Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

Vývoj aplikace pro konfiguraci vozidel

Bakalářská práce

Tomáš Kotnour

Školitel RNDr. Jaroslav Icha

České Budějovice 2014

Bibliografické údaje

Kotnour, T., 2013: Vývoj aplikace pro konfiguraci vozidel
[Development of application for vehicles configuration Bc. Thesis, in Czech.] – 34 p.,
Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Abstrakt

Předmětem této bakalářské práce je navrhnutí a vyvinutí systému pro konfigurování a zobrazování vozidel, který používá formáty 3D souborů společnosti Siemens PLM, programovací jazyk JavaFX 2.2 a Java SE 8. Pro navržení tohoto systému byly použity nástroje UML. Zobrazování modelů nákladních automobilů vytvořených tímto programem bude realizováno prostřednictvím programu Siemens JT2Go. Tento systém je složen ze tří oddělených aplikací prodejce, správce a databáze.

Abstract

The main focus of this bachelor thesis is designing and developing system for configurating and displaying trucks, which uses Siemens PLM 3D file formats, Java SE 7 and JavaFX 2.2 programming languages. For designing this system were used tools of Unified Modeling Language. Displaying models of the trucks created by this software is provided by Siemens JT2Go program. This system is composed of three parts dealers application, administrators application and database.

Klíčová slova

Java, JavaFX, PLM XML, XML, JT2Go, UML, složení modelu

Keywords

Java, JavaFX, PLM XML, XML, JT2Go, UML, model composition

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 24. dubna 2014

…….....

Tomáš Kotnour

Poděkování

Rád bych zde poděkoval panu RNDr. Jaroslavu Ichovi za vedení mé bakalářské práce a poskytnuté rady pro řešení této problematiky. Dále pak panu Ing. Pavlu Karbanovi a Petru Kopeckému za vstřícnost při jednání a realizaci mých návrhů.

Obsah

1. Úvod a cíle práce

1.1 Úvod do problematiky

V současné ekonomické situaci je velice složité prodat nákladní automobil, který není určen k provozu v rámci silniční kamionové dopravy. Výrobci po celém světě se snaží zaujmout a získat zákazníky pomocí vlastností jejich výrobku, jeho ceny, servisní sítě a vlastní propagace vozidla.

Ta zahrnuje vše od tiskových zpráv o novinkách, které výrobce uvedl na trh a jejich prezentaci na výstavách, přes udržování kontaktu se zákazníky, ukázkové jízdy vozidel doplněné o propagační videa, tiskoviny a dárkové předměty, až po účast automobilky v dálkových soutěžích. Toto jsou prostředky, které používají všechny přední automobilky a z pohledu zákazníka se jeví jako povinnost. Jak tedy probudit jeho zájem?

Společnost TATRA TRUCKS, a. s. produkující stejnojmenné nákladní automobily se rozhodla jít cestou obchodní strategie TATRA IS THE SOLUTION, kterou lze krátce charakterizovat jako vytvoření nákladního automobilu přesně odpovídajícího specifickým přáním a potřebám zákazníka. Nejlepším způsobem předvedení takového vozidla zájemci je jeho zapůjčení na krátkou dobu do provozu, ale tento ideální stav má několik úskalí. Společnost by musela disponovat velkým množstvím prezentačních vozů, které znamenají značnou finanční zátěž podniku, a přesto nemusí žádný z těchto automobilů přesně odpovídat potřebám zákazníka.

Řešením by bylo poskytnutí co nejvíce podobného vozidla, které by zákazníkovi posloužilo k prověření užitných vlastností nákladního automobilu, a jeho následnou modifikací na základě přání a postřehů by byla vytvořena finální varianta, která by pak byla představena. Toto představení dnes běžně probíhá většinou za pomoci několika obrázků různých vozidel, které mají jednotlivé prvky konfigurace toho finálního. Toto řešení ovšem podle mého názoru není optimální.

To si uvědomuje i společnost TATRA TRUCKS, a. s., se kterou byla tato práce konzultována prostřednictvím pana inženýra Pavla Karbana, šéfa oddělení IT a CAD specialistou panem Petrem Kopeckým, jejíž požadavky vytvořily základ zadání této práce,

která má za úkol navrhnout a vytvořit systém, který by prodejcům poskytl možnost vizualizovat vozidlo v podobě 3D modelu a který by umožnil tuto konfiguraci dílčím způsobem upravit.

1.2 Cíle práce

Cílem této práce je na základě poznatků o objednávání, konfiguraci, konstrukci a životního cyklu modelové řady nákladních automobilů TATRA navrhnout a vytvořit systém, který by dokázal zobrazovat konfiguraci nákladního automobilu v podobě vozidla složeného z modelů jednotlivých dílů a konstrukčních celků. Tento systém bude odpovídat požadavkům společnosti TATRA TRUCKS, a. s. a bude využívat dohodnuté datové struktury.

2. Formulace požadavků, analýza a návrh řešení

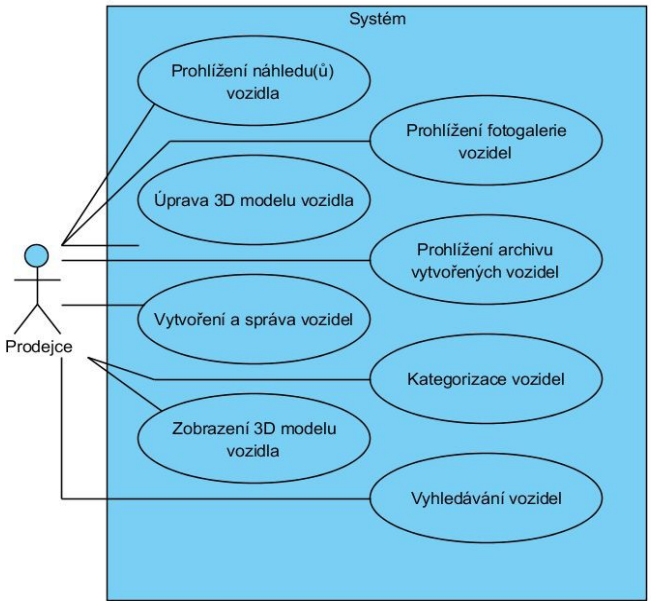
Pro vytvoření celého systému bylo potřeba formulovat požadavky společnosti TATRA TRUCKS, a. s., zhodnotit dostupné prostředky a navrhnout řešení celé aplikace.

2.1 Formulace požadavků

Jak již bylo řečeno, cílem této bakalářské práce je vytvořit systém, který poskytne prodejci možnost prezentovat zákazníkovi provedení požadovaného vozidla zcela unikátním způsobem. Aby byl tento systém v praxi použitelný, bylo třeba určit jeho jednotlivé komponenty a začlenit do něj další dohodnuté funkce. Ty byly specifikovány na základě jednání s panem Ing. Karbanem a panem Kopeckým za pomoci jazyka UML.

2.1.1 Požadavky prodejce

Požadavky na aplikaci prodejce byly stanoveny následovně.



2.1 Požadavky prodejce

Aby systém byl skutečně konfigurátorem, je třeba do něj začlenit možnost měnit jednotlivé volby vozidla na základě přání zákazníka. Vzhledem k tomu, že model je vytvořen až po specifikaci vozidla, musí být možné modifikace zahrnuty do modelu předem. Tyto změny budou menšího charakteru a budou mít za cíl doladit vozidlo do konečné podoby, která bude vyhovovat zákazníkovi. Takto vytvořený model vozidla bude uložen mimo seznam vozidel a po ukončení aplikace nebude dostupný. Aby mohl prodejce tuto konečnou podobu objednat, musí mu systém nabídnout vozidlo ve webovém IS, které odpovídá vozidlu sloužícímu jako základ pro provedení zákazníkových úprav.

Prodejce potřebuje při jednání se zákazníkem prezentovat různá konstrukční řešení nákladních automobilů, z toho důvodu vznikl požadavek na rozhraní schopné zobrazit paralelně několik náhledů vozidel tak, aby je bylo mezi sebou možno porovnávat a přesněji popisovat. Tento náhled sestává ze specifikace vozidla tvořené jeho označením, modelovou řadou, znakem pohonu, provedením podvozku, verzí vozidla, popisem a začlenění do kategorií, dále jeho fotografiemi, volbou zobrazení modelu vozidla a rozhraní pro upravení jeho konfigurace s možností zobrazení modelů jednotlivých konstrukčních skupin a voleb.

Prohlížení fotografií v tomto rozhraní má umožňovat jejich zobrazení jak ve variantě náhledu celého obrázku v režimu maximalizovaného okna, tak zobrazení fotografie v přirozeném rozlišení s možností rychlého pohybu v ní.

Samozřejmostí je možnost vytvoření nákladního vozidla a jeho úprava. Minimální záznam obsahuje jeho označení, modelovou řadu, znak pohonu, provedení podvozku, verzi vozidla a odkaz na vozidlo ve webovém informačním systému. Tento záznam lze dále rozšířit o soubor s konfigurací vozidla, která umožní vytvoření modelu, jeho fotografie a zařazení do libovolného počtu kategorií. Pokud došlo k uložení chybného záznamu vozidla, bude při změně modelové řady, znaku pohonu, provedení podvozku, verze vozidla či konfiguračního souboru provedeno opětovné vygenerování nového modelu.

Kategorizování nákladních automobilů je plně pod kontrolou každého prodejce. Je zde zahrnuta možnost tyto kategorie vytvářet, mazat, přejmenovávat a skrývat. Při mazání bude kategorie odstraněna ze všech vozidel stejně tak jako podkategorie, které obsahuje. Pokud je kategorie označena jako skrytá, nezobrazí se při prohlížení vozidel ani v ní nelze vyhledávat.

