СОФИЙСКИ УНИВЕРСИТЕТ "Св. КЛИМЕНТ ОХРИДСКИ"
ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА
КАТЕДРА ИНФОРМАЦИОННИ ТЕХНОЛОГИИ

ДИПЛОМНА РАБОТА

На Ангел Веселинов Великов
ФН М21318
Специалност Информатика
Специализация Био- и медицинска информатика

Тема: Класификация на административни документи в лекарската практика

Научен ръководител: Антоний Тодоров Попов
Консултант: Яко Пилософ

София
15.06.2005 г.

I. Увод

I.1. Лекарската практика в България

Здравната реформа в България промени установената безплатна здравна помощ. Целта на здравната реформа бе да се приведе здравното обслужване от социалистически към пазарен начин на работа. Така социалната задача на държавата се ограничи до частично заплащане на разноските на населението по здравеопазване. За съжаление този преход не премина успешно. Българското население не бе готово да заплаща за услугите по здравеопазване. Групите нуждаещи се от най-голяма помощ бяха същевременно групите разполагащи с най-малки финансови възможности.

Здравните работници и здравните заведения не бяха готови за предизвикателствата на пазарната икономика и конкуренцията. Те трябваше да оцеляват с намалено субсидиране и да търсят самостоятелно правилният начин на организация и работа за да се поддържа рентабилност. В съгласие с пазарните закони се появиха частни здравни заведения и практики, които да конкурират държавните, като по този начин привличаха с по-качественото си обслужване по-платежоспособните клиенти. Така пазарът действаше със своите еволюционни, но антисоциални закони. Задачата на държавата в тази ситуация бе да се компенсира тази разлика с подходящи субсидиране, законова основа, организация и контрол.

Законовата основа бе положена като начало на самата реформа под формата на Закона за Здравното Осигуряване (ЗЗО). Като основен орган по контролиране на организацията и качеството на здравното осигуряване остана Министерството на здравеопазването (МЗ). Бе създадена нова институция, наречена Национална Здравно-Осигурителна Каса (НЗОК). Тази институция получи задачата да разпределя средствата постъпили чрез Националният Осигурителен Институт (НОИ).

В момента НЗОК е йерархична структура, покриваща територията на цялата страна. На централната НЗОК са подчинени Регионални Здравно-Осигурителни Каси (РЗОК), които са реалните представители на тази институция.

Взаимоотношенията на НЗОК и здравните работници и здравните заведения се уреждат с Национални Рамкови Договори (НРД) с продължителност от една година. В тези договори са посочени задълженията на лекарите към техните пациенти и към НЗОК и съответния начин на изчисляване на субсидията, която те ще получават на месечна основа. Рамковите договори се подписват от Българския Лекарски Съюз (БЛС) и Съюза на Стоматолозите в България (ССБ) като представители на лекарската професионална общност.

Ще цитирам Чл.1. от НРД 2003 изясняващ предмета на договора:

„**Чл. 1**. (1) Предмет на Националния рамков договор (НРД) са правата и задълженията по оказването на медицинска и стоматологична помощ в рамките на чл. 55 от Закона за здравното осигуряване (ЗЗО) на:

1. Националната здравноосигурителна каса (НЗОК);
2. Българския лекарски съюз (БЛС) и Съюза на стоматолозите в България (ССБ);
3. изпълнителите на медицинска помощ (ИМП);
4. изпълнителите на стоматологична помощ (ИСП);
5. задължително здравноосигурените лица (ЗЗОЛ)."

В договора са определени правата и задълженията на страните, заплащаните от НЗОК услуги, изискванията към изпълнителите и самото изпълнение на здравните услуги, начините на контрол, изисквания документооборот, безплатните и частично изплащаните лекарства и изследвания, и санкционирането при установена нередност.

Като изпълнители на първична извънболнична медицинска помощ (ПИМП) са определени общопрактикуващите лекари (ОПЛ), познати също и под английското наименование general practitioner (GP). Те имат отговорността да се грижат за своите пациенти и нужните им документи и при нужда да ги пренасочват със съответен документ към изпълнители на специализирана извънболнична медицинска помощ (СИМП) или болнична помощ (БП). Всеки общопрактикуващ лекар има официален списък на записаните при него пациенти, като движенията на пациенти между лекарите се съобщава своевременно в съответната РЗОК и тя обновява списъците.

I.2. Административни взаимоотношения на лекарите

Лекарите, които нямат взаимоотношения с НЗОК обикновено работят срещу директно заплащане от пациента. Документооборотът, който те водят е съставен главно от документи свързани със здравното обслужване на клиента и са близки до лекарското ежедневие и професия.

За разлика от тях здравните работници и заведения получават основния си приход по своя договор с НЗОК. При тях услугата и заплащането са разделени логически, във времето и пространството. Това води до нуждата от документално обосноваване на извършената услуга. Така лекарската практика се обременява с воденето на множество документи, които се отчитат пред съответната РЗОК. Освен самите документи лекарите са задължени да предават и отчети и справки базирани на документите.

Най-често ползваните от лекарите първични документи са:

- Амбулаторен лист
- Месечен отчет на общопрактикуващ лекар (в електронен вид, заменя амбулаторните листи)
- Медицинско направление
- Медицинско направление за високоспециализирани дейности
- Направление за медико-диагностична дейност
- Рецептурна бланка
- Талон за ЛКК
- Направление за хоспитализация

С тези документи лекарите доказват извършената от тях дейност. Въз основа на тях те са длъжни да приготвят финансови отчетни документи, в които да агрегират информацията от първичните документи. Такива са:

- Спецификация за извършена медицинска дейност от лечебно заведение за първична извънболнична медицинска помощ
- Отчет за извършена медицинска дейност от лекар в лечебно заведение за първична извънболнична медицинска помощ
- Месечен отчет на общопрактикуващ лекар
- Опис към месечния отчет
- Спецификация за извършена медико-диагностична дейност
- Аналогични документи за стоматолозите, специалистите и болничните заведения

Тези документи се предават на съответните РЗОК ежемесечно. Въз основа на тях на лекарите се заплаща извършената дейност по определените в договора цени. За да няма забавяния и проблеми с изплащането на възнагражденията им лекарите трябва да са изключително внимателни и прецизни в попълването на тези документи, съпътстващите ги изчисления и класификации.

При попълването на документите се използват множество класификации, групирания и кодировки специфицирани от НЗОК. Те би трябвало добре да се познават

от този, който попълва документа за да не бъде допусната дори техническа грешка. Това обикновено е непосилно за един човек и попълването е съпроводено с чести справки с документите описващи нужните списъци.

Изчисленията в справките изготвяни от лекарите не са по сложни от сумиране и умножение, но се основават на голям брой първични документи, което прави вероятността за грешка много голяма.

Класификацията на документите се основава на това, което е попълнено в тях и е основа за много от изчисленията в справките. Правилата за класифициране обикновено са сложни и често имат различни специални случаи и изключения.

Някои видове дейности са ограничени като бройка, която се заплаща, като например прегледите на диспансеризирани пациенти, а други имат задължителен минимум, например профилактичните прегледи.

Освен до неизплащане на сумите, грешно попълнени документи могат да доведат до глоби и административни наказания. Тези мерки са взети от НЗОК за да се осигури качествено обслужване на пациентите съгласно определени норми. Същевременно това прави административната работа на лекарите изключително важна.

I.3. Компютъризация на лекарската практика

В системата на здравеопазването преди здравната реформа нямаше развито информационно осигуряване като глобално решение. В по-голямата част от здравните заведения се работеше единствено на хартия. Здравната реформа предвиждаше промяна на работата в здравеопазването към по-съвременен начин чрез използването на компютъризирана обработка на документите в електронен вид. За целта беше предвидено да се осигури техническо обезпечаване на здравните работници във формата на хардуер и софтуер помагащ на тяхната дейност.

Като хардуер общопрактикуващите лекари трябваше да получат съвременни компютри, позволяващи работа в мрежа и достъп до Интернет. В конфигурациите беше включено записващо компактни дискове устройство, както и лазерен принтер. За лекарите работещи в отдалечени райони беше предвидена радиовръзка. Всяка конфигурация съдържаше четец на електронни карти, чрез който да могат да се идентифицират пациентите и съответно в тях да се запазват техните здравни досиета.

Техническото оборудване бе закупено и раздадено на лекарите по приоритет, който се оценяваше въз основа на броя на записаните при тях пациенти. Въпреки, че беше предвидено и имаше техническа осигуреност на лекарите, не беше извършено свързването им в мрежа и включване към Интернет. Също така въвеждането на електронен подпис в България се забави. На пациентите не бяха раздадени електронни карти и тази част от процеса на здравно осигуряване, предвиден в реформата не е изпълнена.

Всяка РЗОК получи сървърни конфигурации и лицензи за ползване на Oracle база данни. Специалистите, които трябваше да се грижат за тях, бяха обучени. За съжаление на много места поддръжката не е на нужното ниво.

За софтуерната поддръжка на здравната реформа НЗОК подписа договор, който по-късно беше скандално прекъснат. Лекарите не получиха никакво програмно осигуряване на работата си, а здравните каси имаха само няколко готови модула. Така инициативата по намирането на софтуер беше загубена от НЗОК и се децентрализира и остана като лично решение на здравните работници и здравните заведения.

Това създане на пазара за софтуер търсене, на което българските разработчици бързо откликнаха. Към края на 2004 година се предлагаха множество решения за общопрактикуващите лекари. Някои от тях са:

- BetaGP

- Global Medics
- GP Soft
- GPTOOL
- Nurse
- АКСИОМ
- Медик35
- Медика
- Хипократ GP

I.4. Програмата „Хипократ”

Програмата „Хипократ GP” е разработена от фирма Контракс за обслужване на административните нужди на общопрактикуващите лекари. Тя е лицензирана от НЗОК и в момента с нея работят над 1300 лекари в цялата страна.

С нея се поддържа практиката с всички лекари и записаните при тях пациенти. С времето за всеки пациент се натрупва досие от първоначално въведената за него информация и история и документите от всички извършени прегледи.

При преглед лекаря има възможност бързо и възможно най-лесно да въведе информация и да разпечати съответните документи. В съответствие с изискванията на НЗОК документите могат да се печатат на бял лист или на официалните бланки. За улесняване на работата лекарите могат да копират стари документи и да поправят данните като въведат само разликите. За текстовите полета е предвиден избор от набор шаблони, които лесно могат да се настройват, така че да са най-удобни за лекаря – например най-често попълваните текстове или готови изрази, в които да се въведат само данни. Програмата съдържа списъци, от които може да се избира, когато става дума за стандартни полета, като например специалност на лекаря. Сред тях са списъците с лекарствата, кодовете на болестите, имунизациите, диспансеризациите и т.н. За удобство при избиране на някои типове програмата предлага автоматично да попълни някои от другите полета или предупреждава при попълването на определени данни, които не отговарят на изискванията на НЗОК.

