SLOVENSKÁ TECHNICKÁ UNIVERZITA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-5384-50777

GENEROVANIE FORMULÁROV PRE WORKFLOW MANAŽMENT SÝSTÉMY
DIPLOMOVÁ PRÁCA

Študijný program:	Aplikovaná informatika
Študijný odbor:	9.2.9 Aplikovaná informatika
Miesto vypracovania:	Ústav informatiky a matematiky
Vedúci bakalárskej práce:	doc. RNDr. Gabriel Juhás, PhD.

Bratislava, 2012 **Bc. Richard Koháry**

Čestné prehlásenie:

Čestne prehlasujem, že som predloženú bakalársku prácu vypracoval samostatne pod vedením vedúceho práce doc. RNDr. Gabriela Juhása, PhD. za pomoci uvedenej literatúry.

Bc. Richard Koháry

.....

Abstrakt

Názov školy: Slovenská technická univerzita

Názov fakulty: Fakulta elektrotechniky a informatiky

Študijný program: Aplikovaná informatika

Meno autora: Bc. Richard Koháry

Názov diplomovej práce: Generovanie formulárov pre workflow manažment systémy

Vedúci bakalárskej práce: doc. RNDr. Gabriel Juhás, PhD.

Odovzdané: Máj 2012

Diplomová práca nadväzuje na bakalársku prácu *Tvorba dátového modelu pre formuláre*. V prvej časti sa táto práca zaoberá analýzou uvedenej problematiky diplomovej práce. V druhej časti je popísaná konkrétna implementácia aplikácie DataModelCreator 2 Web, pomocou ktorej si užívateľ môže vytvoriť svoju vlastnú web stránku pre každý dátový model . Aplikácia umožňuje užívateľovi tvorbu dátových modelov, prehľadnejšie a viac užívateľsky priateľskejšie ako v predchádzajúcej bakalárskej práci. Aplikácia je doplnená a vylepšená o množstvo funkcionalít, ktoré sa ukázali ako potrebné na základe predchádzajúcej práce. Tvorba dátových modelov je robená formou vytvárania web stránky. Aplikácia kladie dôraz na jednoduchosť a prehľadnosť používania. Jednoducho Drag & Drop si užívateľ zvolí požadovaný atribút, vloží si ho na stránku a aplikácia automaticky vytvára dátové modely s požadovanými atribútmi, ktoré budú vstupom do *workflow manažment systému* a zároveň prislúchajúcu web stránku.

Táto diplomová práca je modulom generickej aplikácie. Pomocou vytvorenej aplikácie si užívateľ vytvorí dátové modely, ktoré budú ďalej poskytnuté workflow engine- u Následne si užívateľ pomocou tejto aplikácie navrhne graficky spracovanú web

stránku spolu s formulárom pre dátový model. Cieľom nadväzujúcej diplomovej práce je zaistenie validity pre zadané údaje do formulára pomocou jazyka Javascript. Výstupom aplikácie DataModelCreator je dátový model exportovaný formátom .xml , web stránka vo formáte .html spolu s kaskádovými štýlmi (CSS). Výstupom nadväzujúcej práce na zaistenie validity formulára je súbor formátu .js. Tieto súbory tvoria vstup do Workflow manažment systému.

Kľúčové slová: dátový model, atribút, formulár, workflow manažment systém, Petriho sieť, rola, task, aplikácia, Java.

Abstract

University: SLOVAK UNIVERSITY OF TECHNOLOGY

Faculty: Faculty of Electrical Engineering and Information Technology

Department: Applied informatics

Author: Bc. Richard Koháry

Title: Generation of form for workflow management systems

Adviser: doc. RNDr. Gabriel Juhás, PhD.

Delivered: May 2012

The thesis builds on the bachelor thesis Creating a data model for forms. In the first part of this work deals with the analysis of that issue of the thesis. The second part describes the specific implementation of a DataModelCreator by which users can create their own web page for each data model separately. The application allows users to create data models, clearer and more user friendly than in the previous BSc work. The application is enhanced by amount of functionalities, which proved to be the basis of previous work. Creating data models is done by creating web pages. The application emphasizes simplicity and transparency of use. Just Drag&Drop the user selects the desired attribute, put it on the page and the application automatically creates the required data models with attributes that will be entering the workflow management system, while associated with the relevant website.

This work is part of a generic application. Application for creation of data models is proposed for workflow management system in which the user creates processes in Petri nets. Then the user using this application proposes graphically processed web page with the form of a data model. The follow-up of the thesis is to ensure validity of data entered into the form using Javascript. The output is a datamodel , DataModelCreator´s exported format. ".xml", web page format, HTML with cascading style sheets (CSS). The

output for a follow-up work to ensure the validity of the form is ".js" file format. These files are input to the workflow management system.

Key words: data model, attribute, form, workflow management system, Petri net, role, task, application, Java.

Zoznam skratiek

WFMS – Workflow management system

DBMS – Database management system

CSS - Cascading Style Sheets

XML - Extensible Markup Language

DMC – DataModel Creator

Zoznam obrázkov

Úvod

Formuláre v rôznych formách sa stali súčasťou každého databázového, informačného, alebo podnikového systému. Bez formulárov by ich chod a správa nebola možná. S vypĺňaním formulárov sa stretávame veľmi často. Či už na internete (informačné systémy, portály, mailové schránky, a podobne.), alebo v papierovej forme. Každé vypĺňanie formulára môžeme považovať za vykonanie procesu. Formuláre sú tvorené z formulárových atribútov - vstupov. Každý atribút ma vlastnú charakteristiku a názov. Napríklad uvedieme políčko „Meno“ s jednoduchým textovým vstupom. Pri prihlasovaní do portálu užívateľ musí vložiť svoje meno do tohto políčka. V tomto prípade systém očakáva reťazec znakov do formulárového vstupu. Po potvrdení formulára sa užívateľovi zobrazí vstupná stránka a systém sa posunul do ďalšieho stavu. Tieto stavy a procesy môžeme modelovať pomocou Petriho sietí.

Cieľom tejto diplomovej práce je rozšíriť existujúcu bakalársku prácu teda vytvoriť novú aplikáciu, ktorá umožní používateľovi vytvoriť web stránku, ktorá bude obsahovať formulár reprezentujúci dátový model. Táto web stránka bude formátovaná pomocou kaskádových štýlov (CSS). Aplikácia bude umožňovať jednoduchou drag&drop metódou pridávať atribúty(textfield, textarea, checkbox, combobox,....) na plochu. Užívateľ bude môcť definovať asociácie medzi prechodmi v Petriho sieťach namodelovanými v PNeditore a jednotlivými dátovými modelmi. Na základe tejto nadväznosti aplikácia automaticky vygeneruje web stránky s prislúchajúcimi CSS, ktorá bude vo workflow manažment systéme slúžiť ako užívateľské rozhranie pre prechod v vytvorenej logike v Petriho sietí.

Nadväzujúca diplomová práca bude zaisťovať validitu tohto formulára prostredníctvom regulárnych výrazov. Aplikácia bude komunikovať s aplikáciou na zistenie validity prostredníctvom rozhrania, ktoré mu umožní získať údaje o formulároch. Jej výstupom bude súbor formátu JavaScript.

Súbory vygenerovanej webovej stránky spolu s css súborom, xml dátovým modelom a súborom z nadväzujúcej diplomovej práce –JavaScript budú zbalené do

formátu ZIP. Tento zbalený súbor bude vstupom pre ďalšiu nadväzujúcu diplomovú prácu *Workflow manažment systém*. Ten bude slúžiť na spracovanie namodelovaných web stránok a ich následne prepojenie na servery prostredníctvom logiky nakonfigurovanej v Petriho sieťach pomocou PNeditoru.

Výsledkom bude komplexná generická aplikácia, v ktorej si užívateľ môže namodelovať prakticky ľubovoľnú web aplikáciu bez nutnosti programovania. Logika aplikácie bude definovaná Petriho sieťou, v ktorej prechody (Transitions) predstavujú business procesy, ktorých formulár , respektíve web stránku si môže užívateľ namodelovať prostredníctvom aplikácie, ktorá bola cieľom tejto diplomovej práce. Následne si užívateľ môže ošetriť vstupy formulára v nadväzujúcej aplikácií.

1 Analýza

1.1 Workflow manažment systém

Proces manažment môžeme vnímať ako aplikovanie cyklu riadenia organizačných procesov. *Gaitanides ET. Al* definuje proces manažment ako súbor plánovania organizovania a kontrolných aktivít pre cieľovo orientovaný manažment so zreteľom na faktor kvality, času , a zákazníckej spokojnosti. Hlavným cieľom proces manažmentu je dosiahnutie transparentnosti vzhľadom na štruktúru procesov. [2]

Becker a *Vossen* poukázali na tri aspekty vplývajúce na vývoj workflow manažment systému. Rozrastaním sa mailových systémov, workflow manažment systémy dokážu rýchlejšie komunikovať pozdĺž workflow modelu a spolupracovať s užívateľmi z rôznych kútov zeme. WFMS má určité podobnosti s aktívnymi databázovými systémami , ktoré dokážu monitorovať stavy systému a aktivít spúšťačov - *triggerov*. Z toho môžeme posúdiť, že WFMS súvisia s federatívnymi databázami a s konceptom rozšírených transakcií. Workflow môžeme vnímať ako dlho trvajúce transakcie a vyžaduje sofistikované odchytávanie chýb a záchranného (*recovery*) mechanizmu. Na druhej strane vývoj *document management techonology* môže byť nástrojom komerčného úspechu workflow technológie. [2]

1.1.1 Document management system

Mnoho úspešných workflow projektov je založených na kombinovaných workflow metódach. Dokumentovo orientovaného WFMS sú zamerané na **process object** (vo väčšine prípadov elektronický dokument). Z tohto pohľadu ma vplyv na paradigmy procesového modelovania obsiahnutého v systéme. Zatiaľ čo modely založené na aktívnom prístupe reprezentujú proces ako sekvenciu úloh, dokumentovo orientované aplikácie sú založene na základných modelovacích metódach, popisujúce procesy vo formulároch platné stavové zmeny biznis objektu.[2]

1.1.2 Databázové WFMS aplikácie

WFMS sú postavené na databázových technológiách, ktoré zabezpečujú ukladanie workflow –u a organizačného modelu. Je veľmi dôležité aby pri vytváraní WFMS databázy sa kládol dôraz na samotnú analýzu požiadaviek. Najdôležitejšie časti pri analýze *workflow database manažmente* sú:

- **Prechodový manažment**
- **Aktívne pravidlá**

Z pohľadu prechodového manažmentu môže byť workflow inštancia vnímaná ako dlho trvajúca transakcia s viacerými sub- transakciami. Workflow často krát spolupracuje so vonkajšími aplikáciami (na základe rôznych rozhraní). Udržuje svoje *business* dáta vo vnútri svojho kontextu. Vo väčšine prípadov však je potrebné aby k týmto dátam mali prístup aj ostatné aplikácie bez strát efektivity v celom informačnom systéme. Z perspektívy databáz môže byť workflow implementovaný na základe *triggerov* a aktívnych databázových pravidiel, ktoré monitorujú systém podmienky a získavajú údaje z *triggerov* na detekciu špecifických udalostí. Sekvencia databázových aktivít môže byť implementovaná ako súbor databázových akcií. Prvá aktivita môže byť spúšťaná externým *triggerom* a môže vytvoriť novú udalosť. Spúšťajú sa nové aktivity pozdĺž celého workflow modelu.[2]

1.1.3 Informačný systém pre biznis procesy

Organizácia práce , zvnútra , ako aj medzi spoločnosťami sa stáva stále viac a viac komplikovanejšou. Z tohto dôvodu sa čoraz častejšie zavádzajú do firiem informačné systémy, ktoré napomáhajú tieto biznis procesy riadiť a koordinovať. V tejto časti si najprv zatriedime informačné systémy a načrtneme vývoj informačných systémov z minulosti až po súčasnosť.

1.2 Kategorizácia informačných systémov

Informačné systémy môžeme kategorizovať z viacerých pohľadov. V nasledujúcom prehľade si spomenieme niektoré druhy informačných systémov

v závislosti od ich funkcionality. Prvý typ spomenutých systémov obsahuje len veľmi málo poznatkov o systémoch, používajú sa skôr pre ľudí čo pracujú vo firmách, zatiaľ čo v poslednej môžeme spravovať procesy bez akéhokoľvek ľudského zásahu.

1.2.1 Kancelársky informačný systém

Tento typ systému asistuje zodpovedným zamestnancom pri vykonávaní a manažovaní procesov so základnými informačnými procesmi , ako je napríklad písanie, kreslenie, počítanie, vypĺňanie a komunikácia. Tieto systémy zahŕňajú notesy, kresliace balíčky, tabuľky, jednoduché dáta systémy a elektronickú poštu. Tieto systémy samotné nezahŕňajú žiadne poznatky o procesoch. Systém môže obsahovať biznis informácie, ale v rámci systému s nimi nič nevykonáva.

1.2.2 Transakčný systém

Tieto systémy sa tiež nazývajú aj **registračné systémy.** Tieto systémy registrujú a komunikujú na základe príslušných aspektov zmien v životnom cykle procesu a zaznamenávajú tieto zmeny. Systém, čo sa špecializuje na komunikáciu medzi organizáciami sa nazýva tiež **inter- organizačný informačný systém**. Tieto systémy často používajú elektronické dáta (EDI – Electronic data interchange), používajúce štandardy pre výmenu dát, ako napríklad XML. Srdcom celého tohto systému vo všeobecnosti je *database management system* (DBMS), ale v súčasnosti sa už používa aj WFMS ako základný komponent.

1.2.3 Knowledge – management systém

Tento typ systému je charakteristický tým, že už čiastočne vie samostatne spracovávať informácie, vie samostatne interpretovať prichádzajúce transakcie a rozhodnúť, kde a akým spôsobom budú tieto dáta uchovávané. Systém zabezpečuje akvizíciu a distribúciu dát, ktoré budú použité pre zaškolených zamestnancov. Tieto údaje sú priamo poskytované v elektronickej podobe. Jeden z najjednoduchších takýchto systémov je vyhľadávací mechanizmus spojený s **dokumentovo manažovaným systémom**. Pracovník je schopný vyhľadávať časti textu, vytvoreným ním, alebo niekým iným jednoduchým vložením kľúčových slov. Pokročilejšie zariadenia sú **prípadové**

systémy, ktoré sú schopné prehľadávať databázy a nájsť prípady s najvyššou podobnosťou s aktuálnym prípadom. Riešenie prezentuje konkrétnym prípadom (case). Manažéri sa zaujímajú najviac o agregované dáta procesu prípadu, alebo o prípad samotný. Často používame dátové sklady (*data warehouses*) , ktoré sú napojené na nástroj pre štatistické vyhľadávanie. Tieto dátové sklady sú databázy , ktoré uchovávajú agregované dáta v tabuľkách.(Uchovávajú číslo zákazníka, typ produktu, názov, cena, geografickú polohu a podobne.).

1.2.4 Rozhodovacie informačné systémy (Decision – support system)

Tieto systémy vypočítavajú rozhodovanie na základe interakcií s ľuďmi. Sú dva typy rozhodovacích systémov. Prvý typ je založený na matematických modeloch. Napríklad rozpočtové, investičné systémy a produktovo – plánovacie systémy. Druhý typ je založený na logicko – zdôvodňovacích systémoch (**logical reasoning systems**). V niektorých literatúrach sa nazývajú aj expertné systémy (*expert systems*). Jedným z príkladov je systém na stanovenie príčiny zlyhania stroja. Tieto systémy sa používajú na všetkých úrovniach riadenia (prevádzkových, taktických aj strategických).

1.2.5 Riadiace systémy (Control system)

Známe aj ako programované rozhodovacie systémy. Tieto systémy prepočítavajú a implementujú rozhodovanie úplne automaticky. Sú založené na zaznamenávaní stavov procesov. Napríklad automatické objednávky, kontrola klímy a podobne.

Informačné systémy sa často kombinujú z týchto štyroch typov, ktoré sme si popísali. Z hľadiska efektivity je ideálny riadiaci systém (**Control system**), pretože nevyžaduje žiadneho zamestnanca. V praxi však býva počet aplikácií systémoch obmedzený a nie každý problém je jednoznačne rozhodnuteľný. Rozhodovacie systémy (Decision – support) majú najväčší potenciál pretože spája ľudskú predstavu s výpočtovou silou. V praxi sa najčastejšie používajú Kancelárske informačné systémy a prechodové informačné systémy.