Seznam vozidel bude zobrazen za pomoci ikon jednotlivých vozidel obsahujících jednu zmenšenou fotografii vozidla a jeho označení. Toto jednoduché zobrazení bude možné změnit do zobrazení vozidla obsahující jeho specifikaci, odkaz do webového IS, možnost zobrazení modelu vozidla a přechod do paralelního zobrazení. Množina vozidel zobrazená v náhledu odpovídá aktuálně vybrané položce ze seznamu kategorií, nebo struktury určující provedení vozidla zobrazena v sestupném pořadí jako modelová řada, znak pohonu, provedení podvozku a verze vozidla, nebo výsledku vyhledání či filtrace vozidel.

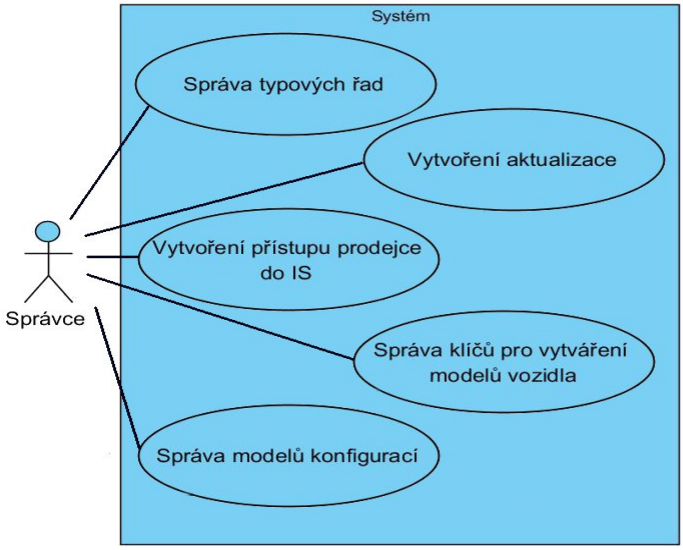
Vyhledávání v seznamu vozidel má být možné prostřednictvím volby vyhledání vozidla a filtrace vozidla. Tyto dvě volby si budou velice podobné, obě budou využívat výběr základní množiny vozidel z výše zmíněné struktury určující provedení vozidel. Z této množiny jsou pak dále vybrána vozidla, která splňují zařazení do kategorií. Konečná množina vozidel určených k zobrazení pak obsahuje pouze ta vozidla, která odpovídají svým označení tomu zadanému a popis vozidla obsahuje zadané hodnoty. Rozdíl mezi vyhledáním a filtrací vozidla je v povinnosti zahrnutí vozidla ve všech vybraných kategoriích při vyhledávání vozidla.

Vzhledem k tomu, že v průběhu života této aplikace dojde k zavádění nových vozidel, uprav dílů, zaniknutí vozidel a změn v konfiguračních možnostech, musí být aplikace schopna na tyto změny reagovat a zpracovat je. Pokud dojde ke změně, která bude ovlivňovat modely dílů a konstrukčních celků, ze kterých je vozidlo složeno, musí systém nabídnout prodejci možnost tyto změny zahrnout do jeho modelu.

Během vývoje aplikace se ukázalo jako velice vhodné vytvořit instalátor, který by provedl správné překopírování provozních dat, nainstalovat by podpůrný software, pokud by byl potřeba, a zároveň by zavedl do systému souborový formát tatracf obsahující konfiguraci vozidla z webového informačního systému, který by byl asociován s aplikací prodejce. Na základě spuštění tohoto souboru by se automaticky provedlo uložení vozidla a vytvoření a zobrazení jeho modelu.

2.1.2 Požadavky správce

Aplikace na správu má následující požadavky



2.1.2 Požadavky správce

Společnost TATRA TRUCKS, a. s. si samozřejmě nepřeje, aby mohl k tomu konfiguračnímu systému přistupovat každý, proto je nutné zabezpečit přístup k tomuto systému pouze prodejcům. Tyto prostředky bude vytvářet správce systému a budou poskytnuty prodejcům prostřednictvím jiných distribučních kanálů. Správce dále požaduje možnost monitorování činnosti systému při poskytování změn prodejcům.

Správa modelových řad zahrnuje závádění nových modelových řad, znaků pohonu, provedení podvozků a verzí vozidel. Tyto změny budou prováděny ručně za pomoci editačního rozhraní nebo automaticky při nahrávání modelů voleb konstrukčních celků vozidla.

Vytvoření aktualizace bude inicializováno na základě pokynu správce. Bude zahrnovat vytvoření unikátně pojmenované složky obsahující všechny modely, které byly do aplikace nahrány od poslední aktualizace, soubor klíčů pro vytváření modelů vozidel a seznam modelových řad. Obsah této složky pak bude distribuován prodejcům.

Správa klíčů pro vytváření modelu vozidla v sobě zahrnuje vytváření nových a správu existujících klíčů s omezením možnosti mazaní těch, které už byly dány k dispozici

k aktualizaci. Ruční přidávání a editace klíčů bude probíhat po jednom klíči nebo úpravou všech klíčů, které používají stejné modely.

Aby se zrychlilo vytváření a úprava klíčů určených pro vytvoření modelů, bude aplikace vybavena rozhraním pro hromadné nahrání modelů. Ty budou uloženy ve struktuře odpovídající té, která je zveřejněna v návrhu řešení.

2.2 Analýza možných řešení

2.2.1 Grafické soubory

Společnost TATRA TRUCKS, a. s., používá pro navrhování a konstrukci nákladních automobilů software společnosti Siemens. Proto musí použitý datový formát vycházet z rodiny formátů podporovaných touto společností.*

Jednotlivé konstrukční díly jsou vytvářeny v programu Solid Edge, který je ukládá do souborového formátu prt. Ten kromě vlastního zobrazení modelu obsahuje i další údaje, jako je geometrie, metadata a informace o produktu a výrobě (PMI)**, které systém nepotřebuje.

Další volbou by mohl být soubor typu JT (Jupiter Tessellation). *JT je 3D datový formát vyvinutý společností Siemens PLM Software (dříve UGS Corp.) a používá se k vizualizaci produktu, spolupráci a výměně CAD dat. JT soubory jsou ze své podstaty "lehké" (~ 1-10% velikosti souboru CAD),*[1] mohou obsahovat PMI data, ale ne nezbytně, jsou tedy ideální jak pro přenos po internetu, tak pro uchování velkého množství modelů.

2.2.2 Model nákladního automobilu

Nákladní automobil je tvořen souborem jednotlivých konstrukčních skupin a jejich voleb. Obvykle této volbě odpovídá jeden soubor, nemusí to být ale žádný, nebo jich může být i více. Aby datová náročnost nebyla tak velká, bylo rozhodnuto zobrazovat pouze ty volby, jejichž modely jsou viditelné z vnějšku, zákazník tedy neuvidí díly jako jsou například

---

* Dostupný na http://www.plm.automation.siemens.com/en_us/products/open/vis/#lightview%26uri=tcm:1023-11334%26title=PLM%20Vis%20-%20Software%20Toolkit%20for%20Collaborative%20View%20and%20Markup%20-%20PLM%20Components%20Fact%20Sheet%20-%206875%26docType=.pdf

** Více na http://en.wikipedia.org/wiki/Product_and_manufacturing_information

[1] JT (visualization format). In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-04-25]. Dostupné z: http://en.wikipedia.org/wiki/JT_(visualization_format)

ozubená soukolí převodovky a sestupu, rozvody a diferenciály v nosné rouře, elektrorozvody a hadice nesouvisející s nástavbou a interiér kabiny.

První možností typu souboru pro uložení modelu nákladního automobilu je JT. *Společnost Siemens dovoluje vytvořit a distribuovat software, jehož vstupem nebo výstupem je soubor, který přesně odpovídá specifikaci uvedené v popisné příručce.*[2] Pokud by byla tato možnost realizována, bylo by nutné začlenit do aplikace metody pro čtení a zápis těchto souborů, ať už vyvinuté v rámci této práce, nebo získané od třetích stran. Tím by byly získány širší možnosti konfigurace modelu (např. barevné schéma vozu), na druhou stranu by tato implementace přinesla další zvýšení nároků na kapacitu disku pro tento systému.

Program Vis Mockup, který používá pan Petr Kopecký při zpracovávání modelů, umožňuje uložení načtených modelů do souboru typu VF. Je to tzv. *Teamcenter session file - soubor, který obsahuje jak parametry relací stejně tak jako data modelů*.[3] To umožňuje sestavovat komplikované modely bez toho, aby docházelo k opětovnému ukládání modelů do souboru scény a tím snižuje velikost takto složeného modelu. Zároveň pouhým přepsáním odkazovaného modelu dojde k jeho aktualizaci ve všech scénách, kde je zahrnut. Tento souborový formát je však uzavřený a není k němu dostupná žádná dokumentace.

Přesto je filozofie použití souboru, který do scény importuje modely dílů bez nutnosti jejich duplikace v rámci aplikace správná. Společnost Siemens PLM Software nabízí otevřený značkovací jazyk PLM XML, který je vytvořen pro práci s grafickými objekty prostřednictvím aplikací, které nemusí pocházet z jejich dílny. Pomocí následné analýzy poskytnutých schémat, která jsou k dispozici na webových stránkách[*], lze navrhnout minimální datovou strukturu, která tuto zamýšlenou funkcionalitu umožní.

---

[2] CARTER, Michael, Jianbing SASHANK, Ganti HUANG a Bo XU. SIEMENS PLM SOFTWARE. JT File Format Reference Version 9.5 Rev-A. 2010. Dostupné z: http://www.plm.automation.siemens.com/en_us/products/open/jtopen/technology/index.shtml#lightview%26uri=tcm:1023-134827%26title=JT%20v95%20File%20Format%20Reference%20Rev-D%26docType=.pdf

[3] SIEMENS PLM SOFTWARE. PLM Vis – Software toolkit for collaborative view and markup: Supported formats. 2011. Dostupné z: http://www.plm.automation.siemens.com/en_us/products/open/vis/#lightview%26uri=tcm:1023-11334%26title=PLM%20Vis%20-%20Software%20Toolkit%20for%20Collaborative%20View%20and%20Markup%20-%20PLM%20Components%20Fact%20Sheet%20-%206875%26docType=.pdf

[*] Schémata jsou dostupná na http://www.plm.automation.siemens.com/en_us/products/open/plmxml/schemas.shtml

2.2.3 Zobrazení modelu vozidla

Po té, co je vytvořen model vozidla, je třeba ho zobrazit. Vzhledem k jeho vysoké kvalitě a tím pádem velkému množstvím polygonů by aplikace, která by ho dokázala zobrazit i na běžných pracovních stanicích, musela být velice dobře napsána při čemž použití frameworků pro vývoj her, jako například XNA Game Studio, by nemuselo postačovat. Vývoj takovéto aplikace by zahrnoval i vytvoření nástrojů pro čtení jt souborů.

