C., G. Young, *The approximation of one matrix by another of lower rank* Psychometrika, 1936.

29. Eikvil L.. *Information Extraction from World Wide Web* - A Survey. Technical Report 945, Norweigan Computing Center, 1999.

30. Etzioni O., M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S.Weld, A. Yates. *Unsupervised Named-Entity Extraction from the Web*: An Experimental Study. Artificial Intelligence, (1):91–134, 2005.

31. *Examples of PAT applied to the Oxford English Dictionary* Gonnet G. University of Waterloo, 1987

32. Faloutsos C., S. Christodoulakis. *Description and performance analysis of signature file methods*, ACM TOIS 1987

33. FAST PMC - *The Pattern Matching Chip* http://www.fast.no/product/fastpmc.html http://www.idi.ntnu.no/grupper/KS-grp/microarray/slides/heggebo.pdf

34. Finkelstein L., Gabrilovich E., Matias Y., Rivlin E., Solan Z., Wolfman G., Ruppin E. *Placing search in context: the concept revisited*. In Proc. of the WWW10, pp. 406-414, 2001.

35. Foltz P. W., *Using latent semantic indexing for information filtering*. In Proc. of the ACM Conference on Office Information Systems (COIS), pages 40-47, 1990.

36. Freitag D. *Machine Learning for Information Extraction in Informal Domains*. PhD thesis, Carnegie Mellon University, 1998.

37. Furnas G.W., S. Deerwester, S.T. Dumais, T.K. Landauer, R. A. Harshman, L.A. Streeter, and K.E. Lochbaum *Information retrieval using a Singular Value Decomposition Model of Latent Semantic Structure* ACM SIGIR, 1988

38. Furnkranz J., P. A. Flach. *ROC 'n' Rule Learning - Towards a Better Understanding of Covering Algorithms*. Machine Learning, 58(1):39–77, 2005.

39. Glimpse, Webglimpse, *Unix-based search software*, http://webglimpse.org

40. Gottlob G., C. Koch, R. Baumgartner, M. Herzog, S. Flesca. *The Lixto Data Extraction Project - Back and Forth between Theory and Practice*. In Proceedings of the Symposium on Principles of Database Systems (PODS-04), 2004.

41. Harman D. *Latent semantic indexing (LSI) and TREC-2*. In Proc. of the Second Text REtrieval Conference, 1994.

42. Hatzivassiloglou V., L. Gravano, A. Maganti. *An investigation of linguistic features and clustering algorithms for topical document clustering*. In Proc. of the SIGIR'2000, 2000.

43. Hsu C.-N. *Initial Results on Wrapping Semistructured Web Pages with Finite-State Transducers and Contextual Rules*. In Workshop on AI and Information Integration, in conjunction with the 15'th National Conference on Artificial Intelligence (AAAI-98), pages 66–73, 1998.

44. Hsu C.-N., M.-T. Dung. *Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web*. Information Systems, 23(8):521–538, 1998.

45. Jansen B. J., Spink A., Saracevic T. *Real life, real users, and real needs: a study and analysis of user queries on the web.* Information Processing and Management, 36(2):207-227, 2000.

46. Joachims T., *A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization.* In Proc. of the International Conference on Machine Learning (ICML), 1997.

47. Joyce T., R.M. Needham, *The Thesaurus Approach to Information Retrieval American Documentation*, 1958

48. Kleinberg J. M. *Authoritative sources in a hyperlinked environment*. Journal of the ACM, 46(5):604-632, 1999.

49. Kushmerick N. *Wrapper Induction: Efficiency and Expressiveness*. Artificial Intelligence, 118(1-2):15–68, 2000.

50. Kushmerick N., D. S. Weld, R. B. Doorenbos. *Wrapper Induction for Information Extraction*. In Intl. Joint Conference on Artificial Intelligence (IJCAI), pages 729–737, 1997.

51. Landauer T., P. Foltz, and D. Laham. *An introduction to latent semantic analysys*. In Discourse Processes, volume 25, pages 259-284.