1.3 Koncept workflow aplikácií

V tejto časti si zavedieme základnú terminológiu pre workflow aplikácie. Po prehľade terminológie a definícií sa budeme venovať analýze workflow aplikácií.[2]

1.3.1 Workflow manažment systému:

„WFMS vytvára , riadi spúšťanie workflow softvéru, beží na jednom, alebo viac workflow engine , ktorý je schopný interpretovať definície procesov, komunikovať s ostatnými časťami workflow a prípade potreby využiť dostupné IT nástroje a aplikácie „ [3]

Z definícií vyplýva, že workflow systém pozostáva z komponentu na vytváranie workflow modelov, funkcionalít na tvorbu workflow inštancií z workflow modelov a funkcionality na spúšťanie týchto workflow inštancií, ktorý sa nazýva ***workflow engine***. [2]

1.3.2 Workflow model:

Definícia procesu pozostáva zo siete aktivít a ich vzťahov, kriterí začiatku a ukončenia procesu a informácií o individuálnych aktivitách , ako čiastočné asociované IT aplikácie a dáta.

Definícia je limitovaná na aktivity založené na modelovaní procesov. Z funkčného hľadiska úlohu WFMS môžeme združiť do“ plánovania, implementácie, stanovenie a hodnotenie workflowu. Fáza plánovania a implementácie sa často krát nazýva aj časová fáza. Fázy stanovenie a hodnotenie workflowu sú považované za fázu *runtime* [2]

1.3.3 Fáza plánovania

Počas tejto fázy je vytvorený koncepčný model procesu a tiež prepojenie (linky), ktoré sú uplatnené počas spúšťania aktivít externých aplikácií. V tejto fáze je zvažovaný aj dátový model. [2]

1.3.4 Fáza implementácie

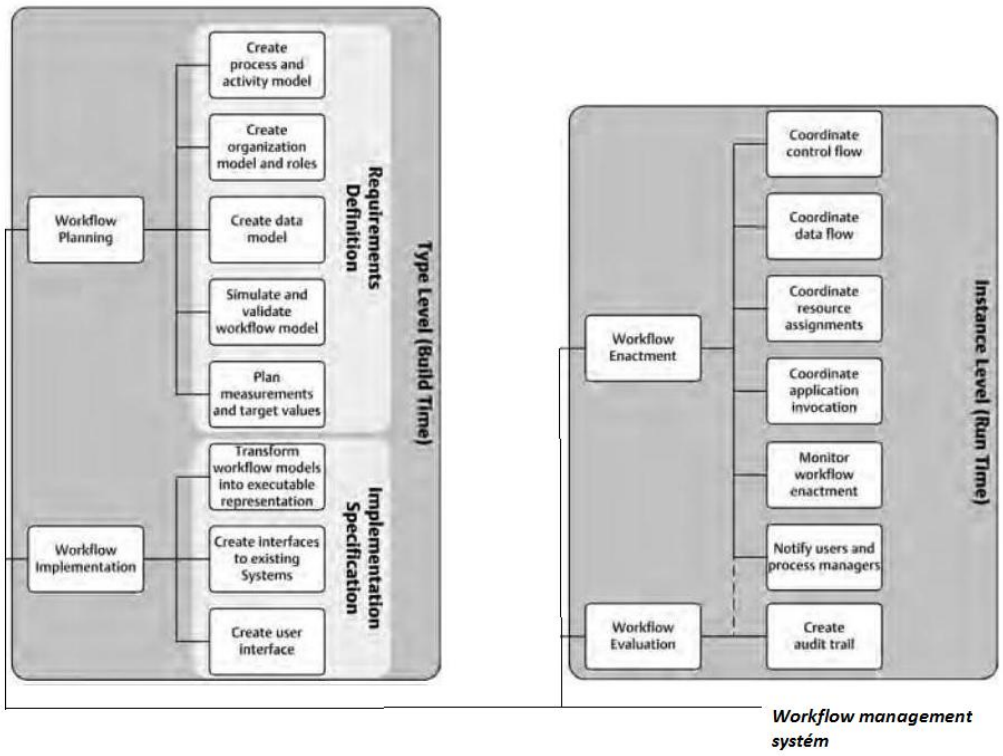
Koncepčný workflow model je transformovaný do spustiteľnej reprezentácie. Niektoré systémy požadujú preklad z ich vnútornej reprezentácie workflow modelu. Ostatné systémy sa spoliehajú na zdieľanú databázu . Modelovanie a spúšťanie fáz a ich spustiteľný workflow model sú identifikované atribútmi. Z technického hľadiska rozhrania do externých systémov musia byť implementované na tejto úrovni za účelom realizovania komunikácie medzi workflow manažment systémom a volanými aplikáciami podobne ako front- endové systémy komunikujú s užívateľom. [2]

1.3.5 Koordinačna a monitorovacia fáza

Táto fáza odkazuje na určité formy a prevedenie jednotlivých workflow inštancí z predchádzajúceho nadefinovaného modelu. (inštancia sa niekedy nazýva aj **Workflow case** , budeme sa jej venovať nižšie). Workflow engine koordinuje kontrolu toku (*flowu*) regulovaním aktivovaním a spúšťaním aktivít zavíslých na celkovom stave workflowu. Koordinujú dáta posielané medzi prechodmy – jednotlivými aktivitami a medzi workflow systému a uplatnenými aplikáciami. Celková integrita workflow inštancie je monitorovaná WFMS. Napríklad procedúry sú spúšťané , prave vtedy keď je dosiahnutý určitý zadefinovaný deadline a informácia o súčasnom stave z workflow inštancie je poskytnutá zájmovým častiam. (napríklad zákazník je informovaný o stave objednávky).[2]

1.3.6 Fáza hodnotenia

V tejto časti sa kontroluje, správanie systému a spúšťanie workflow inštancí. Táto funkcionalita môže byť tiež časťou koordinačnej a monitorovacej fázy.



Obrázok 1 Bloková schéma workflow manažment systému

1.3.7 Aplikačný systém workflow

Je aplikácia , ktorá používa workflow engine na koordináciu komponentov aplikácie, užívateľov, dát. Aplikácia pozostáva z jedného , alebo viacerých workflow engine- ov, aplikácií, administračných komponentov, užívateľských rozhraní a dát. [2]

1.3.8 Case (Workflow inštancia)

Case je individuálna reprezentácia procesu založeného na workflow modely, ktorý je spracovávaný workflow engine-om. Aktivita modelov z workflow modelu sú reprezentované inštanciami aktivít. Každá časť workflow *(workflow item*) je asociovaná s jednou, alebo viacerými úlohami.- work list, ktorý slúži na uchovanie všetkých *work itemov* pripojeným k jednotlivým častiam workflowu. [1]

Case sa vždy nachádza v istom stave, ktorý je zložený z týchto častí [3]:

1. *Case attributes* – dáta v určitom stave (vyplnené meno pacienta, diagnóza , a podobne..)
2. Podmienka splnenia – napríklad vo workflow manažment systéme vyplnenie formulára a jeho potvrdenie.
3. Obsah úlohy

Case-u má pridelený rozsah premenných (*variables*). Na príkad *case atribút* v našom príklade s pacientom je *„diagnóza všeobecného lekára“.* Na základe

tejto premennej , môže workflow systém rozhodnúť na ktoré oddelenie pacienta pošle. Hodnota *case* atribútu sa mení postupne podľa toho, ako *case* postupuje.[3]

1.3.9 Case condition

Tieto podmienky nám pomáhajú určiť, ktorý *task* sa vykonal a ktorý sa ešte stále vykonáva. Podmienky môžu nadobúdať hodnoty ako *napríklad „prijať pacienta, neprijať pacienta“*. [1][3]

„*Workflow systém* neobsahuje detaily ohľadom obsahu úloh, ale len atribúty a podmienky. Obsah je zahrnutý v dokumentoch, súboroch, archívoch, databázach, ktoré nie sú riadené *workflow manažment systémom*.“ [3]

1.3.10 Task

Ďalšia dôležitá jednotka workflow manažment systému je ***Task***. *Task* je vratná logická jednotka, je nedeliteľná a musí byť vždy vykonateľná. Pokiaľ niečo počas vykonávania *tasku* zlyhá úloha sa jednoducho vráti späť na začiatok. Táto operácia sa nazýva podobne ako v databázových systémoch *ROLLBACK* . *Tasky* môžu mať rozličný charakter. Za *task* môžeme považovať vyplnenie vstupného formulára do IS, alebo vyplnenie žiadanky o vyšetrenie, spustenie liečenia vírusov antivírusom v počítači, alebo ručne dať skenovať disk. Môžeme na spomínaných úlohách sledovať ich súvislosť s mierou zásahu človekom. Na základe ich môžeme deliť na [1][3]

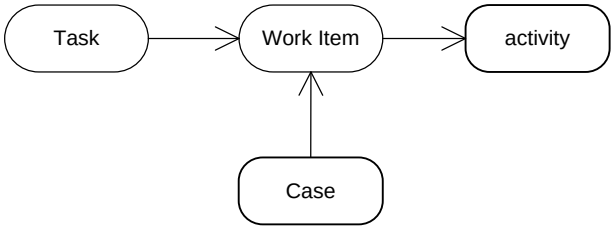
1. Manuálne
2. Automatické
3. Polo-automatické

„**Manuálny *task*** sa vykonáva výlučne na základe ľudského zásahu a ľudskej činnosti. Za manuálny task môžeme považovať napríklad. vykonávanie fyzickej kontroly“ [3]

„**Automatické *tasky*** sa realizujú bez ľudského zákroku. To znamená, že program analyzuje dáta na základe údajov bez zásahu človekom“ [3]

Špeciálny prípadom sú **poloautomatické tasky**, V tomto prípade na task má vplyv ako aj aplikácia , tak aj osoba. Napríklad: *„Kontrola zadaných údajov a ich validácia do počítača“*. [3]

Ďalšie dôležité termíny vo workflow systéme sú *„pracovná jednotka“* ***(work item)*** a *„aktivita“****(activity).*** *Work item*, vzniká kombináciou casu a tasku. Táto jednotka vzniká hneď v okamžiku keď to umožní case. *Work item* je tak určitá časť práce , ktorá sa bude vykonávať. Aktivita je konkrétny výkon pracovnej jednotky - *work item“*. V okamihu keď sa začína práca *work itemu*, začína sa aj aktivita [1][3]



Obrázok 2 Vzťah medzi *taskom, case*-om a *aktivitou*

1.3.11 Proces

„***Proces*** je metóda, ktorá popisuje kde môžu byť jednotlivé kategórie *case*-ov uskutočňované. *Proces* indikuje, ktorý task potrebuje byť spustený a taktiež poradie, v ktorom budú *tasky* uskutočnené.„ [3, strana 14]

„*Procesy* môžeme tak isto chápať aj ako procedúry pre jednotlivé typy *case*-ov. Vo všeobecnosti množstvo odlišných *case*-ov je ovládané používaním jedného *procesu* .

Poradie, v ktorom sú *tasky* vykonávané môže tiež meniť závislosť na vlastnostiach *case*-u. O poradí, v ktorom nasledujú, obvykle rozhodujú podmienky. Podstata procesu je z tohto dôvodu založená na *taskoch* a podmienkach.“ [3, strana 14]

„Taktiež je možné používať predchádzajúci definovaný proces ako časť ostatných *procesov*, takže okrem *taskov* a podmienok, môžu *procesy* pozostávať z viacerých podprocesov, alebo nemusia pozostávať zo žiadneho (*subprocess*). Každý z podprocesov znovu obsahuje *tasky*, podmienky a môže obsahovať ďalšie podprocesy. Explicitne identifikované a oddelene popisované podprocesy môžu byť znovu použité. To znamená, že komplex *procesov* môže byť *hierarchicky štruktúrovaný*.“ [3]

„Na najvyššom stupni popisovaných procesov vidíme konečný počet podprocesov. Môžeme tieto podprocesy otvoriť a vidieť pripadný podproces podprocesu. Teda podprocesy sú *„zoomovateľné“* (*zoom in,zoom out*)*.“* [3]

„Životný cyklus *case*-u je definovaný procesom, pretože každý *case* má konečnú dobu expirácie (*lifetime*) so začiatkom a koncom. Z toho vyplýva, že každý proces má tiež svoj začiatok a koniec.“ [3]

1.3.12 Klasifikácia organizačných procesov

Procesy v organizáciách sú združované do niekoľkých kategórií v závislosti od pôvodu predmetu procesu. Zatiaľ čo „*logistic process*“ sú vykonávané s cieľom manipulácie s fyzickým objektom alebo vykonávanie služieb. (napríklad sťahovanie určitých produktov z miesta na iné miesto a podobne.). Napríklad Finančný proces sa vykonáva v okamžiku , keď dve strany si vymieňajú peňažné hodnoty. Každý tento proces je sprevádzaný *informačným procesom*, ktorý reprezentuje tok dát v informačných

systémoch spoločností, ktorý je spôsobený dôležitými logistickými, alebo finančnými procesmi. [2]

Účastníci procesov

Účastníci procesov môžu byť buď ľudia, alebo stroje, alebo kombinácia obidvoch. Teda podobne ako *tasky* (úlohy), môžu byť aj procesy manuálne, automatické a poloautomatické. V automatických procesoch môžeme rozlišovať medzi hardvérovými a softvérovými procesmi. V závislosti od typu a od priradenia aktivít môže byť proces vykonávaný automaticky (*push*) a manuálne (*pull*). Od kapacity a technickej dostupnosti technického zásahu môžeme určiť automatickosť procesu vo viacerých uhloch. Automaticky priradený algoritmus môže zamestnať hardvér, zatiaľ čo pull- stratégia je bežnejšia pre ľudskú činnosť. [2]

Štruktúra

Ak je štruktúra procesov a aktivít známa dopredu , potom je možné ich definovať prostredníctvom formálnych metód. Táto špecifikácia môže byť interpretovaná workflow manažment systémom. [2]

Klasifikácia atribútov procesov

Stupeň koordinácie kontroly wokrflow manažment systému je priamo úmerný stupňu jeho konkrétnosti a presnosti namodelovanej štruktúry. [2]

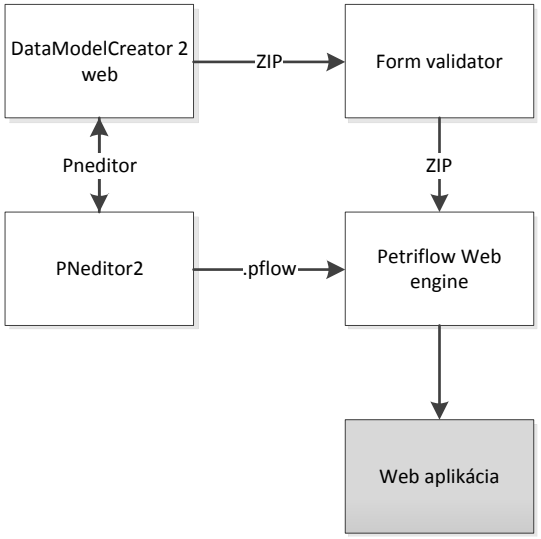
Ad – hoc procesy

Ad hoc procesy sú tie procesy, ktorých štruktúra je neznáma, alebo počas výkonu nie je jasná. Táto situácia môže vzniknúť v procese, ktorý sa z rôznych vonkajších príčin mení, alebo analýza a dokumentácia štruktúry procesov je považovaný za drahý. Aktivity , s ad hoc štruktúrou, alebo je príliš nákladná na zistenie, ktorý zdroj (ľudský, technický, finančný, alebo informačný) je požadovaný na spustenie kompletnej aktivity. [2]

Rozsah procesu

Rozsah jednotlivých procesov ohraničuje procesy. Procesy medzi organizáciami sa nazývajú medzi – organizačné procesy. Procesy medzi organizáciami sú buď prichádzajúce – *inbound processes*, alebo odchádzajúce – *outbound processes*, ktoré sú spúšťané vonkajšími faktormi spoločnosti. Typický príkladom takéhoto procesu sú dodávkové procesy na zákazníckej strane a dokončenie procesu na dodávanej strane. [2]

1.4 Náhľad, vstupy a výstupy modulov generickej aplikácie



Obrázok 3 Architektúra

1.4.1 Pneditor2

Pneditor 2 poskytuje rozhranie PNeditor, pomocou ktorého môžeme získavať inštancie potrebných tried tj. (getTransitions, getPlaces, getRoles, getArcs,..). Tento výstup Pneditora je zároveň vstupom do aplikácie , ktorá je cieľom tejto diplomovej práce. Na základe informácií získaných o namodelovanej bussiness logike v Pneditore (Petriho sietí) sme schopný asociovať tieto PN údaje s funkcionalitami DataModel Creatora (ďalej len DMC).