Další možností by bylo využití existující aplikace. Jednou z nich je JNetCAD[*]. Jejím autorem je Johannes Raida. Kromě zmíněného typu dokáže otevírat a zobrazovat další typy souborů jako je 3DS, DXF a STL. Její využití pro nekomerční použití je zdarma, pro komerční by bylo nutné zakoupit licenci. Při testování této aplikace používající pro zobrazování technologii JavaOpenGL byly zjištěny problémy při načítání modelů s větším množstvím polygonů. Při otevření modelu došlo k postupnému narůstání množství paměti využívané tímto programem bez jakékoliv zpětné vazby.

Další cestou pro zobrazení souborů formátu JT je použití aplikace napsané v jazyce Java s názvem JTViewer[**], který je šířen dle licence GNU Lesser General Public License. Ten používá pro zobrazení JReality toolkit. Při testování se však vyskytly podobné problémy jako v předchozím případě.

Společnost Siemens PLM Software nabízí v rámci podpory rozšíření svých technologií program JT2Go. *JT2Go je bezplatné zobrazovací řešení, které obsahuje základní 3D prostředí, základní 3D vnitřnímu členění a PMI.*[4] Jeho technologie zobrazení je dostatečně kvalitní na to, aby zvládla načíst a zobrazit množinu modelů dílů vozidla i na běžných počítačích prodejců.

2.2.4 Zabezpečení systému

Společnost TATRA TRUCKS, a. s. poskytuje interní data pouze smluvním partnerům. Do této kategorie patří i modely komponent vozidla, proto není vhodné, aby k nim měl přístup každý. Jak již bylo zmíněno, tyto modely ve formátu JT obsahují jenom nejnutnější data pro jeho zobrazení. Samotná komponenta je vymodelována tak, aby vzhledem odpovídala

[*] Dostupný na http://www.johannes-raida.de/index.htm?jnetcad

[**] Dostupný na http://code.google.com/p/jt-java/

[4] JT2Go. SIEMENS PLM SOFTWARE. JT2Go: Siemens PLM Software [online]. 2012 [cit. 2012-04-25]. Dostupné z: http://www.plm.automation.siemens.com/en_us/products/teamcenter/lifecycle-visualization/jt2go/index.shtml

skutečné předloze, neobsahuje však její vnitřní strukturu. Tím je zabráněno jejímu možnému reinženýringu a nabídnutí na trhu náhradních dílů, čímž se snížila atraktivita těchto dat.

Dalším nebezpečím je využití chyby v zabezpečení serverové aplikace pro ovládnutí počítače, na kterém tato aplikace běží, a jeho využití pro získání přístupu do firemní sítě společnosti TATRA TRUCKS, a. s. To by mohlo vést k získání citlivých informací, nebo alespoň k poškození serverové aplikace poskytující data prodejcům. Podobným způsobem lze útočit i na aplikace prodejců.

Systém by měl umožňovat vytvoření pouze důvěryhodných připojení, která musí být odolná proti odposlechu dat. Dalším požadavkem je zaznamenávání, komu byla data poskytována, a tím odhalit narušení bezpečnosti systému. Zároveň by mělo býti vytváření prostředků pro umožnění přístupu dostatečně uživatelsky přívětivé a jejich použití zbytečně nezatěžovat počítač.

Zabezpečení přenosu dat proti odposlechnutí je možné realizovat dvěma způsoby. Prvním z nich je šifrování dat, která jsou přenášena, druhým je šifrování kanálu pro přenos dat. To je možné realizovat za použití SSL spojení mezi klientem a servem za použití certifikátu generovaného například nástrojem keytool. Toto připojení bude chránit všechna data během přenosu, ale nezabezpečí důvěryhodnost připojení. Klientská aplikace sice pozná, že bylo ustanoveno spojení se serverovou aplikací konfigurátoru, server ale nikoliv. Aby byla zajištěna komunikace serveru s autorizovaným klientem, musela by být aplikace doplněna autorizačním rozhraní.

Další možností je použití dvou veřejných a soukromých klíčů. Jeden veřejný a soukromý klíč by byl vygenerován pro serverovou aplikaci a další pro konkrétního prodejce. Tomu by pak byl distribuován veřejný klíč serverové aplikace a jeho soukromý klíč. Serverová aplikace pak bude mít její soukromý klíč a veřejný klíč každého prodejce. Šifrování přenášených modelů pomocí asymetrické šifry by bylo samozřejmě příliš náročné, proto by před každým zasláním souboru byl vytvořen symetrický klíč, kterým by byl obsah zprávy zašifrován. Komunikace by pak probíhala následujícím způsobem, odesílatel by vygeneroval symetrický klíč, kterým by zašifroval zprávu. Tento klíč by pak zašifroval veřejným klíčem adresáta. Z obsahu zprávy by pak vytvořil hash, který by byl zašifrován soukromým klíčem odesílatele a odeslán. Adresát pak svým soukromým klíčem dešifruje symetrickým klíč, kterým pak použije pro získání obsahu souboru. Z této zprávy pak vytvoří hash, který pak porovná s obdrženým hashem, který byl získán dešifrováním pomocí

veřejného klíče odesílatele. Takto navržené zabezpečení přenosu zabezpečí šifrování přenášených dat, pozná, zda data byla změněna, a zda pochází od autorizovaného zdroje.

Na stejném principu by pak pracovalo zabezpečení využívající místo vygenerovaných privátních a soukromých klíčů rovnou celé certifikáty od komerčních certifikačních autorit.

Kvůli zabránění vzniku možných škod by serverová aplikace měla běžet v uzavřeném prostředí sandboxu s právy pouze na čtení souborů modelů dílů, souboru klíčů pro vytvoření modelu a typových řad. Jediný soubor, do kterého by měla mít možnost zapisovat, je pro logování úspěšných připojení aplikací prodejců a poskytovaných souborů. Všechny vstupní parametry metod by měly být kontrolovány, aby bylo zabráněno poslání kódu v Assembleru, který by mohl být vykonán.

Aby bylo zabráněno zneužití zdrojového kódu aplikace prodejce, která po uvolnění mezi ně může být získána třetí stranou, budou její zdrojové kódy zatemněny. *K tomu slouží zatemňovač (obfuscator), jenž je sestaven za jediným účelem: zmaření zpětného překladu.*[5]

2.3 Návrh řešení

Systém bude tvořen třemi aplikacemi, které budou implementovány za pomoci jazyka Java a JavaFX 2.2. Aplikace prodejce bude se serverovou automaticky komunikovat a sdílet data v podobě seznamu modelových řad, klíčů a samotných modelů dílů vozidla.

2.3.1 Aplikace prodejce

Vytvoření vozidla – bude možné dvěma způsoby, vytvořením nového vozidla, nebo modifikací existujícího. Jak bylo zmíněno v dříve, minimální záznam vozidla obsahuje označení vozidla, modelovou řadu, znak pohonu, provedení podvozku, verzi vozidla a odkaz na vozidlo ve webovém informačním systému. Ten je možné dále doplnit popisem vozidla, zařazením do kategorií a fotografiemi vozidla.

Z důvodu zabránění překlepů a chyb při zadávání modelové řady, znaku pohonu, provedení podvozku a verze vozidla bude jejich výběr realizován pomocí zvolení volby ze skupin voleb. Na základě změny volby bude provedena změna obsahu všech nižších stupňů tak, aby odpovídaly dostupným možnostem. Tyto možnosti budou vybírány ze seznamu

[5] TAYLOR, A. Hacking bez tajemství: java a J2EE. Vyd. 1. Brno: Computer Press, 2003, 409 s. ISBN 80-722-6868-6, s. 156.

modelových řad, který obsahuje všechna provedení sériově vyráběných vozidel. Základním prvkem pro odlišení vozidel je jeho označení dle systému označování vozidel Tatra.[*] Vzhledem k tomu, že značení vozidel modelové řady T 158 PHOENIX je realizováno částečně odlišným způsobem, byla po domluvě s panem Petrem Kopeckým zavržena možnost automatického generování části označení vozidla.

Kategorie, do kterých lze vozidlo zařadit, budou odpovídat seznamu kategorií, které prodejce před tím v aplikaci vytvořil s možností vytvořit další prostřednictvím tohoto rozhraní. V tomto seznamu budou odlišeny kategorie, ve kterých je toto vozidlo už zařazeno a nebudou moci být opětovně vybrány. Výběr obrázků vozidla bude prováděn prostřednictvím rozhraní, které bude filtrovat soubory, které nebudou odpovídat domluvenému formátu. Formáty obrázků, které bude aplikace podporovat, jsou následující: tif, gif, jpeg, png. Každý obrázek, který byl vybrán, bude v rozhraní přidávání/editace vozidla reprezentován miniaturou. Každý tento obrázek může být použit jako hlavní obrázek zobrazovaný v přehledu vozidla. Pokud vozidlo neobsahuje žádné obrázky, poslouží jako hlavní obrázek logo značky TATRA. Po uložení vozidla dojde k vytvoření složky, jejíž název bude odpovídat označení vozidla, do ní pak budou tyto obrázky duplikovány.