52. Lewis D. and M. Ringuette. *A comparison of two learning algorithms for text categorisation*. In Proc. of the Third Annual Symposium on Document Analysys and Information Retrieval, pages 81-93, 1994.

53. Blagoev D., G. Totkov, Visual Parser Builder, RANLP'05, Borovetz, 112-116.

54. Korečko S., B. Sobota, C. Szabó, Performance analysis of processes by automated simulation of Coloured Petri nets, 10th International Conference on Intelligent Systems Design and Applications (ISDA), 2010, Cairo, Nov. 29 2010-Dec. 1 2010, p176-181.

55. Totkov G., D. Blagoev, V. Angelova, SLOG 1.0: Bulgarian text to speech transformation system, International Joint Conf. on Computer, Information, Systems Sciences and Engineering (CIS2E'05), Bridgeport (CT), Dec 10-20, 2005.

56. T. Dufresne, J. Martin. *Process Modelling for E-Business*, INFS 770 Methods for Information Systems Engineering: Knowledge Management and E-Business, Spring 2003.

57. S. Williams. *Business Process Modelling Improves Administrative Control*, In: Automation, December, 1967, pp. 44 - 50.

58. D. Blagoev, G. Totkov. *Regular Expressions Builder and Parser for Unicode Systems*, Information Technologies and Control, Year III, №1/2005, 40-45.

59. G. Totkov, D. Blagoev, V. Angelova. *SLOG 1.0: Bulgarian text to speech transformation system*, International Joint Conf. on Computer, Information, Systems Sciences and Engineering (CIS2E'05), Bridgeport (CT), Dec 10-20, 2005.

60. Hr. Indzhov, D. Blagoev, G. Totkov. *Executable Petri Nets: Towards Modelling and Management of e-Learning Processes*, ACM International Conference Proceeding Series; Vol. 375, Proc. of the 10th Int. Conf. on Computer Systems and Technologies CompSysTech'09, Rousse, Bulgaria, June 18th - 19th, 2009, pp. IIIA.12.1 - IIIA.12.6.

61. Hr. Indzhov, R. Doneva, D. Blagoev, G. Totkov. Modelling and Carrying on e-Learning Processes with a Workflow Management Engine (this volume).

62. D. Sampson, C. Karagiannidis, C. Kinshuk. *Personalised Learning: Educational, Technological and Standardisation Perspective*, Interactive Educational Multimedia, number 4 (April 2002), pp. 24-39.

63. D. Lamboudis, A. Economides, C. Papas. *Applying Work Flow Reference Model in Adaptive Learning*, In G. Richards (Ed.), Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education, 2005, pp. 856-859.

64. *Workflow*. Wikipedia, http://en.wikipedia.org/wiki/Workflow.

65. Totkov G., E. Somova, M. Sokolova, *Modelling of e-Learning Processes: an Approach Used in Plovdiv e-University*, Int. Conf. on Computer Systems and Technologies (e-learning), CompSysTech'04, 17-18 June 2004, Rousse, IV.12.1 – IV.12.6.

66. *LAMS Users Guide v1.0.1*, http://lamsfoundation.org/, 2005.

67. Lin J., Ch. Ho, W. Sadiq, M. E. Orlowska, *On Workflow Enabled e-Learning Services*. icalt, pp.0349, Second IEEE International Conference on Advanced Learning Technologies (ICALT'01), 2001.

68. Lin, J., Ch. Ho, M. E. Orlowska, W. Sadiq, Using workflow technology to manage flexible e-learning services. Educational Technology and Society. 2002, 5 4: 1-10.

69. Mangan, P., S. Sadiq, *On Building Workflow Models for Flexible Processes*. In Proc. Thirteenth Australasian Database Conference (ADC'2002), Melbourne, Australia. CRPIT, 5. Zhou, X., Ed. ACS. 103-109.

70. Sadiq W., M. E. Orlowska, *On Correctness Issues in Conceptual Modeling of Workflows*. In Proceedings of the 5th European Conference on Information Systems ECIS '97, Cork, Ireland, June 19-21, 1997.