Vstup: užívateľom namodelovaná busines logika petriflowu

Výstup: .pflow, PNeditor interface

1.4.2 DataModelcreator 2WEB

Modul DMC tieto údaje získava v reálnom čase. Teda aká koľvek zmena na PN sieti je detekovaná automaticky v moduly DMC. Rozhranie PNeditor je zároveň aj jediným externým vstupom do DMC modulu. Po spracovaní údajov DMC modulom a vytvorením

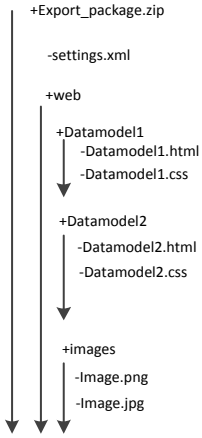
dátových modelov a namodelovanými web stránkami užívateľom DMC modul poskytuje 2 typy súborových **výstupov**:

1. Súbor ZIP (DMC export)
2. Súbor typu .dmc

Súbor typu ZIP má špecifickú a dopredu dohodnutú štruktúru s ostatným členmi tímu (Tomáš Zúber – Petriflow web engine, Peter Baranec – Zabezpečenie validáciu namodelovaných formulárov). Tento súbor po rozbalení bude obsahovať konfiguračný súbor settings.xml, ktorý špecifikuje premenné Petriflow. Premenné Petriflow sa asociujú s formulárovými vstupmi, vďaka čomu je web engine schopný spracovávať vstupy koncových užívateľov (Osoby pracujúce s výsledným namodelovaným produktom).

Ďalej bude obsahovať zoznam názvov dátových modelov (Názov konkrétnej web stránky obsahujúcej formulár) pre potreby web engine –u. Na základe zoznamu je web engine schopný lokalizovať v súborovej štruktúre konkrétnu zložku s cieľovou web stránkou.

V adresárovej štruktúre sa bude ďalej nachádzať zložka „web“. V tejto zložke budú všetky dátové modely vo vlastných zložkách, ktoré obsahujú html stránku formulára dátového modelu a súbor „.style.css“ pre každú stránku. Štýly budú generované DMC . a zložka „images“, ktorá bude obsahovať obrázky na zobrazované stránkach. Táto zložka bude v adresárovej štruktúre rovnocenná so zložkami dátových modelov, kvôli odstráneniu duplicity obrázkov.



Obrázok 4 Adresárová štruktúra výstupného súboru

Pre úplnosť si uvedieme príklad. Užívateľ si na dvoch stránkach dátový modelov pridá dva rovnaké obrázky. Keby každý obrázok bol priradený ku každému dátovému modelu zvlášť vznikla by duplicita. Na servery by museli byť uložené dva rovnaké obrázky , čo by pri hromadnom používaní systému mohlo spôsobiť nemalé kapacitné ťažkosti.

Súbor typu .dmc je súbor do ktorého budú vkladané *serializované* objekty z projektu. Tento súbor sa generuje pri ukladaní a obsahuje všetky potrebné údaje na znovu obnovenie rozpracovaného projektu. Užívateľ má tak možnosť kedykoľvek svoju prácu uložiť a vrátiť sa k svojej práci neskôr, prípadne svoju prácu upraviť podľa potrieb.

Vstup: PNeditor interface

Výstup: export.zip, project.dmc

1.4.3 Form validátor (Bc. Peter Baranec)

Úlohou tohto modulu je validácia správnosti zadaných údajov do formulára dátového modelu. Modul dostane na vstup súbor „.zip“, ktorý si rozbalí spracuje „.html“ súbory. Z rozparovaných súborov následne aplikácia získa *id-ečka DIV-ov.* Na základe id-čiek si validátor nadefinuje o aký typ validácie pre tom- ktorom poli pôjde. Aplikácia vygeneruje súbor „.js“ a pripojí ho do „.zip“ súboru. Výsledný „.zip“ súbor je zároveň výstup aplikácie.

Vstup: export.zip

Výstup: export.zip (doplnené o validations.js)

1.4.4 Petriflow Web engine (Bc. Tomáš Zúber)

Webový engine zabezpečuje spracovanie všetkých workflow procesov generickej aplikácie, spracovanie údajov a vytvorenie webovej aplikácie ako finálny produkt.

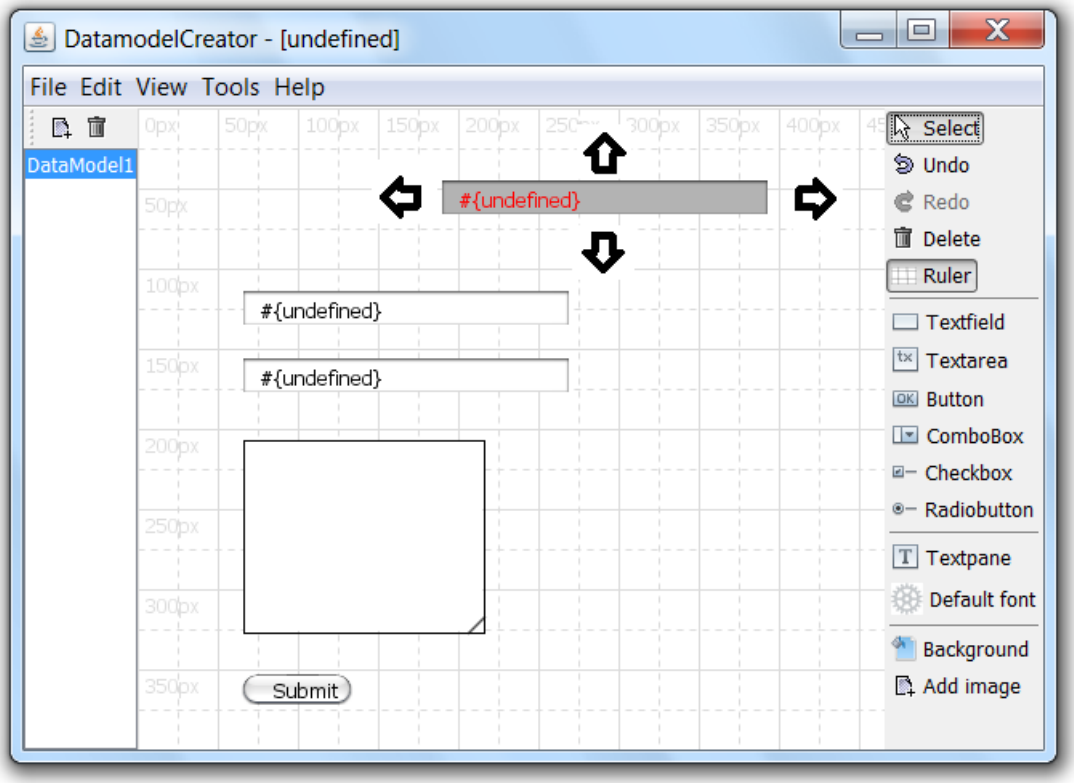
Vstup: export.zip, pnet.pflow

Výstup: web aplikácia

2 Opis riešenia

Aplikácia DataModel Creator 2WEB je riešená ako vnorená aplikácia do PNeditora. Toto riešenie vychádza z požiadaviek kladených na aplikáciu. Tieto aplikácie budú navzájom komunikovať prostredníctvom rozhrania PNeditor implementovaného v PNeditore. Toto rozhranie poskytne aplikácií potrebné dáta na prácu pri vytváraní dátových modelov a prislúchajúcich web stránok.

Táto práca nadväzuje na bakalársku prácu [3] , v ktorej som sa venoval problému vytváraniu dátových modelov, tj. štruktúry formulárov (dátový interpret-textarea, textfield, ..., dátový typ- string, integer, boolean,..., prístupové práva, asociácie s PN prechodmy). Cieľom tejto diplomovej práce je nadviazať na bakalársku prácu. Umožniť užívateľovi nie len vytváranie dátových modelov, ale hotových formulárov formou WEB stránky. Tieto stránky si užívateľ môže štýlovať podľa seba. Jednotlivé atribúty formulárov (textfield, textarea,...) môže ľubovoľne umiestňovať na ploche (Canvas),



Obrázok 5 Demonštrácia lokalizácie atribútu na stránke

Užívateľ môže nastavovať potrebné atribúty jednotlivých vstupových interpretov, rozlíšenie stránky farbu pozadia, pridávanie obrázkov, ich presúvanie, mazanie, asociácie stránok s PN prechodmi formou užívateľsky prehľadného a jednoduchého rozhrania, exportovanie ZIP balíčka s dopredu dohodnutými formátmi súborov.

2.1 Špecifikácia požiadaviek

Riešenie problému spočíva v niekoľkých fázach:

1. Analýza problému a návrhy riešenia
2. Výber najvhodnejšieho programovacieho jazyka a výber vhodných a moderných technológií (v čase návrhu)
3. Navrhnúť komunikačné rozhrania, abstraktné triedy a základný singleton aplikácie
4. Navrhnúť štruktúru triedy dátového modelu, predchádzajúca špecifikácia nie je postačujúca vzhľadom na požiadavku generovanie web stránky
5. Vytvoriť GUI, v ktorom užívateľ si vytvorí dátový model priamo formou vytvorenia formuláru metódou drag&drop a na základe neho aplikácia vygeneruje príslušnú web stránku.
6. Vytvoriť užívateľské rozhranie pre tvorbu atribútov jednotlivých dátových modelov (formulárov),
7. Navrhnúť riešenie a vytvoriť GUI asociáciu dátového modelu s prechodmi PN.
8. Export projektu, generovanie xhtml + css , export dátového modelu v XML pre jadro workflow manažment systému

2.2 Návrh

Samotný návrh aplikácie vychádza z analýz na požiadavky aplikácie. Úlohou bolo vytvoriť aplikáciu, pomocou ktorej užívateľ na základe definovaných procesov (taskov) si môže vytvárať dátové modeli pre procesy definované v PNeditore. Aplikácia užívateľovi umožní rýchlo a jednoducho si namodelovať dátové modely formou vytvárania formulárov na ploche (canvas). Jednotlivé atribúty sa dajú presúvať, mazať, kopírovať. Pre konkrétne atribúty užívateľ môže definovať jeho špecifické vlastnosti. Pre *CheckBoxi* môže určiť defaultnú booleovskú hodnotu, pre *TextField* si môže vybrať prislúchajúci dátový typ v atribúte formulára (integer, string, boolean, double, date), *TextArea* (textové pole) je riešená tak, že užívateľ si môže meniť jej veľkosť jednoduchým ťahaním za jej koniec.

Architektúra aplikácie by mala byť rozdelená do viacerých logických blokov, nielen kôli prehľadnosti, ale aj pre lepšiu prístupnosť k vytvoreným inštanciám v jednotlivých triedach.

- Blok „Document“ –obsahuje všetky dáta o vytvorených dátových modeloch, ich stránkach, štýloch
- Blok „DefaultSettingManager“ – Obsahuje inicializačné údaje , ako nastavenie štandardného fontu, šablóny, rozlíšenia pre optimalizáciu stránky a podobne.
- Blok „SelectionManager“ – slúži na určenie ktorý nástroj sa práve nad plochou používa (výber, pridávanie atributov, ...)
- Blok „UndoManager“ – Obsluhuje funkcionalitu Undo-Redo.
- Blok „MainFrame“ – Základný rámec aplikácie.

Tieto bloky predstavujú triedy, ktorých inštancie sú vytvorené v *singletóne DataModelCreator* , pomocou ktorého môžeme pristupovať k dátam oveľa efektívnejšie ako pri predávaní referencií z triedy na triedu.

2.2.1 Blok Document

```
public class Document implements Serializable {

        private Datamodels dataModels = new Datamodels();

        private String projectName;

        private List<Dvariable> variables = new LinkedList();

        +getters and setters
```

Ukážka triedy Document

Pomocou triedy *Document* vieme pristupovať k jednotlivým častiam aplikácie okamžite bez predávanie inštancií. Trieda je serializovateľná –trieda implementuje rozhranie *Serializable* kvôli ukladaniu dátových štruktúr do bytovej mapy, pomocou ktorej sme schopný zapisovať objekty do súborov. Ukladaniu sa budeme venovať neskôr.

Ako vidíme z ukážky, trieda zahŕňa 2 dôležite inštancie – Datamodels a List<Dvariable> .

Datamodels

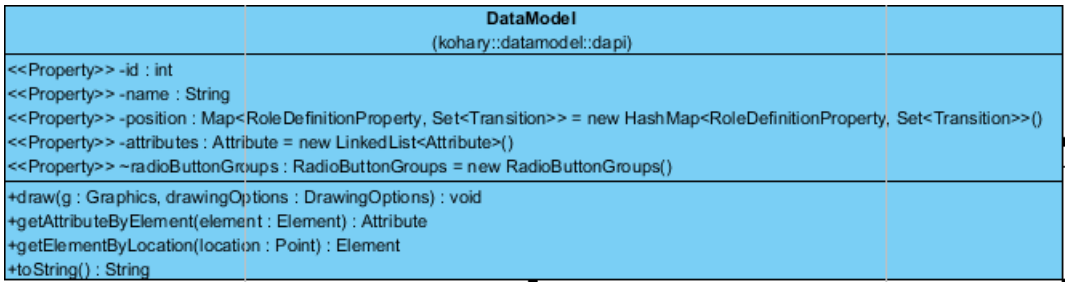
Je potomkom triedy ListModel<Datamodel>. V triede sú implementované metódy na pridávanie nových dátových modelov , respektíve ich mazanie. Trieda tiež obsahuje viacero interných (private) metód na sprehľadnenie kódu a výpočtov jej výstupov. Ostatné metódy trieda dedí po svojom predkovi.

V predkovi je definovaný aj samotný list, ktorý je definovaný ako generický list elementov. Ďalej sú tu implementované základné metódy nad dátovým typom list. (getSize(), getElementAt(int i),add(E element),..)

Ako už vyplýva z predchádzajúcich vysvetlení trieda Datamodels predstavuje list Dátových modelov, ktorých štruktúru si vysvetlíme nižšie.

Datamodel

Trieda *DataModel* je jednou z tried , ktoré tvoria kostru celej aplikácie. Jej atribútmi sú:

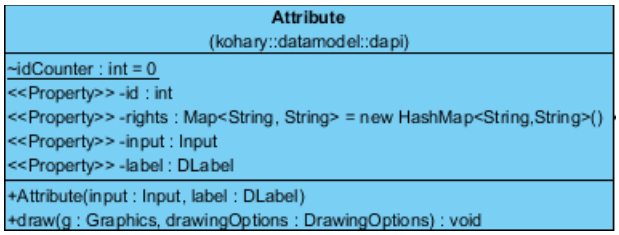


Obrázok 6 Class diagram Datamodel

- **Id**: identifikátor dátového modelu
- **Name**: názov dátového modelu. Užívateľ si pri vytváraní zadá meno sám, alebo na základe identifikátora mu názov bude pridelený DatamodelXX, kde XX je jeho id
- **Position**: Pozícia dátového modelu. Určuje pre ktorý prechod (proces), namodelovaný v PNeditore bude dátový model aktívny.
- **Attributes**: sú atribúty dátového modelu. Atribúty predstavujú riadok formulára a reprezentuje ho dvojica Input-Label
- **RadioButtonGroups**: je množina všetkých skupín *radio* - tlačitok, ktoré sú evidované ako atribúty

Attribute

Je trieda reprezentujúca atribút dátového modelu. Inak povedané reprezentuje riadok formulára namodelovaného užívateľom aplikácií.