Vytvoření modelu vozidla – jak již bylo řečeno, model vozidla se skládá z modelů jednotlivých konstrukčních celků a dílů. Ty budou vybírány na základě určení specifikace vozidla a souboru s konfigurací vozidla. Cesta k těmto jt souborům bude zahrnuta do struktury souboru plmxml. Model vozidla bude vytvořen po uložení vozidla s konfiguračním souborem nebo prostřednictvím rozhraní konfigurátoru. Pokud dojde k modifikaci specifikace vozidla nebo konfiguračního souboru bude model znovu vygenerován. Rozhraní konfigurátoru nabídne seznam konstrukčních skupin, které obsahuje konfigurace zdrojového vozidla. Jednotlivé konstrukční skupiny je možné přidávat a odebírat. Při přidání konfigurační skupiny bude její výběr realizován z množiny skupin, které vozidlo ještě neobsahuje. Z konstrukční skupiny může být vybrána jenom jedna možnost. Do souboru vozidla lze přidat i ostatní možnosti, při čemž jejich modely budou zobrazeny pokynem uživatele.

Vyhledání a filtrování vozidel – bude realizováno jedním grafickým rozhraním, které bude sloužit pro výběr parametrů. Bude možné vybrat celou modelovu řadu, provedení pohonu v jejím rámci, dále provedení podvozku, která jsou dostupná v konkrétním provedení

[*] Dostupný na http://partners.tatra.cz/exter_pr/vp/img/oznacovanivozidel.pdf

pohonu a seznam verzí vozidel, ve kterých se provedení podvozku vyrábí. Pokud nebude vybrána žádná z těchto voleb, budou implicitně vybrána všechna vozidla. Dále následuje výběr kategorie, do které musí být vozidlo zařazeno. Výběr kategorie, která obsahuje podkategorie, neznamená automatický výběr těchto podkategorií. Jak bylo zmíněno, při volbě filtrace musí být vozidlo alespoň v jedné kategorii, při vyhledání vozidla ve všech. Následovat bude možnost vyhledat vozidlo podle VDS. *Zkratka VDS (Vehicle Descriptior Section) znamená popisný kód vozidla.*[6] Formát tohoto kódu je společný pro všechny typové řady a prodejce na jeho základě dokáže určit další vlastnosti vozidla. Poslední volba umožňuje vypsat řetězce, která musí obsahovat popis vozidla. Pro větší počet vyhledávaných řetězců budou jednotlivá spojení oddělena znakem '+'.

Rozhraní pro paralelní zobrazení vozidla – jak bylo určeno v zadání, aplikace bude umožňovat paralelní zobrazení vozidel. To bude tvořeno malou aplikací spouštěnou z rozhraní aplikace prodejce obsahující tři karty. První z nich bude sloužit na zobrazení informací o vozidle, jeho zařazení do kategorií a popis. Druhá karta bude sloužit pro zobrazení galerie fotografií vozidla. V horní části této karty bude zobrazen náhled aktuálního vozidla, v dolní části pak všechny obrázky, které jsou k vozidlu připojeny. Po kliknutí na obrázek dojde k zobrazení obrázku přes celou obrazovku, kromě horní lišty aplikace a lišty operačního systému. Dalším kliknutí dojde k zobrazení obrázku vozidla v nativním rozlišeni. Pohyb po zobrazované ploše bude možný pomoci posuvníků nebo klikáním levého tlačítka myši na pozici, která by měla být posunuta ke středu obrazovky. Kliknutí pravým tlačítkem myši pak vrátí původní zobrazení seznamu obrázků s náhledem aktuálně vybraného. Poslední karta bude sloužit pro konfiguraci modelu vozidel dle části vytvoření modelu vozidla**.**

Zobrazení seznamu vozidel – seznam zobrazených vozidel bude reprezentován rozhraním složeným z dlaždic, z nichž každá zastupuje jedno vozidlo. Na této dlaždici bude zobrazen hlavní obrázek vozidla a jeho zkrácené označení. *Toto označení se zavedlo pro snazší a jednodušší orientaci v určení provedení vozidla, které je tvořené první a druhou číselnou skupinou označení úplného, například T 815-2DOS13.*[7] Rozměry a obsah této dlaždice půjde kliknutím změnit tak, že bude obsahovat hlavní obrázek vozidla ve větší

[6] ROSENKRANZ, Karel. Nákladní automobily Tatra. Vyd. 1. Kopřivnice: TATRA TRUCKS, 2007, 661 s. ISBN 978-80-239-9877-1.

[7] ROSENKRANZ, Karel. Nákladní automobily Tatra. Vyd. 1. Kopřivnice: TATRA TRUCKS, 2007, 661 s. ISBN 978-80-239-9877-1.

velikosti, plnohodnotné označení vozidla, jeho typovou řadu, typ pohonu, provedení podvozku, verzi vozidla, dále jeho popis a seznam kategorií, do kterých je vozidlo zařazeno. Tlačítko pro zobrazení modelu vozidla a pro přechod do rozhraní paralelního zobrazení. Stiskem levého tlačítka myši lze vozidlo označit a tím ho zpřístupnit pro úpravu nebo smazání prostřednictvím volby z menu.

2.3.2 Aplikace správce

Hromadné nahrání souborů – tato volba bude umožňovat nahrání velkého množství modelů a vytvoření klíčů pro modely vozidla. Pro provedení volby bude třeba vybrat složku s dohodnutou strukturou. Tato složka se bude jmenovat models. Další úroveň členění bude zahrnovat jednotlivé modelové řady. Každá modelová řada pak bude členěna na jednotlivé složky verzí pohonu. Tyto složky budou mít název odpovídající vzoru počet kol x počet hnaných kol (např.: 6x6), dále budou na stejné úrovni složky pojmenované spolecneVsem a spolecne_verze pohonu_další verze pohonu (například spolecne_4x4_6x6). Tyto společné složky budou obsahovat přímo modely voleb konstrukčních skupin. Složky typů pohonů budou obsahovat pouze složky jednotlivých provedení podvozku, které mají tříznakové názvy. V těchto složkách jsou pak uloženy modely voleb konstrukčních skupin. Pokud je při procházení nalezena verze vozidla, která není v seznamu modelových řad, dojde k jejímu založení. Po té následuje podobné projití modelů, které jsou uloženy v aplikaci správce. Díky tomu dojde k naplnění modely, které odpovídají společným dílům pro danou modelovou řadu, provedení pohonu, provedení podvozku a verzi vozidla. Původní název souboru modelu byl navržen ve tvaru XXX_YYY_popis_dilu_bez_diakrity_ZZZZZZZ.jt, kde znaky XXX zastupují tři číslice konstrukční skupiny. YYY pak označuje verzi vozidla, pokud má hodnotu 000 je použit pro všechny dostupné. ZZZZZZZ pak označují konkrétní volbu. Během vývoje aplikace a jejího testování byl nahrazen novým, neboť se ukázalo, že některé různé konstrukční celky mají stejné označení, nebo docházelo k duplicitě stejných modelů. Proto byly názvy všech souborů modifikovány, aby odpovídaly následujícímu tvaru. AAAAAA_BBB_CCC_DDDDDDDDDDD_nazev_dilu_bez_diakritiky_ZZZZZZZ.jt.
AAAAAA označuje barvu dílu, ta je uvedena ve formátu RALXXXX, kde X zastupuje pouze číslo. Tato část je jednonásobná a nepovinná. BBB označuje číselné označení konstrukční skupiny. Tento parametr je jednonásobný a povinný. CCC značí verzi vozidla, ta

může být ve tvaru 000, která označuje všechny varianty vozidla. Pokud je verze vozidla tvořena jiným číslem, jedná se o model pro konkrétní variantu. Poslední možnost tvaru této části je YYX, kdy YY jsou čísla označující rozvor podvozku a písmeno X všechny verze vozidla pro tento rozvor dostupná. Parametr je vícenásobný a povinný. DDDDDDDDDDD obsahuje takzvanou řídící hodnotu, což je číslo konstrukční skupiny a volby, které se musí v souboru s konfiguraci vozidla objevit, aby byl použit tento konkrétní model. Hodnota je ve formátu RHBBBZZZZZZZ viz předchozí popis zástupných znaků. Tento parametr je vícenásobný a nepovinný.

Vytvoření aktualizace – je inicializováno vybráním patřičné položky z menu aplikace. Dojde tím k vytvoření složky, jejíž jméno obsahuje datum a čas vytvoření aplikace. Obsah této složky pak tvoří modely, jejichž název obsahuje původní cestu k modelu ve složce s modely a variantu dílů. Ta slouží k odlišení dílů, které byly nahrazeny novými. Dále pak aktuální soubor modelRanges.xml a jeden, nebo dva soubory s klíči, aktuální a minulý. Složka s modely vozidel totiž obsahuje pouze ty, které jsou aktuální. Při vzniku aktualizace dojde ke zjištění zda existují předešlé aktualizace. Pokud ano, je vybrána nejmladší z nich a na základě porovnání souboru klíčů dojde k vybrání nahrazených modelů voleb. Soubor s klíči z poslední aktualizace je pak překopírován do nové se jménem rozšířeným o verzi a nakopírování aktuálního souboru keys.xml. Obsah složky této nové aktualizace je pak připraven k nakopírování do odpovídající složky serverové aplikace.

2.3.3 Serverová aplikace

Poskytování dat – na základě příchozího spojení dojde k vytvoření nového vlákna, které bude zajišťovat distribuci dat prodejci. Serverová aplikace pak odešle potvrzení o úspěšném připojení klienta. Ten pak zašle serveru číslo verze souboru modelRanges.xml, ten ho porovná s verzí svého souboru. Pokud číslo odpovídá, je mu zasláno potvrzení o aktuálnosti souboru. Pokud ne, aplikace prodejce přejde do části pro příjem souboru a přijme soubor. Ten je uložen do složky temp a po ukončení odesílání souboru je klientovi zaslána kontrolní hodnota hash souboru. Pokud jeho hodnota neodpovídá hodnotě na straně serveru, bude soubor zaslán znovu. Pokud se hodnoty shodují, dojde k překopírování souboru do složky xmls a ověření aktuálnosti souboru keys.xml. Pokud verze zaslaná klientem odpovídá verzi zaslané serverem, bude spojení ukončeno. Pokud ne, serverová aplikace načte soubor

odpovídající verzi klienta a na základě toho určí, které modely mu budou zaslány. Zaslání modelů bude probíhat podobnou metodou jako před tím popsané zaslání souboru modelRanges.xml, ale jako první zašle server název souboru. Jako poslední pak bude zaslán soubor keys.xml.