71. Hollingsworth D., *The Workflow Reference Mode.*, The Workflow Management Coalition Specification, Document No. TC00-1003, Issue 1.1, 19-Jan-1995, http://www.wfmc.org/standards/ docs/tc003v11.pdf

72. P. Avgeriou, S. Retalis, and N. Papaspyrou,. *Modeling learning technology systems as business systems*. Software and Systems Modeling, 2(2):120--133, July 2003.

73. IEEE Learning Technology Standards Committee: *Draft standard for Learning Technology Systems Architecture* (LTSA), Draft 9, November 2001

74. Kruchten P., *The Rational Unified Process*. *An introduction*. Addison-Wesley, Reading, MA, 1999

75. *The Rational Unified Process*, v. 2001.03.00.23, Rational Software Corporation, part of the Rational Solutions for Windows suite, 2000.

76. Cesarini, M., M. Monga, R. Tedesco, *Carrying on the e-Learning Process with a Workflow Management Engine*. in Proceedings of the 2004 ACM Symposium on Applied Computing (Nicosia, Cyprus, March 14-17, 2004). SAC'04. ACM, New York, NY, 940-945. DOI= http://doi.acm.org/10.1145/ 967900.968091.

77. Lin, J., *A Method for Modeling Service Management of e-Learning*. WSEAS Trans. Info. Sci. and App. 5, 7 (Jul. 2008), 1251-1261.

78. Indzhov Hr., D. Blagoev, G. Totkov, *Executable Petri Nets: Towards Modelling and Management of e-Learning Processes*, ACM International Conference Proceeding Series; Vol. 375, Proc. of the 10th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing 2009, Rousse, Bulgaria, June 18-19, 2009. IIIA.12.1-IIIA.12.6.

79. Тотков Г., Д. Денев, Р. Донева, *BEST: български образователен портал за е-проекти (с демонстрации)*, Втора национална конференция с международно участие по електронно обучение във висшето образование, 14-17 септември 2006, Китен, 69-79.

80. Благоев Д., *Извличане на контактна информация от документи на български език*, в Сборник доклади на 3-та научна конференция за студенти, докторанти и млади научни работници, 25.4.2009, Пловдив, 339-343.

81. Костадинова Хр., Г. Тотков, М. Райкова, *Към автоматизирано генериране на тестове по Блум*, 40-та Юбилейна конференция на СМБ, Боровец, 5 – 9 април 2011 г., 413-422.

82. Bauer M., R. Maier, *Metadata Generation for Learning Objects: An Experimental Comparison of Automatic and Collaborative Solutions*. E-Learning, 2010, pages181-195.

83. Blagoev D., G. Totkov, M. Staneva, Kr. Ivanova, Kr. Markov, *Indirect Spatial Data Extraction from Web Documents*, International Book Series Information Science & Computing, New Trends in Intelligent Technology, Number 14, ITHEA, 2009, 89-100.

84. Cardinaels K., M. Meire, E. Duval, *Automating Metadata Generation: the Simple Indexing Interface*. In Proceedings of the 14th international conference on World Wide Web, ACM Press, 2005. 548-556.

85. Duval, E., F. Neven, *Reusable Learning Objects: a Survey of LOM-based Repositories*. MULTIMEDIA ’02 Proceedings of the tenth ACM international conference on Multimedia, 2002.

86. *Final LOM Draft Standard*, http://ltsc.ieee.org/wg12/20020612-Final-LOM-Draft.html.

87. Greenberg, J., K. Spurgin, A. Crystal, *Functionalities for Automatic-Metadata Generation Applications: A Survey of Metadata Experts’ Opinions*. International Journal of Metadata, Semantics, and Ontologies. 2006, *Vol. 1, No. 1, 2006* 3 http://www.inderscience.com/storage/f121932106117458.pdf.

88. Handschuh S., S. Staab, F. Ciravegna, *S-CREAM - Semi-automatic CREAtion of Metadata*. EKAW '02 Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web 2002.

89. McCalla, G., C. Brooks, *Towards Flexible Learning Object Metadata*. Engineering Education and Lifelong Learning, (16):50–63, 2006.

