Т Е Х Н И Ч Е С К И У Н И В Е Р С И Т Е Т – В А Р Н А

Самуил Владимиров Николов

Програмни среди за генериране на приложения

А В Т О Р Е Ф Е Р А Т
на дисертация за получаване на образователна и научна степен "Доктор"

Научен ръководител: доц. д-р инж. Анатолий Антонов

Рецензенти:

1. .....

2. .....

Варна, 2013 г.

Дисертационният труд е обсъден на 23.10.2013 г. в катедра "Компютърни науки и технологии" и насочен за защита.

Автор: Самуил Владимиров Николов
Заглавие: Програмни среди за генериране на приложения

Т Е Х Н И Ч Е С К И У Н И В Е Р С И Т Е Т – В А Р Н А
Самуил Владимиров Николов

Програмни среди за генериране на приложения

А В Т О Р Е Ф Е Р А Т

на дисертация за получаване на образователна и научна степен "ДОКТОР"

Варна, 2013 г.

Дисертационният труд съдържа 139 страници, включително 34 фигури и 6 таблици, оформени в 4 глави, заключения и списък на използваната литература от 288 заглавия, от които 1 на кирилица и 287 на латиница.

Защитата на дисертационния труд ще се състои на …..…….г. от ….. ч. в …… на открито заседание на жури сформирано със заповед на Ректора №.....…/2013 г.

Материалите по защитата (дисертацията, рецензиите и становищата) са на разположение на интересуващите се във ФД "Докторанти", стая 318 НУК.

Актуалност на проблема

В софтуерната индустрия има набор често повтарящи се проблеми свързани с разработването на програмни системи. Това са излизане на разработвания проект извън бюджета, времевата рамка, грешки в крайния продукт или лошо качество на програмата - което води до трудна поддръжка. За решаването на тези проблеми, софтуерната индустрия разработва нови и нови методи и средства за разработка, които да подпомогнат разработчиците при изпълнение на поставените им задачи. Създаването на такива средства за разработка, улесняващи работата на програмистите и ускоряващи разработването на нови продукти е от първостепенно значение за успеха при реализацията на всеки един проект.

Цел и задачи на изследването

Цел: Да се изследват възможностите и проблемите пред една среда, която генерира приложения и по-специално - информационни системи. Идеята е те да бъдат генерирани на базата на скриптов език, описващ онтологии, като същността на разработването на приложения посредством средата да се състои в моделиране на онтологичната структура на информационната система и формалното й описание посредством скриптовия език.

Задачи:

1. Разработване на метод за създаване на информационни системи, основани на онтологични класове;
2. Създаване на програмен модел за имплементация на онтологични класове в средата за генериране на информационни системи;
3. Създаване на подход за запис на инстанции на онтологични класове в релационна база от данни;
4. Създаване на метод за проверка и актуализиране на записани инстанции на онтологични класове;
5. Създаване на модел за представяне на проблемите за анализ на съответствие и имплементация на изчислител, основан на средата за генериране на информационни системи.

Обект и предмет на изследването

Обект на изследването: Софтуерните приложения - структурата и начините за разработването им.

Предмет на изследването: Автоматизираното разработване на приложения посредством помощни среди.

Методи на изследване

Извършени са теоретични изследвания на моделите за създаване на информационни системи и начините за представяне и обработка на данните в тях. Разгледани са недостатъците на съществуващите методологии. Анализирани са възможностите за моделиране на информационни системи посредством онтологии. Направени са изводи относно възможностите за създаване на среда за генериране на информационни системи, основана на онтологични данни.

Разработена е експериментална среда за генериране на информационни системи на база на C++ ядро и продукционна система. Разработената среда е внедрена в български и чуждестранни фирми и банки.

Научна и практическа новост

В резултат от проведените теоретични и експериментални изследвания в съответствие с целта и задачите на дисертационния труд могат да бъдат дефинирани следните основни приноси:

1. Създадени са метод и архитектура на среда за създаване и поддържане на информационни системи, основани на онтологични класове.
2. Към средата за генериране на информационни системи е създаден метод, синтаксис за представяне на ограничителни условия и имплементация на система за проверка съответствието на потребителските действия с тези условия.
3. Създаден е домейн-специфичен език за бързо моделиране на онтологични класове и метод за свързването им в информационни системи, генерирани от средата.
4. Създаден е подход за съхранението в релационна база от данни на твърдителната част от онтологичните данни на средата за генериране на информационни системи.
5. Разработен е метод за поддръжка на различни версии на софтуера в средата за генериране на информационни системи чрез автоматично опресняване на инстанции и автоматично адаптиране към структурни промени на онтологиите.
6. Чрез средата за генериране на информационни системи са реализирани информационни системи в областта на финансовите изчисления и анализи и проверки на ограничения и съответствия, които се основават на онтологични класове.

Реализация на резултатите

На базата на теоретичните разработки в дисертационния труд е разработена среда за генериране на информационни системи и са реализирани следните информационни системи:

- Система за оценка на финансови инструменти;
- Система за управление на пазарен риск;
- Система за оценка на капиталова адекватност на застрахователни компании (Solvency II);
- Система за оценка на съответствие на финансови портфейли към регулаторни изисквания;
- Система за оценка на частен и корпоративен рейтинг;
- Система за управление на финансови активи и пасиви;
- Скоринг система;
- Система за оценка на банков операционен риск.

След отчисляването с право на защита, разработените информационни системи са внедрени в банки от България и Босна и Херцеговина, както и използвани от швейцарски, български и босненски фирми за което има приложени документи към дисертацията.

Апробация на резултатите

Резултатите от теоретичните и експериментални изследванията са представени посредством десет публикации на шест международни конференции към които са издадени сборници с доклади и в едно научно списание. Статиите са на английски език.

Обем на дисертацията

Дисертационният труд съдържа 139 страници, включително 34 фигури и 6 таблици, оформени в 4 глави, заключения и списък на използваната литература от 288 заглавия, от които 1 на кирилица и 287 на латиница. 18 източника са в електронен вариант.

Съдържание на дисертационния труд

Увод

В софтуерната индустрия има набор често повтарящи се проблеми свързани с разработването на програмни системи. Това са излизане на разработвания проект извън бюджета, времевата рамка, грешки в крайния

продукт или лошо качество на програмата - което води до трудна поддръжка. В почти всички случаи проблемът на проектите е един и същ - сложността им. Най-често срещаните проблеми са грешките в продукта, недофинансирането, забавянето, невъзможността да се разбере изцяло работата му, както и текучеството на персонал.

За решаването на тези проблеми без да се прави компромис с качеството на продуктите, времето им за достигане до пазара и да се влезе във все по-намаляващите бюджети, софтуерната индустрия разработва нови и нови методи и средства за разработка, които да подпомогнат разработчиците при изпълнение на поставените им задачи.

Целта на дисертацията е да се изследват възможностите и проблемите пред една среда, която генерира приложения и осигурява качествено и бързо разработване, както и лесна поддръжка при висока сложност на приложенията.

Глава 1. Концепции при изграждането на среди за генериране на приложения.