2.3.4 Návrh zakomponování zabezpečení s veřejnými a soukromými klíči

V menu aplikace prodejce by byla vytvořena volba pro vygenerování sady klíčů a identifikačního řetězce. Pro generování klíčů by byl použit objekt třídy `KeyPairGenerator`, který by byl inicializován parametrem 2048, který značí délku klíče. Tato hodnota je v současné době doporučována jako bezpečná. Provedením metody genKeyPair() na instanci třídy `KeyPair`. Z té by šlo získat soukromý a veřejný klíč za pomoci metody getPublic a getPrivate. Následně je třeba přes objekt třídy `KeyFactory` získat `RSAPublicKeySpec` a `RSAPrivateKeySpec`. Z těchto objektů se pak dají získat hodnoty modulu a exponentu veřejného a soukromého klíče. Tyto hodnoty jsou uloženy v podobě objektu typu `BigInteger`. Pro načtení těchto klíčů v aplikaci je třeba vytvořit vlastní metodu, která ze souboru získá modul a exponent a ty jsou pak použity jako parametry konstruktoru instance třídy `RSAPublicKeySpec`. Z této instance lze pomocí objektu `KeyFactory` zavoláním metod generatePublic a generatePrivate opětovně získat veřejný a soukromý klíč. Pro šifrování dat je používán objekt třídy `Cipher`, který je vytvářen pomocí metody Cipher.getInstance, parametr této metody určuje typ klíče.

Zatemnění definované v analýze možných řešení neposkytuje dostatečnou ochranu kódu této aplikace menšího rozsahu. Lepší zabezpečení by poskytlo zakódování souborů tříd a vytvoření vlastního zavaděče tříd. Jednotlivé zásady zabezpečení Java aplikace jsou zaváděny spuštěním správce zabezpečení. *Spuštění správce zabezpečení je velmi jednoduché. Nejprve bychom si měli ověřit, že již není nějaký správce zabezpečení používán, a pak pomocí statické metody setSecurityManager() třídy System přiřadit aplikaci správce.*[8] Zásady zabezpečení mají podobu potomků třídy `java.security.Permission`. Záznam umožňující pouze čtení souborů ve složce filesForDistribution *permission java.io.FilePermission "\\filesForDistribution*","read";*

[8] PAVLÍČKOVÁ, Jarmila a Luboš PAVLÍČEK. Vývoj klient/server aplikací v Javě [online]. Vyd. 1. Praha: Oeconomica, 2004, 94 s. [cit. 2012-04-25]. ISBN 80-245-0791-9. Dostupné z: http://java.vse.cz/Java/Skripta

3. Použité nástroje

Pro vývoj aplikací bylo zvoleno prostředí programu NetBeans IDE 8.0 (Build 201403101706) na platformě Java 1.8.0_00 a JavaFX 2.2.21.

Pro vytvoření tohoto systému byla použita metodika vývoje řízeného vlastnostmi (Feature-Driven Development, FDD), *která se zaměřuje na vývoj po malých kouscích - vlastnostech, rysech, což jsou elementární funkcionality přinášející nějakou hodnotu uživateli. Vývoj probíhá v pěti fázích, první tři jsou sekvenční, poslední dvě pak iterativní. Iterace trvají zpravidla 2 týdny. Začíná se vytvořením modelu, ten se převede do seznamu vlastností, které se postupně implementují. Velice dobře se měří pokrok ve vývoji projektu, FDD umožňuje detailně plánovat a kontrolovat vývojový proces.*[9]

Pro zpracování požadavků zadavatele byl použit jazyk UML v prostředí programu Visual Pardigm for UML ve verzi 8.3.

Instalátor byl vytvořen v jazyce C# pomocí WPF ve vývojovém prostředí Visual Studio 2010 Express pro běhové prostředí .Net Framework 3.5 a vyšší.

Program policytool je možné použít pro vytvoření souboru zásad zabezpečení aplikace. *Ten je ve formátu prostého textu, je tedy možné ho vytvářet a editovat pomocí textového editoru. Nástroj policytool je poskytován jako součást Java SDK. Má grafické rozhraní a je možné ho spustit z příkazové řádky pomocí příkazu policytool.*[10]

Pro vytvoření izolovaného prostředí, které zabrání změnám ostatních programů a dat, určeného pro běh serverové aplikace je plánováno využití komerčního programu Sandboxie.

Návrhy xml souborů pro uložení dat byly tvořeny prostřednictvím xsd schémat v prostředí aplikace <oXygen/> XML Editor ve verzi 12.2, build 2011062910. Pro zatemnění aplikace by bylo možno použít java balík JODE, který je šířen pod GNU GPL licencí.

---

[9] HAJDIN, Tomáš. *Agilní metodiky vývoje software*. Brno, květen 2005. Dostupné z: http://is.muni.cz/th/39440/fi_m/. Diplomová práce. Masarykova Univerzita. Vedoucí práce Barbora Bühnová.

[10] PAVLÍČKOVÁ, Jarmila a Luboš PAVLÍČEK. Vývoj klient/server aplikací v Javě [online]. Vyd. 1. Praha: Oeconomica, 2004, 94 s. [cit. 2012-04-25]. ISBN 80-245-0791-9. Dostupné z: http://java.vse.cz/Java/Skripta

4. Implementace

Vzhledem k omezenému rozsahu práce byly popsány pouze některé implementace navržených řešení.

4.1 Grafické uživatelské rozhraní

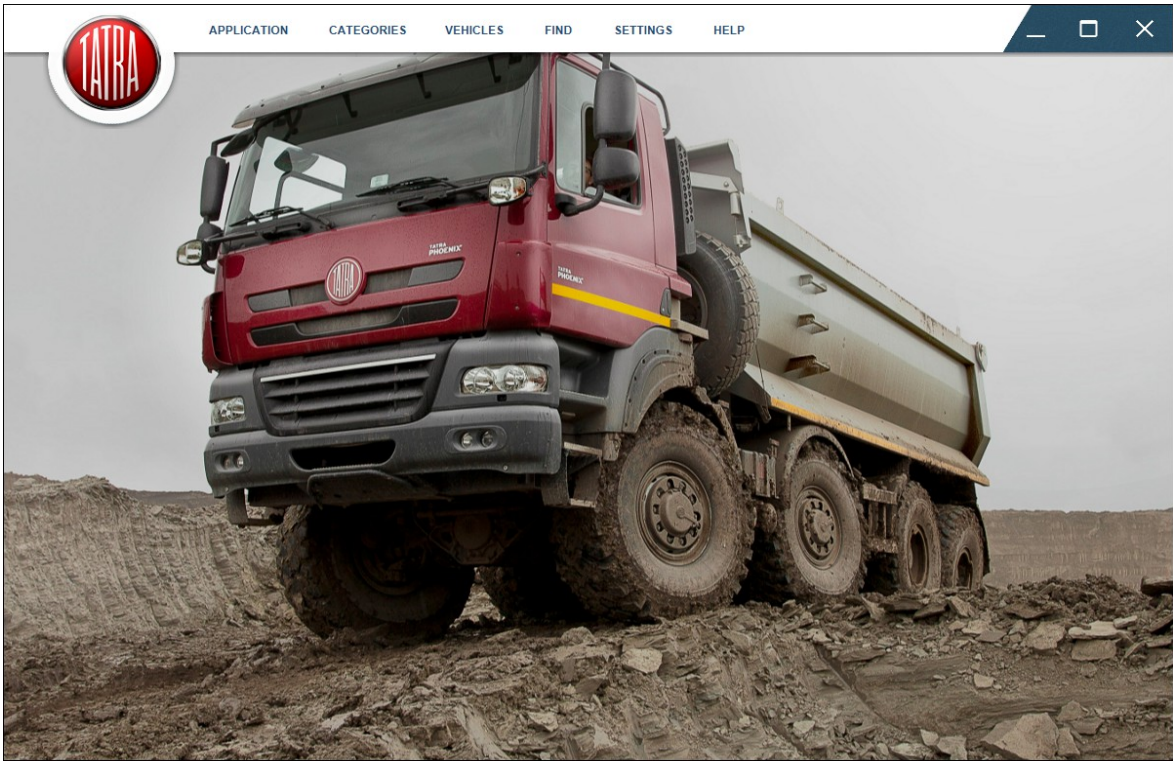
Grafické uživatelské rozhraní aplikace prodejce je vytvořeno spuštěním metody main v třídě `TCTClient`, která dědí ze třídy `Application` z prostředí JavaFX. Tato metoda pak spustí aplikaci zavoláním metody launch z třídy `Application` s parametry tctclient.TCTClient.class a args pro zajištění startu z příkazové řádky s parametrem. Metoda launch poté využívá překrytou metodu start s parametrem typu `Stage`. Vzhledem k plánovanému nasazení aplikace v zahraničí je implementováno načítání zobrazovaných textů z externího souboru. O toto načítání se stará objekt třídy `Language`, jež načte všechny dostupné lokalizace a textu pro konkrétně vybranou ze souboru languages.xml. Vzhled zobrazovaného okna je upraven odebráním ohraničení okna a tlačítek pro minimalizaci, maximalizaci a zavření okna. Toto bylo provedeno na základě volání metody initStyle s parametrem StageStyle.UNDECORED na objektu typu `Stage`.