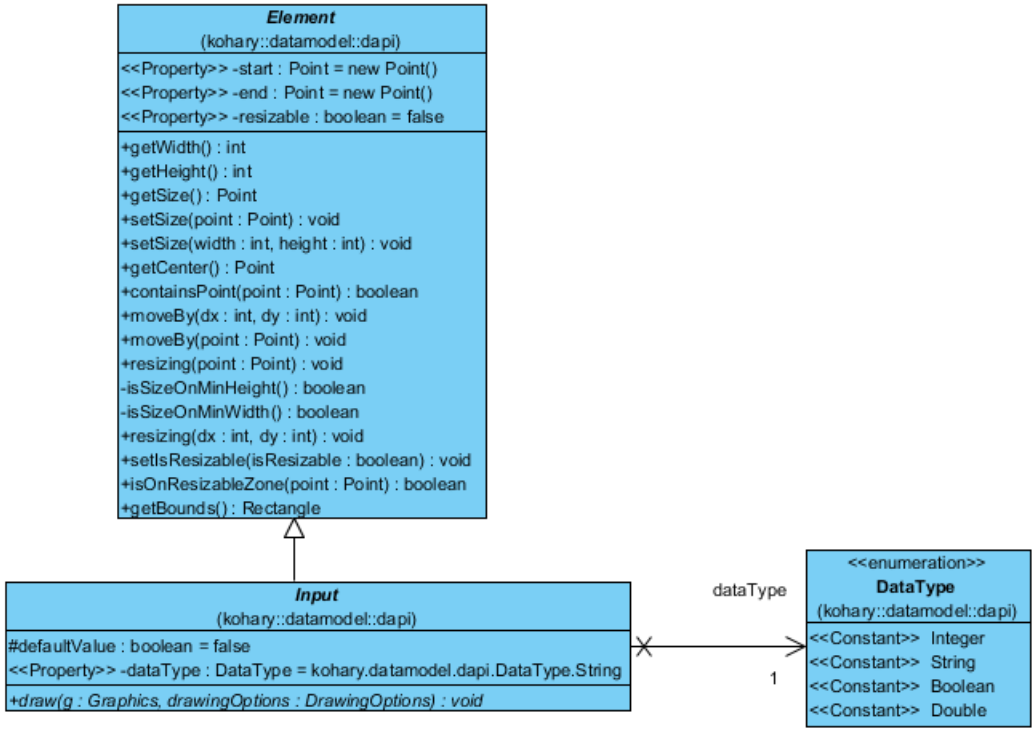


Obrázok 7 Class Diagram Attribute

- Statický atribút idCounter slúži na prideľovanie *id* jednotlivým atribútom.
- **Rights**: sú prístupové práva (Read-Only, Read-Write, None) pre jednotlivé role v procesoch
- **Inputs**: reprezentujú vstupy do formulára
- **Dlabel**: je názov (Label) atribútu

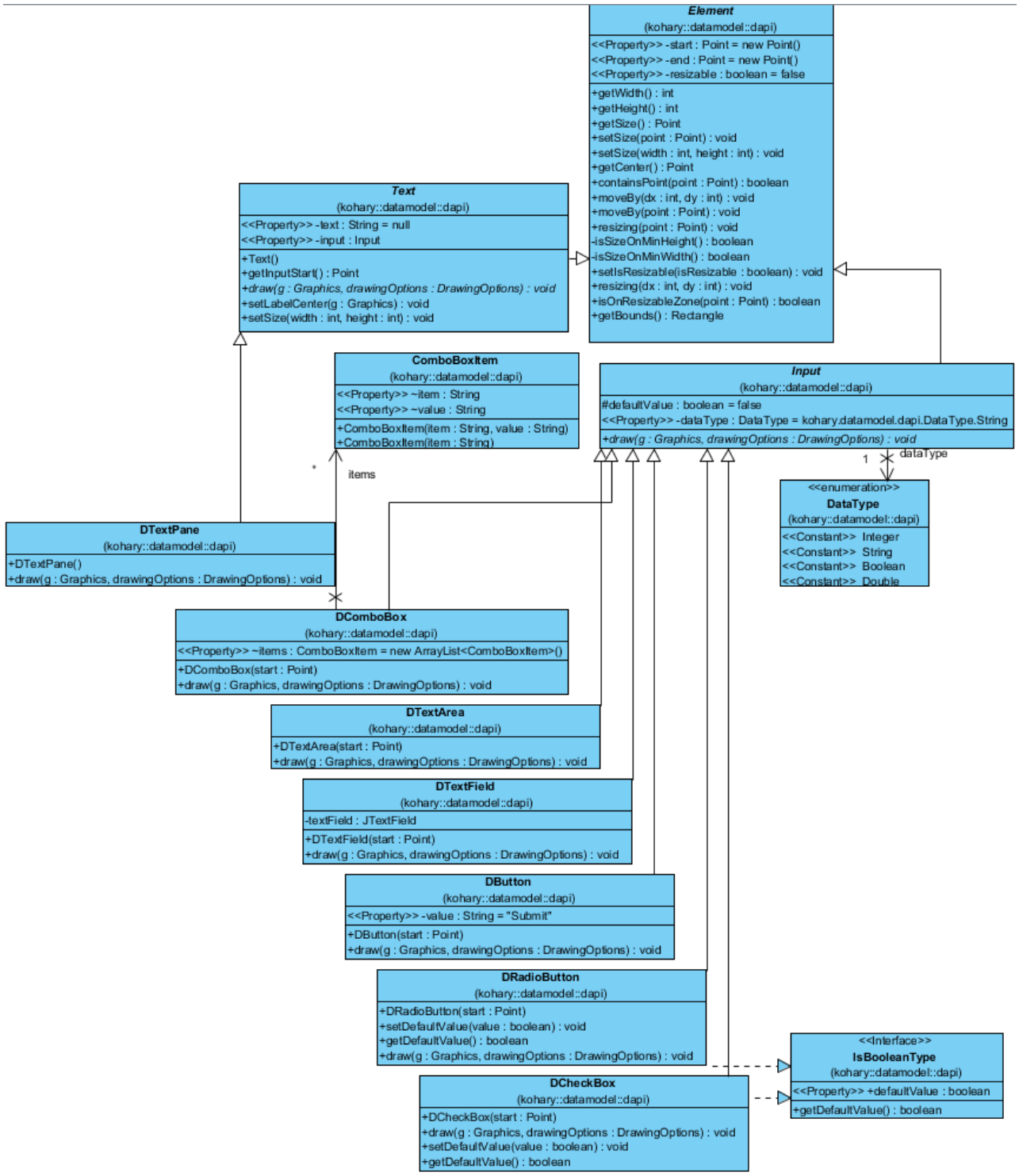
Input

Táto trieda reprezentuje vstup do formulára dátového modelu.



Obrázok 8 Class Diagram DInput

Ako môžeme vidieť na obrázku 5 (*Class Diagram Input*) , trieda rozširuje abstraktnú triedu Element, ktorá obsahuje atribúty a metódy spoločné pre všetky prvky, ktoré užívateľ pridá na plochu (*Canvas*). Napríklad užívateľ chce vytvoriť formulár s atribútmi : textArea, textField, checkbox, ... viď obrázok 2. Užívateľ si vyberie v pravom pracovnom nástroji prislúchajúci výber a následne vyberie miesto na ploche kde tento element chce umiestniť. Aplikácia po tomto úkone vytvorí novú inštanciu triedy, podľa vybraného elementu, následne ho uloží medzi atribúty dátového modelu. Enumeračná trieda DataType slúži výber dátového typu, ktorý jednotlivej podtriede Input-u prislúcha. Na obrázku 6 môžeme vidieť diagramy podtried triedy Input, ich atribúty a metódy.



Obrázok 9 Class Diagram triedy Input

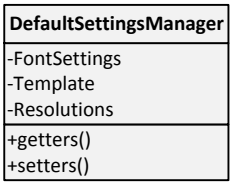
Na predchádzajúcom obrázku sa nachádzajú diagramy tried Element a Input a jeho priamych podtried. Jednotlivé podtriedy Inputu reprezentujú vstupy formulára s ich špecifickými vlastnosťami vzhľadom na ich požiadavky.

- **DTextPane** - voľný text na stránke. Môže sa jednať o nadpisy, poznámky, legendy,
- **DComboBox** – predstavuje roletové menu podobne ako html tág <select>. Tento element sa používa v prípade, ak dopredu poznáme hodnoty a len tie hodnoty, ktoré môže element nadobúdať.
- **DRadioButton** – konkrétna osoba s prístupovými právami pre prislúchajúcu rolu v procese určí jednu voľbu z viacerých. Každý DRadioButton musí patriť do konkrétnej skupiny DRadiobuttonGroup, ktorá sa automaticky vytvorí pri pridaní prvého elementu DRadioButton. Následne je užívateľovi poskytnutá voľba, či chce nový DRadioButton vložiť do už existujúcej skupiny, alebo chce vytvoriť novú.
- **DCheckBox** – prislúchajúci tág je <input type=”checkbox”/> . Tento vstup určí , či daná hodnota označená ako label platí, alebo nie. Teda či je True alebo False. DataModelCreator umožní nastaviť aj jeho defaultnú hodnotu. Label v tomto prípade tiež určuje názov (name)
- **DButton** – v prípade workflow manažment systému má toto tlačítko dôležitú funkciu, a to je spúštanie prechodov namodelovaných v Petriho sieťach, tým, že užívateľ potvrdí formulár, tak po splnení určitých podmienok sa celý proces posunie do ďalšieho stavu, vráti na predchádzajúci, alebo ostane na tom istom. Posledný prípad je možný napríklad v prípade, že formulár neprejde validáciou, workflow logika je namodelovaná tak aby po spustení ostala v tom istom stave a podobne.

2.2.2 Blok manažéry

Ako už samotný názov napovedá, v tomto bloku si popíšeme triedy zabezpečujúce riadenie aplikácie. Každá z nich má špecifickú úlohu.Aplikácia obsahuje štyri typy manažérskych tried: *DefaultSettingsManager, DatamodelSelectManager, SelectionManager a UndoManager.* Viac o týchto triedach si povieme nižšie.

DefaultSettingManager



Obrázok 10 Class diagram DefaultSettingManager

Táto trieda má za úlohu nastaviť inicializačné parametre po naštartovaní aplikácie. Obsahuje v sebe vytvorené inštancie tried *FontSetings, Template, Resolution*, ich gettre a settre.

Trieda *FontSettings* definuje typ používaného fontu, hrúbku písma, veľkosť písma. Pri vytváraní novej inštancie typu *DTextPane* (text na ploche) sa jeho parametre nastavujú podľa tejto triedy. Defaultné hodnoty si užívateľ môže nastaviť podľa seba.

Trieda *Resolution* (rozlíšenie) definuje šírku a výšku monitora, pre ktorý bude stránka optimalizovaná. Pri vytváraní nového dátového modelu aplikácia dá automaticky na výber z predefinovaných rozlíšení. Užívateľ si jedno z nich vyberie ak nie použije sa východzie nastavenie.

Aplikácia má predprípravu na pridávanie a úpravu šablón dizajnu atribútov formulára. Defaultne sa používa „business template“ . Template je súbor pozadí elementov formulára, definícia pozadia, typ písma a podobne.

DatamodelSelectManager

Aplikácia má vlastný pracovný nástroj, na ktorom si užívateľ vyberá akciu ktorá sa bude aplikovať na ploche (výber, pridávanie formulárových atribútov, text, obrázky,...). Táto trieda nám umožňuje jednoduchý prístup na zistenie , ktorý výber užívateľ uskutočnil. Táto trieda je typu *enum* , má predefinované možnosti výberu.

UndoManager

Akcie ktoré sa vykonávajú nad aplikáciou DatamodelCreator 2 web môžu byť vratné a nevratné. Napríklad užívateľ presunie formulárový element z určitého miesta, nakoniec sa ale rozhodne, že tento element chce naspäť tam kde bol. Ďalším presúvaním by hrozilo, že tento element by už nevrátil na to isté miesto a vznikli by súradnicové rozdiely. Z tohto dôvodu ako aj z dôvodu jednoduchšieho pracovania s aplikáciu sú isté funkcionality vratné. Medzi tieto funkcionality patrí napríklad aj vymazávanie formulárových atribútov a pridávanie atribútov. Na to aby aplikácia umožňovala funkciu REDO, UNDO, sme museli vytvoriť mechanizmus, ktorý by túto špeciálnu požiadavku spĺňal. Najprv sme urobili analýzu akcií, pre ktoré má byť táto akcia aktívna a pre ktoré nemusí byť aktívna, keďže akcia je jednoducho vratná.

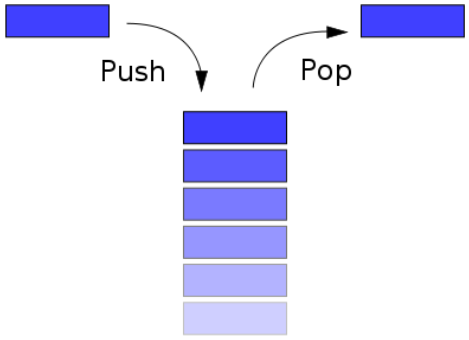
2.2.3 Commands

Command je rozhranie, ktoré špecifikuje metódy, potrebné na vykonanie redo, resp. undo

```
public interface Command {

    public void execute();

    public void undo();

    public void redo();

    public String getName();

}
```

Toto rozhranie musí implementovať každá trieda, ktorá riadi vratnú funkcionalitu aplikácie. Medzi tieto patria triedy obsluhujúce pridávanie atribútov, mazanie atribútov, presúvanie prvkov a podobne. Diagramy tried implementujúce rozhranie Command z dôvodu rozsiahlosti nebudeme ukazovať, je však možné v prílohe CD nazrieť do zdrojových kódov. Pre úplnosť spomenieme najzákladnejšie ako napríklad MoveElementCommand, AddDTextFieldCommand,.... Tieto triedy implementujúce toto rozhranie majú všetky potrebné metódy na vykonávanie funkcie redo/undo. Systém sme implementovali tak, že užívateľ po kliknutí na plochu s úmyslom pridať prvok, zavolá príslušnú akciu, ktorá vytvorí inštanciu príslušného commandu. V tomto commande si systém uchováva stav pred vykonaním akcie a po vykonaní akcie. Tieto akcie sú uložené v liste akcií a volaním funkcie undo, sa postupne vykonávajú metódy. Tento list

reprezentuje na určitej úrovni abstrakcie zásobník, do ktorého vkladáme inštancie tried implementujúce rozhranie Command(Push- execute). Vykonávaním príkazu undo sa spätne volá (Pop - undo).

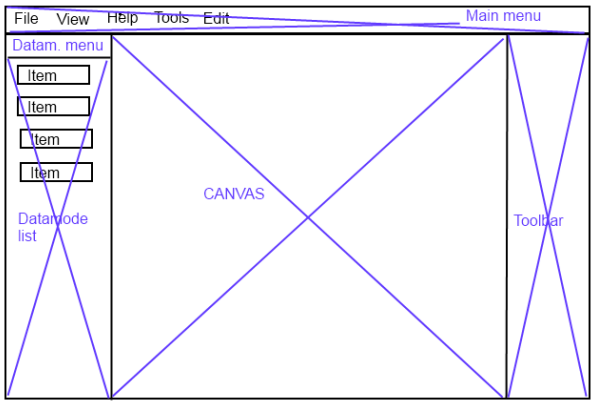


Funkcionality pre elementy obsiahnuté funkciami REDO/UNDO:

- Pridávanie
- Presúvanie
- Vymazávanie
- Kopírovanie

Návrh užívateľského rozhrania

Pri užívateľskom rozhraní sa veľmi dobre uplatní heslo „obal predáva“. Z tohto dôvodu sme zvolili hlavne prehľadný, moderný dizajn. Java Swing, poskytuje programátorovi dostatok priestoru na vytváranie vlastného layoutu.

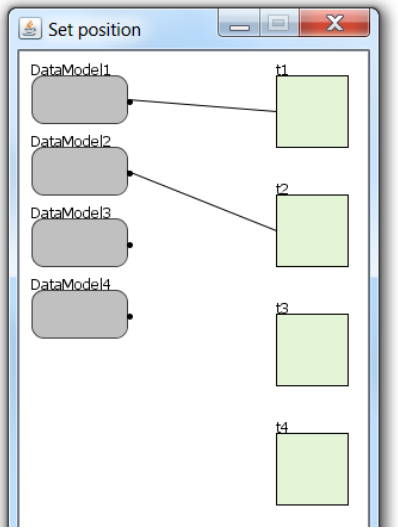


Obrázok 11 Návrh hlavného layoutu

Aplikácia bude obsahovať hlavné menu *topbar*, ktorý bude obsahovať viaceré menu položky (*Menu items*) a to *FILE, EDIT, TOOLS,* a *HELP*. Položka FILE, bude obsahovať ponuku ukladania aktuálneho súboru s ktorým aplikácia pracuje a vytvorenie prázdneho dokumentu. Pre úplnosť aplikácia vždy pracuje s vytvorenou inštanciou triedy DOCUMENT, ktorá v sebe obsahuje všetky dáta o vytvorenom dátovom modely, namodelovaných stránkach, premenných a podobne. Pri otvorení aplikácie sa vytvorí nová (prázdna) inštancia tejto triedy a všetky úkony zo strany používateľa sa ukladá do tejto inštancie. Užívateľ môže exportovať svoje údaje dvomi spôsobmi. Uložiť súbor a export pre workflow. Viac v kapitole export. Ďalšou položkou hlavného menu je položka EDIT, v ktorej sa nachádzajú funkcionality ako UNDO, REDO a DELETE, tieto funkcionality má užívateľ k dispozícii aj v hlavnom pracovnom panely. V položke TOOLS sa nachádzajú aplikácie na prepojenie dátových modelov – *Association* a aplikácia na definovanie premenných *Variables*.

Asociácie

Asociácie zabezpečujú prepájanie vytvorených dátových modelov s konkrétnym prechodom v *Petriho sieti*, namodelovanej v PNeditore. Ak si túto sieť predstavíme v logike workflowu, potom prechod (*transition*) v Petriho sietí môžeme považovať za proces, ktorý musí konečný užívateľ vykonať na to aby sa workflow systém dostal do ďalšieho stavu. V našom prípade bude workflow vytváraný formou web aplikácie, bežiacou na servery a komunikujúcou s užívateľom formou interakcie pomocou web stránok. Na základe týchto predpokladov *DatamodelCreator 2Web* generuje webové stránky, ktoré bude tento systém na servery zobrazovať užívateľovi. Tieto generované stránky je ale nutné logicky prepojiť s logikou namodelovanou pomocou Petriho sietí v PNeditore. Po namodelovaní dátových modelov (a prislúchajúcich web stránok) užívateľ pomocou prehľadnej aplikácie prepojí dátové modely s prechodmi.