В първата глава на дисертационния труд са изследвани проблемите при разработката на приложения и известните в литературата начини за тяхното решаване. Разгледани са основните варианти на генериране на софтуерни приложения и вътрешната им структура. Обърнато е внимание на модерните структурни варианти за създаване на приложения – обектно-ориентирани, базирани на компоненти и базирани на модели. Разгледани варианти за обобщено моделиране на приложения.

В края на главата са направени следните изводи:

- Базираното на модели разработване е мощен и широко разпространен начин за бързо създаване на приложения;
- Разработването на среда за генериране на приложения на тази база би изисквало моделиране на приложения чрез онтологии;
- Онтологиите са подходящи за моделиране на информационни системи (ИС).

Въз основа на направените изводи се дефинира целта на дисертационния труд и задачите на изследването.

Глава 2. Представяне на информационни системи с онтологии

Разгледани са онтологиите като вариант за генериране на ИС и са изучени видовете, начините за създаването и представянето им, вариантите им на приложение. Като пример за модерна среда за генериране е разгледана

Eclipse Modelling Framework, която генерира програмен код по Unified Modelling Language (UML) описание. Разгледани са недостатъците й, които пречат за пълноценното генериране на завършени приложения. Анализирани са начините за разширяване на приложните възможности на онтологиите посредством правила.

Втора глава завършва със следните изводи:

- Използването на онтологиите по време на разработка и по време на изпълнение се считат за изолирани дейности, а анализираните разработки следват стандартни подходи за моделиране на системата или генериране на код чрез конвертиране към UML;
- Онтологиите се използват основно като концептуална база за модел-базирано разработване, а не като директен вход за автоматизиран процес на разработка;
- Според изследваните литературни източници проектирането и разработката на цели приложения, използващи онтологии по време на целия цикъл на разработка, не е реализиран;
- Разширяването на онтологиите посредством допълнителни факти и правила, служещи за дефинирането на потребителски интерфейс, комуникация и управление на софтуерно ядро, може да бъде обект на настоящия труд.

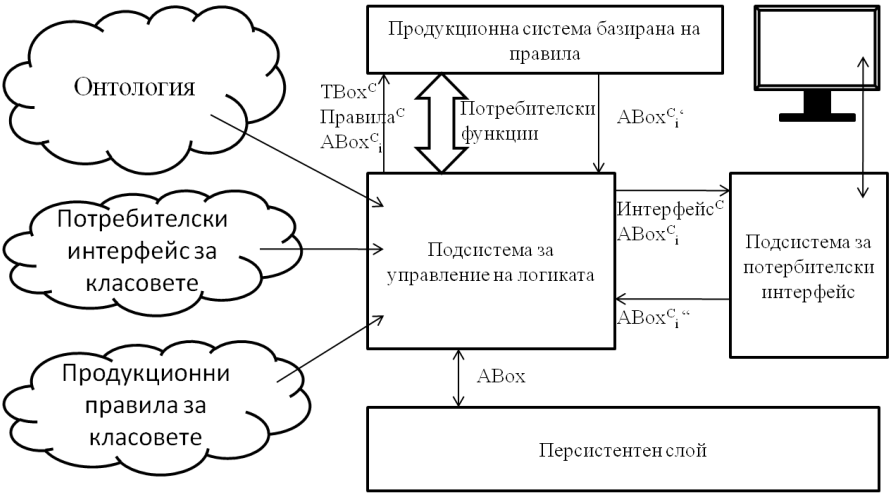
Глава 3. Разработка на среди за генериране на приложения

3.1. Метод за създаване на информационни системи, основани на онтологични класове

3.1.1. Обзор на предлаганото решение

На фигура 1 е показана обща схема на предлаганото решение, представено като ИС, базирана на онтологии. Основната идея на подхода е онтологията да бъде разширена посредством дефиниции на потребителски интерфейс и продукционни правила за класовете, които да управляват системата и да съдържат бизнес логиката на информационната система.

Подсистемата за управление на логиката обособява терминологичните знания и правилата, отнасящи се до един клас C ($TBox^C$ и $правила^C$) от онтологията и ги зарежда в продукционната система (ПС), базирана на правила заедно с данните на една инстанция от разглеждания клас ($ABox^C_i$). Стартирането на ПС позволява да се обработят наличните стойности на променливите, дефиниращи инстанцията, описвана от $ABox^C_i$ чрез правилата. Това се прави като се приложат ограниченията върху свойствата на инстанцията и се извършат изчисленията върху някои от тях, описани в

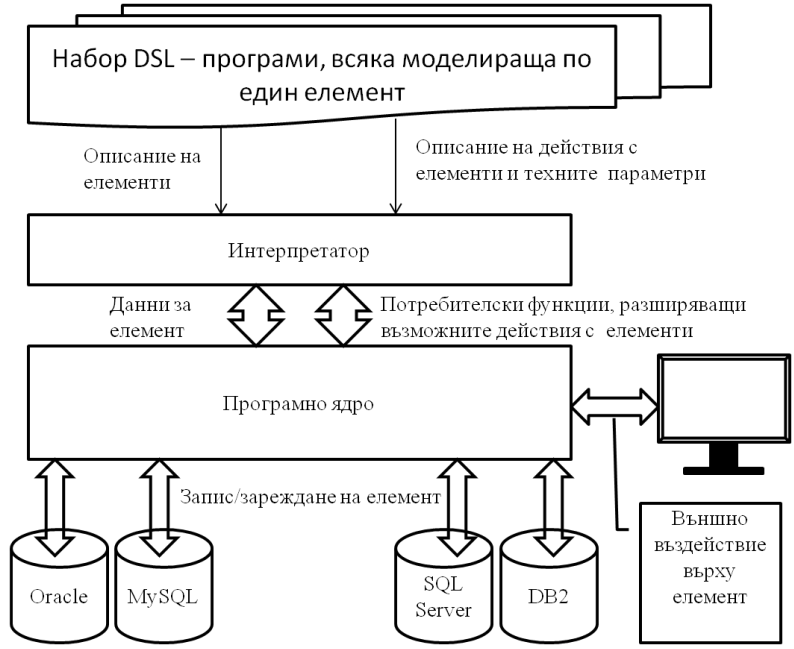


Фигура 1. Обща схема на предлаганото решение разглеждано като информационна система, базирана на онтологии

правилата. Резултатът от стартирането на продукционната система е променената инстанция на класа $ABox^{C}_{i}$'. Подсистемата за управление на логиката добавя и потребителски функции към ПС, като така разширява значително възможните действия, извършвани в правилата, като дори позволява изпълнението на правилата да предизвика промени в цялата ИС.

Потребителският интерфейс се генерира посредством отделеното от подсистемата за управление на логиката описание за един клас и конкретна инстанция на класа (интерфейс^{C} и $ABox^{C}_{i}$). Описанието на потребителския интерфейс за класа специфицира начина, по който всички свойства и връзки на дадения клас трябва да се изобразят на екрана на потребителя. Това се прави с цел крайния потребител на ИС да получи стандартизиран потребителски интерфейс, който да скрие онтологичната същност на данните. След редакция на някои от свойствата, към логическата подсистема се връща променената инстанция на класа $ABox^{C}i$", която може да се запише посредством персистентния слой или да се подаде като вход на ПС.