Velikost okna a jeho obsah je tvořen instancí třídy `Scene` zavolené konstruktorem s parametrem typu `Parent`, který slouží jako zdroj obsahu scény, dvěma parametry typu float určujícími velikost okna a s parametrem typu boolean, který určuje, zda bude vytvořen buffer pro hloubku. Velikost dostupné plochy pro zobrazení okna je určena z pomocného objektu typu `Rectangle2D` za pomoci zavolání metody getVisualBounds s objektem typu `Screen` získaným pomocí metody getPrimary ze třídy `Screen`. Takto získané hodnoty odpovídají aktuálnímu rozlišení zobrazovacího zařízení s odečtením plochy, kterou zabírá lišta operačního systému.

Jak bylo uvedeno, bylo z okna odstraněno ohraničení a tlačítka. Aby tyto obvyklé prvky byly zachovány, byla vytvořena nová tlačítka pro zavření aplikace, zvětšení a zmenšení okna a jeho minimalizaci na lištu. Minimalizace a zavření okna je realizováno

zavoláním metody setIconified s parametrem true na objektu třídy `Stage`, resp. close.

Obsah grafického rozhraní vždy odpovídá instanci některé třídy jejíž jméno končí příponou View. Tyto třídy dědí ze třídy `Viewport`, která obsahuje metody resize, changeLanguage, changeTextSize a close. Hlavní třída byla implementována tak, aby věděla, co je zrovna zobrazeno a mohla tedy grafické prvky minulého rozhraní odstranit.



4.1 Grafické uživatelské rozhraní

4.2 Vytvoření modelu v souboru PLM XML

Pro zapisování modelů vozidla byla vytvořena třída `ModelFileWriter`. Tato třída zapisuje prvky tvořící scénu pro zobrazení modelu vozidla. Obsahuje zdroj světla pro osvětlení scény a jeho nastavení, dále element zobrazení ProductView, a nastavení úvodního pohledu.

Jednotlivé modely jsou reprezentovány elementy InstanceGraph, v něm je definován název, pod jakým bude zobrazen v navigátoru programu JT2Go, formát souboru, jeho umístění ve scéně a cestu k souboru. Každý element v souboru PLM XML má vlastní parametr s identifikátorem. Tyto identifikátory umožňují procházení celou strukturou souboru, jak je vidět v následujícím kódu.

```
<InstanceGraph id="id2" attributeRefs="id3" rootRefs="inst4">
  <ProductInstance id="inst4" name="kabina kratka-cervena" partRef="#id5"> </ProductInstance>
  <ProductRevisionView id="id5" name="kabina kratka-cervena" type="solid">
    <Bound id="id6" values="0 0 0 1 1 1"></Bound>
    <Representation id="id7" format="JT" location="E:\TCTClient\models\001\o\cod.jt"></Representation>
  </ProductRevisionView>
</InstanceGraph>
```

Pro tyto elementy byla vytvořena třída `InstaceGraph`, která svou metodou getInstanceEntry vrátí tuto strukturu. Vytvoření těchto identifikátorů je zabezpečeno metodou ze třídy `ModelFileWriter`. Aby byl model zobrazen ve scéně a v navigátoru, musí se „vyskytnout“ v rámci elementu ProductView mezi elementy Occurrence, kde parametr instanceRefs odpovídá parametru rootRefs elementu InstanceGraph.

Pro určení základního pohledu na scénu je možné použít tyto typy *Perspective, Orthographic a ViewMatrix*[11] Dalšími elementy je pak možné určit bod, ze kterého se díváme, bod, na který se díváme, případně směrový vektor pohledu.

4.3 Konfigurování vozidla

Základním požadovanou funkcionalitou byla možnost měnit konfiguraci vozidla. Tato funkcionalita byla implementována prostřednictvím třídy `AddEditVehicle`. Toto grafické uživatelské rozhraní je používáno pro optimalizaci vozidla podle přání zákazníka. Z toho tedy plyne, že vozidlo v tomto rozhraní zobrazené, musí mít k dispozici pouze a jenom aktuální konstrukční skupiny a volby. Výběr těchto skupin je zajištěn v konstruktoru třídy. Nejdříve jsou vybrány všechny konstrukční skupiny odpovídající této specifikaci vozidla. U těch, které jsou obsaženy v konfiguračním souboru vozidla, je vybrána odpovídající volba. U nezahrnutých konstrukčních skupin pak je konkrétní volba nevybrána.

Každá konstrukční skupina je v grafickém rozhraní reprezentována objektem třídy `ConstructionGroupItem`. Ten obsahuje popisek s číslem konstukční skupiny, její popis, pokud je dostupný, dále objekt třídy `ChoiceBox` jehož jednotlivé položky tvoří volby v rámci konstrukční skupiny a objekty třídy `Button` a `CheckBox`. Tlačítko

[11] SIEMENS PRODUCT LIFECYCLE MANAGEMENT SOFTWARE INC. PLMXML Schema. 2008. Dostupné z: http://www.plm.automation.siemens.com/en_us/products/open/plmxml/schemas.shtml

zobraz model slouží k zobrazení modelu konkrétní skupiny,. Pokud je dostupný a zaškrtnut `CheckBox` Ostatní jako skryté, dojde také k vykreslení modelů ostatních voleb do scény. O tento postup se stará metoda viewModel, která vytvoří `ArrayList` instancí tříd `InstanceGraph`, které reprezentují jednotlivé modely, a použije ho jako parametr pro metodu setInstanceGraphs instance třídy ModelFileWriter, která zajistí vytvoření souboru plmxml. Stejně pracuje i metoda viewVehicle, inicializovaná tlačítkem zobraz model, která vytvoří soubor obsahující model vozidla.

4.4 Distribuce dat

Na základě seznámení se s serverovými prostředky společnosti TATRA TRUCKS a. s., bylo rozhodnuto upustit od plánu vytvořit vlastní serverovou aplikaci a využít některou z dostpných možností od společnosti Microsoft. V tomto případě databáze v prostředí Microsoft SQL Server nebo Microsoft Sharepoint. Z těchto možností je samozřejmě výhodnější použít Sharepoint, avšak jeho propojení s klientem v javě je mnohem náročnější než s databází na SQL Serveru. Proto bylo rozhodnuto dočasně použít SQL Server s tím, že další vývoj tohoto systému využije Sharepoint.

4.5 Struktura souborů

4.5.1 Soubor keys.xml

Tento soubor slouží pro uložení klíčů pro vytváření modelů vozidla. Je vytvářen správcem a prostřednictvím serverové aplikace šířen dále k prodejcům. Jeho struktura odpovídá objektům třídy `Key` doplněné o označení verze souboru.

Element id slouží k uložení identifikátoru konkrétního klíče, ten je typu int a jeho hodnotu ovlivňuje nejenom počet klíčů před ním, ale i počet voleb, neboť číslovací algoritmus používá pro tyto dva prvky stejnou inkrementovanou proměnnou.

Element vehicleVersion typu int slouží pro uložení identifikátoru verze vozidla, pro který je tento klíč určen. ConstructionGroup uchovává označení konstrukční skupiny, pro kterou byl tento klíč vytvořen. Tato konstrukční skupina má vždy číselné označení, proto byl zvolen typ int. Element optionLabels, který obsahuje alespoň jednočlennou množinu

elementů optionLabel typu string slouží pro uložení popisu dané konstrukční skupiny pro každou jazykovou lokalizací. Jejich obsah by měl být neprázdný, ale implementace programu si s prázdným obsahem dokáže poradit.

Element options, který obsahuje elementy option slouží pro uložení všech dostupných voleb pro konkrétní konstrukční skupinu v určené specifikaci vozidla. Element option je tvořen elementem id, comparisonValue, models a labels. Id typu int slouží k identifikaci volby, optionNumber typu string je hodnota, se kterou je porovnávána druhá část záznamu z konfiguračního souboru. Tato hodnota má sedmiznakové označení tvořené písmeny číslicemi. Models obsahuje elementy model typu string, které uchovávají cestu k modelu dílu nebo konstrukční skupiny. Pro volby, které neobsahují model, by přesto měly být uvedeny mezi výběrem voleb. Takovým případem je třeba volba sluneční clony, která má dvě možnosti, ano/ne, jejíž soubor jt bude obsahovat prázdnou scénu. Tato varianta byla zvolena z toho důvodu, aby bylo možné tyto volby automaticky tvořit při hromadném nahrávání modelů. Element labels obsahující jednotlivé elementy label pak slouží podobně jako element optionLabel pro uložení popisu dané volby z konstrukční skupiny. Počet těchto elementů by měl odpovídat počtu jazyků v aplikaci.

Element allModelsViewable typu boolean složí k zamezení možnosti použití všech modelů všech voleb v rámci této konstrukční skupiny do scény. Tato řešení bylo zvoleno na základě testování aplikace, kdy při zobrazení modelů všech voleb u konstrukční skupiny jako je například motor, jehož model má velký počet polygonů, docházelo k dalšímu navýšení požadavků na paměť počítače a prodloužení doby vykreslování.

Element replaced typu int slouží k odlišení aktuálních klíčů od těch, které používají modely dílů nebo konstrukčních skupin, které už neodpovídají současnému provedení. Pokud je hodnota tohoto elementu nula, je klíč aktuální, pokud je větší, odpovídá hodnota identifikátoru klíče, kterým byl tento nahrazen.

4.5.2 Soubor vehicleItems.xml

Tento soubor slouží pro uložení seznamu vozidel. Obsahuje množinu elementů vehicleItem, které se skládají z elementu id typu int, obsahující identifikátor vozidla a elementu vehicleFolder pro uložení názvu složky vozidla.

4.5.3 Soubor vehicle.xml

Soubor vehicle.xml slouží pro uložení informací o vozidle. Tento soubor vzniká spolu s vytvořením vozidla a je uložen do složky vozidla spolu s obrázky, souborem s konfigurací vozidla a jeho modelem. Soubor je tvořen elementem vehicle, který se skládá z elementu id typu int sloužící k identifikaci vozidla. Elementy modelRange, driveVersion, chassisPattern a vehicleVersion typu int určují modelovou řadu, typ pohonu, provedení podvozku a verzi vozidla. Tyto elementy obsahují identifikátory konkrétních prvků ze souboru modelRanges.xml.