Системата поддържа вход и изход за стандартните електронни документи, които използват РЗОК за обмяна на информация с лекарите. С тях се обновява списъка на записаните пациенти и списъка на диспансеризациите. За РЗОК се извеждат файлове съдържащи дейностите по различните програми и новите диспансеризации.

В програмата са включени разнообразни справки помагащи на лекарите в ежедневната им работа и за отчитането в РЗОК. Възможно е потребителят сам да приготви справка, като избира списъците, от които търси записи и въвежда допълнителни условия за да ги ограничи. От тези справки също може да се направи изход в различни файлови формати. Направената справка може да бъде записана за да бъде изпълнена отново в по-късен момент. За отчитането в РЗОК са приготвени специални справки отговарящи на всички изисквания. Те са и предмета на настоящата дипломна работа.

Справките за отчитане в РЗОК представляват най-вече финансови документи. В тях лекарите описват извършената дейност, която подлежи на заплащане от страна на здравноосигурителната каса. Най-общо казано те съдържат информация относно броя на пациентите по групи, броя на прегледите по групи и броя на другите издадени документи, които подлежат на контрол от здравноосигурителната каса. Сложността на задачата по приготвянето на такава справка се крие в класифицирането на документа в определена група. За да бъде направено това, справката придобива изключително сложен вид и често това води до грешки и трудности при промяна на изискванията на здравноосигурителната каса.

I.5. Проблеми на програмното осигуряване на лекарите

Понеже разглежданият софтуер има за задача да помага най-вече за отчитането на лекарите пред здравноосигурителната каса, можем да разделим проблемите на групи съответно на свързани с лекарите и свързани с НЗОК и РЗОК.

Първият проблем на при работата с лекарите е, че те често нямат компютърна грамотност. Усвояването на елементарните умения за работа с компютър е тежък процес за тях. Често те се страхуват от грешки и от незнанието си. Среща се нежелание да се разберат основните принципи залегнали в компютърните системи. Те смятат, че обучение по темата би им отнело твърде много време. Повечето лекари имат много пациенти и са длъжни да наемат помощници и често решават проблема като прехвърлят отговорността за работа с компютъра на тях.

Друг проблем е неправилното възприемане на софтуера от тяхна страна. При попадане в непозната ситуация те често опитват всички начини на излизане от нея без да обръщат внимание на реакциите на компютъра. При грешка отношението е крайно – или търсят вината изцяло в себе си, или обвиняват програмата за всичко. Съответно те често влизат в спорове със служителите на РЗОК, като твърдят, че данните извлечени от техния компютър са абсолютно верни, без да отчитат възможността за лична грешка или обратно да влизат в спорове с представителите на софтуера при всяка спънка в отчитането им пред здравноосигурителната каса.

Често лекарите използват компютъра без да рабират смисъла на действията си. Те наизустяват стъпките, чрез който се изпълнява дадена операция и ги следват сляпо. Това води до неефективно използване на програмата и сериозни проблеми дори при малки промени в интерфейса.

Използването на софтуер позволява върху работата на лекаря да се оказва по-голям контрол. Така излизат наяве грешки на лекарите, които в друг случай има голяма вероятност да бъдат пропуснати поради непълната проверка от страна на служителите на РЗОК. Това не се харесва, макар че всъщност предпазва от глобите при евентуалното откриване на допуснатата грешка.

Обратната връзка от потребителите често не е ефективна. Лекарите нямат изградени навици да съобщават за срещнатите проблеми, нито да записват съобщенията на екрана. Обясненията им обикновено са неточни и непълни. Понякога умишлено укриват информация за допуснатите от тях грешки, без да разбират, че това единствено усложнява проблема.

Изискванията към програмата са противоречиви. От една страна тя трябва да им спестява труд и да подготвя отчети в тяхна полза, максимизирайки приходите им. От друга страна отчетите трябва да бъдат коректни и да издържат на проверките от страна на здравноосигурителната каса, пазейки лекарите от глоби.

Липсва на организация и комуникация между здравните каси. НЗОК има ръководна роля, а всички РЗОК са реалните изпълнители. Подготвяните от НЗОК документи, които трябва да регулират дейността често са пренебрегвани или се изпълняват неточно от някои РЗОК.

Персоналът е неподготвен. Прекъсването на процеса на внедряване на цялостно решение промени ситуацията и служителите на здравноосигурителните каси не са подготвени да изпълняват задълженията си. Изискванията към тях са различни и често се случва те да не се справят със задълженията си по най-добрия начин. За да се справят с непривичните си отговорности те понякога променят изискванията към лекарите за да са им по-удобни за работа. Това означава неоднозначност на приетите формати за обмяна на информация и създава огромна трудност за създаването на универсален продукт. Освен това често електронното отчитане от страна на лекарите е възпрепятствано не по тяхна или на софтуера, който ги генерира, вина.

Липсва добра обратна връзка от лекарите. Техните желания и оплаквания трябва да минат през съответните професионални организации, а това е доста тромаво. Здравноосигурителните каси нямат стимул да решават проблемите на лекарите, въпреки че са стимулирани да санкционират лекарите и да търсят неизрядностите в техните практики. Очевидните неуредици често се решават след неофициалните, но масови протести на лекарите.

Много от изискванията не са формализирани. Рамковият договор и приложенията към него са обширни и описват голяма част от отношенията, правата и задълженията. Но въпреки това в него има много пропуски. Тези пропуски излизат наяве едва при практическото прилагане на рамковия договор. Това означава, че самата нормативна база съдържа подводни камъни за практиката. Тези текущи проблеми се решават чрез допълнителни заповеди и решения от НЗОК. Но обикновено те се приготвят за кратко време под натиска на недоволни лекари. Затова често съдържат неясноти и им липсва формализация, което налага нуждата от допълнителното им разясняване. Това създава допълнително забавяне на времето за реакция и увеличава вероятността за допускане на още грешки не само в разпоредбите, но още повече в тяхното изпълнение.

При вземането на решения здравноосигурителните каси не се съобразяват с практическата дейност на лекарите. Техните решения са продиктувани главно от техните нужди и най-често имат за цел да ограничат злоупотребите от страна на лекарите. Това е доста трудно и затова се случва приетите мерки да са крайни. Обикновено тези мерки поставят допълнителни отчетни задължения за лекарите и с това нарушават работния им процес и отнемат значителна част от времето им.

НЗОК често поставя кратки срокове на изпълнение или въвежда нови неща със задна дата. Това води до нужда от бърза реакция при разработката на софтуер за лекарите. Нужните промени трябва да се разработят и след това да достигнат до потребителите. Освен това информацията, която е въведена до него момент може да има нужда от коригиране за да отговаря на новите изисквания. В тези случаи проблемът може да ескалира, понеже такава корекция често е невъзможно да бъде направена автоматично и е нужно повторното и въвеждане.

II. Изложение

II.1. Цели на дипломната работа

Разработваната дипломна работа разглежда частта от програмният продукт „Хипократ", която изготвя справките и отчетите, нужни на здравните работници. Най-динамичната и същевременно изключително важна част от тях са финансово отчетните документи. Целта на програмния продукт е да служи като информационна система и затова може да приемем, че справките са нейният краен продукт. Като такъв обаче той се влияе най-много от проблемите, които са посочени по-горе. Вероятността дадена грешка да се отрази на справките е много голяма независимо в кой модул е допусната. Затова трябва да се потърсят нови решения, които да са по-добри и да намалят, доколкото е възможно, вредното влияние на проблемите.

В по-общ план целите са:

- да се увеличи гъвкавостта на софтуера;
- да се даде по-голяма свобода на потребителя;

Конкретните цели на дипломната работа могат да бъдат определени като следните:

- да се ускорят справките – да се потърси решение, което да има по-голямо бързодействие спрямо текущото. Това може и да не е възможно, но след като се прави основна промяна на метода на генериране на справките е добре да се оптимизира неговата работа, така че да е не по-бавна от сегашното бързодействие. Като се има предвид праволинейността на текущото решение, би трябвало да е възможно намирането на по-бързи решения.
- да се позволи посочването на тип несъответстващ на логиката на програмата – в момента генерирането на справките следва изцяло логиката, заложена в програмата и не може да бъде променяно от потребителя. Проблемите, които се опитваме да решим, диктуват нуждата от съществуването на възможност за промяна на данните против правилата кодирани в програмата. В миналото тази нужда предизвика премахването на множество твърди забрани от системата и замяната им с предупреждения, но това не бе направено за модула на справките, може би защото техните правила не са пряка забрана, а по-скоро страдат от липса на възможност за коригиране. Затова на потребителя е нужно да се даде не само способност да избира от възможните, но и от невъзможните, т.е. непозволените опции.
- да се позволи лесна промяна на типа на прегледа от потребителя – в настоящото решение на потребителя е позволено да коригира единствено данните в генерираната справка. Едно от неудобствата на това решение е, че промяната на класификацията на даден документ може да има отражение на повече от едно място – в няколко полета и дори в различни справки, и съответно трябва да се коригира навсякъде ръчно от потребителя. В справките, съдържащи списъци, такива промени са дори невъзможни, което понякога може да доведе до тяхната неизползваемост от потребителя.
- да има своевременна информация за типа на прегледа – в момента такава информация може да бъде получена, като се генерират съответните справки и отчети, но това е бавно и неудобно. Затова обикновено потребителите нямат своевременна информация за категоризирането на документите, а я виждат веднъж месечно, когато приготвят отчетните си документи. Своевременното получаване на информация би спестило на потребителя търсенето на допуснатата грешка сред многото документи. Тази грешка може да е на

програмата, но може и да е от негова страна, което означава, че ако той получи тази информация навреме, може да се предотврати проблем, който има не само административен характер и да се подобри качеството на обслужване на пациентите.