Obrázok 12 Ukážka nástroja na vytváraní asociácií

Workflow premenné (Variables)

Pred tým ako budeme priraďovať dátové modely prechodom a exportovať súbor ZIP do workflow systému je potrebné definovať premenné pre workflow manažment systém. Tieto premenné sú asociované s formulárovými prvkami na dátových modeloch. Na príklad nech máme dátový model „Lekár“. Web stránka ktorú uvidí lekár obsahuje formulárové prvky ako textové polia, textové zóny a podobne. Každý tento formulárový vstup musí zastupovať určitú premennú vo workflow systéme. V opačnom prípade by sa tieto dáta stratili hneď po potvrdení formulára. Pomocou týchto premenných je workflow engine schopný preklápať údaje vložené užívateľom do formulára s biznis triedami a následne tieto dáta spracovať. Tieto dáta tiež môžu byť do formulárových vstupov aj vkladané. Napríklad po vyplnení formulára sestrou, sa dáta pomocou premenných spracujú vo workflow engine , ten rozhodne ktorý proces bude nasledovať. Po vyhodnotení zobrazí prislúchajúci formulár, napríklad formulár lekára. Lekár potrebuje vedieť údaje pacienta, ako napríklad meno, priezvisko, rodné číslo a podobne. Pomocou premenných tieto dáta vloží do formulára a lekárovi sa tieto dáta zobrazia.

Na zabezpečenie definovanie premenných workflow -u, slúži interný modul aplikácie DataodeCreator2, pomocou ktorého užívateľ je schopný definovať názov premennej a jej dátovy typ. Definovanie dátového typu slúži Validačnému modulu (viď analýza) na zistenie o aký dátový typ v rámci formulárového vstupu ide. Na základe toho javascript – ový skript validuje vstupy koncového používateľa workflow webovej aplikácie.

Datamodel menu

Na ľavej strane užívateľského rozhrania aplikácie je jednoduchá ponuka dátových modelov. Pomocou *minitoolbar-u* užívateľ môže pridávať a mazať dátové modeli. Pod týmto toolbarom je umiestnený multifunkčný list dátových modelov. Po kliknutí navybraný dátový model sa na ploche (canvas) zobrazí, po načítaní údajov, namodelovaná stránka pre dátový model. Pravím tlačítkom na prvok v liste sa aktivuje menu dátových modelov, kde užívateľ môže upravovať dáta modelov.

Panel nástrojov

Z dôvodu jednoduchšej dostupnosti sme panel umiestnili na pravej strane vedľa modelovacej plochy.Panel je rozdelený grafickým oddeľovačom na viacero častí: editovacia časť, formulárové vstupy, textová a grafická časť. V prvej časti sú umiestnené funkcionality ako REDO, UNDO, DELETE, SELECT a RULER.

Ruler (pravítko) slúži na zjednodušenie umiestňovania formulárových vstupov. Umožňuje ľahšiu orientáciu na ploche a dáva používateľovi informácie o vzdialenostiach od kraja obrazovky, čím je schopný presne umiestniť konkrétny prvok na stránke podľa požiadaviek. Na x/y osi sú zobrazené mierky v pixloch (px) v 25 pixelových blokoch. Po namodelovaní stránky je funkcionalita deaktivovateľná.

Select umožňuje výber konkrétnych formulárových, grafických, alebo textových prvkov na stránke, či už po jednom, alebo viac prvkov naraz. Kliknutím na prvok je prvok automaticky vybratý a ostatné prvky sú doznačené. Kliknutím a podržaním myši sa aktivuje *multiselect* mód a ťahaním myšky sa označia elementy vo vytvorenom výberovom obdĺžniku. Po označení a opätovnom kliknutím a podržaním myši na elemente, (resp. elementoch) môže užívateľ vybrané elementy presúvať. Každé presunutie prvkov je vratná akcia.*(viď Undo manager.*

V druhej časti sa budú nachádzať formulárové prvky, na ktoré keď užívateľ klikne môže pridávať na plochu. Tieto prvky sú TextField – textové polia, TextArea – textová zóna, checkbox- tlačítko s možnosťou výberu, radio button- tlačítka s možnosťou výberu jednej položky z viacerých položiek, submit button – tlačítko formulára, comboboxy- roletový výber.

Ďalšie dve časti sa týkajú grafickej a informatívnej perspektívy web stránky, Užívateľ po zvolení výberu TextPane, môže vkladať ľubovoľný text na plochu a ten upravovať podľa

požiadaviek. Môže vybrať font písma, veľkosť a typ. Užívateľ bude môcť vkladať obrázky na plochu a tiež zvoliť farbu pozadia.

Plocha

Pri vytváraní dátových modelov si užívateľ zvolí rozlíšenie obrazovky, pre akú bude modelovaná stránka optimalizovaná. Plocha reprezentuje webovú stránku a užívateľ v implementovanej aplikácií bude schopný metódou drag&drop vkladať atribúty, presúvať ich. Na plochu sa môže tiež vkladať text, a meniť farbu pozadia.

2.3 Implementácia

Na tvorbu aplikácie sme zvolili vývojové prostredie NetBeans 7[1]. Toto vývojové prostredie umožňuje automatické dopĺňanie kódu, generovanie konštruktorov, gettrov a settrov ako aj prehľadný debbuger.

Programovací jazyk sme zvolili Javu s platformou Java JDK 7[1]. Táto verzia poskytuje zabudovanú triedu JLayer, ktorá slúži na oživenie Swing GUI.

2.3.1 Koreň aplikácie

Za Koreň aplikácie môžeme považovať triedu DatamodelCreator. Pomocou statickej metódy *Properties. userNodeF[1]orPackage(this.getClass());* na jednom počítači sú viacerí užívatelia schopní pracovať súčasne. Táto trieda umožňuje, aby každý užívateľ je schopný volať novú inštanciu tejto triedy, ktorá obsahuje koreňové atribúty, pomocou ktorých sme schopný získať ľubovoľný aktuálny stav modelovania dátových modelov.

2.3.2 Implementácia užívateľského rozhrania

Užívateľské rozhranie sme navrhovali tak, aby práca so samotnou aplikáciou bola intuitívna. Celá aplikácia je postavená na hlavnom rámci (MainFrame), na ktorý sme postupne pridávali jednotlivé panely. Rozhranie môžeme vidieť na obrázku 7 GUI. Na ľavej strane hlavného rámca je umiestnený panel pre manipuláciu, pridávanie, vyberanie, vymazávanie dátových modelov. Tento panel je rozdelený na 2 časti . Prvá časť je *Toolbar* v ktorom je užívateľ schopný pridávať a mazať dátové modeli. V druhej časti je umiestnený list dátových modelov, v ktorom si užívateľ vyberie, s ktorým chce pracovať.

[1] v dobe implementácie najnovšia verzia

Pre tento list je aktívne *PopUpMenu* , ktoré sa aktivuje po kliknutí myškou pravého tlačítka. Toto menu poskytuje užívateľovi asociovať dátový model s prechodmi Petriho siete, premenovávať, vymazávať dátové modely. Na hornej lište sa nachádza hlavný *MenuBar* obsahujúce základné funkcie pre aplikáciu

V strede celého rámca je plocha (Canvas), na ktorej užívateľ modeluje formuláre. Podobne ako pri liste dátových modelov, aj pri ploche je aktívne *PopUpMenu,* ktoré aktivuje vybrané akcie pri kliknutí na konkrétny uzol. Napríklad pri kliknutí pravým tlačítkom myši na dátove pole (DTextField) sa aktivuje výber dátového typu a užívateľ si môže vybrať, aký dátový typ tu bude používaný. Pri kontrolných tlačítkach (DCheckBox) je zrejmé, že sa jedna jedine o hodnotu *TRUE/FALSE* a pri kliknutí pravým tlačítkom na tento element sa aktivuje akcia umožňujúca užívateľovi nastaviť defaultnú hodnotu.

Pravá strana hlavného rámca poskytuje užívateľovi jednoduchý a rýchly výber nástroja. Jednotlivé logické časti sú od seba oddelené oddeľovačom. Do prvej časti sme umiestnili nástroje pre základné operácie nad elementmi dátového modelu. To sú : Výber, Akcia späť/opätovné vykonanie akcie (UNDO/REDO), Vymazávanie elementov, a nástroj pravítko, ktorý uľahčuje zarovnávanie atribútov formulára.

V ďalšej časti si užívateľ môže vybrať s viacerých vstupov pre formulár.

Textfield – textové pole. Jedná sa o jednoriadkový vstup. Pre tento vstup je špecifické, že môžeme špecifikovať dátový typ, ktorý budeme očakávať od koncového užívateľa (osoba , ktorá formulár vypĺňa) a aj jeho rozsah. Napríklad vyberieme že jeho dátový typ bude integer s hodnotou od 0 do 100. Na stránke iná hodnota potom nebude akceptované a validátor na to užívateľa upozorní.

TextArea – Jedná sa o jednoduchú textovú zónu. Jej špecifikom je, že koncový užívateľ sem môže zadávať viacriadkový text. Jej veľkosť sa dá dynamicky meniť a regulovať aj v aplikácií aj na stránke.

Button –Tlačítko môže mať viacero funkcií. Môže potvrdzovať formulár a posúvať proces do ďalšieho stavu. V petriho sieti to znamená, že správne vyplnený formulár predstavuje „enabled transition“ a tlačítkom „submit“ sa vykoná akcia „fire“ Môže sa

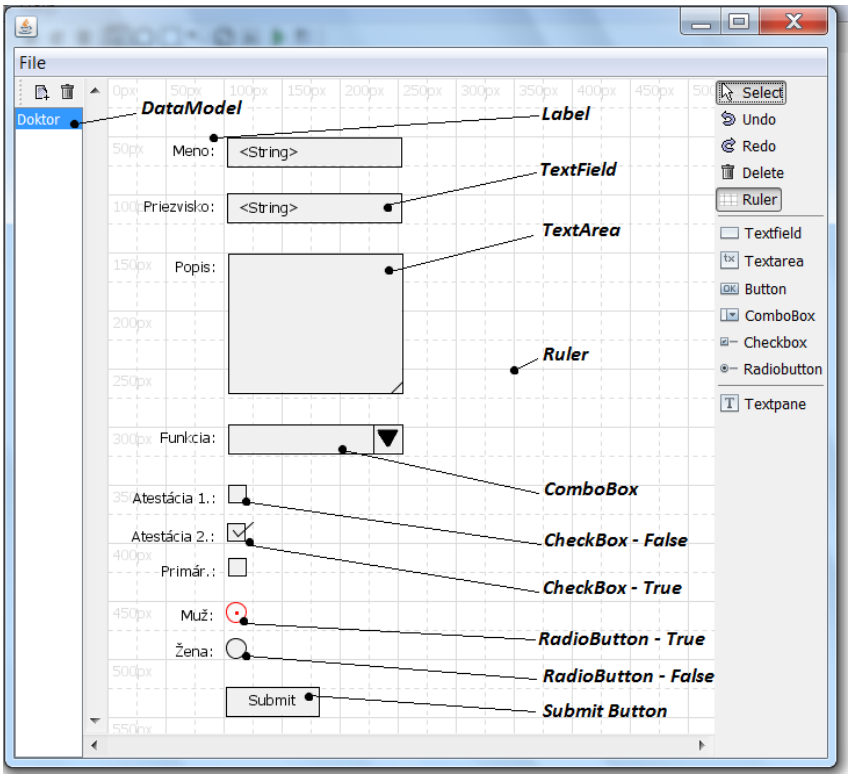
jednať o tlačítko „zrušiť“, ktorý proces posúva späť , respektíve ostáva stáť v tom istom stave.

ComboBox – Roletový výber. Roletový výber sa používa pri atribútoch, o ktorých vieme ktoré hodnoty môže nadobúdať. Napríklad pohlavie- Muž, Žena. Pracovná pozícia : Lekár, Sestra, Rádiologický asistent, ... Výber je rýchly, uľahčuje prácu koncovému užívateľovi a redukuje možnú chybu. Aplikácia umožňuje nadefinovanie týchto možných výberov aj s výberom defaultného.

CheckBox – Kontrolné tlačítko. Používa sa pri jednoduchom zisťovaní či daný atribút platí, alebo nie. Môže nadobúdať len hodnoty T/F Aplikácia umožňuje užívateľovi vybrať defaultnú hodnotu.

RadioButton- Jeho funkcionalita je podobná roletovému výberu. Každé tlačítko je viazané ku skupine tlačítok Pri prvom pridaní tlačítka na plochu sa automaticky vytvára nová skupina (RadioButtonGroup). Pri opätovnom pridaní je užívateľ vyzvaný vybrať s existujúcich, alebo vytvoriť ďalšiu skupinu.

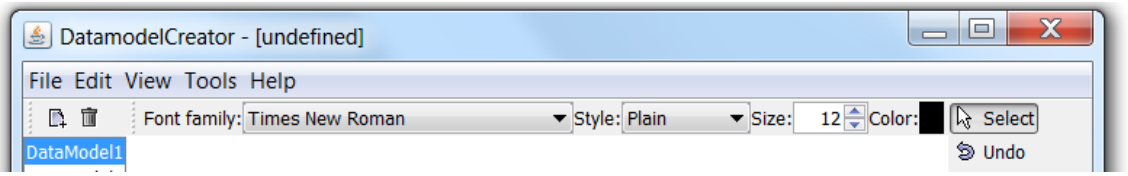
Konkrétnu implementáciu formulárových vstupov, môžeme vidieť na obrázku 11.



Obrázok 13 Príklad dátového modelu 1

Štýl písma

V nasledujúcej časti pracovného panelu, je umiestnené tlačítko TextPane a Default font. Po stlačení aplikácia zachytáva akciu a volá sa príslušný *handler*, ktorý vytvorý na vrchnej časti panel , v ktorom si užívateľ môže definovať parametre písma. To jest môže definovať font, veľkosť písma, a štýl písma. Tento nástraj tak umožňuje užívateľovi vytvárať text podľa svojich požiadaviek na dizajn stránky, na ktorý sa v súčasnej dobe kladie čoraz väčší dôraz. Konkrétnu implementáciu tohto nástroja môžeme vidieť na obrázku 14

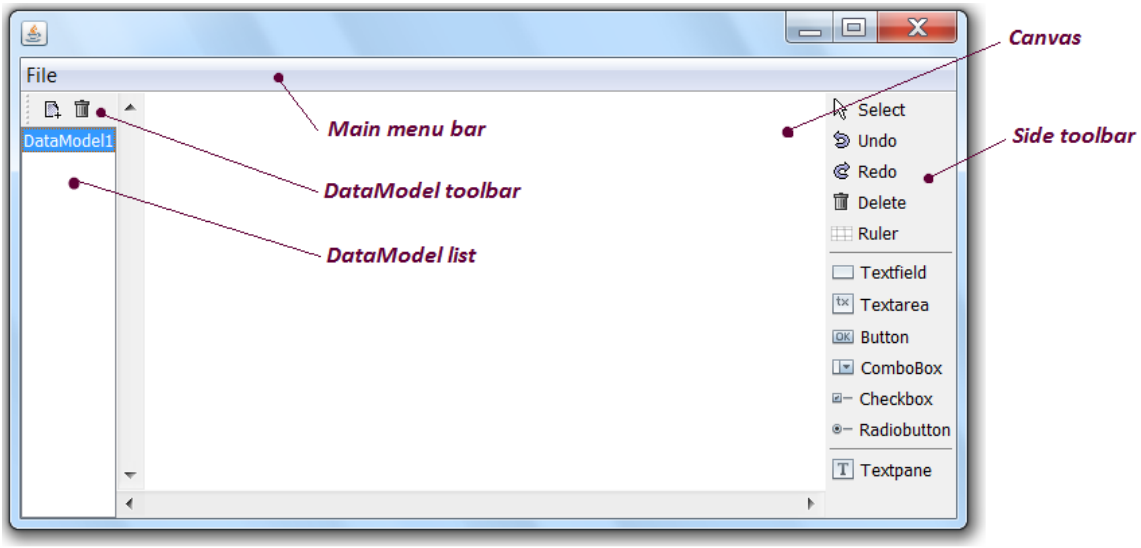


Obrázok 14 Ukážka GUI nastavovania parametrov textu

Typy fontov sú real-timeovo získavané z triedy GraphicsEnvironment, ktorá poskytuje list dostupných fontov prostredníctvom metódy *getAvailableFontFamilyNames()*. Farba textu sa mení prostredníctvom užívateľsky prehľadného nástroja , ktorý poskytuje priamo Java Swing. Prostredníctvom vytvorenej inštancie triedy JColorChooser, je užívateľ schopný presne definovať farbu ktorú požaduje viacerými spôsobmi. Užívateľ si môže priamo vybrať farbu z dopredu nadefinovaných základných farieb, HSV metódou (Hue, Saturation, Value), HSL (Hue, Saturation, Lightness), RGB (Red, Green Blue), CMYK (Cyan, Magenta, Yellow).