Другата гледна точка върху системата - като модел-базирана среда за генериране на ИС - е представена на фигура 2. Основата на средата е съставена от интерпретатор - ПС базирана на правила - зареждащ специфични за сферата - Domain Specific Language (DSL) програми. Това са моделите, които са комбинация между онтологичните данни, описанието на потребителския интерфейс и правилата за обработка на всеки клас от



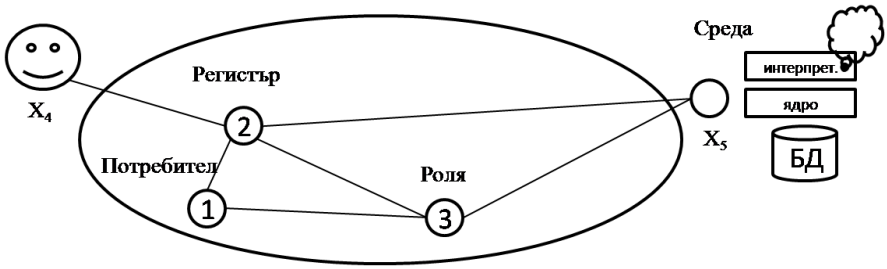
Фигура 2. Обща схема на предложеното решение разглеждано като модел-базирана среда за генериране на приложения

онтологията. Средата съдържа и програмно ядро, което е обединение между логическата, персистентната и системата за потребителски интерфейс. Ядрото предоставя възможности за манипулиране на инстанциите на класовете по време на изпълнение, генерира потребителския интерфейс, управлява зареждането и менажирането на моделите в средата. Интерпретаторът зарежда и парсира моделите и изпълнява правилата заложени в тях.

3.1.2. Проектиране на информационни системи посредством средата

Проектирането на ИС за средата започва с проектиране на онтология, която моделира съответната сфера (домейн). На фигура 3 е показана примерна онтология за реализиране на система за сигурност. Тя се състои от три класа - регистър за съхранение на наличните роли, потребители и системи в средата и отделни онтологични класове за потребителите и ролите.

При появата на нов служител на организацията или добавянето на нова система към средата, те трябва да бъдат регистрирани в регистъра 2 (обозначен по-долу с X_2). При регистриране на потребител в регистъра, трябва да му бъде създаден вътрешен модел, показан на фигурата с възел 1



Фигура 3. Система за сигурност

(обозначен по-долу с X_1) като към вътрешния модел се асоциират n на брой роли в средата. При необходимост от нови роли, те трябва да бъдат добавени в регистъра. Системата за сигурност на фигурата бележим с $\sigma_s=(C,B_C)$, като:

$C(\sigma_s)=\{X_1, X_2, X_3\}$

$B_C(\sigma_s)=\{(X_1,X_2),(X_2,X_3),(X_1,X_3)\}$ **(3.1.1)**

Където $C(\sigma_s)$ е множество от елементи и $B_C(\sigma_s)$ са множество зависимости между елементите на системата. Върху свойствата на елементите в системата за сигурност важат множество валидиращи правила, съответно $L_1(X_1)$, $L_2(X_2)$, $L_3(X_3)$. С $S_i(X_i)$ се обозначават всички възможни стойности, които свойствата $V_i^{1..n}$ на елементите X_i могат да приемат, а с $S_{Li}(X_i)$ – това подмножество на $S_i(X_i)$, за което:

$\forall l_i^j(X_i) \in L_i(X_i) \mid l_i^j(X_i) \rightarrow \{валидно\}.$ **(3.1.2)**

За обясненията се използват следните определения:

състав на системата σ_s в t_0 е $\tilde{C}$ (σ_s,t_0)

външна среда на σ_s в t_0 е $\tilde{E}(\sigma_s,t_0)$

структура на σ_s в t_0 е $\tilde{S}(\sigma_s,t_0)$

вътрешните зависимости са $\bar{B}$ (σ_s,t_0)

външните зависимости са $\hat{B}(\sigma_s,t_0)$

Нека състоянията на елементите във времето се нарича история. Историята на σ_s в момента t_0 е $h(X_i)=\{<t_0,F_i^0>\}$, като в последващите моменти t_1, t_2 и т.н., тя е съответно $\{<t_0,F_i^0>, <t_1,F_i^1>\}$, $\{<t_0,F_i^0>, <t_1,F_i^1>, <t_2,F_i^2>\}$, …, където F_i^j е функция на състоянието на класа X_i (j е номер на състоянието).

Всеки елемент $X_i \in C(\sigma_s)$ се моделира посредством една DSL програма за ПС – съдържаща факти и правила. Всяко свойство V_i^k на X_i (k е номер на свойството) се моделира посредством шаблони на факти в DSL програмата, описващи име, тип и стойност на свойството, като се указва и стойност по подразбиране. Освен шаблони на факти, към програмата се добавят и факти, указващи начина, по който свойството V_i^k трябва да се визуализира върху екрана на потребителя. Валидиращите правила $L_i(X_i)$ се моделират посредством правила в DSL програмата. Зависимостите между двойки елементи от $B_C(\sigma_s)$ - (X_i, X_j) се моделират посредством съставни свойства (

факти с много стойности – multifields) и правила, които ги обработват. Всяко въздействие X_i D X_j се моделира посредством свойство $V_i^k \in X_i$, което указва набор имена на функции на състоянието на X_j, които са зависими от текущата функция на състоянието на X_i. При необходимост се добавя и правило, съдържащо потребителска функция за презареждане на текущо изпълняваната DSL програма в програмата, описваща X_j. Потребителската функция има вида:

$\psi(X_j,x_j^r) \mid \quad x_j^r \in V_i^k$ **(3.1.3)**

където r е номер на идентификатор на инстанции на X_j , $1 \le r \le n$, а n е броя имена на инстанции, съдържани в свойство V_i^k.

Външните въздействия $\hat{B}$ се извършват посредством потребителския интерфейс – чрез него се добавят нови идентификатори на инстанции към системата (примерно потребители) или чрез достъп до базата данни.

Историята $h(X_i)$ на елемента X_i се разглежда в два аспекта. Разпознаваеми са както различните променливи (функции на състоянието) на X_i – x_i^r, така и различни техни версии в различните моменти на времето $x_i^r(t_{0..n})$. Към името на идентифицируемата инстанция – x_i^r се асоциират и всичките й различни състояния във времето.

Моделирането на събития за преминаване от състояние s_i към състояние s_i' с обозначение $<s_i,s_i'>$, където $s_i, s_i' \in S_i(X_i)$ в средата става със стартиране на правилата, описани в DSL програмата за елемента X_i. Резултат от събитието може да бъде и зареждането на нова DSL програма, описваща X_j, която да моделира свързаността между X_i и X_j.

Сред предимствата на средата за генериране на информационни системи е лесната структурна разширяемост. Добавянето на нов елемент към системата се състои в дефинирането на DSL програма, моделираща елемента и установяване на връзка към него посредством свойства в съществуващите модели. За системата по време на изпълнение може да се твърди, че:

$C(\sigma_s) \neq const$ **(3.1.4)**

$B_C(\sigma_s) \neq const$

, което е и една от целите на разработката - устойчивост на промени по време на изпълнение.