- да се централизира логиката на справките – една и съща класификация се използва в няколко справки. Те се различават като заявки, макар че части от тях съвпадат. Направено е обединяване на тези логически идентични части във функции, но въпреки това логиката им не е напълно централизирана и е нужно при всяка промяна да се открият всички използващи ги заявки. Те трябва да се прегледат внимателно и да се тестват дали не са се появили отклонения от логиката в следствие на промените. Централизирането на промените е възможно чрез по-добра организация, структуриране и формализиране на начина, по който се генерират справките. Точно това е една от целите на дипломната работа.
- да се улесни обновяването на логиката – един от проблемите, които бяха споменати, е честата промяна на класифициращите правила. Понеже програмата „Хипократ" се обновява на месечна основа, всяка нужда от спешна промяна по време на месеца изисква допълнителни ресурси и логистика за доставянето на обновяването на всички клиенти. Понякога новите изисквания влизат в сила в прекалено малък срок или дори със задна дата, което изисква още по-малък срок за доставяне на обновен софтуер.
- начинът на работа с програмата да не се променя – основната задача е промяната да не обърква потребителите, които вече са свикнали с някаква схема на работа. Не бива да се увеличава броя на действията при попълване на данните. Проблем ще е и промяната на типичната последователност от действия. Това означава да се ползват автоматични изчисления и стойности по подразбиране. Потребителят не бива да бъде прекъсван от допълнителни съобщения, но трябва да получи информация за автоматично взетото от програмата решение. Случаят, за който можем да си позволим отклонение е, когато потребителят, след като е получил информация за избраната класификация, желае да промени този избор по свое усмотрение. Това е отклонение от типичния работен процес, но е по изричното желание на потребителя.
- да няма допълнителни ограничения или задължения за потребителя – направените промени не трябва да премахват вече съществуваща функционалност на програмата. Възможностите, с които потребителите са разполагали досега, трябва да останат. Използването на новите функции не трябва да е за сметка на стари. Освен това потребителят не трябва да бъде натоварван с допълнителни отговорности. Ако той не желае да ползва новата функционалност неговите задължения не трябва да се променят.

II.2. Идея за решение

За да е възможно избирането на класификация от потребителя и по-късното и използване при генерирането на справките е нужно данните за този избор да се записват от програмата. Този запис може да бъде направен по различен начин – във файл, в регистъра на операционната система, в базата данни и т.н. За да е възможно избора да се вижда от всички потребители при многопотребителска среда, най-добрият подход е базата данни. Тя е обща за всички потребители и решава проблемите с конкурентния достъп до тази информация и синхронизирането и. Освен това по този начин данните остават заедно и няма опастност да бъдат да бъдат разделени при създаването на и възстановяването им от архив или при преместването им на ново място. Минималното съдържание на един запис би трябвало да включва идентификатор на прегледа и избраната класификация.

Изискването за съществуване на запис за всяка избрана от потребителя класификация води до идея за възможно ускоряване на справките. При дотук избрания подход при генериране на справка е нужно освен прилагането на използваната класифицираща логика да се проверява и за направен избор от потребителя. Това означава, че дотук предложеното решение ще забавя генерирането на справките. Факт е обаче, че една и съща класификация често се ползва при генерирането на две, а понякога и на повече справки. Така системата ненужно изчислява вече получени резултати. Затова може да оптимизираме генерирането на справките като изпълняваме класифициращата логика еднократно върху цялото множество от документи и записваме резултатите. Това класифициране е нужно само за документите, за които не е направен избор от потребителя. Ако това не се вземе под внимание ръчният избор на класификация ще бъде загубен – заменен от логиката на програмата.

След като сме сигурни, че за всички документи има избрана класификация – от програмата или от потребителя – можем да сведем генерирането на справките до просто броене на документите от даден клас. Този метод е близък до методите на динамичното оптимиране и кеширането. Те не гарантират ускорение на алгоритъма, освен при наличието на информация за повторения във входните данни. както бе отбелязано по-горе ние имаме такава информация. Понеже повторението в повечето случаи е едва двойно, двойно е и ускорението от ползването на записания резултат. Самото ползване обаче, отнема време. Допълнителното време идва от записването и след това от прочитането на записа. В конкретния случай всички класифициращи правила са достатъчно сложни и би трябвало да отнемат по-голямо от това време, така че можем да очакваме, че ще имаме ускорение на времето за генериране на всички справки.

Използването на алгоритъм, основан върху запазени готови резултати изисква разглеждането на един потенциален проблем. Този проблем е синхронизирането на записания резултат с входните данни, от които той е генериран. Той има два аспекта. Първият е нуждата при промяна на входните данни веднага да се направи промяна на изчислените резултати. Вторият е, че трябва да се знаят всички резултати, които зависят от дадени входни данни.

Последният може да бъде отстранен лесно, понеже знаем, че при промяна на данните за даден преглед е нужно да се преизчислят само неговите класификации. Не е сигурно, че те ще са различни, така че определянето на връзката между класифициращото правило и определени данни от прегледа може да бъде възможност за допълнително ускорение на програмата.

Първият аспект води до извода, че при създаване на нов или редакция на стар документ е нужно генериране на класификации съответно логиката на програмата. Това

трябва да става автоматично и без участието на потребителя. За целта съществуващите стари класификации се изтриват при редакция на документа, а при запис те се генерират автоматично. За да се позволи на потребителя да избере класификацията по време на редактиране на документа, при запис първо трябва да се провери дали не съществува класификация и ако не съществува, тогава да я генерира. Така ако потребителят е направил избор, той няма да бъде загубен, а в останалите случаи класификации не би трябвало да има, понеже те са изтрити при започване на редакцията на документа. Това решение съдържа един недостатък. Той е възможността избрана от потребителя класификация при предишна редакция на документа да бъде загубена при последваща такава. Погледнато логически, документът след като е редактиран представлява нов документ. Следователно дори направените от потребителя решения за класифициране е възможно да са вече невалидни. Затова ще приемем, съгласно целите на дипломната работа, че главната задача на програмата е да поддържа класификацията да отговаря на зададената логика, за да бъде съвместима с начина на работа на предишните версии, с които са свикнали потребителите, а новата възможност за корекция на направения избор да остане като изключение, поддържано от потребителя. Една друга възможност е да се маркират класификациите, избрани от потребителя и при тяхна промяна да се извежда съответно съобщение.

За да се улесни обновяването на логиката с нова такава е нужно тя да се отдели от кода на програмата. Логиката трябва да се може да се пренася и инсталира както заедно, така и отделно от новите версии на програмата. Възможно е това да стане под формата на DLL (dynamic-link library). Тази библиотека би съдържала всички функции и процедури за класификация и обновяването и би се свеждало до замяна на файла на библиотеката с по-нов. Един по-гъвкав вариант е пазенето на логиката в базата данни. В момента тя е под формата на SQL заявки, така че е възможно тя да бъде пазена като текст и при нужда изпълнявана от програмата. В този случай пренасянето на новата логика би ставало чрез обновяващ SQL скрипт – метод, който е неведнъж използван при разработката на програмата и обновяването и.

Гъвкавостта на този подход се крие в лесната възможност за коригиране на логиката на място от специалист или от потребител, насочван в действията си от специалист отдалечено – по телефона или по друг начин. За улесняването на тази дейност е нужно да се създаде администраторски екран на програмата или отделна програма, чрез която да могат да се извършват тези корекции. Освен това улеснение би било съществуването на автоматично генериране на обновяващия скрипт. Това ще сведе обновяването на логиката до натискането на няколко бутона.

Запазването на логиката в базата данни под формата на SQL заявки би нарушило досегашната структура изградена в програмата. Кода, който формира правилата за класифициране използва и множество функции, в които е отделена общата логика. Това позволява по-централизирано коригиране при нужда и съответно избягване на опастността от човешки грешки. За да бъде пренесена и йерархията е нужна допълнителна формализация на правилата. В момента те представляват SQL заявки, които се сглобяват от различни парчета. Тези парчета представляват общи изрази, заместени от споменатите функции, така че да могат централно да бъдат променяни, и специфични изрази, които се ползват само от това правило. Допълнителното формализиране на правилата би довело да по-неефективни заявки и затова досега не е направено. В текущата дипломна работа ще се опитаме да го развием. Понеже решихме изпълнението на правилата да става по време на запис на документа, те ще бъдат изпълнени само върху данните на един документ. Така, макар времетраенето на класифицирането да се увеличи като сума, то ще бъде разпределено във времето на малки порции, което позволява прилагането на такъв метод.

Първата възможност, която имаме за формализация, е да опитаме директно да отразим досегашните правила в псевдо SQL заявка, която да се обработва допълнително. Функциите ще бъдат записани поотделно в базата данни. При допълнителната обработка определени ключови думи ще бъдат заменяни от съответстващите им функции и така подобно на досегашния подход ще се получава заявка, която да връща търсените стойности.

В текущата дипломна работа искаме да използваме по-формален подход. Неговата цел е да разбием всяка от заявките на отделни изрази и да премахнем специфичните части. Така след приготвянето на елементарните изграждащи блокове, всяко правило ще може да се сведе до списък от използваните функции, които трябва да бъдат залепени една за друга с логически оператори.

За да направим избраният подход по-обобщен, можем да генерираме водещата част от SQL заявката автоматично, стига да разполагаме с името на таблицата от базата данни, в която се пазят документите, и полето от нея, което е уникален идентификатор. Това е възможно, понеже всички правила трябва да имат логическа стойност като резултат – дали документа спада към класификацията или не. Таблицата, от която се извличат входните данни ще се получава като параметър, а като знаем полето, което идентифицира уникално документа, можем да ограничим изпълнението на справката само до него.

Едно очевидно подобряване на формализацията е пазенето на изразите само в позитивната им форма. Тяхното противоположно значение може лесно да се получи, като се поставят в скоби и пред тях сложим оператор за логическо отрицание. Така не се дублират логически идентични изрази и се улеснява корекцията им. За да се ползват в този вид, списъкът на изразите, съставящи правилото трябва да се промени, като му се добави една допълнителна стойност за всеки запис. Тази допълнителна стойност ще обозначава дали се ползва позитивната или негативната им форма.

Разглежданата формализация свързва отделните елементарни изрази с логическия оператор И. Това ограничава възможностите за създаване на правила, като ни принуждава при наличие на правила, съдържащи логически оператор ИЛИ да ги преработваме, като вкараме оператора в елементарен израз или приложим някакви преобразувания.

За да решим генерално този въпрос и за да дадем една по-добра гъвкавост на правилата можем да изградим структура, подобна на нормалните форми на логическите изрази от булевата алгебра. Можем да разделим елементарните изрази на групи. Групите ще бъдат номерирани и всеки израз ще принадлежи на една група. Изразите във всяка група ще са свързани с логически оператор И, а групите ще са свързани помежду си с логически оператор ИЛИ. Така получаваме пълна изразителност за представяне на правила. Единствен компромис в така построената схема е липсата на възможност да се влагат логическите изрази един в друг (суперпозиция). За да бъде направена и тази възможност е нужно изградената структура да бъде йерархична. Това ще доведе до нуждата от допълнителни проверки за ацикличност. Затова ще приемем, че тази възможност излиза извън рамките на дипломната работа.