Farba pozadia

Podobne ako aj farbu textu môžeme definovať prostredníctvom nástroja ColorChooser ako sme popísali v predchádzajúcom odstavci. Po vybratí užívateľom farby pozadia, sa farba uloží do inštancie konkrétneho dátového modelu. Farba pozadia sa prejavý na stránke prostredníctvom vygenerovaného „css“ súboru, ktorému sa budeme venovať v časti export.



Obrázok 15 GUI

2.3.3 Export

Najdôležitejšia časť aplikácie je jej export. Bez výstupu by žiadna aplikácia nemala význam. Výstup aplikácie sme navrhovali tak aby čo najoptimálnejšie a najefektívnejšie spĺňal požiadavky na workflow engine. Samotný výstup aplikácie je súbor „.zip“ . Aplikácia najskôr vygeneruje súbor „settings.xml“. Tento formát sme zvolili pre jeho čoraz väčšie využívanie v praxi, je ľudsky čitateľný. Nevýhodou tohto formátu je rýchlosť strojového spracovania a však v súčasnej dobe táto negatívna stránka xml formátu bola prirodzene odstránená čoraz výkonnejším hardvérom.

Súbor settings.xml v sebe nesie informácie ohľadom namodelovaných premenných workflowu a tiež definovaných asociácií. Pre obraznosť uvedieme jednoduchú ukážku:

```
<settings>
  <variables>
    <variable name="name" type="String"/>
    <variable name="surname" type="String"/>
    <variable name="num" type="Double"/>
  </variables>
  <dataModels>
    <dataModel name="DataModel1">
      <transition>t1</transition>
      <transition>t2</transition>
    </dataModel>
```

```
</settings>
```

Po rozparsovaní tohto „.xml“ súboru workflow engine definuje premenné do svojich interných databázových tabuliek a mapuje jednotlivé premenné s atribútmi formulára prostredníctvom relačno-objektového mapovania.

Ďalšou časťou exportu je zložka „web“, v ktorej sa nachádzajú aplikáciou vygenerované „.html“ stránky s prislúchajúcimi „.css“ súbormi.

Knižnica na generovanie html a css súborov

Na generovanie html a css stránok sme si vytvorili vlastnú jednoduchú knižnicu. Táto knižnica pozostáva z týchto základných tried : Atrribute, Attributes, CssAttribute, CssDocument, CssGlobalUnits, CssUnit, HtmlExporter, HtmlGenerator, HtmlPage ,Tag, TagType, ktoré si podrobne popíšeme.

Trieda **Tag** – Táto trieda reprezentuje html tág a je potomkom triedy LinkedList. Má viacero konštruktorov, ktoré umožňuje flexibilitu použiteľnosti. Umožňuje užívateľovi definovať atribúty, určiť či ide o párový, alebo nepárový tág. Nesie v sebe zabalené inštancie triedy Attributes, meno tágu a definícia párovosti.

Trieda **Attribute** – Táto trieda predstavuje atribúty tágu, V praxi to znamená napríklad atribút „name, id, a podobne“. Pomocou týchto atribútov sme schopný generovať identifikátory tágov nielen pre workflow systém, a css súbory, ale aj pre modul validácie. S touto triedou súvisí aj trieda Attributes - množina atribútov tágu, reprezentuje ju reťazec do ktorého sú tieto atribúty zapísane.

Trieda **CssUnit** – táto trieda reprezentuje jednu definíciu kaskádového štýlu. Pre obraznosť uvedieme príklad:

```
.label{
        position:relative;
        float:left;
        margin-right:10px;
}
```

V tomto prípade je definícia „.label“ jediná v definícií kaskádových štýloch. Bodka (.) reprezentuje typ css jednotky, ktorá v tomto prípade znamená „class“ teda triedu. Táto definícia sa používa pri viac násobnom používaní definície v html súbore.

Táto trieda v sebe nesie vytvorené inštancie mapy Map<String, TagType>, kde String predstavuje názov definície a enumeračná trieda TagType dovoľuje definovať typ definície (id, class, normal). V praxi sa poväčšine používa len jediná definícia css jednotky, môže sa však definovať aj viac naraz napríklad div.h1,div.h2 {color:white}. V tomto prípade by v mape boli uložené dve položky. A to {(div.h1,normal),(div.h2,normal)}. Trieda umožňuje definovať aj to či má byť jednotka aktívne len v prípade udalosti keď užívateľ príde na element myšou. V kaskádových štýloch sa táto voľba definuje pridaním atribútu „:hover“ za hlavičku jednotky. Trieda ďalej v sebe nesie informácie o atribútoch teda v predchádzajúcom príklade by to bol „color:white“. Tieto položky má uložené v inštancí listu List<CssAttribute>

Trieda **CssAttribute** – Trieda reprezentuje dvojicu názov – hodnota. Názov znamená meno atribútu v predchádzajúcom príklade by to bolo “color”, jeho hodnota by bola “white”.

Trieda **CssGlobalUnit** – Táto trieda je podobná ako trieda CssUnit. Obsahuje však v sebe referenciu na DPage- modelovanú stránku, a metódy, ktoré slúžia na inicializovanie základných css jednotiek opakujúcich sa vo všetkých stránkach. Na príklad nastavenie farby pozadia podľa namodelovaných farieb v aplikácií, definovanie relatívnej pozície labelu(štítku), a vstupu v rámci balíčkového divu, v ktorom sa obe nachádzajú. Následne tento „div“ bude mat v rámci stránky absolútnu pozíciu. Toto nám umožní vložiť na stránku tieto atribúty tam kde si ich aj užívateľ v aplikácií namodeloval. To jest absolútna pozícia od ľavého horného okraja počítaná v pixloch.

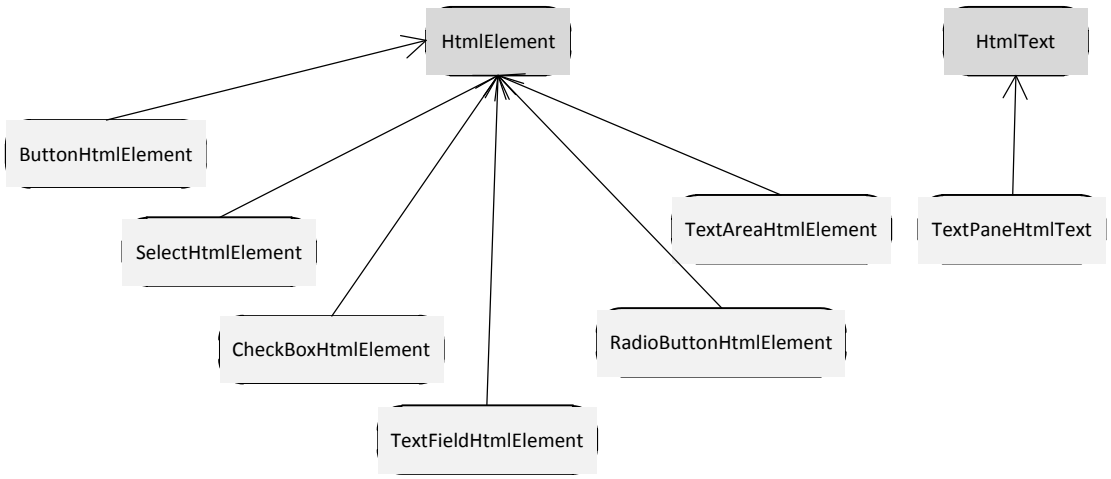
Trieda **CssDocument** – Táto trieda balí inštancie tried CssGlobalUnit a vytvára list CssUnit. Inak povedané táto trieda objektovo reprezentuje celý „.css“ súbor, Preťažuje metódu „toString“ a tým podáva celý súbor vo forme reťazca ktorý sa zapíše do súboru.

Trieda **HtmlPage** – Táto trieda je objektovým odrazom html stránky. Obsahuje inicializačnú metódu na vytvorenie štandardnej hlavičky html stránky – „<head>“, ktorá zahŕňa tág „<title>“, ktorá určuje názov stránky, „<meta>“ tágy , pomocou ktorých definujeme obsah stránky a jej kódovanie. V našom prípade sme zvolili kódovanie UTF-8, ktoré zabezpečuje správne zobrazenie všetkých znakov na stránke. Po inicializácií hlavičky, kódovanie stránky a nastavenie názvu, vytvárame „<form>“ tág. Tento tág obsahuje dva základné atribúty „action“ a „method“. Pomocou atribútu „action“ definujeme kam nasmerovať workflow po potvrdení formulára. Toto smerovanie zabezpečuje workflow engine, ktorý definuje logiku na základe petriho siete namodelovanej pomocou PNeditora. Na základe tejto siete a asociácií vytvorených v DataModelCreatore workflow engine vyhodnotí ktorá stránka (proces) má nasledovať. Atribút „method“ popisuje metódu posielania údajov zo stránky. V našom prípade budeme kvôli bezpečnosti prenosu dáta prenášať POST – metódou. Každý formulár obsahuje pod svojou hlavičkou. Skrytý vstup , ktorý slúži na definovanie nasledujúceho prechodu, pre potreby workflow enginu. Trieda HtmlPage ďalej zisťuje aké formulárové vstupy užívateľ vložil na stránku a tie vkladá na stránku roztriedené podľa typu. Vytvorí pre každý vstup „div - tág“, ktorý má definované ID atribút. Podľa ktorého sa určuje absolútna poloha tohto tágu. Do vytvoreného tágu sa zabaľuje formulárový vstup (textfield, textarea, ...) a jeho názov – label. V ďalšej metóde „setUpCss“ táto trieda vytvára súbor „.css“ na základe vytvorenej inštancie triedy CssDocument.

Trieda **HtmlGenerátor** – úlohou tejto triedy robiť rozhranie medzi objektovo uchovávanými informáciami o stránkach a ich súborovou reprezentáciou. Prostredníctvom DatamodelCreator singletonu, sme získali aktuálne definované dátové modely. Do konštruktora táto trieda dostáva adresár do ktorého sa budú vkladať html stránky. Tento adresár má v adresárovej štruktúre názov „web“. Do tohto adresára sa budú generovať zložky. Každá zložka nesie názov po dátovom modely, ktorý užívateľ vytvoril. V každej zložke táto trieda vytvorí dva súbory. „.html“ dokument a „.css“ dokument jemu prislúchajúci.

Html elementy

Aby sme dodržiavali štandardy pre objektovo – orientované programovanie, vytvorili sme pre každý „tág“ reprezentujúci formulárový vstup triedu , ktorá tento tág reprezentuje, a dedí metódy zo svojho predka HtmlElement.



Obrázok 16 Hierarchia interných html tried

Trieda **HtmlElement** obsahuje metódy opakujúce sa pre každý formulárový prvok. Vytvára štandardný „div“ element do ktorého sa formulárový prvok zabaľuje, tomu tágu priraďuje triedu „label“, ktorá bola vytvorená do „css“ súboru ako štandardná. Trieda vytvára vo svojej *private* metóde novú inštanciu triedy CssUnit , v ktorej definuje id atribútu, a jeho absolútnu pozíciu.

Trieda **HtmlText** je nadtrieda všetkých prvkov obsahujúci text. Táto trieda vytvára inštanciu CssUnit, v ktorej definuje veľkosť písma , vybraný font, štýl písma a jeho pozíciu.

Trieda **ButtonHtmlElement** – táto trieda reprezentuje tlačítka na stránke. Vytvára tág „<input type=”submit”/>“, Nesie v sebe pomocné atribúty ako meno a id.

Trieda **CheckBoxHtmlElement** – trieda reprezentuje formulárový výber. Užívateľ sa rozhodne či potvrdí tento jedno výber, alebo nie. Trieda zabezpečuje aj preddefinované nastavenie tohto elementu, teda či bude na stránke sa zobrazovať užívateľovi preddefinovane potvrdený alebo nie.

Trieda **RadioButtonHtmlElement** – Tlačítka typu „radiobutton“ sú riešené prostredníctvom grúp. Jednotlivá grupa v sebe nesie súbor inštancií „radiobutton“ tlačítok. Jednotlivé grupy sú od seba nezávisle. Táto trieda umožňuje podobne ako pri

triede **CheckBoxHtmlElement** predefinovať ktoré tlačítko bude preddefinovane vybraté jednotlivo v každej grupe.

Trieda **TextAreaHtmlElement** táto trieda reprezentuje jednoduchý formulárový vstup. Môže nadobúdať hodnoty rôznych dátový typov. Preddefinovaný dátový typ je „string“. Užívateľ tento dátový typ definuje priradením premennej k tomuto formulárovému vstupu.

2.4 Systémové požiadavky

Aplikácia je implementovaná na platforme Java Sun. Odporúčané systémové požiadavky:

- Operačný systém: Windows XP/Vista/7
- Platforma: **Java SE 7**
- RAM: min. 256 MB
- Procesor: min. 1GHz

3 Zhodnotenie

Na začiatku tejto práce sa venujeme podrobnej analýze, v ktorej si postupne rozoberáme problematiku workflowu, jeho časti, informačné systémy a jeho nadväznosť s workflow manažment systémom. Podrobne sme si rozobrali základné stavebné jednotky workflow aj s príkladmi.

V ďalšej časti tejto práce sme sa venovali opisu riešenia danej problematiky. Zadefinovali sme si požiadavky na aplikáciu. Následne sme sa venovali návrhu optimálneho riešenia, pre tieto požiadavky. Navrhli sme „business“ triedy aplikácie a podrobne sme si popísali ich metódy, implementácie a dedičnosti.

Následne sme sa venovali implementácií aplikácie, popis funkcionality a prepojení medzi triedami. Popísali sme užívateľské rozhranie aj s obrázkami.

V tejto diplomovej práci som sa venoval vytvoreniu aplikácie, pomocou ktorej je užívateľ schopný rýchlo a bez potreby znalosti programovania vytvoriť stránky, ktoré budú súčasťou workflow systému. Pomocou tejto aplikácie si užívateľ namodeluje dátové modely , na základe ktorých sa tieto stránky vygenerujú. Užívateľské rozhranie je navrhnuté prehľadne takže užívateľ je schopný pracovať s aplikáciou aj bez predošlého „veľkého“ zaúčania. Aplikácia užívateľa nabáda na dodržovanie postupu pri vytváraní dátových modelov, a jeho funkcionalít. Napríklad užívateľ nemôže pridávať atribúty formulára, pokiaľ nie je vytvorený dátový model. Jednoduchým spôsobom „drag&drop“ užívateľ vie vytvoriť asociáciu, medzi dátovým modelom a prechodom Petriho siete. Aplikácia obsahuje aj nástroj na vytváranie premenných workflow, ktoré sa prideľujú jednotlivým atribútom formulárov v dátových modeloch.

Otestovaním aplikácie na jednoduchej prípadovej štúdií sme overili, že aplikácia je navrhnutá a implementovaná správne, čo potvrdilo „chodivosť“ a „živosť“ celého systému.

4 Záver

Úlohou tejto práce bolo analyzovať problematiku workflow manažment systémov a na základe poznatkov navrhnúť aplikáciu - nástroj, v ktorom užívateľ môže jednoducho vytvoriť dátový model, na základe ktorého aplikácia bude generovať webové stránky pre workflow engine , nadväzujúcej práce. Dôraz sme kládli hlavne na samotný návrh aplikácie, na základe ktorého sme aplikáciu implementovali. Aplikáciu sme implementovali na platforme Java^{tm} Sun 7, ktorá bola v dobe implementácie najaktuálnejšou verziou. Navrhli a implementovali sme prehľadné užívateľské rozhranie aplikácie prostredníctvom ktorého užívateľ jednoduchou drag&drop metódou si namodeluje workflow web stránku (vloží formulárové vstupy, nadefinuje premenné worflow,...), reprezentujúcu workflow proces. Aplikácia umožňuje asociácie týchto web stránok s prechodmi vytvorených v PNeditore. Užívateľské rozhranie pre tieto asociácie je intuitívne. Užívateľ len čiarou prepojí vo vytvorenom nástroji dátové modely s prechodmi. Vytvorili sme tiež nástroj na vytváranie workflow premenných. Tieto premenné sú neodmysliteľnou časťou formulárov. Premenné užívateľ prideľuje pomocou nástroja, ktorý sme vytvorili za týmto účelom. Po namodelovaní požadovaných workflow web stránok, aplikácia generuje worflow výstup – súbor „.zip“, ktorý je vstupom pre workflow engine. Tento súbor zahŕňa všetky namodelované web stránky vo formáte „.html“ s prislúchajúcimi „.css“ súbormi a obrázkami na stránkach, konfiguračný súbor „setting.xml“, pomocou ktorého workflow systému sú predané informácie o workflow premenných a asociáciách.