90. Manber U., G. Myers, *Suffix Arrays: A New Method for On-line String Searches* 1st ACM-SIAM Symposium on Discrete Algorithms, 1990

91. Maron M.E., Kuhns J.L. *On relevance, probabilistic indexing and information retrieval.* Jornal of the ACM, No. 7, 1960, pp. 216-244.

92. Muslea I., S. Minton, C. Knoblock. STALKER: *Learning Extraction Rules for Semistructured, Web-based Information Sources*. In Proceedings of AAAI-98 Workshop on AI and Information Integration. AAAI Press, 1998.

93. *Natural Language Information Retrieva*l, Tomek Strzalkowski (ed.) Kluwer Academic Publishers, 1999

94. *Open Text Corporation* http://www.opentext.com

95. Papka R., J. Allan. *Document classification using multiword features*. In Proc. of the ACM International Conference on Information and Knowledge Management (CIKM-98), pages 124-131, New York, November 1998.

96. Riloff E. *Automatically Constructing a Dictionary for Information Extraction Tasks*. In Proc. of AAAI-93, pages 811–816, 1993.

97. Riloff E. *Automatically Generating Extraction Patterns from Untagged Text*. In AAAI/IAAI, Vol. 2, pages 1044–1049, 1996.

98. Riloff E., J. Shoen. *Automatically Acquiring Conceptual Patterns without an Automated Corpus*. In Proceedings of the Third Workshop on Very Large Corpora, pages 148–161, 1995.

99. Robertson S.E., S. Jones K., *Relevance Weighting of Search Terms* JASIS, 1976

100. Sokolova M., G. Totkov, *Accumulative Question Types in e-Learning Environment*, International Conference on Computer Systems and Technologies - CompSysTech'2007, IV.21-1 - IV.21-6.

101. Sokolova, M., G. Totkov, Extended IMS Specification for Accumulative Test System, ACM International Conference Proceeding Series; Vol. 374, Proc. of the 9th Int. Conf. on CompSysTech, Gabrovo, Bulgaria, June 12-13, 2008, V.14-1–V.14-6.

102. Salton G., Fox E., and Wu H. *Extended Boolean information retrieval*. Communications of the ACM, Vol. 26, No. 4, December 2001, pp. 35-43.

103. Scott A. Weiss, Simon Kasif, and Eric Brill. *Text classification in USENET newsgroups*: A progress report.

104. Sedgewick R., *Algorithms in C++*, Addison-Wesley, 1992

105. Shivakumar N., Garcia-Molina H. *Finding Near-Replicas of Documents on the Web*. In Proc. of the WebDB'99, 1999.

106. Singhal A. *Modern Information Retrieval: A Brief Overview*. Data Enginering Bulletin, IEEE Computer Society, Vol. 24, No. 4, December 2001, pp. 35-43.

107. Soderland S. G.. *Learning Text Analysis Rules for Domain-Specific Natural Language Processing*. PhD thesis, 1997.

108. Soderland S. *Learning Information Extraction Rules for Semi-Structured and Free Text*. Machine Learning, 34(1-3):233–272, 1999.

109. Soderland S., D. Fisher, Jonathan Aseltine, and Wendy Lehnert. *CRYSTAL: Inducing a Conceptual Dictionary*. In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, pages 1314–1319, 1995.

110. Stata R., K. Bharat, F. Maghoul, T*he Term Vector Database: fast access to indexing terms for Web pages* WWW9, 2000 http://www9.org/w9cdrom/159/159.html.

111. Turtle H. R. *Inference Networks for Document Retrieval*. Dissertation. University of Massachusetts. Department of Computer and Information Science. February 1991.

112. Yang Y., J. Pederson. *Feature selection in statistical learning of text categorization*. In Proc. of the ICML'97, pages 412-420, 1997.

113. Warpechowski M., M. A. M. Souto, J. P. M. de Oliveira, Techniques for Metadata Retrieval of Learning Objects. Workshop on Applications of Semantic Web Technologies for e-Learning (SW-EL@AH'06), 2006.

114. Warpechowski M., P. de Oliveira et al., Adaptive Hypermedia in the AdaptWeb Environment In: First EAW. Eindhoven. In: AH 2004 Workshop Proceedings, 68 – 73.