За да е по-лесно прехвърлянето на правилата от вид на код към новия формализъм е нужно елементарните изрази да могат да изпълняват ролята на функциите в кода. Освен липсата на начин да се ползват други изрази те имат и друг съществен недостатък. Той е липсата на възможност да се предадат параметри към израза. Добавянето и би довело до допълнителна централизация и лекота на поддържането на правилата. Затова изглежда целесъобразно добавянето на такава функционалност. Това може да се реализира чрез добавяне на списък с параметри към записа за използвания в правилото израз. Преди добавянето на израза към правилото

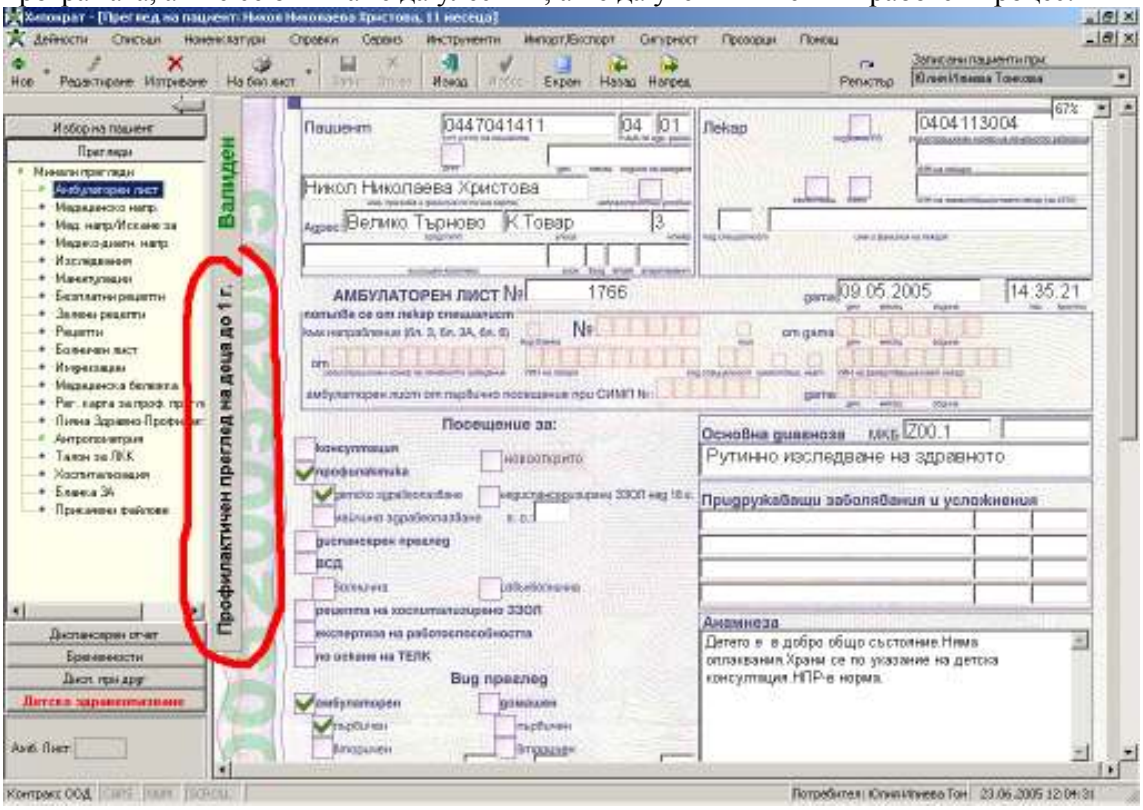
той ще се обработва със списъка на параметрите. В този дух можем да добавим и възможност за задаване на параметри по подразбиране. Те ще се ползват ако не са посочени такива в списъка с изразите в правилото.

Направените промени поставят една формализирана структура за представяне на правилата за класификация. Тя няма изразителността и функционалните възможности на вградените в кода правила, но за сметка на това притежава гъвкавост и лекота на доставка на обновяванията до потребителя. Чрез нея се дава възможност за бърза редакция на правилата на място и дори от самия потребител. Освен това ограниченията, поставени от формализма, дават едно друго предимство. При така зададената структура на представяне на правилата лесно може да се отговори на въпроса защо даден документ не отговаря на изискванията за определена класификация. Естествено този отговор може да не носи достатъчно информация. Но ако правилата се структурират добре отговорите могат да бъдат напълно достатъчни и разбираеми за обикновения потребител. Ако за всеки елементарен израз се осигури описание, разбираемо за хората, то от тези описания може да се изградят текстове, отговарящи на въпросите какво още е нужно за да получи документа дадена класификация и кои изисквания вече са изпълнени. Възможно е също да се направят бъдещи допълнителни разширения, които да анализират потребителските корекции на класификациите и да извличат новите правила от тях, които важат за този регион.

II.3. Интерфейс

II.3.1. Потребителски

Една от целите на дипломната работа е да се добавят новите функционалности с минимални корекции в интерфейса. Това е продиктувано от изискванията на потребителите. За тях всяка промяна предизвиква страх и неувереност в работата с програмата, а ние се опитваме да улесним, а не да утежним техния работен процес.



фигура 1

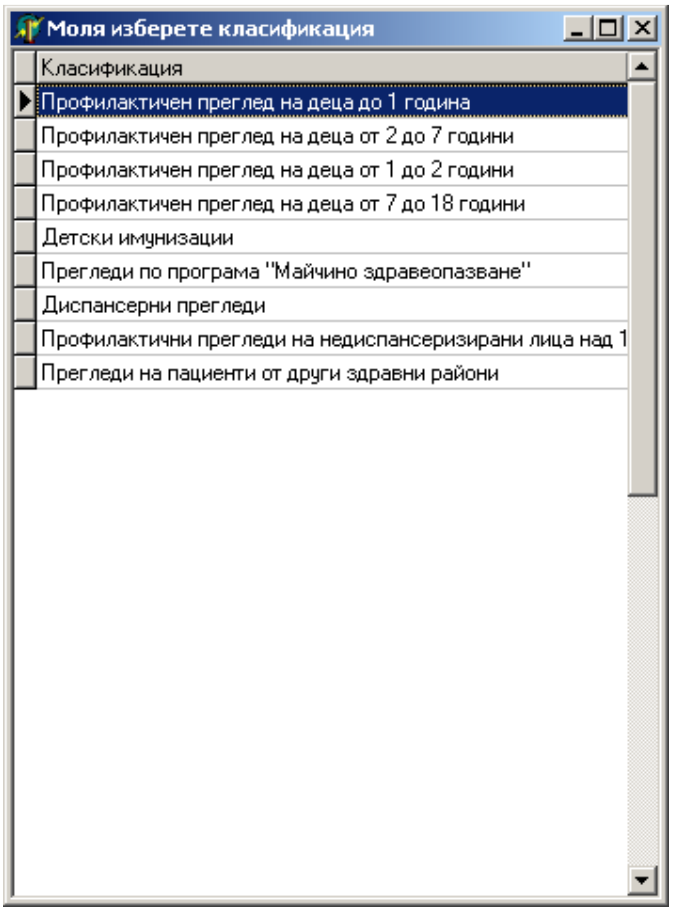
Затова бе избрано решението да се постави допълнителен бутон, който да бъде позициониран дискретно в лявата част на документа под вече съществуващ друг бутон. Другият бутон служи за анулиране на издаден документ и също е рядко използван.

Дизайна на двата бутона е синхронизиран за добиване на по-добра композиция на интерфейса. Текста е раположен вертикално и размера е съобразен със съществуващите описания на класификации.

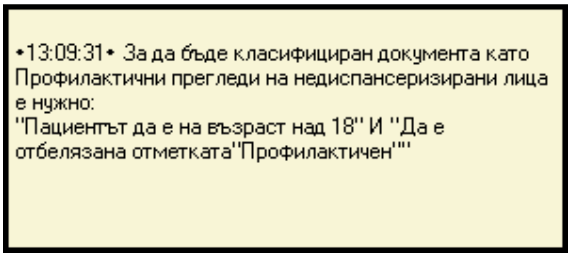
Текста на бутона се променя съобразно избрания документ. Така потребителят получава своевременна информация без това да му пречи излишно.

Класификацията на документа се извършва автоматично при запис на документа.

В някои други части на програмата също се изпълнява прекласификация на документа, но това остава незабелязано от потребителя. Тези части са свързани с начина на класификация, например имунизациите и диспансеризациите. Ако такава прекласификация е извършена, информацията за нея се получава при последващо отваряне на документа.



фигура 2



фигура 3

При щракане с мишката върху бутона, който е добавен и показва текста на класификацията, се дава възможност на потребителя да коригира съществуващата класификация по свое усмотрение.

Това не е отклонение от задачата за минимална разлика в работния процес, понеже стандартния работен процес е прекъснат по желание на потребителя.

Потребителят избира новата класификация от нов прозорец, който се появява след щракането с мишката.

Този прозорец съдържа списъка на всички класификации включени в тази група.

Изборът на нова класификация се извършва чрез двойно щракване върху желания ред. След двойното щракване прозорецът за избор автоматично се скрива. Потребителят се връща към екрана на документа, като вече се вижда избраната класификация. При запис на документа няма да се направи прекласификация, понеже вече има избрана класификация.

В повечето случаи изборът на класификация, направен ръчно от потребителя е в противоречие с правилата, които са настроени в системата. Затова след избора на класификация от страна на лекаря програмата визуализира екран със съобщение, което съдържа описание на разликите между избраната класификация и текущия документ.

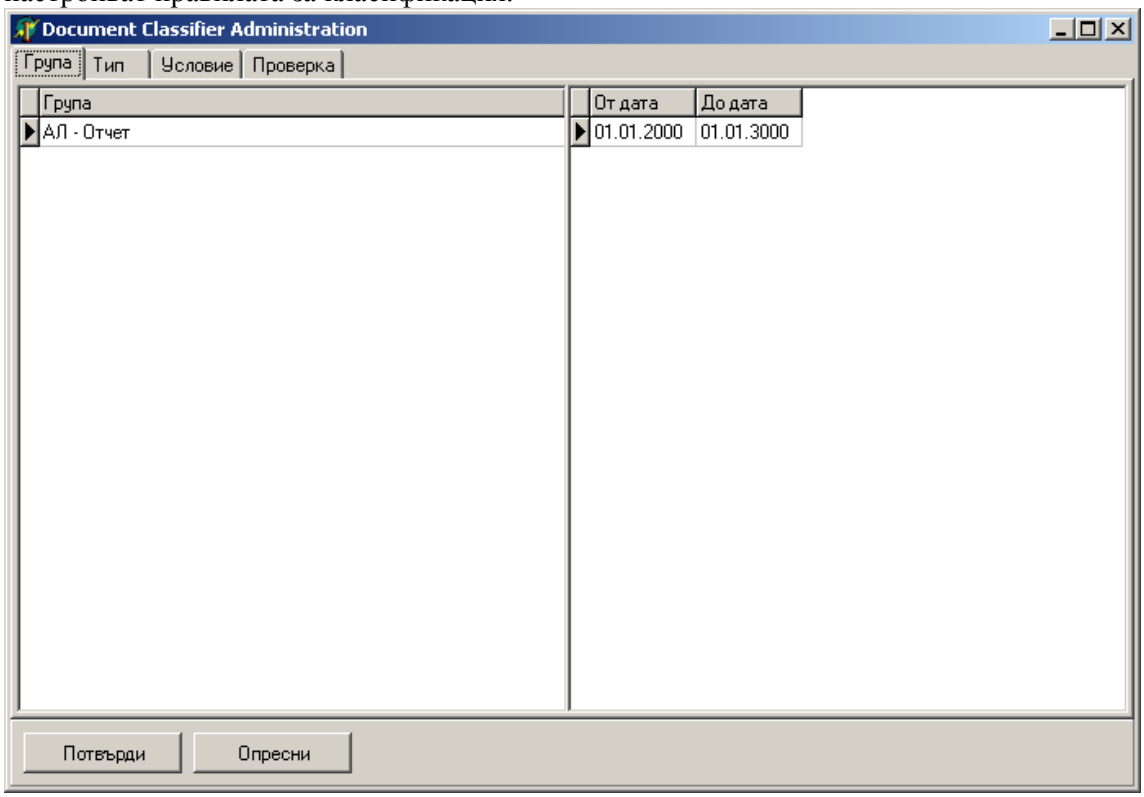
Екранът за съобщенията се използва в системата за множество цели. Основното му предимство е, че не придобива фокуса на системата и по този начин не прекъсва работата на потребителите.