Táto práca je súčasťou generickej aplikácie a nadväzuje na bakalárske práce autorov. Aplikácie prešli úplnou zmenou návrhu, ako aj implementácie, poskytujú prehľadnejšie GUI, ako aj predchádzajúce aplikácie. Aplikácia DatamodelCreator 2 web využíva rozhranie PNeditor, poskytujúce informácie o namodelovanej sietí. Pomocou ďalšej nadväzujúcej aplikácie Petra Baranca, užívateľ môže vytvoriť validácie pre každý formulárový vstup , výstupom tejto práce je súbor „.js“, ktorý je vložený do „.zip“ súboru a poslaný workflow engine- u Tomáša Zúbera.

5 Zdroje

[1] Aalst, W., Hee, K.: Workflow Management Models, Methods, and Systems. The MIT Press Cambridge, Massachusetts London, England,1997, ISBN 0-262-01189-1

[2] Michael zur Muehlen, Workflow –based process controling, Berlin ISBN 3-8325-0388-9

[3] KOHÁRY, Richard. 2010. Tvorba dátového modelu pre formuláre: Bakalárska práca, 53strán

[4] Cardoso, J.,Camargo, H.: Fuzziness in Petri Nets. Heidelberg, 1997. ISBN 3-7908-1158-0

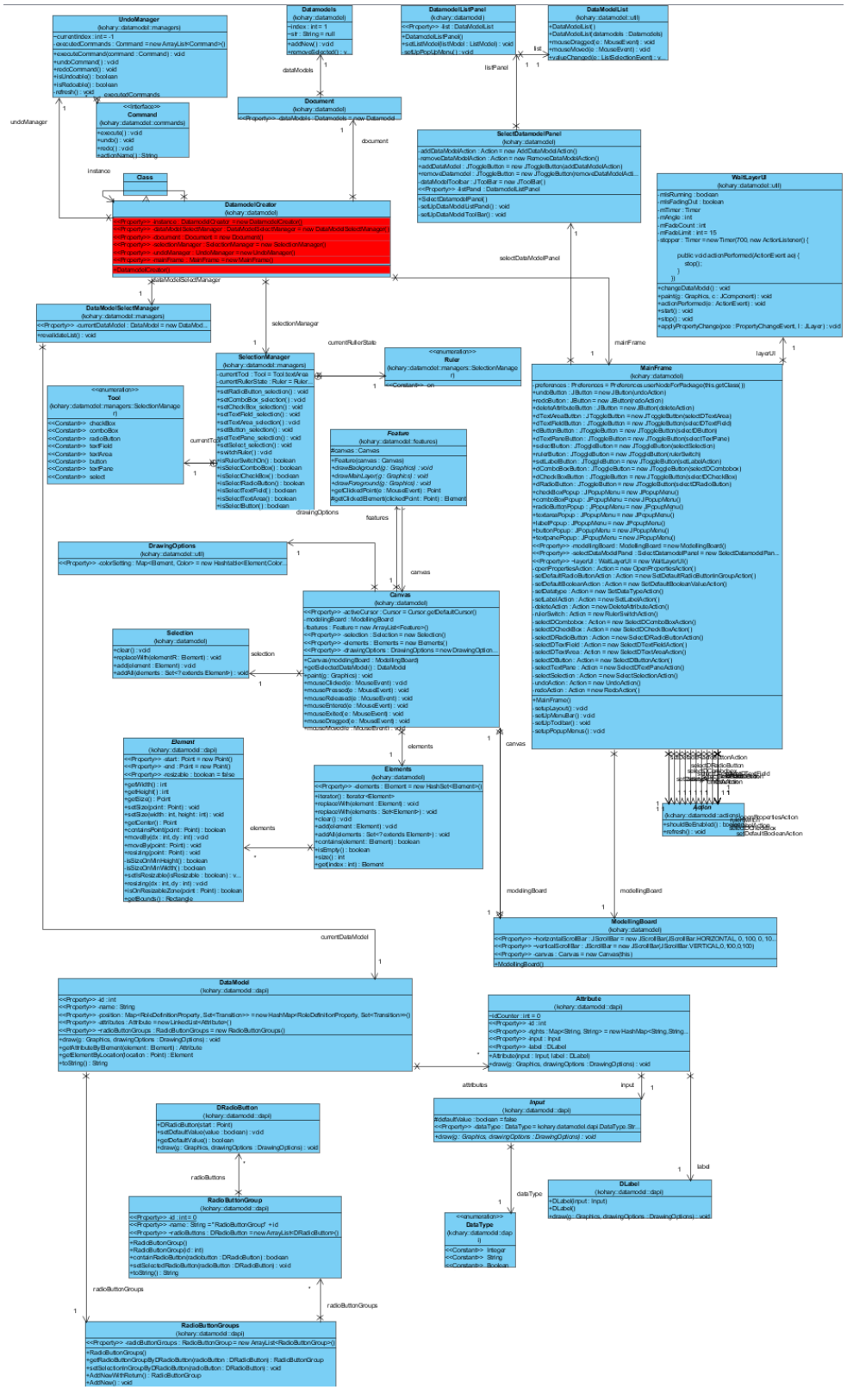
[5] S. Zakahour,S Hommel, J. Royal, I.Rabanovitch, T.Risser, M. Hoeber: Java6 Výukový kurz, Vydavateľstvo Computer Press, a.s.Vol. 2555, ISBN 978-80-251-1575-6

[6] Ehrig, H., Juhás, G., Padberg, J., Rozenberg, G., (Eds.) : *Unifying Petri Nets. Advances in Petri Nets.* Lecture Notes in Computer Science. Vol. 2128, Springer, 2001, 485pp. ISBN 3-540-43067-9

6 Zoznam príloh

1. Príloha A: Class diagram DatamodelCreator a jej väzieb
2. Príloha B: Užívateľská príručka
3. Príloha C: Architektúra vstupov a výstupov generickej aplikácie
4. Príloha D: PNeditor_datamodel_Creator.jar (CD)
5. Príloha F: Zdrojové kódy aplikácie DataModel Creator (CD)

Príloha A Class diagram DatamodelCreator a jej väzieb



Obrázok 17 Class diagram DatamodelCreator a jej väzieb

Príloha B Užívateľská príručka

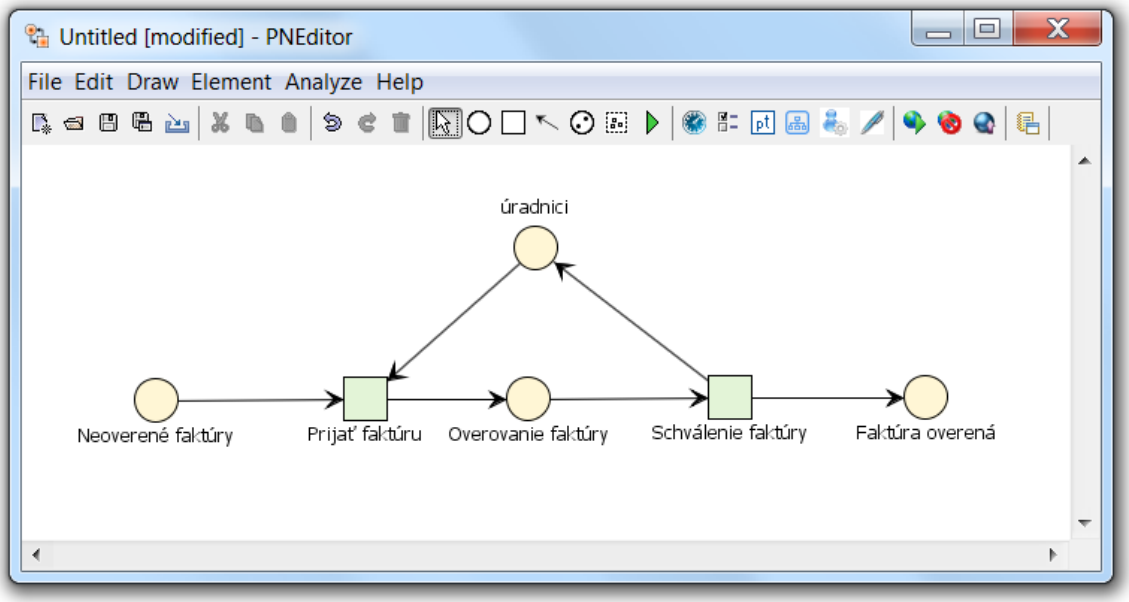
DataModel Creator 2 WEB

Užívateľská príručka

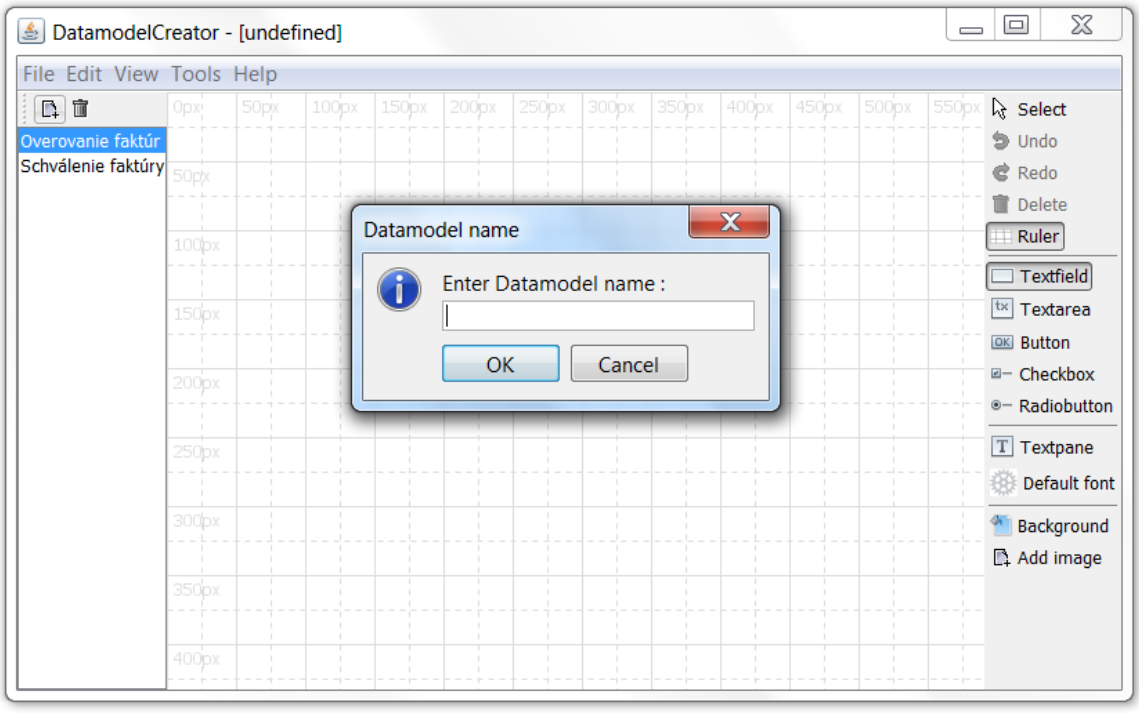
(Platná pre verziu:1.0)

Java(TM) 7

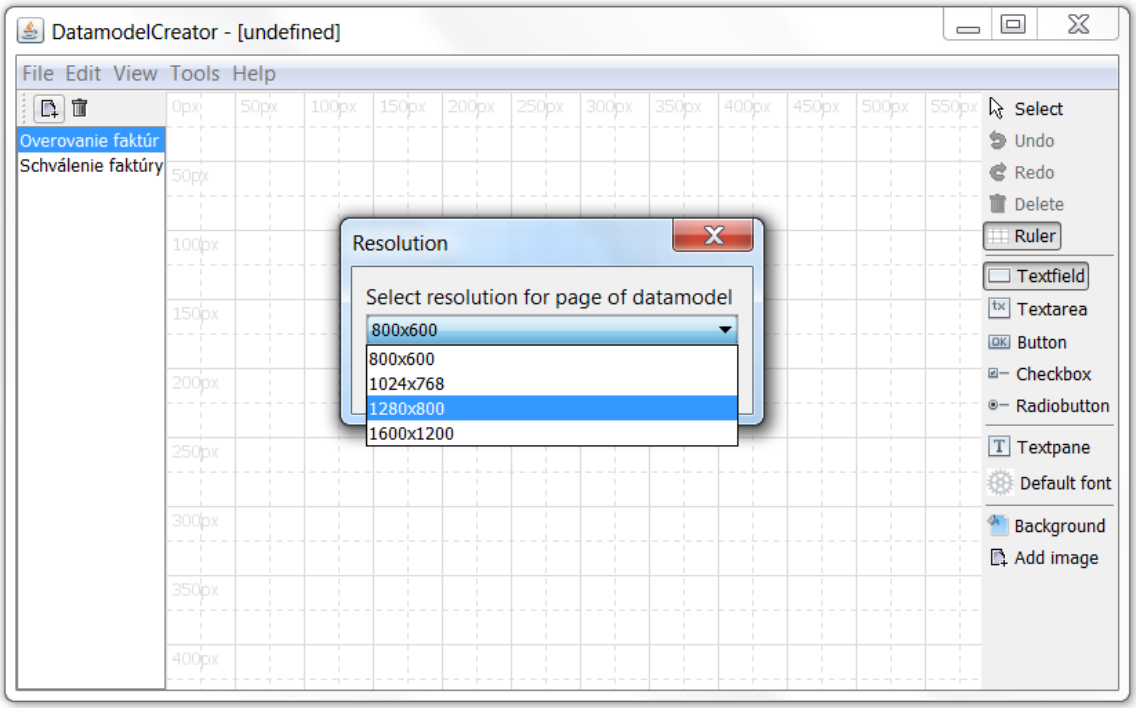
1. Vytvoriť logiku Petriho siete v PNeditore



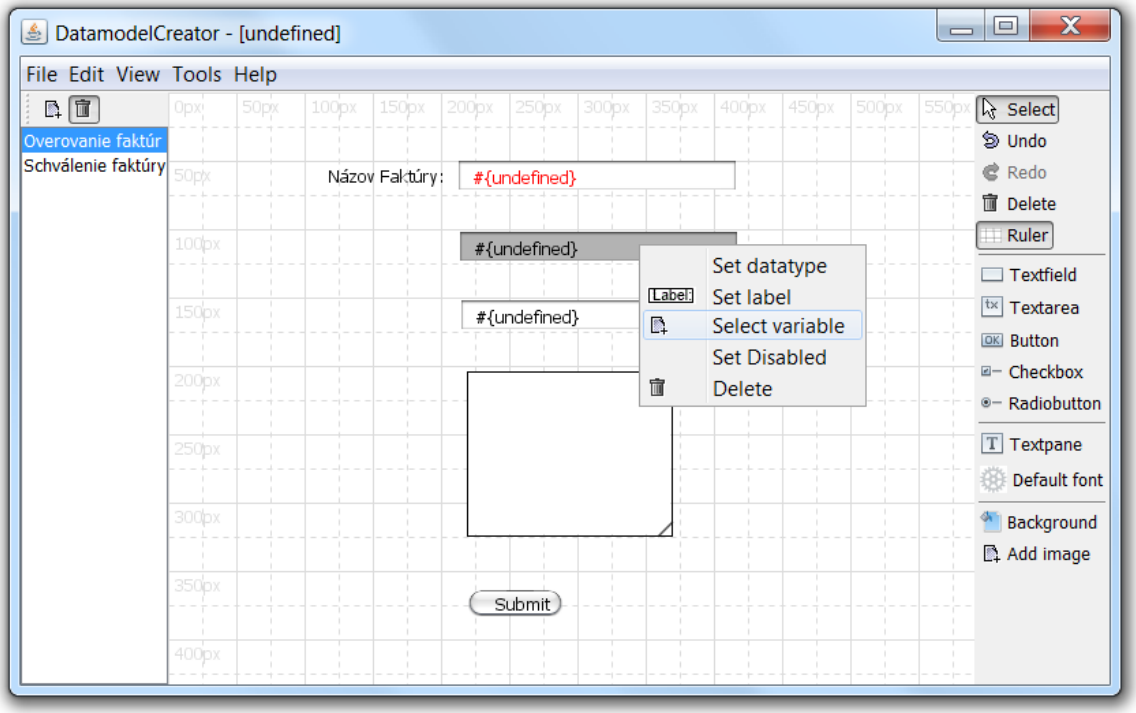
2. Vytvoriť dátový model, vložiť meno dátového modelu