3.1.3. Генериране на информационни системи посредством средата

За да се осигури интеракцията на потребителите с така дефинираната среда, в нея се въвеждат два интерфейсни компонента - редактор на инстанция и списък с инстанции. Редакторът показва на екрана на потребителя функционалната схема на една инстанция на елемента $x_i^r(t_m) \in \tilde{C}(\sigma_s,t_m)$, използвайки описанията на потребителския интерфейс в DSL програмата, съответстващ на елемента X_i. За всяко свойство $V_i^k \in X_i$ в

редактора на инстанции се генерира съответстващия му контрол, описан в DSL, като стойността в него се взима от конкретната стойност на свойството V_i^k на инстанцията x_i^r. Списъкът с инстанции показва подмножество от всички налични инстанции на X_i или част от историята на X_i. Показват се всички x_i във времето, за които:

$x_i(t) \in h(X_i) \cap f(x_i)$ **(3.1.5)**

, където $f(x_i)$ е функция, която връща "истина" ако x_i присъства в множеството подадено като втори параметър на предходното извикване на потребителската функция $\psi(X_i, x_i^r)$. В точки 3.1.3.1 и 3.1.3.2 са разгледани по-подробно функционалните възможности на двата интерфейсни елемента.

3.1.4. Изводи

Създадена е архитектура (фигура 1 и 2), която позволява бързо разработване на информационни системи, защото процеса на създаване се свежда до форматиране на проектираните онтологични данни във факти за ПС и добавянето на правила към тях. Същата е причината и за по-лесното модифициране на ИС, генерирани чрез средата - то се свежда до корекцията на факти, описващи потребителския интерфейс или поправката на дадено правило в модела. Съществено предимство е, че това може да става дори по време на изпълнение. Като заключение може да се подчертае и предимството, че потребителския интерфейс е организиран по начин, който скрива онтологичната същност на средата за крайния потребител.

3.2. Програмен модел за имплементация на онтологични класове в средата за генериране на информационни системи

В настоящата точка се разглежда начина, по който се представят онтологичните класове като DSL програми в разглежданата среда за генериране на информационни системи. Домейн-специфичният език е от типа вграден, като разширява възможностите на продукционната система, свързана в средата за генериране на информационни системи и позволява дефинирането, визуализирането и изчислението на един клас от онтологията в нея посредством специализирани конструкции, използвани от ядрото на системата.

3.2.1. DSL програма за представяне на онтологии

Всяка DSL програма, описваща даден клас в средата съдържа:

Шаблони на факти, описващи свойствата и връзките на класа;

Факти, описващи как се визуализират свойствата при генериране на потребителския интерфейс;

Правила за обработка на декларираните свойства, за манипулиране на потребителския интерфейс и за издаване на команди към програмното ядро.

Описаните части на програмата са представени в ПС посредством специфични структури. Тези структури позволяват ефективна комуникация с ядрото на средата за генериране на ИС - лесно създаване на прототип на потребителския интерфейс и последващото му свързване с бизнес логиката на генерираната информационна система. За целта от ядрото се предоставят набор потребителски функции, които правилата в DSL програмата може да извикват, за да управляват потребителския интерфейс, базата данни на генерираната информационна система и взаимодействията между класовете в нея.

3.2.1.1. Представяне на свойства и връзки между класовете

Първата част на програмата е декларирането на свойствата и връзките на класа, който се представя. Функционалните свойства (отношенията с ограничение 1) се моделират в DSL програмата посредством опростени именувани факти - deftemplate конструкции (шаблони на факти), които съдържат четири слота - тип, низ, число с плаваща запетая и списък. Името на шаблона съдържа името на свойството на класа. Типът указва как да бъдат третирани другите слотове в конструкцията. Низовият и численият слотове съдържат стойностите на свойството - като в някои случаи са нужни и двата типа. Списъкът служи за налагане на ограничения върху стойностите в предните два слота и за дефиниране стойностите на фиктивни класове. В шаблоните могат да се указват стойности по подразбиране, които се използват при началното създаване на инстанции на класа, както и в случаите, когато за даден слот няма подходяща стойност.

Отношенията с ограничения по-големи от 1 се моделират в описвания DSL посредством комплексни шаблони deftemplate. Най-често последните се визуализират като таблици на потребителския интерфейс, като колоните на таблиците се представят чрез полета на продукционната система, съдържащи много стойности (multifield). По този начин се декларират и връзки между класове. На ниво инстанции, всяка връзка между два класа се представя посредством списък от идентификаторите на инстанциите на свързания клас.

3.2.1.2. Дефиниране на данните за потребителския интерфейс

Особеното случая е представянето на интерфейса посредством факти и свързването на част от дефинирания потребителски интерфейс към шаблоните, описващи свойствата на класа. Описанието на потребителския интерфейс съдържа набор факти, всеки от които се състои от идентификатор

на контрол, тип на контрола, координати и размер на контрола, както и име на шаблонния факт, който представя свойство.

3.2.1.3. Дефиниране на правилата в DSL програмата

Като пример в точката се разглежда начинът за дефиниране на правила в продукционната система CLIPS.

Валидиращите правила променят текущите факти в базата знания на CLIPS посредством действията в десните си части по такъв начин, че да се постигне валидно състояние. Вътрешните зависимости между елементите в системата - X_i D X_j се моделират посредством свойство в X_i, което съдържа списък с идентификаторите на инстанциите на X_j, които са свързани с него. Събитията от вида $<s_i,s_i'>$ могат да се случат, когато данните в свойствата на моделирания клас се модифицират по някаква причина от ядрото. Това може да стане вследствие на промяна на стойностите на контролите, визуализирани върху потребителския интерфейс или поради зареждане на нови данни от базата. При настъпване на събитие, в базата знания се добавят нов набор конкретни стойности на свойствата на класа и събитието се обработва от съответния набор правила.

3.2.2. Алгоритъм на работа на DSL програмите

При зареждане на DSL програма, тя се разделя на трите си съставни части. Описанието на потребителския интерфейс се прочита от ядрото и по него то генерира съответстващите диалогови, групови кутии и контроли на екрана на потребителя в редактора на инстанции. Следва инициализиране на стойностите на свойствата на класа, описван от DSL програмата, като за това има два варианта: да се използват стойностите по подразбиране, декларирани в шаблоните или да се зададат от ядрото. То може да ги вземе от контролите на екрана на потребителя, ако последния е имал възможността да ги модифицира, или да ги зареди от данните в отворената инстанция посредством списъка с такива. След като се инициализират всички свойства, се стартират правилата на DSL програмата. Резултатът от това стартиране е или зареждането на нов модел, или нови стойности на свойствата, които се прочитат и изпращат към ядрото за визуализация, редакция и повторна калкулация.

3.2.3. Изводи

Чрез описания DSL се управлява програмното ядро на средата за генериране на информационни системи. Представената структура на DSL програмата позволява бързото моделиране на класове и свързването им посредством преходи в информационни системи, генерирани от средата.

3.3. Подход за запис на инстанции на онтологични класове в релационна база от данни

В точката се представя подход за персистиране на твърдителната част от онтологията (ABox) - на инстанциите на онтологичните класове. Базата е проектирана така, че да поддържа хисторизация на данните, да записва информация за потребителите, които работят с тях и действията им и да поддържа външни средства за генериране на отчети като crystal reports.