II.3.2. Администраторски

За целите на администрирането на правилата, използвани за класификация, е приготвен модул, с който тази дейност се улеснява. Този модул е неделима част от предложеното решение.

За удобство при интегрирането на решението в системата е приготвена функция, чиято задача е да стартира административния модул на класифициращия обект. Тази функция може да се използва на произволно място в системата и дава гъвкавост на предложеното решение.

Самият административен модул се състои от няколко екрана, в които се настройват правилата за класификация.

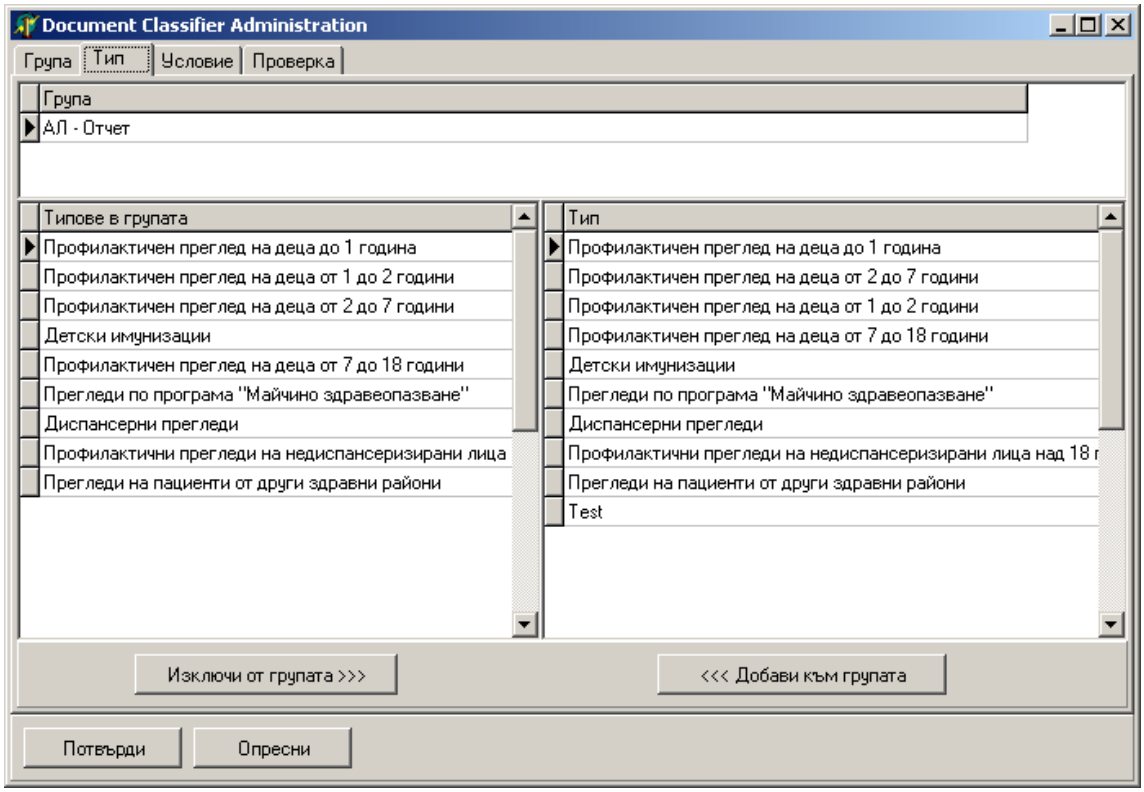


фигура 4

В първият екран на административният модул има две таблици в лявата и дясната част.

Лявата таблица съдържа списък със всички групи, които са настроени в тази база данни. За да се добави група е достатъчно да се натисне бутон стрелка надолу за да се появи празен ред и да се попълни името на новата група. Изтриването на група се извършва чрез комбинацията Ctrl+Del. Естествено, ако се опитаме да изтрием група, в към която има свързани други данни, това няма да бъде позволено.

Дясната таблица съдържа историята на избраната в ляво група. При смяна на групата автоматично се попълват нейните исторически групи. Самите исторически групи са представени като интервали, дефинирани от две дати. Интервалът е затворен от ляво и отворен от дясно. Избраният ред от таблицата с интервалите обозначава историческата група, която се настройва в следващия екран.



фигура 5

Във втория екран на административната форма имаме три таблици.

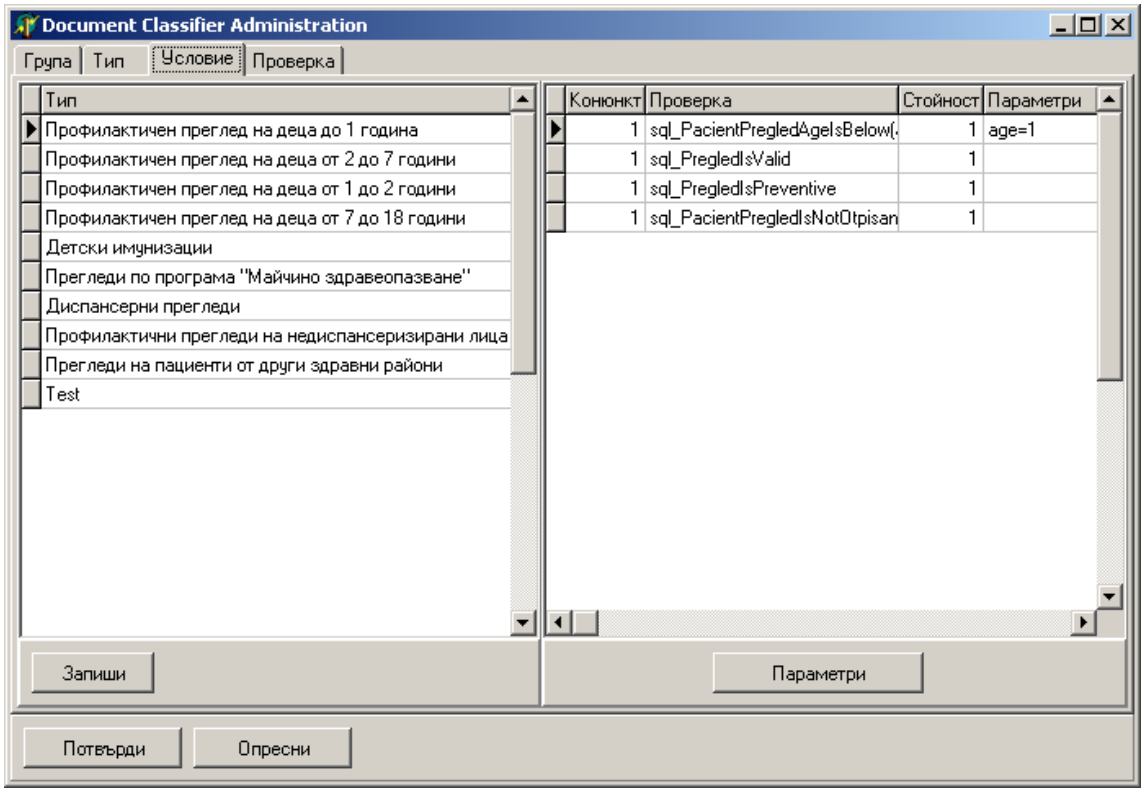
Таблицата отгоре съдържа името на групата, която е текущо избрана.

В долната част има две таблици, чрез които се настройват класификациите, които се включват в дадената група.

Лявата таблица съдържа списъка на всички класификации, които са включени в избраната група.

Дясната таблица съдържа всички класификации, които са дефинирани в базата данни.

С помощта на двата бутона разположени под таблиците се извършва включването и изключването на класификацията към избраната група.



фигура 6

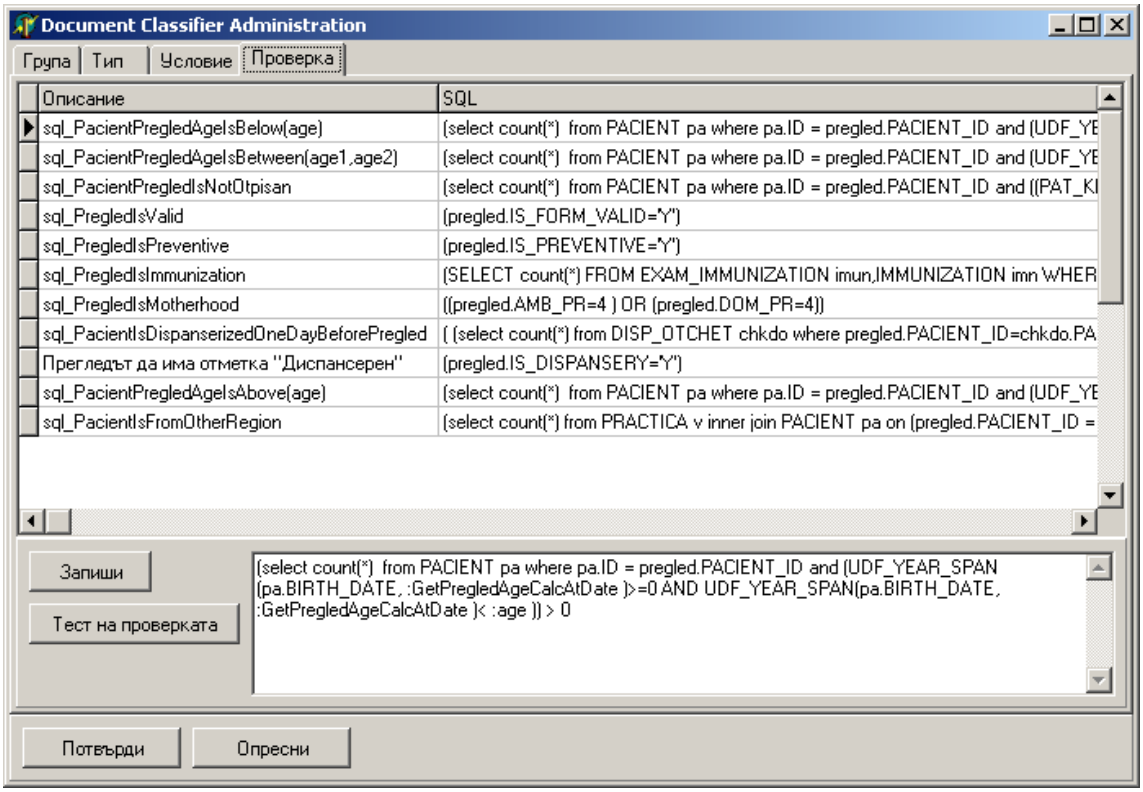
Третият екран съдържа настройките на условията.