Dalším elementem je vehicleCoding typu string pro uložení označení vozidla. Toto označení odpovídá systému *označování vozidel TATRA*.[12] Element vehicleDescription typu string obsahuje textový popis vozidla, který je vytvářen prodejcem. Zařazení do kategorií je realizováno pomocí elementu categories, který se skládá z množiny elementů category typu int. Tyto elementy obsahují identifikátor kategorie ze souboru categories.xml. Element options, který se skládá z elementů option zabezpečuje propojení se seznamem klíčů. Každý element option obsahuje identifikátor klíče keyId typu int a identifikátor volby z toho klíče optionId taktéž typu int. Tyto elementy vznikají při uložení vozidla se souborem s konfigurací vozidla a slouží pro urychlení modifikace vozidla prostřednictvím aplikace prodejce, kdy odpadá nutnost získávat množinu klíčů a voleb z konfiguračního souboru.

Element images obsahující množinu elementů image typu string slouží jako seznam pro uložení obrázků vozidla. Ty jsou uloženy v podobě jejich názvu. Tento název se skládá z názvu složky vozidla a původního názvu obrázku. S elementem images souvisí element mainImage typu int. Jeho hodnota určuje obrázek, který je používán pro obrazovou prezentaci vozidla. Příkladem je zobrazení v přehledu vozidel, obrázek použitý v zobrazení informací o vozidle ve `VehicleInformationViewport` či při editaci vozidla. Pokud je hodnota elementu -1, je místo obrázku vozidla zobrazeno logo společnosti, v případě, že je hodnota větší, je zobrazen obrázek ze seznamu obrázků vozidla na odpovídající pozici.

Element webConfiguration, který je typu string, obsahuje adresu vozidla ve webovém informačním systému. Element configurationFile obsahuje název souboru obsahující konfiguraci vozidla. Poslední element actual typu int slouží k určení, zda je toto vozidlo dostupné. Pokud má element hodnotu 0, vozidlo je aktuální, pokud 1, model vozidla

[12] SMOLKA, Radomír. TATRA TRUCKS, a. s. Označování vozidel TATRA. 15. vyd. Kopřivnice, 2006. Dostupné z: partners.tatra.cz/exter_pr/vp/img/oznacovanivozidel.pdf

obsahuje modely, které už nejsou aktuální. Hodnota 2 udává, že specifikace vozidla už není dostupná a hodnota 3 určuje, že specifikace vozidla už není dostupná a zároveň model vozidla obsahuje neaktuální modely.

4.5.4 Soubor modelRanges.xml

Tento soubor slouží jako seznam dostupných provedení vozidel. Je vytvářen správcem a distribuován prodejcům prostřednictvím serverové aplikace. Je tvořen elementem version typu int označujícím verzi a množinou elementů modelRange.

Element modelRange se skládá z elementu name typu string, který v sobě uchovává název modelové řady, elementem id typu int sloužícímu k identifikaci typové řady a elementem driveVersions, který obsahuje seznam typů pohonů dostupných v konkrétní typové řadě. Každý typ pohonu je zastoupen elementem driveVersion, který se skládá z elementu id typu int identifikujícím provedení pohonu, počet kol je uložen v elementu numberOfWheels typu int. Element numberOfDrivenWheels taktéž typu int určuje počet hnaných kol. Element chassisPatterns pak reprezentuje množinu provedení podvozků. Každé provedení podvozku je zahrnuto v elementu chassisPattern, který je složen z elementu id typu int sloužícím jako identifikátor provedení podvozku. Dále elementem name typu string pro uložení názvu provedení podvozku. Tento název je třímístní o odpovídá poslednímu třem znakům z druhé sekce systému pro označování vozidel Tatra. Element vehicleVersions obsahuje množinu elementů vehicleVersion reprezentujících jednotlivé verze vozidel dostupné pro konkrétní provedení podvozku vozidla. Element vehicleVersion je složen z elementu id typu int pro identifikování verze vozidla. Elementem size typu int pro vlastní hodnotu rozvoru a elementem active typu boolean označujícím, zda je daná verze vozidla je dostupná pro objednání.

4.5.5 Soubor categories.xml

Seznam kategorií vytvořených prodejcem je uložen v souboru categories.xml. Tento soubor je tvořen elementy typu category.

Každý tento element zastupuje jednu kategorii ze seznamu kategorií. Tento element je tvořen elementem id typu int identifikujícím kategorii, dále elementem structure typu string určujícím zanoření kategorie. Tento řetězec má podobu identifikátorů jednotlivých

kategorií oddělených tečkou simulujících zanoření kategorie. Je ukončen identifikátorem této kategorie a svislou čarou.

Element name typu string obsahuje název kategorie. Posledním elementem je hidden typu boolean, který složí k označení kategorie jako skryté. Tato kategorie a její podkategorie pak budou zobrazovány pouze v editačním modu kategorií při případné upravě vozidla.

4.5.6 Soubor languages.xml

Společnost TATRA TRUCKS, a. s. plánuje využití aplikace i v zahraničí. Jedná se především o trhy v západní Evropě a Austrálii. Z toho důvodu bylo nutné implementovat do aplikace možnost změnit její jazykovou lokalizaci. Zdrojové texty pro tyto lokalizace jsou uloženy v souboru languages.xml.

Ten se skládá z elementu languages reprezentujícího množinu jazyků dostupných v této aplikaci prostřednictvím jednotlivých elementů language, které jsou dále tvořeny množinou elemntů typu string zastupujících konkrétní textový obsah, jedinou výjimkou je element id reprezentující identifikátor konkrétního jazyka. Ten odpovídá stejné hodnotě z elementu id uzlu cultureInfo ze souboru cultureInfos.xml. Tento soubor obsahuje seznam podporovaných jazyků a informace o něm, jako je směr psaní textu, hodnota pro porovnání s nastavením operačního systému.

5 Testování aplikace

Pro zjištění připravenosti nasazení aplikace do provozu je třeba provést její testování. Plnohodnotné testování aplikace je stejně náročné, někdy i náročnější než její samotný vývoj. Vzhledem k omezenému rozsahu této práce nebylo možno toto plnohodnotné testování provést. V tomto případě se testování skládá ze tří částí. První z nich je testování její funkčnosti aplikace, kde hlavním cílem je zjisti, zda nedochází ke vzniku chyb při běhu aplikace, a ověřit schopnost aplikace reagovat na nastalé situace. Dalším krokem je verifikace. *Verifikace představuje proces, jehož cílem je potvrdit, že něco – v našem případě software – vyhovuje zadané specifikaci. Při validaci se pak kontroluje, jestli vyhovuje požadavkům uživatele.*[13] Tedy jestli je pro uživatele použitelný. Může se lehce stát, že software odpovídá zadávací dokumentaci, a přesto je nevhodný pro nasazení.

Jak bylo uvedeno, vývoj aplikací probíhal za použití metodiky vývoje aplikací řízeného vlastnostmi. Díky postupnému vytváření jednotlivých základních funkcionalit systému bylo zjednodušeno jejich testování za pomoci Junit.

5.1 Identifikace problémových míst a jejich testování

Na základě studia problematiky testování byly vybrány oblasti, kterým byla věnována větší pozornost. Jedná se především o načítání dat ze souboru, posílání a přijímání dat mezi serverem a klientem a zobrazování grafických prvků s proměnlivým obsahem jako jsou obrázky, tlačítka s texty atp.

5.1.1 Konfigurování vozidla a generování modelů

Testování generování modelů v sobě zahrnovalo načítání seznamu klíčů ze souboru keys.xml, na kterém byly testovány stavy neexistence souboru, existence prázdného souboru, volby bez modelu, neexistence popisku konstrukční skupiny pro zvolený jazyk aplikace,

[13] PATTON, Ron. Testování softwaru. Vyd. 1. Praha: Computer Press, 2002, 313 s. Programování. ISBN 80-722-6636-5, s. 42.

stejně tak jako popisku volby. Místo skutečných modelů dílů nákladních automobilů TATRA byly použity modely, které poskytuje společnost Siemens pro ukázku možností jejich grafického systému. Odkazy na vozidla ve webovém IS byly nahrazeny typovými listy.

5.1.2 Znakové sady a test lokalizace

V této části bylo nejdříve testováno, jestli dokáže aplikace načítat, ukládat a zobrazovat znaky z jazyků zemí, ve kterých je plánované její nasazení. To bylo vyřešeno speciálním souborem, který obsahoval množinu znaků typických pro konkrétní lokalizaci. Výsledkem tohoto testu bylo objevení problému se zobrazením názvu modelu v jazyce nepoužívajícím latinku v prostředí aplikace JT2Go. Jako protiopatření bylo rozhodnuto zakomponovat do názvu modelu taktéž číslo konstrukční skupiny a volby. Dalším problémem je prodloužení textů. *Angličtina nebo Čeština se může zdát někdy poměrně rozvláčná, ale při překladu do jiných jazyků se přesto ukazuje, že pro vyjádření stejné věci je potřeba často většího počtu znaků. Dobré praktické pravidlo říká, že je u jednotlivých slov vhodné počítat až se 100% nárůstem počtu znaků.*[14] Výsledkem tohoto testování bylo nejdříve doplnění tlačítek popiskem, aby v případě zkrácení textu na tlačítku bylo možné obsah textu získat, posléze vytvořením grafického rozhraní, u kterého nedochází k těmto problémům.

5.1.3 Přenos dat

Jak bylo zmíněno, o distribuci dat se na přání zákazníka stará databáze běžící v prostředí Microsoft SQL Server 2012. Bylo otestováno, jak se bude aplikace správce a prodejce chovat v případě, že nebude možné navázat spojení, či v průběhu přenosu dojde k přerušení spojení. Toto testování probíhalo v lokálním prostředí pomocí zapínání a vypínání běhu jednotlivých aplikací. Výsledkem bylo zjištění, že zvolená implementace postupu aktualizace je funkční.