3.3.1. Начини за персистиране на онтологични данни

В точката са разгледани основните начини за персистиране на онтологични и обектни данни в литературата - в обектни бази, във файл с формата Resource Description Framework (RDF) и в релационни база данни. От своя страна, начините за запис в релационна база са таблица за свойство, таблица за клас и вертикална таблица, която съдържа тройки от вида <обект, свойство, субект(стойност)>.

Подходът, който е използван при разработката на средата за генериране на ИС е базиран на модификация на варианта “вертикална таблица” и използва четири таблици, три от които са вертикални, а четвъртата съдържа допълнителна информация за подпомагане на имплементирането на специфични изисквания и бързо откриване и реконструкция на записаните инстанции. Вертикалните таблици позволяват разширяемост и лесни промени по структурата на записаните обекти. Подходът е подходящ за менажиране на големи класове с много свойства, какъвто е разглежданият в дисертацията случай.

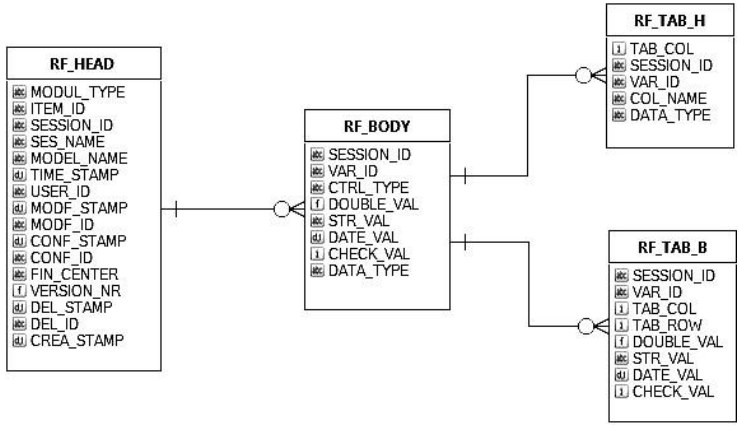
3.3.2. Обща схема на решението

Основните операции върху данните в средата за генериране на информационни системи се състоят в извличането или записването на една или повече инстанции на класовете от/в базата. Поради тази причина реконструкцията на инстанция е от първостепенно значение. Нужен е и достатъчно бърз запис на нови данни в базата. Реконструкция на обект може да става чрез намиране на идентификатори на инстанции по зададени име на клас, идентификатор на обект, домейн или организационна единица. С намерените идентификатори е лесно да се реконструират инстанциите посредством вертикална организация на главната таблица. Фигура 4 показва общ план на използваната структура на базата. Таблицата RF_HEAD се използва като таблица за реконструкция на инстанции, като съпоставя на най-често търсените параметри идентификатор на инстанция. Таблицата RF_BODY е вертикална и съдържа функционалните (с кардиналност 1)

свойства на записаните инстанции, индексирани според идентификаторите им. Другите две таблици, които също са вертикални, са индексирани по идентификатора на инстанцията (SESSION_ID) и идентификатора на свойство (VAR_ID). Те съдържат свойствата в онтологията, които имат кардиналност по-голяма от 1 – представени в средата за генериране на информационни системи като таблици и метаданни за тях. Таблицата RF_TAB_H съдържа информация за колоните на табличните контроли – техните тип и значение, а таблицата RF_TAB_B съдържа данните от клетките им.

3.3.3. Съхранение на метаданни за класовете

В информационните системи е важно да се знае кой, кога и какво е направил с даден набор данни. Също така е добре и да се поддържат различни версии, както и да има възможности за логическо отделяне на данните. Това може да се обобщи като метаданни за всяка една инстанция на онтологичните класове, които в представяното решение са отделени в специална таблица – RF_HEAD.



Фигура 4. Схема на релационната база данни, ползвана за съхранение на обекти в средата за генериране на информационни системи

3.3.4. Представяне на наследяване

Към всеки идентификатор на обект могат да се асоциират няколко различни идентификатора на инстанции. Те могат да представляват времевото развитие на обекта, описван от идентификатора в ITEM_ID – ако са създадени от еднакви класове. Когато обаче данни от различни класове са

асоциирани към един и същ идентификатор на обект, то те представят части от описание на обект от съставен клас. Такива съставни класове се реконструират като се изтеглят всички идентификатори на инстанции, асоциирани към един и същ идентификатор на обект и след това се изтеглят данните, асоциирани към тях от другите три таблици.

В точка 3.3.5 на дисертацията са разгледани метаданните, служещи за представяне на историческото развитие на записаните обекти, а в точка 3.3.6 на дисертацията е представен начинът, по който полето "версия" от метаданните се ползва за разрешаване на конфликти при паралелен достъп до данните. В точка 3.3.7 на дисертацията е описан начина, по който се реконструират класовете посредством вертикалните таблици.

За да се генерират смислени отчети от записаните в системата данни за табличните контроли се използва на вертикалната таблица с метаданни - RF_TAB_H, която съдържа информация за колоните на таблиците, чиито данни са описани в таблицата RF_TAB_B. Тя е представена в точка 3.3.8 на дисертацията.

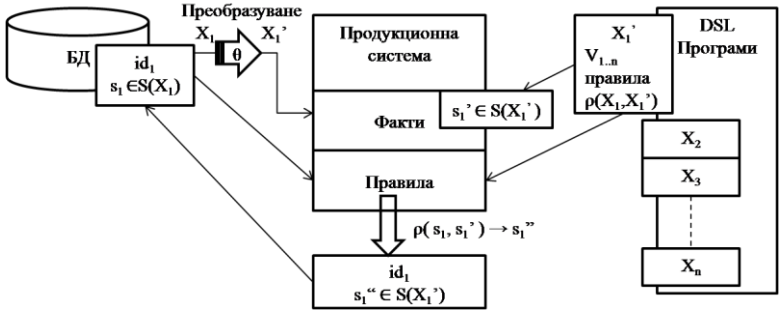
3.3.9. Изводи

В точка 3.3 е използвана модификация на подход за съхранение наречен “вертикална таблица”. Използваната архитектура на базата данни подсигурява бързата адаптивност и повишава гъвкавостта на генерираните ИС, като позволява съхранението на обекти от един и същи клас в различни моменти от време и с различен състав. Структурата спомага за по-лесно генериране на външни репорти.

3.4. Метод за проверка и актуализиране на записани инстанции на онтологични класове

В точка 3.4 е разгледан начина за поддръжка на различни версии на софтуера в средата за генериране на ИС. Управлението на версиите се извършва без да се прекъсва работата на потребителя - чрез подмяна на DSL програмите, моделиращи текущата система. Тъй като DSL програмите описват и структурата на базата данни, една промяна на версията на софтуера на практика се свежда до актуализиране на вече записаните в базата данни инстанции на онтологичните класове. Точка 3.4.1 на дисертацията представя обзор на литературата за различните начини за справяне с разглеждания проблем.

3.4.2. Общ вид на решението



Фигура 5. Авоматично опресняване на записана в базата данни инстанция.