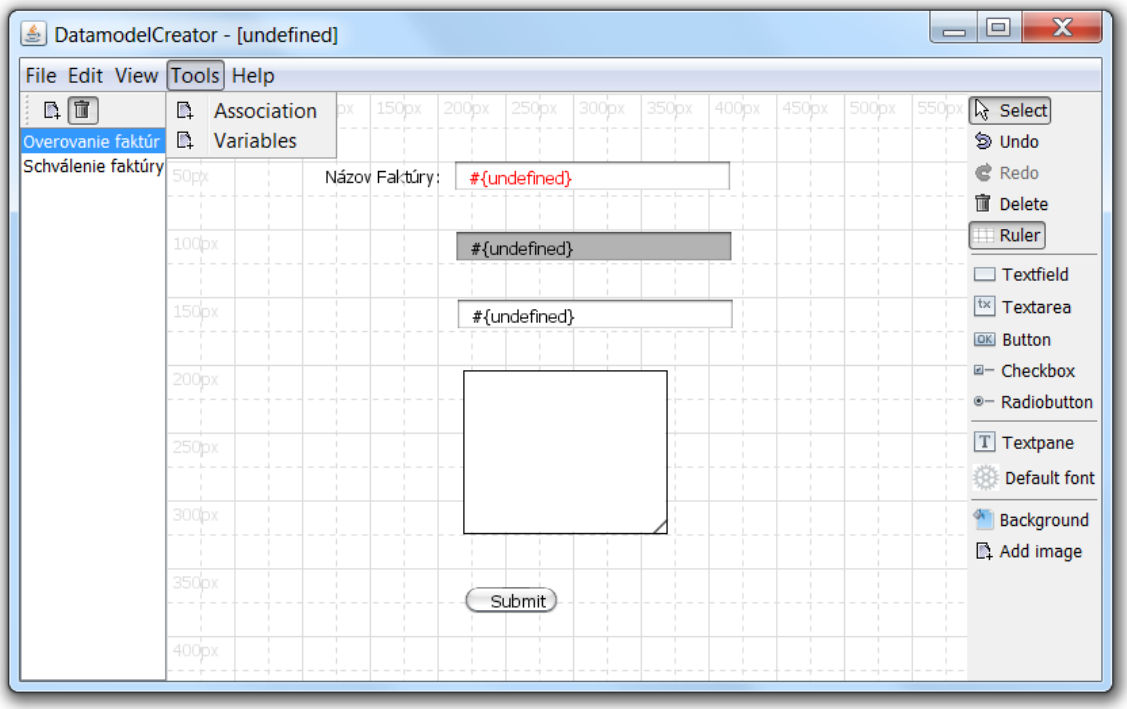
3. Zadať rozlíšenie pre ktoré budú web stránky optimalizované



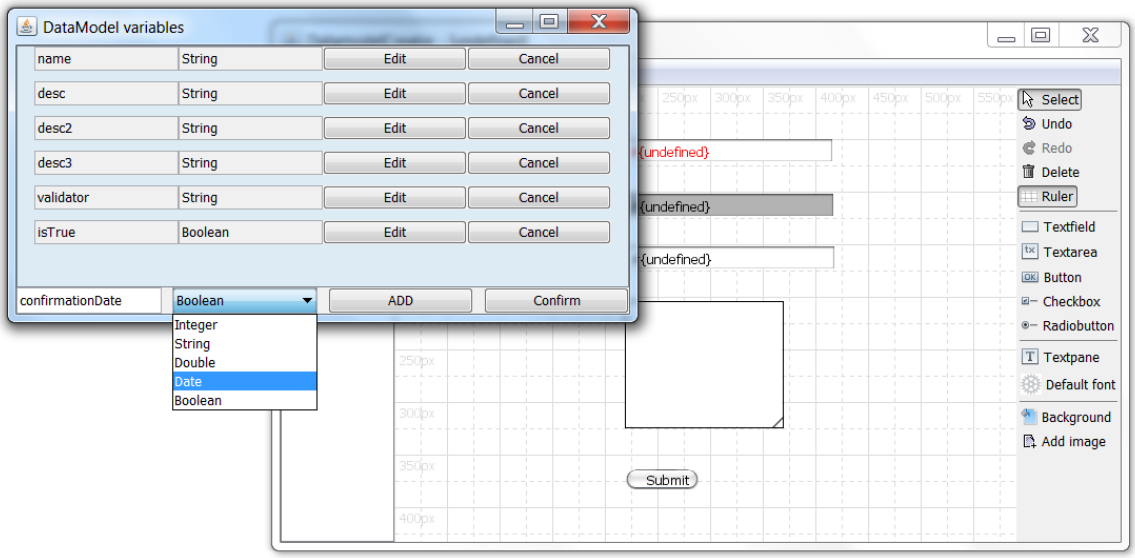
4. Vytvoriť formulár, drag&drop z pravého panela nadefinovať atribúty.
5. Pravým tlačítkom na atribút a zadať názov atribútu (Set label)



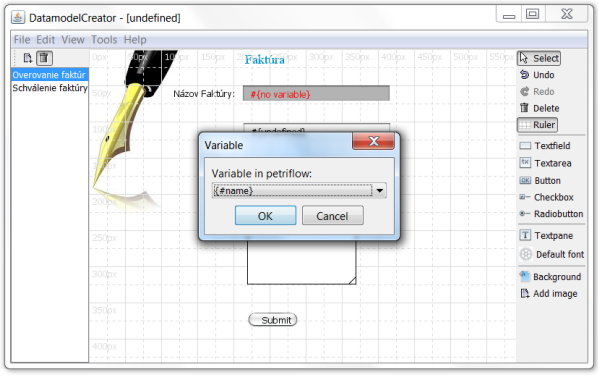
6. Vytvoriť premenné workflow – u. V položke TOOLS vybrať „Variables“



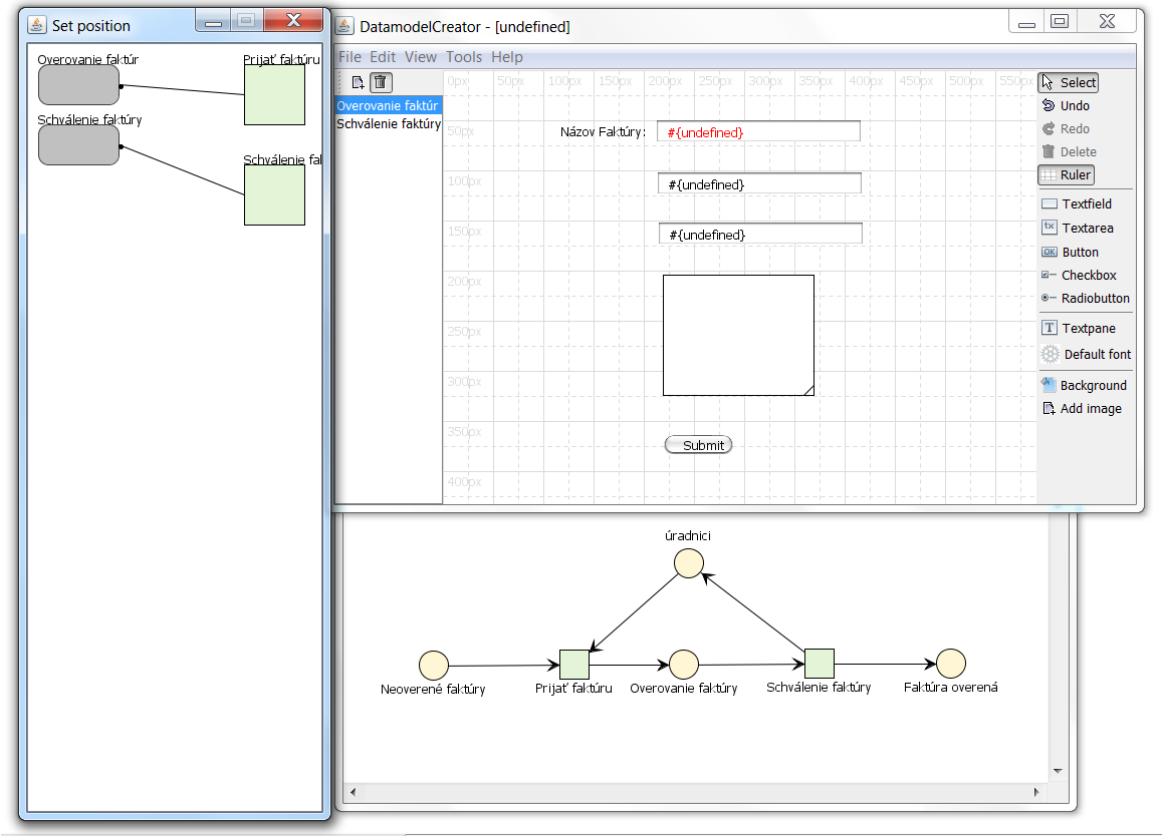
7. Definovať premenné. Zadať názov premennej a jeho dátový typ



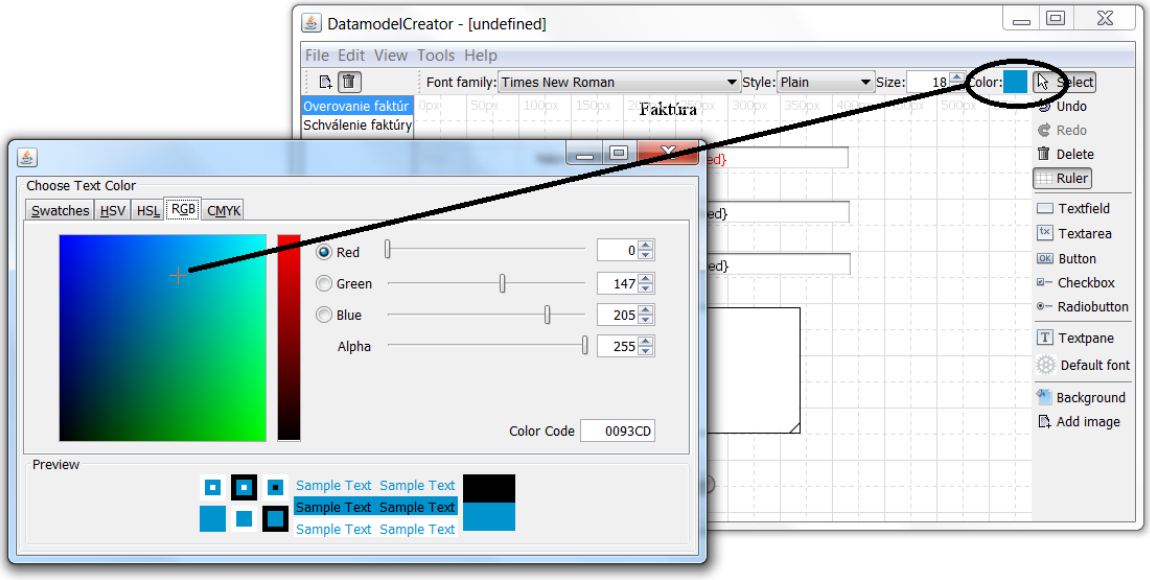
8. Prideliť atribútom definované premenné. Pravým tlačítkom na atribút a vybrať (Select Variable)



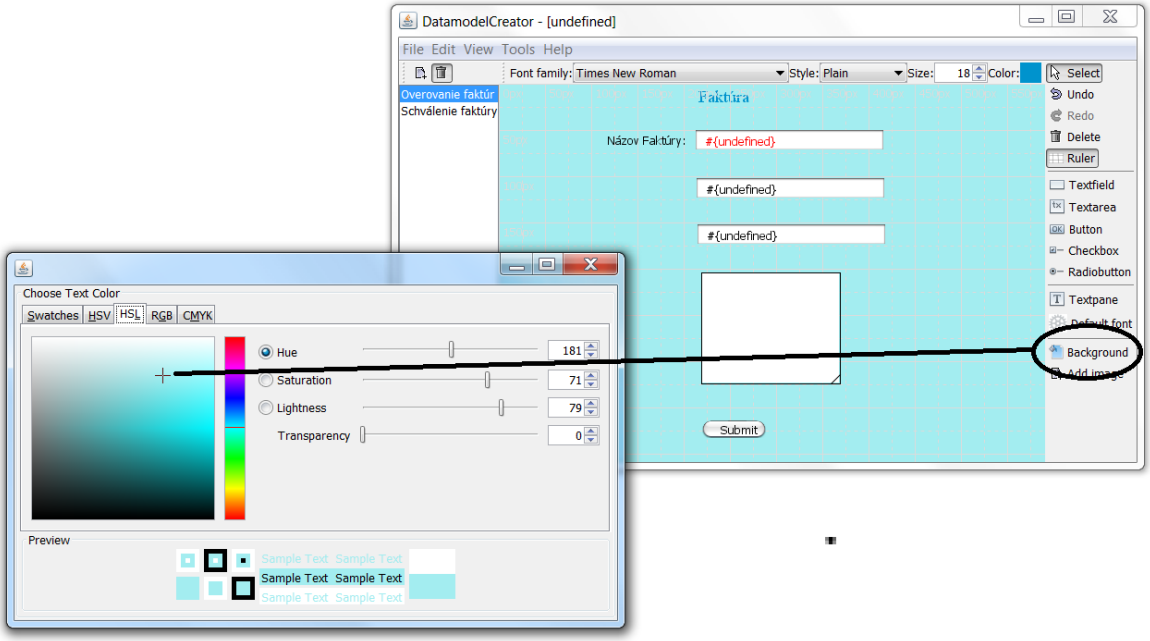
9. Vytvoriť asociácie dátových modelov s prechodmi petriho siete. Zvoliť položku „Association“, drag&drop vytvoriť asociáciu



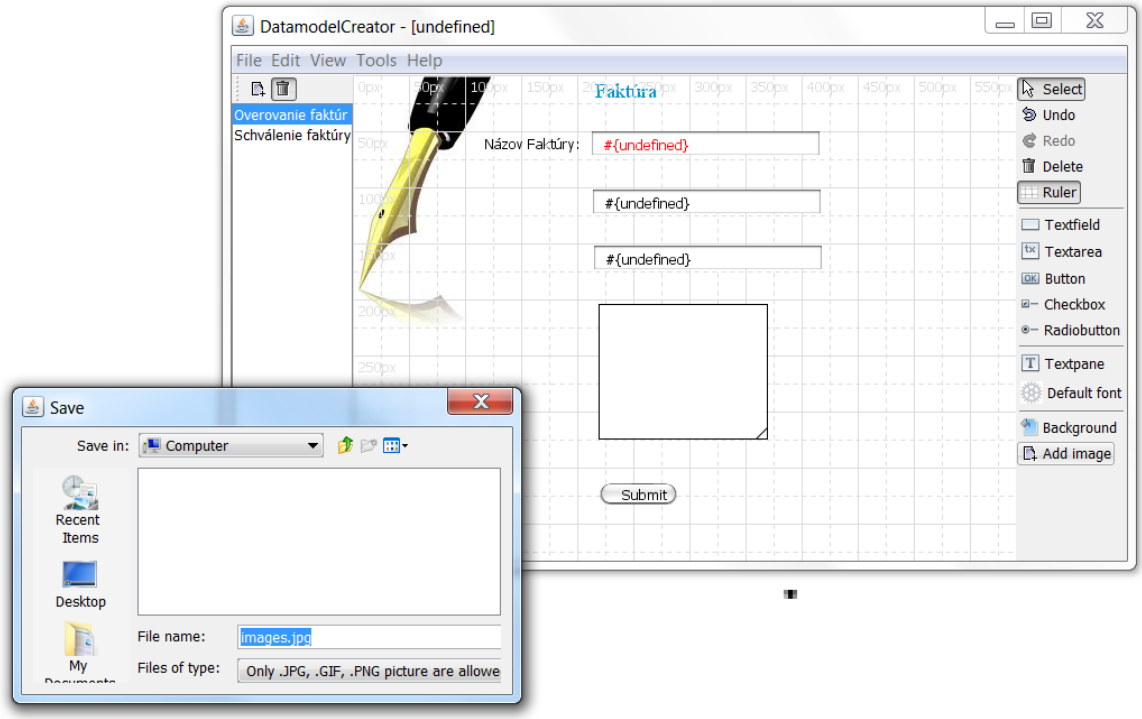
10. Nastaviť font pre doplňujúci text. Štýl písma, veľkosť, farba, ... (nepovinné)



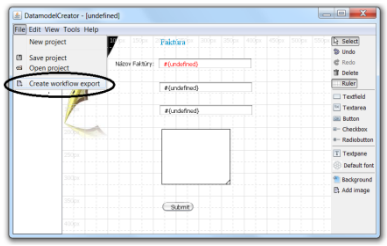
11. Nastaviť farbu pozadia stránky (nepovinné)



12. Pridať obrázok na stránku (nepovinné)



13. Exportovať web stránky pre workflow.



Príloha C Architektúra vstupov a výstupov generickej aplikácie