5.2 Verifikace systému

Díky použité metodě vývoje docházelo k postupnému ověřování jednotlivých funkčních komponent aplikace během jejich představení zadavateli, na jejichž základě se ukázalo

[14] PATTON, Ron. Testování softwaru. Vyd. 1. Praha: Computer Press, 2002, 313 s. Programování. ISBN 80-722-6636-5, s. 135.

nesprávné definování a absence některých funkcionalit, jako například hromadné nahrávání modelů do aplikace spojené s automatickým vytvořením klíčů pro vytvoření modelu vozidla, aplikování změn jednoho z těchto klíčů na všechny, které používají stejné modely, a při konfiguraci modelu vozidla, kdy nabídka konstrukčních skupin nabízela kromě aktuální i ty už nahrazené. Při verifikaci systému před testováním, zda je validní, odpovídal tento systém všem dohodnutým požadavkům.

5.3 Validace systému

Toto testování, které probíhá u zadavatele systému, má prokázat, zda je vytvořený systém připraven pro operační nasazení. Během tohoto zkušebního provozu simulujícího reálné nasazení byly zjištěny problémy s platformou JavaFX, přesněji s uvolňováním objektů, které už nejsou součástí scény, z paměti. Tento problém nebyl odhalen předem, neboť nedocházelo k testování celé aplikace, ale vždy jenom jedné její funkcionality. Podobný problém se objevoval i u ukázkových projektů společnosti Oracle. Závažnost tohoto problému neumožňovala chod aplikace po delší dobu. Jedno z uvažovaných řešení se zaobíralo možností opustit platformu JavaFX a vytvořit grafické uživatelské rozhraní prostřednictvím knihovny Swing. Toto řešení by však neumožnilo naplnění časového plánu. Jako východisko se jevilo použití trochu jiného způsobu práce s jednotlivými objekty grafického rozhraní s příchodem nové verze jazyka JavaFX se ukázalo, že některé problémy přetrvávají.

6 Návrhy na další rozvoj systému

6.1 Integrace webového informačního systému

Několikrát zmíněné jednosměrné propojení webového informačního systému pro objednávání vozidel s aplikací prodejce za pomoci souboru s konfigurací vozu taktéž není ideální. Zakomponování tohoto systému do aplikace prodejce by umožnilo provádění změn konfigurace vozidla na základě modifikace konstrukčních skupin a voleb při konfiguraci modelu vozidla. I když se tento návrh může jevit v současném trendu jako zpátečnické řešení. Datový objem jednotlivých dílů modelu vozidla je i při stále rostoucích rychlostech připojení příliš velký.

6.2 Využití textur pro barevné provedení vozidla

Dalším prvkem, který by mohl být zakomponován do aplikace, by bylo využití textur pro barevné provedení vozidla. V současném stavu je barva dílu určena v modelu vozidla. Proto pro každé barevné provedení dílu je nutné mít zvláštní soubor. Pro toto rozšíření by bylo třeba modifikovat strukturu souboru keys.xml a tříd `Key` a `Option` pro uložení názvu souboru obsahující texturu pro tento model, stejně tak jako tříd, které s nimi pracují. Umožnění využívání této funkce by si vyžádalo taktéž investici společnosti TATRA TRUCKS, a. s. do rozšíření konstrukčního softwaru v rámci desítek tisíc korun.

6.3 Vytvoření aplikace pro zobrazení modelu vozidla

Model vozidla vzniká na základě obsahu souboru plmxml. Ten je pevně stanoven a při zobrazení do něj není možné přidat aplikací prodejce žádné další modely. Jak je popsáno, existují programy, které dokáží soubory typu jt zobrazit, avšak jejich implementace není dostatečně kvalitní na to, aby dokázaly zobrazit vizualizaci vozidla. Pokud by se podařilo vyvinou prostředí, které by dokázalo nahradit program JT2Go, bylo by možné ho zakomponovat do aplikace prodejce a tím umožnit vkládat a odebírat modely přímo do scény. Vlastní prostředí by mohlo poskytovat bohatší nástroje jako je například možnost

posunu modelů ve scéně. Takový systém by s dostatečnou podporou CAD pracovníků tvořících modely nejrůznějších dílů a zařízení, které se na nákladních automobilech objevují nebo by se mohly objevit, dal do rukou prodejců velice silný nástroj, který by umožnil prezentovat naprosto přesné řešení potřeb zákazníka.

7 Závěr

V této práci jsem se pokusil navrhnout a vytvořit prototyp systému, který by poskytl společnosti TATRA TRUCKS, a. s. a prodejcům jejich nákladních automobilů prostředek k prezentaci a modifikaci provedení vozidel nabízených zákazníkům způsobem, který nemá k dispozici žádná jiná automobilka (založeno na tvrzení zaměstnanců společnosti). Už při zadání této práce jsem byl varován školitelem o náročnosti tohoto projektu, což se potvrdilo při realizaci základní funkčnosti, tj. generování modelu vozidla z konfiguračního souboru. Další vývoj aplikace odhaloval stále další a další problémy a návrhy na zlepšení, které znamenaly několik dílčích změn následovaných prakticky přepracováním aktuálního stavu. Prokázalo se, že vývoj pomocí metody FDD dokáže při dostatečném množství zdrojů nejlépe splnit skutečné požadavky uživatele.

Návrh struktury XML dokumentů pro propojení konfigurační a zobrazovací části

Toto bylo splněno. Jak prokázalo testování těchto datových struktur, především souboru keys.xml, poskytuje aplikaci prodejce veškerá data k zajištění požadované funkcionality, tedy vytvoření modelu vozidla.

Vytvoření serverové aplikace poskytující data pro konfiguraci a zobrazení vozidla

Jak bylo uvedeno v textu, aplikace na přání zákazníka nahrazena databází a testování potvrdilo správnost její implementace. Přesto byl zpracován návrh aplikace včetně jejího zabezpečení za pomoci veřejných a soukromých klíčů.

Vytvoření aplikace pro prodejce umožňující zobrazování a konfiguraci vozidel

Navržená a vytvořená aplikace prodejce neodpovídá požadavkům na ni kladených v zadaní. Umožňuje pouze generování modelu vozidla ze souboru. Zbylá funkcionalita byla

implementována v předchozích verzích této aplikace používající nevzhledné GUI, avšak drobné změny vedly k nekompatibilitě datových objektů mezi nimi, tudíž neumožnily jejich rychlou implementaci.

Vytvoření aplikace pro správu dat umožňující konfigurování vozidel

Funkcionalita navržené a vytvořené aplikace pro správu dat byla na základě jejího testování rozšířena o funkce zvyšující efektivitu práce. Takto upravená aplikace umožňuje rychlejší správu klíčů pro vytváření modelu vozidla a je připravena pro generování aktualizací do databáze.

Seznam použité literatury

CARTER, Michael, Jianbing SASHANK, Ganti HUANG a Bo XU. SIEMENS PLM SOFTWARE. JT File Format Reference Version 9.5 Rev-A. 2010. Dostupné z: http://www.plm.automation.siemens.com/en_us/products/open/jtopen/technology/index.shtml#lightview%26uri=tcm:1023-134827%26title=JT%20v95%20File%20Format%20Reference%20Rev-D%26docType=.pdf

HAJDIN, Tomáš. Agilní metodiky vývoje software [online]. Brno, 2005 [cit. 2012-04-25]. Dostupné z: http://is.muni.cz/th/39440/fi_m/. Diplomová práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Barbora Bühnová.

JT (visualization format). In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-04-25]. Dostupné z: http://en.wikipedia.org/wiki/JT_(visualization_format)

JT2Go. SIEMENS PLM SOFTWARE. JT2Go: Siemens PLM Software [online]. 2012 [cit. 2012-04-25] Dostupné z http://www.plm.automation.siemens.com/en_us/products/teamcenter/lifecycle-visualization/jt2go/index.shtml

PATTON, Ron. Testování softwaru. Vyd. 1. Praha: Computer Press, 2002, 313 s. Programování. ISBN 80-722-6636-5.

PAVLÍČKOVÁ, Jarmila a Luboš PAVLÍČEK. Vývoj klient/server aplikací v Javě [online]. Vyd. 1. Praha: Oeconomica, 2004, 94 s. [cit. 2012-04-25]. ISBN 80-245-0791-9. Dostupné z: http://java.vse.cz/Java/Skripta

ROSENKRANZ, Karel. Nákladní automobily Tatra. Vyd. 1. Kopřivnice: TATRA TRUCKS, 2007, 661 s. ISBN 978-80-239-9877-1.

RSA encryption in Java. COFFEY, Neil. JAVAMEX UK. RSA encryption in Java [online]. 2011 [cit. 2012-04-25]. Dostupné z: http://www.javamex.com/tutorials/cryptography/rsa_encryption.shtml

SIEMENS PLM SOFTWARE. PLM Vis – Software toolkit for collaborative view and markup: Supported formats. 2011. Dostupné z: http://www.plm.automation.siemens.com/en_us/products/open/vis/#lightview%26uri=tcm:1023-11334%26title=PLM%20Vis%20-%20Software%20Toolkit%20for%20Collaborative%20View%20and%20Markup%20-%20PLM%20Components%20Fact%20Sheet%20-%206875%26docType=.pdf

SIEMENS PRODUCT LIFECYCLE MANAGEMENT SOFTWARE INC. PLMXML Schema. 2008. Dostupné z: http://www.plm.automation.siemens.com/en_us/products/open/plmxml/schemas.shtml

SMOLKA, Radomír. TATRA TRUCKS, a. s. Označování vozidel TATRA. 15. vyd. Kopřivnice, 2006. Dostupné z: partners.tatra.cz/exter_pr/vp/img/oznacovanivozidel.pdf

TAYLOR, A. Hacking bez tajemství: java a J2EE. Vyd. 1. Brno: Computer Press, 2003, 409 s. ISBN 80-722-6868-6.