На фигура 5 е показана работата на средата при зареждане в ПС на данни от инстанция, чийто клас X_1 е бил подменен с по-нова версия - X_1'. В базата данни съществува инстанция с идентификатор id_1, която описва конкретно състояние s_1 на свойствата на класа X_1 : $s_1 \in S(X_1)$. При зареждане в ПС обаче, DSL програмата която съответства на по-новата версия на този клас X_1' зарежда шаблони на факти, които съответстват на новия набор свойства ($V_{1..n}$) и правила за обработката им. Свойствата в DSL програмата, описваща класа X_1' не съответстват на записаните данни в базата и съответно последните не могат да бъдат добавени като стойности на шаблонните факти в ПС. За целта те се преобразуват по тривиален начин, означен с θ от свойства, съответстващи на описанието на X_1, към свойства, съответстващи на X_1':

$$\theta(X_1', s_1) \mid s_1 \in S(X_1), s_1' \in S(X_1') \rightarrow s_1' \qquad \textbf{(3.4.1)}$$

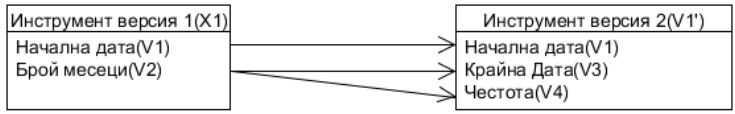
След това преобразуване, правилата в продукционната система се стартират с набора факти, съответстващи на стойностите s_1'. Към правилата в новата версия на класа X_1' има добавени такива, които да се стартират при непълнота на данните, съответстваща на разликата в атрибутите между X_1 и X_1' - $\rho(X_1, X_1')$. Тези правила извличат записаното в базата състояние s_1 и чрез прилагане на набор преобразуващи операции върху старите данни s_1, се постига ново състояние s_1'', което адекватно съответства на еволюцията на схемата, описана чрез прехода от X_1 към X_1':

$$\rho(s_1, s_1') \rightarrow s_1'' \mid \quad s1'' \in S(X_1') \qquad \textbf{(3.4.2)}$$

В точка 3.4.3 на дисертацията е описан тривиалния начин на преобразуване илюстриран с пример.

3.4.4. Правила за преобразуване на инстанции

Тези правила имат пълните възможности на езика на ПС, което отличава метода за опресняване от други подобни от литературата. Например, при нужда от съхранение на по-комплексен модел на финансов инструмент в базата данни, както е показано на фигура 6, полето "Брой месеци" трябва да бъде преобразувано в две нови – "Крайна Дата" и "Честота". Адекватна трансформация на схемата не може да стане без да бъде изчислена крайната дата чрез началната дата и броя месеци. .



Фигура 6. Нетривиално преобразуване на инстанция

Средата за генериране на информационни системи позволява опресняването на група от инстанции. Такова е необходимо, когато промяна в свойствата на един клас води до промяна в свойствата на свързаните с него класове от онтологията. Преобразуването е описано в точка 3.4.5 на дисертацията.

В точка 3.4.6 на дисертацията се представят възможните операции за промяна на онтологията и действията, които се предприемат от средата или DSL програмиста.

3.4.7. Изводи

Прехода към нова версия в средата на практика се свежда до определяне влиянието на подменени терминологични знания (TBox) върху твърдителните знания (Abox) на онтология. Описаният подход с правила постига максимално съответствие между данните на различните версии, защото позволява сложни операции и преобразувания. Използването на правилата е улеснено от структурата на информационните системи, генерирани посредством описваната среда. Представеният метод за опресняване позволява лесно и незабележимо за крайния потребител опресняване на версиите на софтуера, генериран посредством средата.

3.5 Модел за представяне на проверка за съответствие и имплементация в средата за генериране на информационни системи

В тази глава се разглежда използването на средата за генериране на информационни системи при решаване на конкретен практически проблем. Показва се, че средата е в състояние да генерира информационна система,

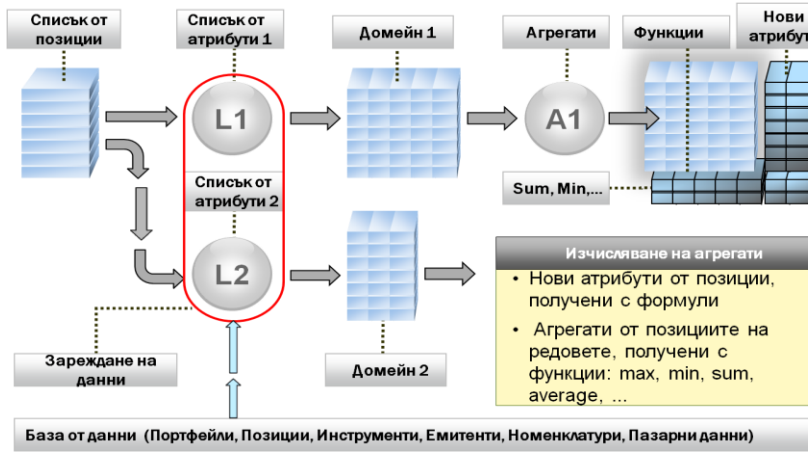
чиято основна цел е оценяване и проверка на съответствие с изисквания или стандарти (compliance check system).

В точки 3.5.1 и 3.5.2 на дисертацията са описани постановката и анализ на проблема.

3.5.3. Спецификация на дефиниционната област на ограничителните условия

Данните за позициите са разпределени в топологична структура – свързана дървовидна структура, в която се представят чрез списъци от инстанции, всяка от които е асоциирана към връх на тази структура. Списъците, които се асоциират към всеки подвръх на структурата се получават по два основни принципа- чрез дефиниране от потребителя и като резултати от операции върху вече дефинирани списъци. Използват се оператори за обединение, сечение, разлика и симетрична разлика.

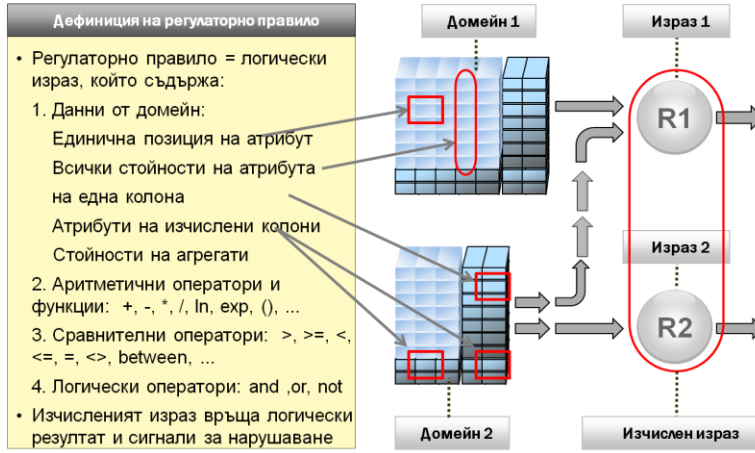
Всяка позиция, участваща в един списък се характеризира със списъци от атрибути - L1, L2 .., от всеки списък се продуцира отделен домейн от базата данни – Домейн1, Домейн 2, както е показано на фигура 7. Върху конкретен домейн могат да се приложат агрегационни функции и да се изчислят нови синтетични атрибути, показани на фигурата. Така се разширява съвкупността от атрибути на позицията, при това динамично. Върху съвкупност от позициите на двата вида атрибути могат да се приложат изчислителни функции-max,min,sum, avg…, с които да се получат допълнителни елементи и същите да се добавят към домейна.