В него има две таблици разположени хоризонтално една до друга.

Таблицата в ляво съдържа списъка със всички дефинирани класификации. Чрез тази таблица могат да се добавят нови дефиниции и да се премахват ненужните.

В таблицата в дясната страна на екрана се съдържа най-сложната таблица, която съдържа правилото за класификация на избрания в лявата таблица тип.

Всеки ред в нея съдържа проверка, която се използва в правилото. Колонката „Стойност” обозначава дали се ползва позитивната или негативната форма на проверката. С цифрата 1 се обозначава позитивната форма. В първата колона се попълва номера на конюнкта, в който се включва проверката. По-късно всички проверки с еднакъв номер ще се свързват помежду си с И в правилото. Самата проверка се избира във втората колонка. Ако е нужно да се предадат някакви параметри, те се посочват в четвъртата колона. Те могат да бъдат въведени като текст, или да се използва бутонът „Параметри”, с който се отваря описаният по-долу екран за въвеждане на параметрите.



фигура 7

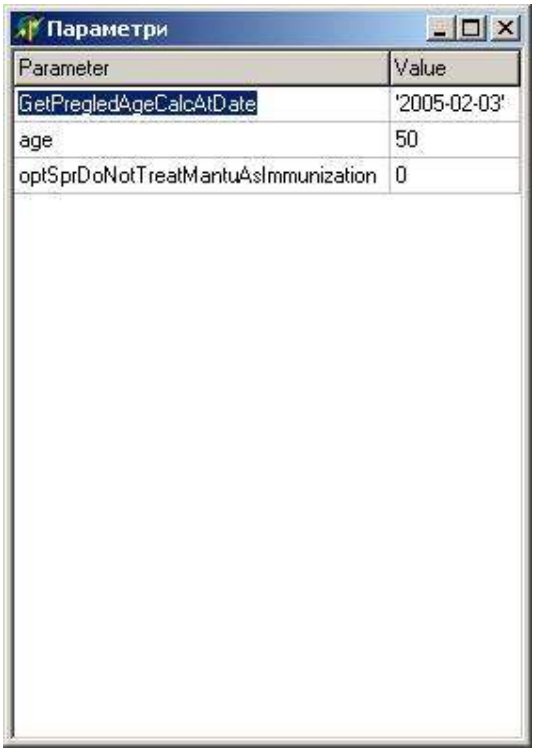
В последният екран се съдържа списъкът с проверките, използвани за класификацията. Чрез комбинирането на тези проверки се генерира правило, с което да се определи класификацията на документа.

В таблицата в горната част на екрана могат да се добавят нови, изтриват ненужните и редактират съществуващите проверки.

В първата колона на таблицата се записва описанието на проверката. Това описание се използва не само за идентификация на логиката, скрита в тази проверка. Тя е нужна за създаването на информационното съобщение, което се показва на потребителя, когато между съдържанието на документа и класификацията има несъвпадение.

Втората колона съдържа най-важната част на проверката – изразът, който се включва в справката за класификация. Този израз трябва да е написан на SQL. Препоръчва се той да е затворен в скоби, въпреки че при използването му те автоматично се поставят. Всички параметри, които се използват в него трябва да бъдат във формата „:<име на параметъра>”.

В третата колона се записват параметрите по подразбиране. Тези параметри се ползват, ако за тях не са подадени стойности.



фигура 8

За въвеждането на параметрите се използва специалният екран, показан на фигура 8.

Той се състои от таблица с две колони. Първата колона съдържа името на параметъра като текст. Това име по-късно се търси в израза на проверката и се заменя със стойността във втората колона. Тя също е във вид на текст. Замяната на параметъра с неговата стойност се извършва директно. Затова текстовите стойности трябва да бъдат поставени в кавички (``). Същото се отнася и за датите.

Понеже стойността се заменя в SQL израз без да се правят допълнителни проверки, това предоставя някои допълнителни предимства, но има и недостатъци.

Предимство е възможността стойността да не е обикновена константа, а да е по-сложен израз. По този начин системата става още по-гъвкава. Също не е задължително текста да представлява израз, който се свежда до константа. В зависимост от това къде в израза на проверката е разположена препратката към параметъра, изразът, който се получава като стойност на параметъра може да бъде също име на таблица, израз за стойност в подзаявка, част от условието на подзаявка и т.н. Принципно е възможно стойността на параметъра да бъде дори произволно изрязана част от израза за проверка, ако не и целия израз. Също е възможно в самата стойност да има препратка към параметър, чрез което да се постигне йерархична структура на параметрите и съответно още по-голяма гъвкавост.

Това същевременно е и недостатък на това решение. Възможно е да се направи цикъл в йерархията на параметрите и така един параметър в крайна сметка да съдържа себе си. За такива случай не е направена проверка и програмата ще изпадне в безкраен цикъл. Друг проблем е липсата на възможност за използване на специалния символ двоеточие „:”, с който маркираме параметрите. Ако имаме нужда израза на проверката да съдържа текст, който включва двоеточие следвано от текст, който съвпада с името на някой параметър, това би довело до неговата замяна.

II.4. База данни

II.4.1. Таблица на групите класификации

В тази таблица ще пазим различните групи, по които ще се класифицира. Примери за разделяне на групи е по признака в коя справка е нужна дадената група класификации.

Името на таблицата е DOC_CLASS_GROUP. Полето за идентификация на групата е DOC_CLASS_GROUP_ID. Описанието на групата се пази в полето GROUP_DESCRIPTION. Идентификаторът на групите се попълва автоматично и е първичен ключ на таблицата.

II.4.2. Таблица на история на групите

Тази таблица служи като едно допълнително ниво на абстракция на групите. Тя свързва всяка група с определен времеви интервал. Така можем да разграничаваме една и съща група, която се ползва на различни дати. По този начин групите имат исторически запис на състоянията, през които са минали. Това позволява правилното класифициране назад във времето и за по-големи, или просто пресичащи две версии на групите, справки.

Името на таблицата е DOC_CLASS_HISTORY_GROUP. Уникален идентификатор в нея е полето DOC_CLASS_HISTORY_GROUP_ID. Това поле се попълва автоматично и служи като първичен ключ на таблицата. В полетата FROM_DATE и TO_DATE се пазят границите на времевия интервал за действие на тази версия на групата. По подразбиране крайната дата е 01.01.3000 г. Полето DOC_CLASS_GROUP_ID сочи групата, на която е зададен времевия интервал, като запис от таблица DOC_CLASS_GROUP.

II.4.3. Таблица на класовете документи

В тази таблица се държи списък на всички класификации, които се използват от групите. Предвидено е класификациите да могат да участват в повече от една група, затова записите в нея не представляват разбивка на групите.

Името на таблицата е DOC_CLASS_TYPE. Неин уникален идентификатор е полето DOC_CLASS_TYPE_ID. Полето се попълва автоматично и е първичен ключ на таблицата. Описанието на класа се пази в полето TYPE_DESCRIPTION.

II.4.4. Таблица на връзките между групите и класовете

Това е таблицата, която служи като връзка между групите и класификациите във всяка група, понеже е предвидено връзката между тях да е от типа много към много. От тук се взема списъка с всички класификации, които са включени в дадена група.

Името на таблицата е DOC_CLASS_GROUP_TYPE. Връзка към таблицата с групите е полето DOC_CLASS_GROUP_ID, а към таблицата с класовете е полето DOC_CLASS_TYPE_ID. За тази таблица е поставено ограничение комбинацията от групата и класа да е уникална, което забранява поставянето на един клас в дадена група повече от един път. Особеното тук е, че групата, към която се сочи е от таблица DOC_CLASS_HISTORY_GROUP, т.е. се избира от историческата разбивка на групите. По този начин се посочват различните версии на групата за различните времеви интервали.

II.4.5. Таблица на класификациите на документите

Тази таблица съдържа класификациите на всички документи. За да се позволи разграничаването на дадена класификация в зависимост от това за коя група е нужна се посочва и групата. Така се предвижда преодоляване на случая, при който имаме една и съща класификация в две групи, но документът да има тази класификация само в едната група, но не и в другата.

Името на таблицата е DOC_CLASS. Уникалният идентификатор на класифицирания документ се пази в полето DOC_ID. Класификацията на документа е посочена в полето DOC_CLASS_TYPE_ID. В полето DOC_CLASS_GROUP_ID се посочва групата, от която е избрана тази класификация.

II.4.6. Таблица на елементарните проверки

В нея се съдържат основните градивни елементи на правилата за класифициране. Те трябва да са във формата на SQL изрази с резултат логическа стойност.

Името на таблицата е DOC_CLASS_CHECK. В нея уникален идентификатор е полето DOC_CLASS_CHECK_ID. То се попълва автоматично и служи като първичен ключ. Полето CHECK_DESCRIPTION съдържа описание на проверката. Това поле е специфично, понеже се използва не само при композиране на правилата от администраторите, но и за генериране на информативни съобщения за потребителя. Самият израз на проверката се пази в полето CHECK_SQL като текст. В полето DEFAULT_PARAMETERS се записва списъкът с параметрите по подразбиране, ако има такива.

II.4.7. Таблица на списъците с проверки за всяка класификация

Чрез тази таблица се дава смисъл на отделен клас. В нея са описани правилата, чрез които се класифицира документа. За всеки клас е представен списък от проверките, които съставят правилото и начина, по който те се свързват за да се получи цялостен израз.