Фигура 7. Метод за дефиниция на домейни

3.5.4. Дефиниция на регулаторни правила и дефиниция на дърво на оценяването

Правилата за регулация се представят като логически изрази, дефинирани върху множествата данни за позиции. Аргументи на регулаторния израз са съвкупност от стойности на даден атрибут, единични стойности на атрибут, атрибутни стойности, добавени чрез агрегационни



Фигура 8. Дефиниция на регулаторно правило

изчисления и агрегирани стойности от функции, както е показано на фигура 8. Между аргументите могат да се прилагат алгебрични функции, условни или циклични операции и оператори и да се описват изрази за включване в сравнителни условия. Изчислените стойности имат логически резултат удовлетворено и неудовлетворено, представени с true или false, като при неудовлетворимост се сигнализира потребителя за нарушаване на регулаторните правила. За всяко едно логическо условие може да се изчисли и степен на удовлетвореност / неудовлетвореност. Така дефинираните логически условия се включват в ограничителен израз (формула) посредством операторите and, or, not, or not, and not и скоби. В точка 3.5.5. на дисертацията е представен синтаксис на езика за дефиниране на ограничителните условия в Бакус-Наур форма. В него са дефинирани отделните елементи на регулаторното правило, както е показано на фигура 8. С помощта на този синтаксис потребителят на системата може да специфицира множества от данни и изрази, дефинирани между тях с използване на константи от различен тип.

3.5.6. Имплементация на изчислител на ограничителните условия в средата

След дефинирането на израз посредством средата или въвеждането му на ръка от потребителя, последния може да стартира компилация и проверка за удовлетворимост на израза. При това изразът се парсира от специален набор правила в текущо отворената програма за ПС. За домейните, включени в него се генерират факти, а за операциите помежду им – правила. След това последните се изпълняват и дават очаквания от потребителя резултат – дали текущо въведения му в системата портфейл отговаря на изискванията, които е вложил в израза.

При генерирането на правилата се ползва принципа на поддръжка на съжденията (reason maintenance). Така при премахване на някоя от предпоставките на дадено правило, ПС премахва автоматично и резултата от правилото, поддържайки логичността на базата си от знания.

3.5.7. Работа на изчислителя в реално време

При търгуване в реално време на борсата, описаните чрез факти свойства на домейните могат да се променят, да се унищожават или да се появяват нови. Всяка една функция в израза се реализира с правило, а не чрез използване на вградените еквивалентни по синтаксис функционалности на езика. Позицията, чиято стойност е подменена, влияе върху ограничен набор от правила. При задействането само на тези правила, се преизчислява удовлетвореността им. При неудовлетвореност или близки до

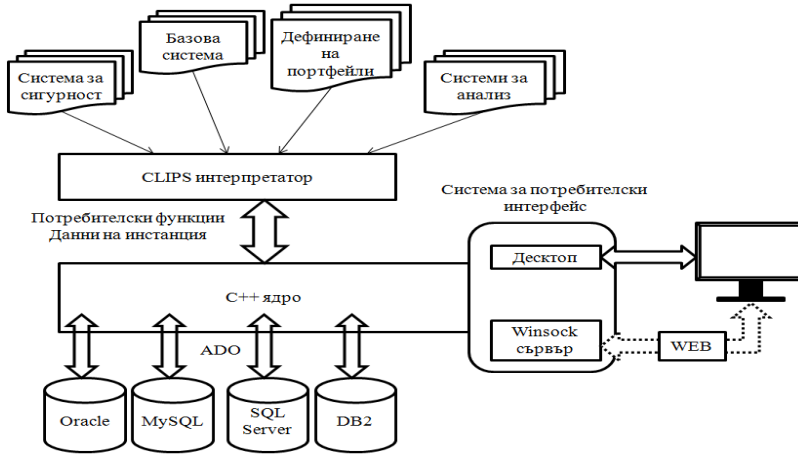
неудовлетвореност стойности, потребителят се уведомява за проблематичността на финансовата транзакция, която иска да извърши. Чрез ограничаването на броя на изпълнените правила се осигурява възможност дилърът да провери дали сделката му нарушава регулаторните изисквания преди да се промени текущото състояние на пазарното предлагане. В точка 3.5.8 на дисертацията е представен пример за дефиниция на ограничително условие.

3.5.9. Изводи и заключение

Предложеният подход позволява на втория етап от анализа да се ограничи изчислението само на правилата, които са логически свързани с промяната, без да се преизчисляват всички генерирани правила. Имплементацията се базира на програмните средства, предоставяни от ползваната ПС. Специфициран е проблемно ориентиран език със специфични оператори за сравнение и сравнителни изрази, алгебрични операции и функции. Предложеният подход е приложим в областта на анализите за съответствие с регулаторни правила при решаването на проблеми от други области.

Глава 4. Експериментални изследвания, резултати

4.1. Експериментална среда за генериране на информационни системи



Фигура 9. Структура на експерименталната среда за генериране на приложения

Създадена е експериментална среда - Risk Framework, реализираща описаните в глава 3 подходи. Средата се състои от C++ ядро, CLIPS интерпретатор и набор CLIPS програми, описващи класове от различни онтологии в областта на финансовите услуги. На фигура 9 е показана структурата на експерименталната среда. Ядрото предоставя възможност за работа като десктоп приложение и като Winsock сървърно приложение през интернет.

Посредством средата са генерирани редица информационни системи като например система за дефиниране на финансови портфейли и множество системи за анализ върху дефинираните портфейли.

Следващите точки на глава 4 описват имплементационните аспекти на средата за генериране на информационни системи, описана в глава 3. В точка 4.2 на дисертацията. е описано представяне на онтологичните класове в експерименталната среда. Точка 4.3 на дисертацията е посветена на реализацията на персистентния слой на експерименталната среда. Описани са технологичните детайли при организацията му, като се показва работоспособността на тази част от системата при използването й в многопотребителски режим с различни видове бази от данни. Точка 4.4 на дисертацията описва детайлите при имплементация на метода за управление на версии и актуализиране на записаните данни в базата на експерименталната среда. Показано е как в ядрото е закодирано тривиално преобразуване на записаните инстанции в базата и как чрез връзката им с CLIPS програмите могат да извършат допълнителни поправки при необходимост. Точка 4.5 на дисертацията показва реализиране на модел за представяне на проверка за съответствие в експерименталната среда. Описана е експериментална система за проверка за съответствие с регулаторни изисквания описани в съответни закони в Германия, Швейцария, Люксембург и Австрия. Точка 4.6 на дисертацията описва реализиране на система за анализ на портфейли чрез експерименталната среда.

4.7. Изводи

Описаната експериментална имплементация на средата за генериране на ИС показва, че разработването на информационни системи по предложеният начин става бързо, ефективно и лесно. Времето за разработване на самата среда не е по-дълго от разработката на една информационна система с конкретно предназначение. Използването на провереният и утвърден интерпретатор на CLIPS подсигурява минимални загуби на бързодействие заради интерпретацията на скриптовете, които се минимализират още повече чрез добавяне на потребителски функции. В процеса на разработка е създаден и визуален редактор на ресурси. В крайна сметка разработката на ИС се

свежда до добавяне на правилата за обработка на свойствата към класовете. Времето за разработка на цялостна система по този начин се намалява до няколко човекоседмици. Използваната структура на базата данни се оказва достатъчно ефективна за съхранението на данните, генерирани от информационните системи.

Основни изводи и предложения за практиката

Поставените цели на дисертационната работа са изпълнени. В резултат от работата е разработен метод за създаване на информационни системи, основани на онтологични класове. Създаден е програмен модел за имплементация на онтологични класове в средата за генериране на информационни системи. Създаден е подход за запис на инстанции на онтологични класове в релационна база от данни. Създаден е метод за проверка и актуализиране на записани инстанции на онтологични класове. Създаден е модел за представяне на проблемите за анализ на съответствие и е имплементиран изчислител, основан на средата за генериране на информационни системи. Създадена е програмна имплементация на средата, която е внедрена в български и чуждестранни фирми и банки.

Приноси

А. Научно-приложни приноси:

1. Създадени са метод и архитектура на среда за създаване и поддържане на информационни системи, основани на онтологични класове.
2. Към средата за генериране на информационни системи е създаден метод, синтаксис за представяне на ограничителни условия и имплементация на система за проверка съответствието на потребителските действия с тези условия.
3. Създаден е домейн-специфичен език (DSL) за бързо моделиране на онтологични класове и метод за свързването им в информационни системи, генерирани от средата.

Б. Приложни приноси:

4. Създаден е подход за съхранението в релационна база от данни на твърдителната част от онтологичните данни на средата за генериране на информационни системи.
5. Разработен е метод за поддръжка на различни версии на софтуера в средата за генериране на информационни системи чрез автоматично опресняване на

инстанции и автоматично адаптиране към структурни промени на онтологиите.

6. Чрез средата за генериране на информационни системи са реализирани информационни системи в областта на финансовите изчисления и анализи и проверки на ограничения и съответствия, които се основават на онтологични класове.

Публикации по дисертационния труд

1. Nikolov S., Antonov A., Nikolov V., Building financial evaluation chips in Java using FOTO-Objects, In Proc. 3rd Int. Conf. "Inf. in the scientific Knowledge", 2010, pp.108-120
2. Nikolov S., Antonov A., Framework for building ontology-based dynamic applications, In Proc. of CompSysTech '10, 2010, pp.83-88
3. Николов С., Антонов А., Implementation of dynamic application structure in a real time framework, 2010, Годишник на ТУ-Варна, том 2, стр.120-125
4. Nikolov S., CLIPS representation of ontology classes in an ontology-driven information system builder – part 1, In Proc. 2012 Int. Conf. of Russe Univ. and Union of the Scientists, 2012, Russe, pp.128-133
5. Nikolov S., CLIPS representation of ontology classes in an ontology-driven information system builder – part 2, In Proc. 2012 Int. Conference of Ruse University and Union of the Scientists, 2012, Ruse, pp.134-137
6. Nikolov S., Storing data of otology-based dynamic applications, In Proc. 6th Int. Conf. Computer Science'11, 2011, Ohrid, Macedonia, pp.134-139
7. Nikolov S., Construction of a portfolio analysis system using an ontology-based information system builder, Contemporary Methods and Technologies in Scientific Research, 2012, Varna, Bulgaria, pp.296-301
8. Trifonova S., Nikolov S., XML presentation of financial instruments, Int. Scientific Conf., 2010, Gabrovo, Bulgaria, pp. I-429 - I-434
9. Nikolov S., Ontology structure evolution in a framework for building ontology based information systems, In Proc. of CompSysTech '13, 2013, in press
10. Nikolov S, Nikolov V, Antonov A, A Constraint-based Approach for Analysing Financial Market Operations, In Proc. of CompSysTech '13, 2013, in press

Abstract
A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Sciences
PhD thesis: "Programming frameworks for generation of applications"
PhD Student: Samuil Vladimirov Nikolov

The dissertation analyses the problems of generating information systems using a framework. A specification of a framework that is based on model-driven development and uses ontologies as models is presented. The framework consists of:

- A core that generates dynamic user interface and manages requests from the production system interpreter;
- A production system that loads and executes programs, each one representing an ontology class.

The production system programs contain ontology property and relation definitions as fact templates, user interface definitions as facts and the business logic of the generated information system as rules.
The ontology instance data are stored as multi tier vertical table and updated via a two-phase mechanism consisting of trivial transformation and transformation through rules.
A model for user-extendable compliance check system that uses rules to limit the impact of input parameter changes is presented in the dissertation.
The main results of the dissertation work are as follows:

- A method for building information systems based on ontology classes;
- A program model for implementation of ontology classes;
- An approach for storing instances of ontology classes in a relational DB;
- A method for checking and updating of stored ontology class instances;
- A model for implementation of compliance check problems.

An implementation of the proposed framework is installed in bulgarian and foreign banks. It is used to generate information systems for:

- Calculation of operational risk using Basel II rules and their bulgarian equivalents;
- Pricing of financial instruments and portfolios;
- Asset and liability management;
- Regulatory compliance systems;
- Market risk and Solvency II requirements;
- Private and corporate rating systems.

Резюме
Диссертаций на тему “Програмные среды для генерации приложений”
Аспирант Самуил Владимиров Николов

Диссертация анализирует проблемы генерации информационных систем с использованием програмных сред. В диссертаций представлена спецификация структуры, основанной на разработке моделями и использующая онтологии в качестве моделей. Среда составлена от:

- Ядро которое генерирует динамический пользовательский интерфейс и управляет запросы от интерпретатор продукционной системы;
- Продукционная система которая загружает и выполняет программы, каждая из которых представляет клас онтологии.

Программы продукционной системе содержат свойства и отношения онтологий как шаблонные факты, дефиниция пользовательского интерфейса как факты и бизнес-логике информационной системы, которая генерируется, как правил.

Данные экземплярам онтологических класов хранятся в виде многослойной вертикальной таблице и обновляются через двухуровнего механизма, состоящего из тривиальной трансформации и трансформации с помощю правил.

В диссертации представлена модель расширяемой системы проверки соответствия, которая использует правила, чтобы ограничить влияние изменений входных параметров.

Основные результаты диссертационной работы заключаются в следующем:

- Метод построения информационных систем на основе классов онтологии;
- Модель программы для реализации классов онтологии;
- Подход для хранения экземпляров классов онтологии в реляционной БД;
- Метод проверки и обновление сохраненных экземпляров классов онтологии;
- Модель для реализации проверки соответствия.

Реализация предлагаемой структуры внедрена в болгарских и иностранных банк. Она используется для создания информационных систем для :

- Изчисление операционного риска с использованием правила Basel II и их болгарских эквивалентов;
- Изчисление цен финансовых инструментов и портфелей;
- Управление активам и пассивам;
- Системы соответствия регуляторным требованиям;
- Требования Solvency II и рыночный риск;
- Частные и корпоративные системы рейтинга.