Името на тази таблица е DOC_CLASS_TYPE_COND. Уникалният идентификатор е полето DOC_CLASS_TYPE_COND_ID. То се попълва автоматично и е първичен ключ на таблицата. В полето DOC_CLASS_TYPE_ID посочваме клас от таблицата DOC_CLASS_TYPE. Така различаваме списъците на различните правила. Всички записи за даден клас съставят спипък, който е разделен на групи в зависимост от полето CONJUNCT_NO. Смисълът на това е, че всяка група представя израз, който се свързва с останалите изрази с логическия оператор ИЛИ. Елементарните изрази включени във всяка група са посочени в полето DOC_CLASS_CHECK_ID. Чрез неговата стойност се намира израза в таблицата DOC_CLASS_CHECK. Елементарните изрази в групата се свързват с логически оператор И. В полето CHECK_VALUE посочваме дали съответния елементарен израз се взема с положителното му значение или с отрицанието му. Преди да се използва, израза се обработва, като параметрите в него се заменят със стойности от списъка на параметрите в полето PARAMETERS.

II.5. Логика

II.5.1. Инициализиране

Преди да започне да се използва, класифициращият обект трябва да бъде създаден и инициализиран. При инициализирането му, към обекта се подават няколко параметъра. Те са обектът, който поддържа връзката с базата данни; идентификатор на групата, за която класифицираме; името на полето, което съдържа идентификатора на

документа; името на таблицата съдържаща документите; и датата към която се прави класификацията.

За работа в реално време е нужно обекта за връзка с базата данни, който е подаден да съвпада с използвания за визуализиране или редактиране на документите. Ако това не е така, класификацията ще работи през друга транзакция и не е гарантирано, че ще има своевременна информация за текущото състояние на документа. Възможно е, ако транзакциите са различни, последните данни за документа да не са достъпни за класификацията преди транзакцията за редактиране на документа да бъде затворена и съответно данните да постъпят в базата данни. Това не е задължително да е вярно, понеже е възможно транзакциите да бъдат настроени на по-ниско ниво на защита, което да позволява предварителното четене на непотвърдени данни.

С един класифициращ обект можем да класифицираме само в една група. Това не е ограничение, продиктувано от архитектурата. Този подход е избран като по-удобен в конкретния случай, но при нужда е възможно начина на работа да бъде реорганизиран.

Името на ключовото поле и таблицата на документите за класификация са нужни за работата на класификатора. Те се използват в множеството SQL заявки извиквани от обекта за намиране на данните на документа и за други цели.

Датата, към която се прави класификацията обикновено е част от данните на документа. Затова тя също се подава като име на поле от таблицата с документите. По време на класификацията на документа, данните от това поле се ползват за да се открие подходящото правило.

По време на инициализацията се записват подадените параметри за по-късна употреба. Обекта се настройва да ползва посочената връзка към базата данни. След като вече има достъп до базата данни се прави проверка за съществуването на собствените таблици на класификатора. Липсата на някоя от тях би довела до невъзможност да се извърши класификация.

Проверкат на собствените таблици се извършва чрез просто броене на имената от даден списък, които се съдържат в системната таблица с имената на таблиците. Ако бройката е по-малка се показва предупредително съобщение.

II.5.2. Автоматично класифициране

Функцията за автоматично класифициране има за цел да се класифицират всички документи след като са били редактирани или създадени, без да е нужно изрично извикване на функции на класифициращия обект. Това става чрез записване на специална функция в списъците на събитията AfterPost, AfterCancel и AfterEdit.

Автоматичното класифициране се активира чрез извикване на функция с параметър обекта, който се използва за редакция на документите. Тази функция проверява дали има предишно активно автоматично класифициране, и ако е така отписва функциите от списъците на събитията. При отписването от списъка се възстановява оригиналната функция за обработка, ако има такава.

След това се вземат текущите функции от подаденото събитие и се запазват, а на тяхно място в списъците на събитията на обекта, който е подаден като параметър, се записват специалните функции, с които да се обработи това събитие. За коректната работа на класифициращия обект, както е описано в по-горната точка, е нужно да съвпадат транзакциите, с които се достъпва базата. Затова като обект за достъп до базата данни тук се избира същия обект, който се използва и от обекта, който е подаден като параметър.

Специалните функции за обработка на събитията са две.

Първата се извиква при събитието започване на редакция на документа. Тя изпълнява запазената оригинална функция за обработка на събитието, а след това изчиства съществуващите класификации на документа.

Втората се извиква при събитието запис или отказ от промените на редактирания документ. Тя също предава управлението на запазената оригинална функция за обработка на събитието и след това класифицира документа.

Функциите за изчистване на класификациите и класифициране на документа са описани по-долу.

II.5.3. Ръчна промяна на класификацията

Тази функция има две основни задачи. Първата е да визуализира класификацията на избрания документ, а втората е да предостави възможност за промяна на тази класификация от потребителя.

Функцията има два параметъра. Първият е обект за действие, а втория е логическа стойност, чрез която се указва дали да се предостави възможност за промяна на класификацията.

За нуждите на визуализирането на класификацията, обектът поддържа списък от класовете, като в дадения момент е избран класа, съответстващ на текущия документ. Създаването на списъка и включването на синхронизацията му с документите е първата задача на тази функция.

Една допълнителна възможност, която се предоставя с тази функция е визуализирането на класификацията в обект за действие (TAction). Така визуализацията става изключително гъвкава, понеже обекта за действие може да поддържа множество други визуални контроли, при това едновременно, чрез които да се вижда класа на документа.

Друго важно свойство на обекта за действие е да бъде допълнително ниво на абстракция за действието промяна на класификацията. Ако вторият подаден параметър има стойност истина, то към събитието изпълнение на действието, представено от обекта за действие се добавя изпълнението на специална функция за ръчна класификация, а оригиналната се запазва.

Тази специална функция търси историческата група за датата на документа, като използва вътрешна функция, описана по-долу. За тази група се извличат всички класове и те се представят на потребителя под формата на списък, от който той може да направи избор. След като потребителя избере класификация тя се записва чрез съответната вътрешна функция, която е описана по-долу, и предава управлението на оригиналната функция за обработка на събитието, ако има такава запазена.

II.5.4. Изчистване на класификациите

Изчистването на класификациите е изпълнено елементарно с SQL заявка. Заявката трие всички записи в таблицата с класификациите за подаденият идентификатор на документ и група. Ограничаването до класификациите само до избраната група е нужно, поради факта, че обекта класифицира само по тази група.

II.5.5. Класифициране на документ

В тази функция се извършва най-важната операция, а именно генериране на правилата и изпълняването им за да се получи класификацията на документа.

В началото се намира историческата група, като се използва функцията, описана по-долу.

Първата стъпка по генерирането на правилото е извличане на списъка с елементарните изрази, които го съставят. Това се извършва с SQL заявка, в която са

свързани таблицата на връзките между групите и класовете, за да открием измежду кои класове можем да избираме; таблицата на списъците с проверки за всяка класификация и таблицата на елементарните проверки. Списъка е подреден по клас и логическа група.

След като сме получили списъка на правилата за всяка класификация, той се обработва ред по ред за да се приготви заявката, която ще върне подходящата класификация. Заявката, която се изгражда, представлява обединение (UNION) от заявки съответстващи на правилата за всеки един от класовете. Ако някоя от тези заявки върне ред, това означава, че документа е класифициран към съответния и клас.

Всяко отделно правило се изгражда като в условието на заявката се добавят елементарните изрази на проверките му. Те се поставят в скоби и ако е нужно да се вземе отрицанието им пред скобите се добавя логическия оператор за отрицание. Във всяка от групите елементарните изрази са свързани с логическия оператор И, а самите групи са поставени в скоби и са свързани помежду си с логическия оператор ИЛИ.

За да се използват на елементарните изрази в правилото, те преминават през спомагателните функции за обработка на списъка с параметри и замяна на параметрите със стойности, които са описани по-долу.

Така приготвената заявка се изпълнява и нейният резултат се обработва съответно в следните три случая: ако резултатът е един, това е класификацията на документа; ако няма резултат, то няма класификация; и ако има повече от един резултат се предоставя на потребителя списък за избор на класификация. Този списък се визуализира в отделен прозорец.

Получената класификация се записва с описаната по-долу функция.

II.5.6. Запис на класификацията

Преди самия запис на класификацията се извиква функцията за изчистване на класификациите, за да се гарантира, че няма да се добави втори запис за същите документ и група. Самото записване се извършва с SQL заявка. След запис се обновява съдържанието на списъка с класовете, който се използва за визуализиране, ако е създаден.

II.5.7. Намиране на историческата група

Това става чрез SQL заявка. В към таблицата с историята на групите се свързва таблицата с документите, която е ограничена само до избрания документ. Така е възможно в същата заявка да ползваме полето на документа, в което е датата и така да ограничим историята на групите до тези от избраната група, които включват във времевия си интервал и датата на документа. Поради възможността да се получат повече от един резултат, сортираме по началната дата на интервала и вземаме историческата група с най-късна такава.

II.5.8. Обработка на списъка с параметри

Целта на тази обработка е да се съберат в едно списъка с подадените параметри и списъка с параметрите по подразбиране. За целта списъка на подадените параметри се прехвърля един по един в списъка с параметрите по подразбиране, като ако параметъра е вече включен в този списък, той се заменя с новия. Така полученият списък се връща от функцията.

II.5.9. Замяна на параметрите със стойности

Замяната се извършва, като в елементарния израз се търси всеки от параметрите в списъка със залепено отпреде му двоеточие. Ако той бъде намерен, се заменя със съответната му стойност. Начина, по който е реализирана тази функция показва, че не

можем да имаме два параметъра, за които единия е префикс на другия, понеже това би довело до възможни грешки, ако вместо параметъра се замени само префикса му.

II.5.10. Стартиране на административната форма

Тази инициализираща функция създава административната форма и отваря всички използвани от нея таблици, след което я показва.

II.5.11. Информация за разликите

Чрез тази функция се изпълнява втората основна функционалност на класифициращия обект. Тя получава два параметъра. Първият е идентификатора на документа, за който искаме информация. Вторият параметър е класът, който искаме да сравним с документа. Той е незадължителен и ако бъде изпуснат се взема класът, в който е класифициран документа.

Първо се намира историческата група, като се използва съответната функция. След това се стартира заявка, която да върне списъка от изразите, съставящи правилото за класификация. Този списък се получава като свържем таблицата на връзките между групите и класовете с таблицата на списъците с проверки за класификациите и таблицата с елементарните проверки и ги ограничим за избраната историческа група и клас. Списъкът е подреден по логическите групи на изразите. Специфичното тук е, че се взема отрицанието на изразите в правилото, т.е. пред тях се поставя логическо отрицание, а ако е обозначено, че се използват негативните им форми, не се поставя нищо. Така като резултат получаваме всички изрази от правилото, които не са изпълнени за този документ. От тях изграждаме съобщение, което да е подходящо за потребителя. То се състои от описанията на изразите, свързани с предлога И и с частицата НЕ пред тях, когато трябва да се вземе отрицанието им. Отделните групи се свързват с предлога ИЛИ, който стои на нов ред за четливост.

III.Заключение

III.1. Анализ на предложеното решение

В настоящата дипломна работа се опитахме да изградим един по-семантичен подход към класификацията на документите в лекарската практика. Причина за темата на дипломната работа бе нуждата от гъвкавост, която да помогне за решаване на проблемите в административната дейност. Макар решението да не претендира за пълнота и универсалност, то постигна в голяма степен поставените цели.

Скоростта на генериране на справките бе значително увеличена. В тях останаха някои сложни заявки, но те нямат връзка с класификацията на документите и съответно с предложеното решение. Ускорението бе направено за сметка на забавяне по време на въвеждането на документа. Това забавяне обаче е незабележимо и относително малко като дял от времето, нужно за обработка на документа при запис.

След записа на документа лекаря има своевременна информация за направената класификация. Не е нужно той да генерира отделни справки или да отваря допълнителни прозорци – тя е визуализирана на екрана успоредно с данните на документа. Така той има информацията дори и без да я е изискал изрично от системата.

Имайки информация за класификацията, потребителя има възможност да открие и отстрани евентуални грешки в нея. Това става лесно чрез няколко натискания на бутона на мишката. Възможността за редакция не е ограничена до момента на запис, а потребителя може да се върне към този документ и да редактира неговата класификация дори без промяна на други данни.

Едно неумишлено подобрение на работата с продукта е възможността участието на документа в повече от един клас да се съобщи на потребителя и двусмислието да бъде отстранено ръчно. При така изградената структура появата на двусмислие е избегната и съответно е отстранена вероятността за повторно броене на един и същ документ. Това е важен плюс в светлината на стриктните ограничения поставени при подготвянето на административните документи.

Тази, както и други позитивни промени са пряко следствие от подобрената централизация и формализация на логиката.

Тя бе поставена в базата данни, което доведе до лесната и преносимост. Обновяването и е отделено от обновяването на програмата. Бързината на реакция на промени в правилата е увеличена и същевременно е спомогнато за намаляне на разходите по разпространяването им.

Правилата придобиха един по-формален вид, с което се подпомага работата с тях при обновяване. Възможността да се допуснат пропуски е по-малка. С добавката на историческо записване е позволена правилна работа за по-големи периоди и дори във времеви интервали, които съдържат различни правила. Самите правила са поставени в нормална форма, което е основа за други подобрения в бъдеще.

Едно такова подобрение, което е следствие от нормалната форма на правилата, е показването на информация за разликата между избраната класификация и съществуващите правила. Това е от помощ за потребителя, когато е допуснал грешка; за представителят, който работи на място при клиента; и за програмистите, когато анализират и разработват правилата.

Смяната на начина на работа на класифицирането не промени начина, по който се работи нормално с програмата. Потребителите, които са свикнали с предишния начин на работа няма нужда да бъдат обучавани отново и няма да изпитат трудности да използват системата. Разликите в работния им процес са само и единствено при изричното им желание.

III.2. Възможно бъдещо развитие

Поставените цели на дипломната работа бяха постигнати с реализацията на предложената идея. Като проект, обаче, разглежданото решение може да бъде развито в бъдещето и да реализира нови удобства и функционалности, както и промени в архитектурата, с които да се дадат нови възможности. Ще отбележем няколко такива идеи:

- Маркиране на е избраните от потребител класификации – за подобряване на интерфейса.
- Пакетно класифициране на документи – всички от даден списък, всички некласифицирани или абсолютно всички.
- Справки за документите от базата, които отговарят на дадено правило, интегрирани в административната форма.
- Добавяне на възможност всеки или повечето елементарни изрази да имат съответстващи методи, които да попълват документа, така че елементарният израз да е изпълнен.
- Автоматизиране на генерирането на справки, чрез добавяне на съответствията между променливите в тях и класовете.
- Използването на правила в правилата, за постигане на по-сложни условия и йерархии.

IV. Приложения

IV.1. Скрипт за създаване на таблиците в БД

```
/*
The document groups
*/
CREATE TABLE DOC_CLASS_GROUP
(
       DOC_CLASS_GROUP_ID    INTEGER not
null,
       GROUP_DESCRIPTION     varchar(100)
not null
);

CREATE GENERATOR GEN_DOC_CLASS_GROUP_ID;
SET TERM ^ ;

CREATE TRIGGER DOC_CLASS_GROUP_BI FOR
DOC_CLASS_GROUP
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  IF (NEW.DOC_CLASS_GROUP_ID IS NULL) THEN
    NEW.DOC_CLASS_GROUP_ID =
GEN_ID(GEN_DOC_CLASS_GROUP_ID,1);
END

^

SET TERM ; ^

alter table DOC_CLASS_GROUP
add constraint PK_DOC_CLASS_GROUP
primary key (DOC_CLASS_GROUP_ID);

/*
The history of the groups
*/

CREATE TABLE DOC_CLASS_HISTORY_GROUP
(
       DOC_CLASS_HISTORY_GROUP_ID
       INTEGER not null,
       DOC_CLASS_GROUP_ID    INTEGER not
null,
       FROM_DATE      DATE not null,
       TO_DATE DATE default '3000-01-01'
not null
);

alter table DOC_CLASS_HISTORY_GROUP
add constraint PK_DOC_CLASS_HISTORY_GROUP
primary key (DOC_CLASS_HISTORY_GROUP_ID);

alter table DOC_CLASS_HISTORY_GROUP
add constraint
FK_DOC_CLASS_HISTORY_GROUP_GR
foreign key (DOC_CLASS_GROUP_ID)
references
DOC_CLASS_GROUP(DOC_CLASS_GROUP_ID);

CREATE GENERATOR
GEN_DOC_CLASS_HISTORY_GROUP_ID;
SET TERM ^ ;

CREATE TRIGGER DOC_CLASS_HISTORY_GROUP_BI
FOR DOC_CLASS_HISTORY_GROUP
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  IF (NEW.DOC_CLASS_HISTORY_GROUP_ID IS
NULL) THEN
    NEW.DOC_CLASS_HISTORY_GROUP_ID =
GEN_ID(GEN_DOC_CLASS_HISTORY_GROUP_ID,1);
END

^

SET TERM ; ^

/*
The different class types of documents
*/
CREATE TABLE DOC_CLASS_TYPE
(
       DOC_CLASS_TYPE_ID     INTEGER not
null,
       TYPE_DESCRIPTION      varchar(100)
not null
);

alter table DOC_CLASS_TYPE
add constraint PK_DOC_CLASS_TYPE
primary key (DOC_CLASS_TYPE_ID);

CREATE GENERATOR GEN_DOC_CLASS_TYPE_ID;
SET TERM ^ ;

CREATE TRIGGER DOC_CLASS_TYPE_BI FOR
DOC_CLASS_TYPE
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  IF (NEW.DOC_CLASS_TYPE_ID IS NULL) THEN
    NEW.DOC_CLASS_TYPE_ID =
GEN_ID(GEN_DOC_CLASS_TYPE_ID,1);
END

^

SET TERM ; ^

/*
The types in every group
*/
CREATE TABLE DOC_CLASS_GROUP_TYPE
(
       DOC_CLASS_GROUP_ID    INTEGER not
null,
       DOC_CLASS_TYPE_ID     INTEGER not
null
);

alter table DOC_CLASS_GROUP_TYPE
add constraint UNQ_DOC_CLASS_GROUP_TYPE
unique
(DOC_CLASS_GROUP_ID,DOC_CLASS_TYPE_ID);

alter table DOC_CLASS_GROUP_TYPE
add constraint FK_DOC_CLASS_GROUP_TYPE_GR
foreign key (DOC_CLASS_GROUP_ID)
references
DOC_CLASS_HISTORY_GROUP(DOC_CLASS_HISTORY_G
ROUP_ID);

alter table DOC_CLASS_GROUP_TYPE
add constraint FK_DOC_CLASS_GROUP_TYPE
foreign key (DOC_CLASS_TYPE_ID)
references
DOC_CLASS_TYPE(DOC_CLASS_TYPE_ID);

/*
```

```
The document classification
*/
CREATE TABLE DOC_CLASS
(
       DOC_ID  INTEGER,
       DOC_CLASS_GROUP_ID    INTEGER,
       DOC_CLASS_TYPE_ID     INTEGER
);
/*
The different checks
*/
CREATE TABLE DOC_CLASS_CHECK
(
       DOC_CLASS_CHECK_ID    INTEGER not
null,
       CHECK_DESCRIPTION     varchar(100)
not null,
       CHECK_SQL      varchar(2048) not
null,
       DEFAULT_PARAMETERS    varchar(1024)
);

CREATE GENERATOR GEN_DOC_CLASS_CHECK_ID;
SET TERM ^ ;

CREATE TRIGGER DOC_CLASS_CHECK_BI FOR
DOC_CLASS_CHECK
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  IF (NEW.DOC_CLASS_CHECK_ID IS NULL) THEN
    NEW.DOC_CLASS_CHECK_ID =
GEN_ID(GEN_DOC_CLASS_CHECK_ID,1);
END

^

SET TERM ; ^

alter table DOC_CLASS_CHECK
add constraint PK_DOC_CLASS_CHECK
primary key (DOC_CLASS_CHECK_ID);

/*
Table with all the checks for the types
*/
CREATE TABLE DOC_CLASS_TYPE_COND
(
       DOC_CLASS_TYPE_COND_IDINTEGER not
null,
       DOC_CLASS_TYPE_ID     INTEGER not
null,
       CONJUNCT_NO    INTEGER not null,
       DOC_CLASS_CHECK_ID    INTEGER not
null,
       CHECK_VALUE    SMALLINT not null,
       PARAMETERS     varchar(1024)
);

alter table DOC_CLASS_TYPE_COND
add constraint PK_DOC_CLASS_TYPE_COND
primary key (DOC_CLASS_TYPE_COND_ID);

alter table DOC_CLASS_TYPE_COND
add constraint FK_DOC_CLASS_TYPE_COND_TYPE
foreign key (DOC_CLASS_TYPE_ID)
references
DOC_CLASS_TYPE(DOC_CLASS_TYPE_ID);

alter table DOC_CLASS_TYPE_COND
add constraint FK_DOC_CLASS_TYPE_COND_CHK
foreign key (DOC_CLASS_CHECK_ID)
references
DOC_CLASS_CHECK(DOC_CLASS_CHECK_ID);

CREATE GENERATOR
GEN_DOC_CLASS_TYPE_COND_ID;
SET TERM ^ ;

CREATE TRIGGER DOC_CLASS_TYPE_COND_BI FOR
DOC_CLASS_TYPE_COND
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  IF (NEW.DOC_CLASS_TYPE_COND_ID IS NULL)
THEN
    NEW.DOC_CLASS_TYPE_COND_ID =
GEN_ID(GEN_DOC_CLASS_TYPE_COND_ID,1);
END

^

SET TERM ; ^
```